Abstract—We propose a new way to save energy in adaptive processors. According to an execution context the custom instruction set of an adaptive processor is selectively 'muted' at run time and thus the energy efficiency is significantly increased. Implemented are multiple so-called 'muting modes' each leading to particular leakage energy savings. A key challenge of this work is to determine which of the muting modes are beneficial for which part of the custom instruction set in a specific execution context. We demonstrate the feasibility by means of an H.264 video encoder (although not limited to that) for various technology nodes. The complex and unpredictable processing behavior of an H.264 encoder represents thereby a real-world scenario. Our results show on average more than 30% energy savings compared to state-of-the-art. We claim that adaptive processors (and reconfigurable computing in general) would be far more energy efficient if FPGA vendors would provide a basic infrastructure that is necessary to exert our novel technique.

Keywords: low power, energy minimization, power-gating, leakage, run-time adaptation, reconfigurable computing, adaptive processors, custom instruction

I. INTRODUCTION AND MOTIVATION
The advancement of FPGAs has led to the emerging architecture trends of adaptive processors that couple a processor core with a reconfigurable fabric (i.e. an embedded FPGA) [1, 2]. Adaptive processors [26] bridge the gap between ASICS and GPPs by providing an adaptive custom instruction set which is adapted at run time to meet applications’ demands and constraints.

Besides dynamic and leakage power, adaptive processors suffer from the power consumed when reconfiguring the instruction set. As the fabrication trend crosses the 65 nm technology node, leakage becomes imperative in the energy-aware design of adaptive processors [13]. Hardware shutdown may be performed to reduce the leakage energy of adaptive processors. The following components of a reconfigurable fabric can be individually shut down:

- **Logic**: Configurable Logic Blocks (CLBs) and programmable interconnect switch matrices (i.e. the routing resources that connect various CLBs)
- **Configuration SRAM**: The SRAM1 cells that store the control bits which define the configuration of the Logic

Several academic research projects have proposed low-power approaches in FPGAs (see Section II). They deploy shutdown schemes that statically determine the parts of a reconfigurable fabric (Logic or Logic + Configuration SRAM) that can be powered-off [9, 13, 14]. However, a shutdown infrastructure that can be controlled at run time is yet unavailable in commercial FPGAs (with support of partial reconfiguration).

Moreover, in the above-mentioned shutdown schemes, the usage/state of a particular hardware is monitored and the shutdown signal is issued to the hardware, e.g., after the hardware is idle for a certain threshold time (e.g. [13]). These approaches mainly focus on hardware-oriented shutdown of the reconfigurable fabric irrespective of the application context (e.g. control flow, application priority etc.) and execution length of computational hot spots2. Therefore, idle periods of Custom Instructions (CIs) usage cannot be exploited for the purpose of energy savings.

When targeting adaptive processors, it is no longer efficient to employ the above-mentioned schemes, as it cannot be determined at compile time which CIs will be reconfigured on which part of the reconfigurable fabric. As a result, these hardware-oriented shutdown schemes suffer from the limitation of inflexibility and are highly dependent upon the underlying shutdown policy.

We propose a new technique that shuns the leakage energy at the abstraction level of CIs (i.e. an instruction set oriented shutdown). We name this concept as selectively muting the CIs. Our technique uses a power shutdown infrastructure in order to define what we call CI muting modes (see Table I) each leading to particular leakage energy savings. Our concept relates leakage energy to the execution context of an application, thus enabling a far higher potential for energy savings. This work aims at exploiting this potential.

**TABLE I: CUSTOM INSTRUCTION (CI) MUTING MODES**

<table>
<thead>
<tr>
<th>Logic</th>
<th>Config. SRAM</th>
<th>CI Muting Modes</th>
<th>Use-Case for the CI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ON</strong></td>
<td><strong>ON</strong></td>
<td>I. Non-Muted (NM-CI)</td>
<td>CI is demanded or it is scheduled to be reconfigured soon</td>
</tr>
<tr>
<td><strong>ON</strong></td>
<td><strong>OFF</strong></td>
<td>N/A</td>
<td>N/A (turning the Logic on but the configuration off may lead to undesired system behavior)</td>
</tr>
<tr>
<td><strong>OFF</strong></td>
<td><strong>ON</strong></td>
<td>II. Virtually-Muted (VM-CI)</td>
<td>CI is not demanded, but expected to be reconfigured soon</td>
</tr>
<tr>
<td><strong>OFF</strong></td>
<td><strong>OFF</strong></td>
<td>III. Fully-Muted (FM-CI)</td>
<td>CI is not demanded and it is not scheduled to be reconfigured soon</td>
</tr>
</tbody>
</table>

We use the following definitions:

- **Partially Reconfigurable Containers (PRC)**: Adaptive processors (see [1, 2, 3, 11]) partition their reconfigurable fabric into so-called Partially Reconfigurable Containers (PRCs, see Section III). They are reconfigured at run time to implement hardware accelerators.

- **Hardware Accelerator (HA)**: It is a reconfigurable hardware module that exploits parallelism at the operator level.

**Custom Instruction (CI)**: A CI is composed of one or more HAs. Different kind of HAs may be used in one or more instantiations to realize distinct implementation alternatives of a desired functionality in form of a particular CI. They may differ in execution time and energy efficiency. For each CI functionality one implementation alternative exists that is executed solely on the core processor (i.e. without using HAs). A CI may be muted through one of the following muting modes (see Table I):

- **Mode I: Non-Muted CI (NM-CI)**: CI is active and operational.
- **Mode II: Virtually-Muted CI (VM-CI)**: CI cannot be executed due to the powered-off Logic. No reconfiguration is required in order to deploy this CI as its Configuration SRAM is kept powered-on. Hence, the otherwise necessary reconfiguration ener-

---

1 In Xilinx FPGAs, 38% of the leakage power is consumed by the configuration SRAMs [17].

2 Throughout this paper, a hot spot denotes a computational hot spot that contains compute-intensive application parts (kernels).
gy is not consumed. Therefore, the reduction in leakage energy is lower compared to Mode III (below). Mode II is beneficial when a subset of CIs is not demanded for a rather short period. Mode III: Fully-Muted CI (FM-CI): CI is not operational, as both Logic and Configuration SRAM are powered-off. This significantly reduces the leakage energy. However, in order to deploy this CI, a reconfiguration is required which costs reconfiguration energy and latency. Mode III is beneficial when a subset of CIs is not demanded for a rather long period.

A. Problem and Motivational Scenarios

Various challenging questions arise given the three muting modes from above: For instance, whether VM-CIs or FM-CIs provide more energy reduction when both leakage and reconfiguration energy is considered. This is reflected by the following equation (Eq. 1) where the decision of a muting mode depends upon the execution length of a hot spot:

\[ P_{\text{VM-CIs}} \times T_{\text{HotSpot}} < P_{\text{FM-CIs}} \times T_{\text{HotSpot}} + P_{\text{reconf}} \times T_{\text{reconf}} \]  

(1)

Additionally, it needs to be clarified whether this decision can be determined statically [9, 10, 13, 14] or whether it requires a dynamic decision. As we will see later on, this depends on a specific run-time scenario i.e. it depends on the application’s execution context.

Fig. 1 shows three different run-time scenarios and compares the energy requirements of VM- and FM-CIs. It is noticeable that the energy requirements of FM-CIs are significantly lower for most of the time. However, when a FM-CI is demanded again for executing a hot spot, its muting mode is switched to NM-CI. It increases the energy requirements significantly due to the demanded reconfiguration of these FM-CIs (see the 2nd addend on the right hand side of Eq. 1). This fact is noticeable in Fig. 1a) where the muting duration of FM-CIs is too short to amortize the reconfiguration overhead \( P_{\text{reconf}} \times T_{\text{reconf}} \). An alternate scenario can also been seen in Fig. 1a) where the leakage energy in VM-CIs is large compared to the reconfiguration energy of FM-CIs. Such a scenario may result due to a higher \( V_{\text{dd}} \) in order to support a higher clock frequency, which is required to fulfill the performance constraints. There, VM-CIs lead to the larger energy consumption.

![Fig. 1a](image1a.png)

**Fig. 1a:** Comparing energy requirements of the VM-CIs and FM-CIs in different scenarios

![Fig. 1b](image1b.png)

**Fig. 1b:** Comparing energy requirements of the virtually- and fully-muted CIs in different scenarios

B. Novel Contributions

We address the above-mentioned challenges with what we call selectively muting CIs in an energy-aware adaptive processor. Our technique uses various muting modes that enable leakage energy reduction at the abstraction level of CIs.

Our novel contribution is:

- determining at run-time which subset of CIs should be put into which muting mode (Table I) at which time by evaluating at run-time the possible associated energy benefit (a joint function of leakage, dynamic, and reconfiguration energy).

In effect, we propose to selectively mute a custom instruction set of an adaptive processor. The associated potential energy savings have not been exploited by state-of-the-art approaches. It is especially beneficial for highly flexible custom instruction set architectures, such as RISPP [27].

We have performed a comprehensive evaluation of our technique under distinct performance and area constraints for three technology nodes. We aim with our technique at applications that are (at design- and/or compile-time) hard to predict in behavior.

The paper organization: Related work is discussed in Section II. Section III provides the model of muted CIs. Section IV focuses on the main contribution of this paper, i.e. our selective custom instruction set muting technique. Section V presents results and evaluation followed by a conclusion in Section VI.

II. RELATED WORK

An overview of adaptive processors can be found in [1, 2]. Previous approaches in adaptive processors like PRISC [28], DISC [29], OneChip [30], Chimaera [31], and Molen [11] have mainly concentrated on improving the performance by offering a set of CIs, which is reconfigured at run time to meet applications’ demands and constraints. This reconfiguration process may consume a noticeable amount of energy. Consequently, the major shortcoming of these adaptive processors is their high energy consumption compared to ASICs and lack of efficient energy management features [10].

These adaptive processors deploy an FPGA-like fabric to support adaptivity. Therefore, these processors (and reconfigurable computing in general) would be far more energy efficient if FPGA vendors would provide a basic infrastructure that provides a foundation to exert high-level power and energy management schemes like the one we propose in this paper. Several academic research projects have already made the case for such shutdown infrastructure [13, 14]. In the following, we present the related work for low-power techniques in FPGAs.

An overview of low-power techniques for FPGAs is presented in [4]. A detailed analysis of leakage power in Xilinx FPGAs is...
performed in [17], highlighting the significance of leakage reduction in FPGAs especially when considering mobile devices. An analysis of dynamic power consumption in Virtex-II FPGAs is presented in [5]. An FPGA architecture evaluation framework for power efficiency analysis is presented in [6] predicting leakage to be dominant for future technologies. A fine-grain hardware shutdown scheme has been introduced in [7] using sleep transistors at gate level. It uses three sleep transistors for CLBs and one for routing resources in order to obtain a fine-grained leakage control. The approach in [8] uses body biasing, multi-Vt logic, and gate biasing to reduce the leakage in FPGAs.

Xilinx research labs introduced a 90 nm low-power FPGA Pika for battery-powered applications that supports voltage scaling, hardware shutdown, and a low-leakage Configuration SRAM [10]. However, Xilinx have not yet introduced such infrastructure in their commercial products. A region-constrained placement is proposed in [13] to reduce the leakage energy in FPGAs. At the time of placement, it targets to increase the number of unused regions that can be shutdown at run time while keeping the Configuration SRAM powered-on. Authors in [14] propose fine-grain leakage optimization by shutting down the Configuration SRAM of unused LUTs. A hardware-oriented shutdown technique for four sleep modes (achieved by applying different bias to the footer device) is proposed in [15] to provide a trade-off between wake-up overhead and leakage savings. Authors in [16] show that too-high Vt transistors for leakage reduction in the Configuration SRAM result in increased SRAM write time (i.e. increased reconfiguration time), that leads to higher reconfiguration energy. The approach in [18] reduces the configuration energy in adaptive processors by using voltage scaling on the configuration process. It deploys configuration prefetching and parallelism to generate excessive system idle time. A low-power version of the Warp Processor is introduced in [20] that integrates voltage and frequency scaling in the Warp Processor to dynamically reduce the energy consumption. The design of Vdd-gateable and Vdd-programmable interconnect switches is introduced in [21] for FPGAs with configurable supply voltage.

In summary, related work shuns leakage energy by means of hardware-related shutdown techniques. In contrast, our new technique uses a power shutdown infrastructure in order to define what we call custom instruction muting modes. These muting modes allow us to shun leakage at the abstraction level of custom instructions and hence relate leakage energy to the execution context of an application or algorithm. This bears a far higher potential for leakage energy savings. It is the goal and challenge of this work to exploit this potential.

III. MODEL OF MUTED CUSTOM INSTRUCTIONS

Before proceeding to our Custom Instruction (CI) set muting technique, we present the model of muted CIs. Fig. 2 provides an overview of the infrastructure needed to apply our technique for adaptive processors. Multiple PRCs are connected to a core pipeline. Each PRC is composed of multiple reconfigurable tiles and each tile contains Configurable Logic Blocks (CLBs) and programmable interconnect switch matrices (i.e. the routing resources that connect different CLBs). Control bits define the configuration of logic and routing resources and are stored in local Configuration SRAM, as shown in Fig. 2.

In order to realize different CI muting modes (as shown in Table I), each PRC contains two independent sleep transistors for Logic and Configuration SRAM. Note that these two sleep transistors are used for all tiles of a particular PRC, whereas two different PRCs use different sleep transistors. The control signal for these sleep transistors for a given muting mode is specified in Table I.

This power shutdown infrastructure is currently not available in today’s commercial FPGAs. Therefore, previous work in adaptive processors has not explored such a leakage energy reduction technique at the instruction set level. We believe that if FPGA vendors would provide this simple infrastructure, we would have a great opportunity to exert our CI muting technique. Consequently, adaptive processors would be far more energy efficient.

A. Some Definitions: Hardware Accelerators (HAs) embody Custom Instructions (CIs)

We represent each alternative of a CI implementation (see explanation in Section I) as a vector \( \mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{N}^n \), where \( c_i \) denotes the number of instances of the \( i^{th} \) HA that are required to implement the implementation alternative. The function \( \mathbf{c}.\text{Fastest}(\mathbf{a}) \) returns the fastest implementation alternative of the same CI that is implemented by \( \mathbf{c} \) for a given set of available HAs \( \mathbf{a} \). The determinant of \( \mathbf{c} \) is defined as \( |\mathbf{c}| = \sum_{i=1}^{n} c_i \cdot \text{Lat}[c_i] \), i.e. the total number of HAs that are required to implement \( \mathbf{c} \).

IV. SELECTIVE INSTRUCTION SET MUTING

A. Operational Flow of our Technique

Our technique evaluates a possible energy benefit (a function comprising leakage, dynamic, and reconfiguration energy, see Section IV.B) of different CIs to select an appropriate muting mode for the corresponding PRCs at run time. Fig. 3 presents a time line showing the execution sequence of previous-, current-, and upcoming hot spots along with the point of time where the CI muting mode is selected.

![Fig. 2: Infrastructure necessary to exert our technique](image)

This power shutdown infrastructure is currently not available in today’s commercial FPGAs. Therefore, previous work in adaptive processors has not explored such a leakage energy reduction technique at the instruction set level. We believe that if FPGA vendors would provide this simple infrastructure, we would have a great opportunity to exert our CI muting technique. Consequently, adaptive processors would be far more energy efficient.

A. Some Definitions: Hardware Accelerators (HAs) embody Custom Instructions (CIs)

We represent each alternative of a CI implementation (see explanation in Section I) as a vector \( \mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{N}^n \), where \( c_i \) denotes the number of instances of the \( i^{th} \) HA that are required to implement the implementation alternative. The function \( \mathbf{c}.\text{Fastest}(\mathbf{a}) \) returns the fastest implementation alternative of the same CI that is implemented by \( \mathbf{c} \) for a given set of available HAs \( \mathbf{a} \). The determinant of \( \mathbf{c} \) is defined as \( |\mathbf{c}| = \sum_{i=1}^{n} c_i \cdot \text{Lat}[c_i] \), i.e. the total number of HAs that are required to implement \( \mathbf{c} \).

IV. SELECTIVE INSTRUCTION SET MUTING

A. Operational Flow of our Technique

Our technique evaluates a possible energy benefit (a function comprising leakage, dynamic, and reconfiguration energy, see Section IV.B) of different CIs to select an appropriate muting mode for the corresponding PRCs at run time. Fig. 3 presents a time line showing the execution sequence of previous-, current-, and upcoming hot spots along with the point of time where the CI muting mode is selected.

![Fig. 3: Time-line showing the execution sequence of hot spots and the situation for a CI Muting decision](image)

Fig. 4 presents the flow of the CI muting technique. It is triggered ahead of a hot spot execution. The key inputs are:

- a list of hardware accelerators that are available from the previous hot spot (\( p \) and

---

355
lists of hardware accelerators that are required by the current and the upcoming hot spots (\( c, \bar{c}, n \), see Fig. 5).

The algorithm returns non-muted, virtually-muted, and fully-muted PRCs for the current hot spot (PRCNM, PRCVM, PRCFM).

The four major steps are:

Step 1) The PRCs to fulfill the hardware accelerator requirements of the current hot spot (see Fig. 5) are kept in non-muted mode (PRCNM, i.e. PRCs in active state). Afterwards, the hardware accelerators required for \( \bar{c} \) are checked if they are already available in PRCs (i.e. common hardware accelerators in \( \bar{c} \) and \( \bar{p} \)). If yes, then these PRCs are added to the PRCNM list.

Step 2) Afterwards, the requirements of the upcoming hot spot are predicted (more details in Section IV.C) and virtually-muting PRCs (PRCVM) are determined. At the start of the current hot spot, the hardware accelerators that are available from the previous hot spot are compared to the hardware accelerators required by the upcoming hot spot.

- If a hardware accelerator is currently available, not needed for the current hot spot, but needed again for the upcoming hot spot, it is a candidate for the virtually-muted mode (Fig. 5) and it is added to the candidate list.

- Then, the maximum number of virtually-muted hardware accelerators (HAvM) is computed by considering the requirements of the current and the upcoming hot spots and the total number of PRCs.

- However – depending upon the requirements of the current hot spot – the maximum number of HAvM may be smaller than the total number of virtually-muting mode candidates as some of the PRCs may need to be reconfigured to fulfill the performance requirements of the hot spot. Therefore, the ‘determineHAvM’ function (details in Section IV.B) evaluates the energy benefit of all candidates for the virtually-muting mode. It then chooses the one that provides the highest energy benefit among all candidates.

- If the energy benefit (no additional reconfiguration required) overcomes the overhead (larger leakage) then the PRC of HAvM are added into the PRCVM list. Alternatively, no PRC is put into the PRCVM list.

- The function ‘determineHAvM’ is iteratively executed until maximum number of HAvM is zero.

Step 3) If some of the hardware accelerators required by the current hot spot are not available, then more PRCs are kept in non-muted mode as they will be reconfigured to fulfill the requirements of the current hot spot. These PRCs are added into the PRCNM list. Those PRCs where the non-muted mode is not beneficial or which are not needed in the current and the upcoming hot spots are put into the fully-muted mode (i.e. added to the PRCFM list), as they may be used rather late during the application execution flow.

Step 4) In the last step, PRCNM, PRCVM, PRCFM lists are sent to the Mute-Mode Controller (see Fig. 2).

Now, we will present the details of the ‘determineHAvM’ (from Fig. 4) function for identifying a virtually-muted PRC.

B. Analyzing the Energy Benefit of Muting

Fig. 7 shows the pseudo code for identifying one hardware accelerator for virtually-muting (HAvM) out of all virtually-muting candidates. The key inputs are: virtually-muting candidates (\( \bar{c}, \bar{p}, \bar{n} \)), HAs required for the upcoming hot spot (\( \bar{n} \)), set of implementation al-

![Flow for selecting a muting mode for the Custom Instruction (CI) set](image-url)
tterns for CIs that are expected to be required for the upcoming hot spot (CI\textsubscript{next}), expected execution time of the current hot spot (\textit{t}\text{Exec\textsubscript{curr_hotspot}}), and a table of the CI weighting factors (\(\omega_{ci}\)).

Each hardware accelerator in the candidate list \(\mathbb{s}\) is evaluated for the energy benefit calculation (line 7-16 in Fig. 7). The hardware accelerator that provides the highest energy benefit among all candidates \(\mathbb{s}\) is then chosen as the one HA\textsubscript{VM} (line 13-15). There are four parts for the energy evaluation (line 12):

a) **Reconfiguration Energy Benefit** (\(E_{\text{ReconfBenefit}}\), line 8): When a hot spot starts executing, its hardware accelerators are reconfigured into PRCs. In case a hardware accelerator is still available after the execution of the current hot spot, another reconfiguration is required for the upcoming hot spot. Thus, a HA\textsubscript{VM} provides an energy benefit of one saved reconfiguration. Moreover, it also results in a latency improvement of one reconfiguration (approx. 0.63 ms) compared to the fully-muted PRC.

b) **Leakage Energy Benefit** (\(E_{\text{LeakBenefit}}\), line 9): As the HA\textsubscript{VM} will be available when the upcoming hot spot starts executing, the CIs of that hot spot may execute in a faster implementation alternative compared to the case when it is not available. This results in a performance improvement for the upcoming hot spot compared to the fully-muted hardware accelerator (see Eq. 2).

\[
L_{\text{benefit}}(\tilde{p}, \text{HA}) = \sum_{i\in\mathbb{CI}} \left( \omega_{ci}(\tilde{x}, CI) * (\tilde{x}, CI)_{\text{Fastest}}(\tilde{p}, \text{HA})_{\text{Latency}} - (\tilde{x}, CI)_{\text{Fastest}}(\tilde{p})_{\text{Latency}} \right) \quad (2)
\]

Each HA\textsubscript{VM} may expedite multiple CIs, where each CI has a different weighting factor (\(\omega_{ci}\), see Section IV.C) depending upon its execution frequency and execution pattern in the hot spot. Faster execution of the upcoming hot spot will reduce the overall leakage energy of both the core processor and the reconfigurable fabric. Therefore, leakage savings are computed for each virtually-muting candidate by considering \(\omega_{ci}\) of the CIs that are accelerated by this candidate.

c) **Leakage Energy Overhead** (\(E_{\text{LeakOverhead}}\), line 10): Leakage occurs in a virtually-muted PRC (due to the powered-on Configuration SRAM) for the whole duration of the current hot spot execution. Therefore, this overhead needs to be considered for the energy benefit function (line 12).

d) **Dynamic Energy Difference** (\(E_{\text{DynE diff}}\), line 11): Different implementation alternatives of a CI vary in their dynamic power and energy consumption. Therefore, a HA\textsubscript{VM} may bring an energy benefit or overhead due to a different CI implementation alternative as shown in Eq. 3.

\[
dynE_{\text{diff}}(\tilde{p}, \text{HA}) = \sum_{i\in\mathbb{CI}} \left( \omega_{ci}(\tilde{x}, CI) * (\tilde{x}, CI)_{\text{Fastest}}(\tilde{p}, \text{HA})_{\text{E}}(\text{E}) - (\tilde{x}, CI)_{\text{Fastest}}(\tilde{p})_{\text{E}}(\text{E}) \right) \quad (3)
\]

The computational complexity for calculating the energy benefit is \(O(\text{numMaxHA}\text{VM} * |\mathbb{CI}|)\). Fig. 5 shows that \(|\mathbb{CI}|\) is typically much smaller than the total number of hardware accelerators that fit onto the reconfigurable fabric at a certain time.

Note: the wake-up energy for virtually-muted and fully-muted PRC are 3.5 and 7.0 pWs (for the sleep transistor design of [10]), respectively. However, the energy for reconfiguring one PRC is 147 \(\mu\text{Ws}\), i.e. more than \(10^6\) times bigger (as we will see in Section V). Therefore, we do not include PRC reactivation energy overhead in the cost function, as it does not affect the muting decision.
C. Hot Spot Requirement Prediction: Computing the Weighting Factors for CIs

Different CIs of a hot spot may have different execution patterns (see Fig. 6). These execution patterns depend upon the following three parameters:

- expected execution frequency of CIs
- the time from the start of a hot spot until the first execution of these CIs
- the average time between two executions of the same CI

The expected execution frequency is predicted by a light-weight online monitoring scheme (we have implemented the scheme from [26]), while the other two parameters are obtained using an average case from off-line profiling (to avoid excessive run-time overhead of our scheme).

Depending upon the above three parameters a weighting factor \( \omega_{CI} \) is computed for each CI. It represents the relative contribution of a CI (compared to other CIs) for the accelerated execution of a hot spot. A CI executing earlier in a hot spot has a higher importance. If only one PRC shall not be used in the current hot spot, but two virtually-muting candidates are available, then it may be more beneficial to maintain the hardware accelerator for the earlier executing CI (i.e. setting it to virtually-muted mode).

To calculate \( \omega_{CI} \), the time line is partitioned into multiple slots, each equal to the reconfiguration time of a hardware accelerator. Since the performance of a CI may only change after a reconfiguration is completed, the number of CI executions \( \#EX_{EXEC_{TS}} \) is computed for each time slot \( T_{S} \) independently. Similarly, CIs executing in the earlier time slots have more weight than the later ones in the same hot spot (denoted by \( F_{TS} \)). Considering there are \( n \) time slots, \( \omega_{CI} \) of a CI \( 'X' \) can be computed as shown in Eq. 4.

\[
\omega_{CI}(X) = \sum_{i=1}^{n}(\#XE_{EXEC_{TS}} * F_{TS})
\]  

V. EXPERIMENTS, RESULTS, AND DISCUSSION

A. Test Conditions and Experimental Setup

Table II presents the basic input parameters and test conditions (without corresponding sources of information) used in our experiments. From the measurements on our FPGA-based prototype, we determined that \( P_{record} = 0.234 \text{ W} \) and \( T_{record} = 0.63 \text{ ms}^3 \) (details of power measurements are beyond the scope of this paper and can be found in [25]).

Table III shows the power consumption and latencies of different CI implementation alternatives used in the experiments for two cases of PRCs at 65 nm and 40 nm. The power consumption of CI implementation alternatives on 40 nm (Virtex-6) is less than that on 65 nm (Virtex-5) due to the low-power architectural improvements in Virtex-6 [24]. The proposed scheme is evaluated on the RISPP reconfigurable processor [27] running at 50 MHz.

B. Fairness of Comparison to State-of-the-Art

As discussed in Section I, state-of-the-art approaches employ hardware-oriented shutdown, which has a different abstraction level for shutdown decision compared to our selective CI muting technique (i.e. an instruction set oriented shutdown). In order to provide a fair comparison, we have deployed the hardware-oriented shutdown concept of [13] and [14] (i.e. predetermined the components of PRCs that can be shutdown at run time) to realize two pre-determined muting modes as follows:

1) Predetermined Virtually Muting (Pre-VM) technique based on the hardware-oriented shutdown of [13]: it always puts the temporarily unused CIs into virtually-muting mode as the hardware-oriented shutdown of [13] only supports switching-off of Logic and it always keeps the Configuration SRAM powered-on.

2) Predetermined Fully Muting (Pre-FM) technique based on the hardware-oriented shutdown of [14]: it always puts the temporarily unused CIs into fully-muting mode as the hardware-oriented shutdown of [14] supports the combined switching-off of both Logic and Configuration SRAM.

In the following, we will present the energy consumption comparison of the above-mentioned techniques for given performance constraints, such that in these particular scenarios, all techniques meet their performance constraints. Different performance constraints correspond to changing application contexts.

For further fairness of comparison, we have provided the same set of CIs and hardware accelerators to all of the competitors as follows:

1) Predetermined Virtually Muting (Pre-VM) technique based on the hardware-oriented shutdown of [13]: it always puts the temporarily unused CIs into virtually-muting mode as the hardware-oriented shutdown of [13] supports switching-off of Logic and it always keeps the Configuration SRAM powered-on.

2) Predetermined Fully Muting (Pre-FM) technique based on the hardware-oriented shutdown of [14]: it always puts the temporarily unused CIs into fully-muting mode as the hardware-oriented shutdown of [14] supports the combined switching-off of both Logic and Configuration SRAM.

In the following, we will present the energy consumption comparison of the above-mentioned techniques for given performance constraints, such that in these particular scenarios, all techniques meet their performance constraints. Different performance constraints correspond to changing application contexts.

For further fairness of comparison, we have provided the same set of CIs and hardware accelerators to all of the competitors as follows:

3) Predetermined Virtually Muting (Pre-VM) technique based on the hardware-oriented shutdown of [13]: it always puts the temporarily unused CIs into virtually-muting mode as the hardware-oriented shutdown of [13] only supports switching-off of Logic and it always keeps the Configuration SRAM powered-on.

4) Predetermined Fully Muting (Pre-FM) technique based on the hardware-oriented shutdown of [14]: it always puts the temporarily unused CIs into fully-muting mode as the hardware-oriented shutdown of [14] supports the combined switching-off of both Logic and Configuration SRAM.

In the following, we will present the energy consumption comparison of the above-mentioned techniques for given performance constraints, such that in these particular scenarios, all techniques meet their performance constraints. Different performance constraints correspond to changing application contexts.

For further fairness of comparison, we have provided the same set of CIs and hardware accelerators to all of the competitors as follows:

5) Predetermined Virtually Muting (Pre-VM) technique based on the hardware-oriented shutdown of [13]: it always puts the temporarily unused CIs into virtually-muting mode as the hardware-oriented shutdown of [13] only supports switching-off of Logic and it always keeps the Configuration SRAM powered-on.

6) Predetermined Fully Muting (Pre-FM) technique based on the hardware-oriented shutdown of [14]: it always puts the temporarily unused CIs into fully-muting mode as the hardware-oriented shutdown of [14] supports the combined switching-off of both Logic and Configuration SRAM.

In the following, we will present the energy consumption comparison of the above-mentioned techniques for given performance constraints, such that in these particular scenarios, all techniques meet their performance constraints. Different performance constraints correspond to changing application contexts.
C. Comparison to State-of-the-Art

Fig. 8 provides the energy breakdown comparison of three CI muting techniques where each bar is the average value of 170 experiments (17 area and 10 performance constraints). Fig. 8 shows that the leakage energy is dominant in the Pre-VM technique [13] due to high SRAM leakage, especially in case of CIF encoding. On the contrary, the Pre-FM technique [14] reduces the leakage by shutting down the SRAM but suffers from significant reconfiguration energy overhead. Our adaptive technique overcomes the drawbacks of both of the above techniques. It overcomes the reconfiguration overhead of Pre-FM by using the virtually-muting mode for a selective subset of CIs and eliminates the drawback of Pre-VM by putting the subset of CIs in fully-muting mode that are not used for a rather long period.

Fig. 8 illustrates that our technique is superior to both competitor techniques in all cases. In particular, the benefit of our technique is noticeable in high-resolution encodings due to a rather long muting duration. It is worthy to note in Fig. 8 that the leakage energy of our technique is slightly lower than that of Pre-FM. This is because virtually-muted CIs may bring a leakage reduction due to a faster execution\(^5\) (as a result of the powered-on energy of our technique is slightly lower than that of Pre-FM. This technique is noticeable in high-resolution encodings due to a rather long period.

The complexity for online computation of the CI weighting factor (Fig. 7) that determines the HAVM. The

\[^{5}\text{The larger leakage power does not necessarily lead to larger leakage energy if the execution time is correspondingly shorter.}\]

It is noticeable in Fig. 9 that our technique performs always better than the Pre-FM technique, whereas Pre-VM sometimes performs better. Therefore, Fig. 10 focuses on further comparisons with Pre-VM, showing the energy benefit summary (480 experiments per technology with various combinations of fabric area and performance constraints) of our technique when compared with the [13]-based Pre-VM technique. Fig. 10 shows that, compared to the Pre-VM technique, our technique provides on average 41.64%, 43.11%, 33.75%, and 43.52% energy reduction for 40 nm, 40 nmL, 65 nm, and 90 nm, respectively. When rather few PRCs are available, then the Pre-VM technique performs better than our technique as most of the PRCs are always used. However, more PRCs are required for performance constraint tightening. In such cases the Pre-VM technique performs better than our technique as most of the PRCs are always used. However, more PRCs are required for performance constraint tightening. In contrast, our technique is beneficial for almost all the cases (see Fig. 9). The performance constraints and the amount of available PRCs cannot be predicted at design-and/or compile-time as they depend on run-time specific scenarios, like changing application contexts or multi-tasking interactions.

\[^{5}\text{The larger leakage power does not necessarily lead to larger leakage energy if the execution time is correspondingly shorter.}\]
energy benefit calculation (in line 8-15, Fig. 7) consumes 8.82 nWs, 11.56 nWs for 40 nm, 65 nm, 90 nm, respectively. For ‘n’ virtually-muting candidates there are ‘n(n-1)’ energy benefit calculations. For the above-presented experimental setup – in worst case – there are at most four candidates. The overall overhead of our technique is 105.89 nWs, 138.71 nWs, 261.68 nWs for 40 nm, 65 nm, 90 nm, respectively. However, the energy savings of our technique are in multiples of nWs, i.e. more than 10^6 times bigger. Therefore, the energy overhead is negligible compared to our energy savings.

The worst-case performance overhead of our technique is 1,356 cycles for above-discussed experiments, which is negligible in comparison to the hot spot execution time (<< 1%, depending on performance constraints). Due to the reconfiguration latency overhead, the maximum achievable performance of Pre-FM is on average 8% smaller than that of Pre-VM. Our technique provides a compromise between Pre-VM and Pre-FM techniques.

We envision our muting technique executing on a Microblaze processor (a soft core provided by Xilinx) that – along with monitoring and the reconfiguration controller – requires only 5,564 slices in our current Xilinx Virtex-4-LX160 FPGA based prototype.

Note: our technique also requires a power shutdown infrastructure in FPGAs to realize energy-aware adaptive computing that incurs additional area overhead. Pika [10] (a Xilinx low power FPGA research project) states an 8% area increase due to their power shutdown infrastructure. However, they provide one sleep transistor per CLB, while we suggest two sleep transistors per PRC (a group of 96 CLBs). Therefore, we envision a much smaller area overhead. Currently, such infrastructure is not available in today’s commercial FPGAs. The adaptive processors (and reconfigurable computing in general) would be far more energy efficient if FPGA vendors would provide a basic infrastructure that is necessary to exert our technique.

VI. CONCLUSION AND SYNOPSIS

Our novel Custom Instruction muting technique determines at run time which subset of the CI to temporarily mute and which muting mode (see Table I) to use. It considers leakage as well as reconfigurable energy under run-time varying situations and constraints. Our technique requires a power shutdown infrastructure to realize the selected muting modes. Our results demonstrate that our technique achieves on average more than 30% energy savings compared to hardware-oriented shutdown techniques (of [13] and [14] that represent current state-of-the-art. This holds for various technology nodes (we have evaluated: 40 nm, 65 nm, and 90 nm). Our results corroborate the potential for far higher energy savings of adaptive processors which currently still suffer from a low efficiency as far as energy is concerned.

VII. REFERENCES