# An analysis on power consumption and performance in runtime hardware reconfiguration

Denis S. Loubach

Department of Computer Systems,
Aeronautics Institute of Technology – ITA,
São José dos Campos, SP,
12228-900, Brazil
Email: dloubach@ita.br

**Abstract:** It will be more difficult to continue with Moore's law scaling in the next years without exploring new heterogeneous architectures with application-customised hardware. The expressive employment of customised accelerators, or runtime reconfigurable designs, will be required to deliver power- and performance-efficient systems. In view of recent technology advances, runtime partial reconfiguration has emerged supported by FPGA-based devices. Still, power consumption and performance are the principal concerns when devising new reconfigurable embedded systems. This paper addresses power and performance analysis of the partial reconfiguration process supported by runtime reconfigurable hardware. We introduce a heterogeneous system-on-chip FPGA-based runtime partial reconfigurable platform design along with an experimental and theoretical power consumption and performance models, which are specific to the partial reconfiguration process. The proposed design was implemented, and both experimental and theoretical power consumption and performance analyses were performed, thus providing a formal tool to the decision-making process between power consumption and performance applicable to the runtime reconfiguration phase. Results show an average accuracy of 89.76% for the power consumption model and 94.82% for the performance model.

**Keywords:** reconfigurable computing; embedded systems; power consumption; performance; heterogeneous systems; field-programmable gate array; FPGA.

**Reference** to this paper should be made as follows: Loubach, D.S. (2021) 'An analysis on power consumption and performance in runtime hardware reconfiguration', *Int. J. Embedded Systems*, Vol. 14, No. 3, pp.277–288.

**Biographical notes:** Denis S. Loubach received his PhD in Electronic Engineering and Computer Science from the Brazilian Aeronautics Institute of Technology – ITA in 2012. He is an Assistant Professor in ITA since 2018. From 2014 to 2018, he was an Assistant Professor with the University of Campinas – UNICAMP, working in the area of real-time embedded systems. He worked a couple of years in the embedded systems industry. His research interests include reconfigurable computing, embedded systems, real-time systems, real-time operating systems, and model-based design.

## 1 Introduction

Notable advances were achieved related to the field-programmable gate array (FPGA) area in the past two decades. It goes from rapid prototyping (Bobda, 2007; Chattopadhyay, 2013) to the support of runtime partial reconfiguration (PR) designs. The latter enables the chip's reconfigurable area to be divided into two or more different partitions, and each partition is allowed to be reconfigured independently of each other.

Basically, FPGA fits between general purpose processors (GPP) and application-specific integrated circuits (ASIC) in terms of flexibility and performance. FPGA is more flexible than GPP, and presents less performance when compared to ASIC.

Process technology is also getting better and better. Nevertheless, power consumption and performance are still the top concerns (Beck et al., 2013) when devising new reconfigurable embedded systems.

As discussed by Borkar and Chien (2011) and Chien (2018), it will be more difficult to continue with Moore's law scaling in the next years without exploring new heterogeneous architectures with application-customised hardware. Operation frequency tends to increase slowly, and energy tends to be the main performance limiter. To face such challenges, the intensive employment of customised accelerators, or runtime reconfigurable designs, will be required to deliver power- and performance-efficient systems. Perera and Li (2019) claim that FPGA-based hardware is one of the best options to face these challenges.

Most of the literature addresses power consumption and performance analysis with respect to the execution of an embedded system, i.e., when the system is already executing a given function in software or even in hardware. On the other hand, a runtime reconfigurable design presents another key point to be observed: the *PR phase*.

The total number of PRs a system may achieve during its execution tends to increase whenever that system really wants to take advantage of this feature to overcome the previously mentioned challenges. In view of this, a tool is required able to estimate power consumption and performance levels when a hardware partition is under the reconfiguration process.

Motivated by this context, this paper addresses *power and performance analysis* of the *PR process* supported by runtime reconfigurable hardware. We introduce a heterogeneous system-on-chip (SoC) FPGA-based runtime partial reconfigurable (FRunPR) platform design along with an experimental and a theoretical power consumption and performance models, which are specific to the PR process.

To assess our models, we implemented the FRunPR platform design in an experimental setup, with one partition, and measured the power consumption and performance when that partition was being reconfigured. Next, the data was compared to the proposed analytical models' estimates.

The main contributions of this paper are summarised as follows:

- Analytical power consumption and performance models applicable to the runtime PR phase, independent from the number of partitions in the reconfigurable area.
- Experimental implementation of a heterogeneous SoC FPGA-based runtime partial reconfigurable platform design, showing a trade-off analysis of power consumption and performance.

The proposed estimation models are applicable for runtime partial reconfigurable systems where power and performance constraints may change during execution time, i.e., where trade-offs are acceptable or even required.

## 2 Background

### 2.1 Reconfigurable computing

In reconfigurable computing, a *programmable* processor may differ from a *reconfigurable* one. The former is basically governed by high-level languages, such as C or C++, and its hardware architecture is fixed. The latter can have its low-level switches architecture completely changed allowing different functionality to take place, where a hardware description language (HDL) is generally employed (Chattopadhyay, 2013).

According to Koch (2013), a notable research area, namely reconfigurable computing, was established taking into account the reconfigurable device's introduction. The reconfigurable computing concept dates back to 1960 when Estrin proposed a seminal variable structure computer along with fixed parts (Estrin, 1960).

Generally, a reconfigurable processor always supports *full reconfiguration* (FR). In this case, the entire reconfigurable area is changed at once. Conversely, modern devices also support *PR*. It allows for defining multiple different areas known as *partitions*, besides a static area. Thus, partitions can be independently hardware-changed by PR at runtime, i.e., when the reconfigurable processor is executing and without stopping other partitions or the static area.

Runtime reconfiguration is not regarded as a simple process, especially taking into account partitions input/output interfaces (Bobda, 2007).

For both FR and PR, a given function may be written using HDL. Then, after synthesis and fitting processes a configuration bitstream is generated. This bitstream must be loaded into the device's configuration memory (CRAM). This memory is the one controlling the logic elements, memory, and routing multiplexers in the reconfigurable processor. CRAM bits can be structured in columns, and in that case, an entire column is overwritten when reconfiguration occurs.

There are a number of modes when generating a configuration bitstream, including 'and/or' mode (AO/M) and 'scrub' mode (SC/M). Basically, the mode dictates the way the configuration bitstream is effectively written in the CRAM.

In the AO/M, memory is written in a two-pass fashion. Column regions inside the partition are bitwise ANDed with zero's, and outside the partition they are ANDed with one's. Next, column regions inside the partition are bitwise ORed with new bitstream data, and outside the partition they are ORed with zero's. Column regions outside the partition under PR remain unchanged.

On the other hand, SC/M is based on an one-pass fashion. The entire column is always changed. Column regions inside the partition are overwritten with new bitstream data, and column regions outside the partition are scrubbed back to their original values. Therefore, this mode does not allow partitions to have overlapping columns.

Choosing AO/M or SC/M has an impact on both power consumption and performance, as discussed in later sections.

### 2.2 Power consumption and performance

Power consumption includes the current driven by the elements involved in a given computation, supply voltage, and operating frequency. There are many power consumption metrics taking into account different variables according to a given design and project requirements.

There are also a number of processor performance metrics. Nevertheless, all of them are based on the system behaviour in some given time length (Noergaard, 2013). Considering that, performance can be expressed as a time unit.

The relation between power and time is stated as seen in equation (1).

$$Energy = Power \times Time \tag{1}$$

where $Energy$ is given in joule (J); $Power$ in watt (W); and $Time$ in seconds (s).

According to Hennessy and Patterson (2006), energy is generally a better metric to compare processors efficiency, since energy is tied to a given task and the time required to complete that task. In our case, this specific task is the *reconfiguration process*.

On the other hand, power consumption can be employed as a constraint in the system. Power can be described as seen in equation (2).

$$Power = Voltage \times Current \tag{2}$$

where $Voltage$ is given in volt (V); and $Current$ in ampere (A).

In turn, $Power$ can be divided into dynamic and static power, giving the total power $tp$:

$$Power \equiv tp = Pwr_{DYNAMIC} + Pwr_{STATIC} \tag{3}$$

Dynamic power $Pwr_{DYNAMIC}$ is basically defined by transistors switching and capacitive loads, i.e., signals toggling together with load capacitance being charged and discharged. $Pwr_{DYNAMIC}$ can be minimised by lowering the circuit supply voltage. A well-known technique to handle $Pwr_{DYNAMIC}$ is the dynamic voltage and frequency scaling (DVFS) (Nunez-Yanez et al., 2016; Le Sueur and Heiser, 2010; Kim et al., 2018; Digalwar et al., 2017).

On the other hand, static power $Pwr_{STATIC}$ is basically the current leakage in transistors when in the quiescent condition. To minimise the $Pwr_{STATIC}$ one has to cut-off unused paths in the circuit. A well-known technique to cope with $Pwr_{STATIC}$ is the dynamic power management (DPM) (Paul, 2014).

As observed by Hennessy and Patterson (2006), voltage dropping from 5 V to approximately 1 V has been mainly responsible for both dynamic power and energy reductions in the past two decades.

## 3 Runtime reconfigurable platform design

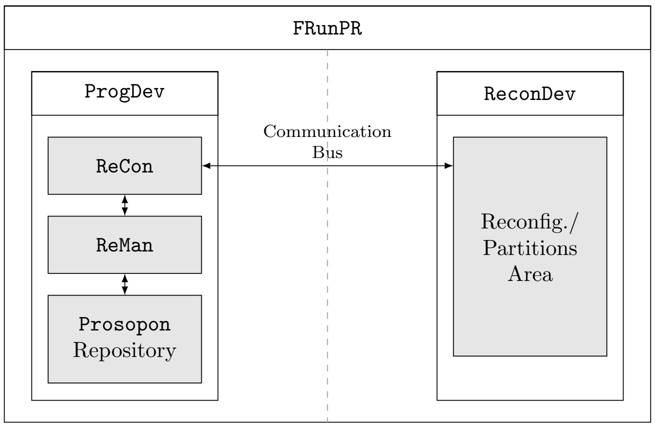
A general reconfigurable processor architecture includes a *fixed* hardware part along with a *changeable* hardware area and a *communication* bus between them. The communication bus can even be divided into data bus and dedicated configuration lines.

To abstract this general architecture, and without loss of generality, we adopted a runtime reconfigurable design proposed by Loubach (2016). That design is further developed and extended in the present paper, where it is introduced as a runtime reconfigurable design named *FPGA-based runtime partial reconfigurable* (FRunPR) *platform*.

FRunPR platform is intended to abstract most of the implementation details and processor specific information. It also aims to be scalable. This scalability aspect is related to the platform characteristic to host a number of different functionality over time enabled by hardware FR, and mainly PR.

Two different areas comprise the FRunPR platform design. A fixed hard-core processor area named *programmable device* (ProgDev), and a reconfigurable area named *reconfigurable device* (ReconDev), as shown in Figure 1. Notice that the definitions of the FRunPR terms follow the idea of Chattopadhyay (2013). Thus, programmable relates to processors governed by high-level languages with fixed hardware architecture, and reconfigurable refers to runtime changeable low-level switches architecture allowing different functionality to take place usually written on a HDL.

**Figure 1** FPGA-based runtime partial reconfigurable (FRunPR) platform design with ProgDev and ReconDev areas



Notes: ReconDev shows the (reconfigurable) partitions area able to be divided into multiple partitions. The communication between these two areas is established by a dedicated bus.

The ProgDev area abstracts the programmable device area of a given heterogeneous hardware architecture. The main component of ProgDev is the *reconfiguration manager* (ReMan). It is a piece of software that implements the reconfiguration logic for the system and abstracts specifics about the underlying reconfiguration process. ReMan also performs the reconfiguration scheduling, which is so far a static scheduling. In the ProgDev area, there is also a memory space named *prosopon repository* dedicated to store the reconfiguration bitstreams used in the reconfiguration process. This repository is new related to the design presented in Loubach (2016).

ReconDev area, on the other hand, abstracts the reconfigurable area, i.e., FPGA, in a general heterogeneous hardware architecture. The main components of ReconDev are *partitions*, which are logically and physically defined areas in the FPGA able to host different hardware functions or implementations. The different hardware functions implementations are named *prosopons* in our design, and they are represented by the symbol Π. In view of this, each partition is able to host different Πs along time. The word *prosopon* comes from the Ancient Greek meaning ‘person’.

Prosopons are regarded as *abstractions of device specific bitstreams*, i.e., implementations of functions that are used to modify a given partition inside the reconfigurable area. Notice that FRunPR platform design does not restrict either the number of partitions on a design or its sizes or interfaces. It is expected that a third-party employed design flow provides means to define those requirements.

As illustrated in Figure 1, ProgDev and ReconDev exchange information by means of a *dedicated communication bus*. That bus encapsulates a $w$-byte width *data bus*, and *control bus lines* containing signals to request the initialisation of reconfiguration and also to monitor the reconfiguration status, e.g., ready to start, done, error.

The *reconfiguration controller* (ReCon) inside the ProgDev is another component responsible for abstracting away the specific device details required for the reconfiguration process, i.e., communication protocol, and provides the implementation of the required software and hardware modules that enable ReMan to control the allocation of prosopons on partitions over the time. ReCon is also new with respect to the design presented in Loubach (2016).

This proposed platform design abstracts a heterogeneous hardware architecture that comprises a programmable device, such as an embedded processor or microcontroller unit, and a reconfigurable device, such as a modern partial reconfigurable FPGA. These abstractions are general enough to capture a number of commercially available off-the-shelf (COTS) devices, from different vendors, and can also be employed to define design requirements for ASIC development. With these characteristics, we intend to address both *generality* and *applicability* in our FRunPR.

In FRunPR, ProgDev is the one controlling which configuration, i.e., which prosopon must be loaded into a given partition inside the ReconDev. This is achieved by ReMan, implemented as an embedded software based on C programming language.

ReMan is basically responsible for:

- ProgDev's central processing unit initialisation
- direct memory access (DMA) setup, so that ReCon is able to efficiently transfer prosopons to ReconDev
- prosopon fetching from the repository, using a static scheduling.

ReMan works together with ReCon, which in turn is responsible for:

- communication initialisation with ReconDev
- FR and PR requestings
- reconfiguration signals monitoring, such as 'configuration ready to start', 'configuration done', or 'configuration error'
- finishing the FR and PR processes.

Before being able to be runtime partially reconfigured, ReconDev must be first full configured once. In this way, it becomes aware of its available partitions. Then, requests for PR can take place.

The hardware implementation process of both FR and PR are out of the scope of this paper. We consider that the reconfiguration bitstreams are ready-made to be used in the proposed platform design. Although, we point some directions.

In a high-level abstraction, the basic steps for hardware implementation generally include:

- describing the hardware function without any partitions at first place
- identifying the parts able to be runtime reconfigured into partitions
- defining the logical and physical areas for partitions
- implementing each partition prosopon
- generating a FR bitstream and a PR bitstreams for each partition's prosopon.

## 4 Power consumption and performance models

Notice that the models, experimental and analytical, do not depend on the number of partitions in the reconfigurable area. Both full and PRs are requested through a communication bus, as illustrated in FRunPR platform design (Figure 1), and are therefore serialised.

In this sense, we are interested in

1 the *voltage lines* responsible to supply the power to the reconfiguration circuitry

2 the *time* to send the configuration bitstream, acknowledge the configuration, and complete it.

### 4.1 Experimental power consumption model

The experimental power consumption model $p_{EXP}$ used in this work is described in equation (4). This first model was applied to calculate the power consumption for the performed experiments introduced in Section 5.

$$\begin{aligned} p_{EXP} &= i_{RMS} \times v \\ &= \frac{i_{PEAK}}{\sqrt{3}} \times v \end{aligned} \tag{4}$$

where $i_{PEAK}$ is the current peak measured when the reconfiguration process takes place; and $v$ is the involved circuit supply voltage. $i_{PEAK}$ was measured from the voltage line, which supplies power to the reconfiguration circuitry, by using an onboard digital-power supply monitor, i.e., LTC2978. $i_{RMS}$ is discussed in the next section.

The following constraint is applied to equation (4).

$$0 < v \leq V_{MAX} \tag{5}$$

where $V_{MAX}$ stands for the maximum supported circuit supply voltage.

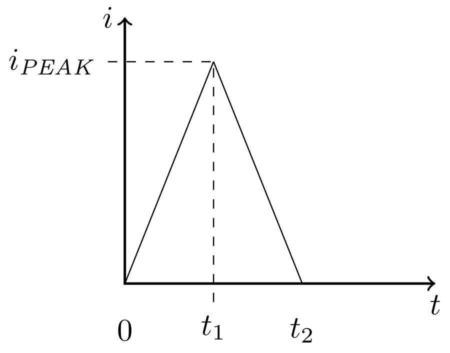
### *4.1.1 Current measurement considerations*

When measuring the power consumption, a current peak $i_{PEAK}$ resembling a triangle waveform (Figure 2) was noticed in the moment of the reconfiguration process. Thus, the root mean square (RMS), given in equation (6), was used to obtain $i_{RMS}$, which is in the experimental power consumption formulation (4), previously described.

$$i_{RMS} = \sqrt{\frac{1}{T}\int_0^T i(t)^2\, dt} \tag{6}$$

where $i_{RMS}$ is the RMS value of the measured current; $T$ is the signal period and $i(t)$ is the function expressing the current over the time.

**Figure 2** A triangle waveform, observed when the reconfiguration process occurs, showing the current rising from time 0 to $t_1$, where $i_{PEAK}$ takes place, and the falling part from $t_1$ to $t_2$



Regarding $i(t)$, there is a linear function from time 0 to $t_1$, and from $t_1$ to $t_2$, as illustrated in Figure 2. This function is expressed in equation (7).

$$i(t) = \begin{cases} i_{PEAK} \times \frac{t}{t_1}, & 0 \le t < t_1 \quad \text{(rising part)} \\ i_{PEAK} \times \frac{t_2 - t}{t_2 - t_1}, & t_1 \le t < t_2 \quad \text{(falling part)} \end{cases} \tag{7}$$

By replacing equation (7) in equation (6), equation (8) is obtained.

$$i_{RMS}^2 = \frac{i_{PEAK}^2 \times t_2}{3T} \tag{8}$$

Considering $T = t_2$, and the duty-cycle equals to 100% for this one-time peak signal in equation (8), the RMS current is finally given by equation (9).

$$i_{RMS} = \frac{i_{PEAK}}{\sqrt{3}} \tag{9}$$

## *4.2 Analytical power consumption model*

The proposed analytical power consumption model $p$ developed and applied in this work is described in equation (10). This model was used to estimate the power consumption for the PR process.

$$p = \frac{1}{2} \times C \times v^2 \times \frac{1}{\tau} \times \pi \times \mu_m \times \gamma \tag{10}$$

where $C$ is the load capacitance in farad (F); $v$ is the involved circuit supply voltage; $\tau$ is the configuration data clock period in seconds; $\pi$ is the prosopon size in bytes; $\mu_m$ with $m \in \{AO/M, SC/M\}$, is given by equations (11) and (12) related to the configuration bitstream mode, i.e., AO/M or SC/M; and $\gamma = 1 \times 10^{-6}$ is a heuristic constant.

$$\mu_{AO/M} = \frac{\Pi_{SC/M}}{\Pi_{AO/M}} \times 1.2 \tag{11}$$

$$\mu_{SC/M} = \frac{\Pi_{SC/M}}{\Pi_{AO/M}} \times 1.8 \tag{12}$$

The $\Pi_{SC/M}$ size represents a fraction $\le 1$ of the $\Pi_{AO/M}$ size. As mentioned in Subsection 2.1, AO/M is based on a two-pass memory writing, and SC/M is based on an one-pass. Thus, AO/M tends to contain more data compared to SC/M. However, SC/M always changes the entire memory column, i.e., regions inside the partition are rewritten, and regions outside the partition are scrubbed back.

Considering this theory and by observing the experimental data values, we tunned our analytical model to capture this behaviour: preserving (AO/M) or not preserving (SC/M) some columns' stored data. Thus, the configuration bitstream mode AO/M represents a number 20% bigger the ratio $\frac{\Pi_{SC/M}}{\Pi_{AO/M}}$, while the SC/M represents 80%, where data are always rewritten.

$C$ is mainly related to the transistors number connected to a given output, and circuit process technology, then giving both the wires and transistors capacitance as an equivalent or lumped capacitance (Hennessy and Patterson, 2006).

The following constraints are applied to equation (10).

$$C > 0 \tag{13}$$

$$V_{MIN} \le v \le V_{MAX} \tag{14}$$

where $V_{MIN}$ and $V_{MAX}$ stand for the minimum and the maximum supported circuit supply voltages.

$$\frac{1}{f_{MIN}} \le \tau \le \frac{1}{f_{MAX}} \tag{15}$$

where $f_{MIN}$ and $f_{MAX}$ stand for the minimum and the maximum supported reconfigurable device's data clock frequencies.

$$0 < \sum_{i=1}^{n} \pi_i \le \pi_{MAX} \tag{16}$$

where $\pi_{MAX}$ stands for the maximum available configuration memory size supported by the reconfigurable device, and $n$ is the total numbers of prosopons.

The model (10) only accounts the term comprising the dynamic power and does not consider the static power term.

In this case, as we are mainly interested in the reconfiguration phase, static power may be neglected because the hardware part responsible for the reconfiguration process can be turned-off when not being used. However, there is still the prosopons repository, which is a memory, that may need power all the time if it is based on a non-persistent memory.

### 4.2.1 *Considerations about the term $\frac{1}{2}Cv^2$*

The capacitance $C$ is given in equation (17), where $q$ stands for the charge in coulombs, and $V$ for the potential difference.

$$C = \frac{q}{V} \tag{17}$$

By integrating the potential difference $V = \frac{q}{C}$, starting from uncharged to charged $Q$, one obtains the stored capacitor energy $E_C$, as described in equation (18).

$$E_C = \int_0^Q \frac{q}{C}\, dq = \frac{1}{2} \times C \times V^2 \tag{18}$$

## 4.3 *Analytical performance model*

Maximising performance means to minimise the time spent to execute a given computation, as described in equation (19). In our case, the given computation is the reconfiguration process. Therefore, performance is about time consumption in the proposed model.

$$Performance = \frac{1}{ExecutionTime} \tag{19}$$

The developed performance model $t$ is described in equation (20). This model was used to estimate the time spent in the PR processes.

$$t = \pi \times \frac{1}{\omega} \times \tau \tag{20}$$

where $\pi$ is the prosopon size in bytes; $\omega$ is the data bus width in bytes; and $\tau$ is the configuration data clock period in seconds.

The following constraint is applied to equation (20).

$$\omega_{MIN} \leq \omega \leq \omega_{MAX} \tag{21}$$

where $\omega_{MIN}$ and $\omega_{MAX}$ stand for the minimum and the maximum data bus widths supported by the reconfigurable device.

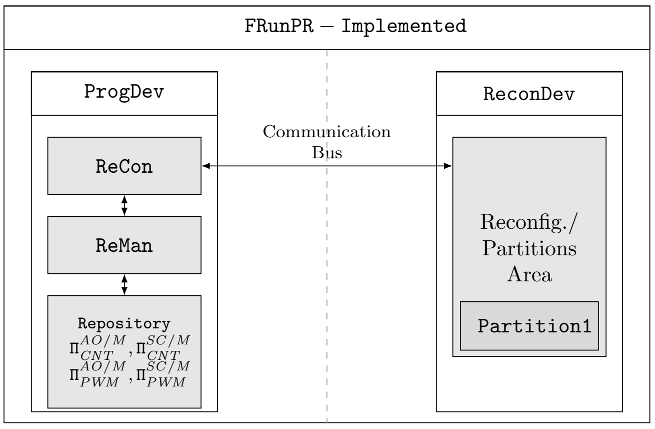
## 4.4 *Power consumption and performance analyses*

Models (10) and (20) describe our analytical power consumption and performance estimates. Notice those two objectives, i.e., minimise power and minimise time, conflicts to each other. Minimising power may cause the time, i.e., reconfiguration time, to increase. Conversely, minimising time may cause the power consumption to increase. In this sense, one may use the Pareto approach to cope with this multi-objective optimisation problem.

## 5 Experimental setup

To test and assess the proposed analytical power consumption (10) and performance (20) models, we implemented the FRunPR platform design in an experimental setup and performed measurements.

**Figure 3** FRunPR platform design implemented considering one partition named `Partition1` able to host different $\Pi$


FRunPR – Implemented
ProgDev
ReCon
ReMan
Repository
$\Pi^{AO/M}_{CNT}$, $\Pi^{SC/M}_{CNT}$
$\Pi^{AO/M}_{PWM}$, $\Pi^{SC/M}_{PWM}$
Communication Bus
ReconDev
Reconfig./ Partitions Area
Partition1

Notes: The prosopons $\Pi^{AO/M}_{CNT}$, $\Pi^{SC/M}_{CNT}$, $\Pi^{AO/M}_{PWM}$, and $\Pi^{SC/M}_{PWM}$ stands for $\Pi$ considering AO/M and SC/M. They are located in the prosopons repository inside ProgDev.

The FRunPR platform design was implemented considering one partition, named `Partition1`, in the ReconDev area, as illustrated in Figure 3.

Two different experiments were conducted intending to exercise the runtime PR.

In the *first experiment*, two different prosopons were developed for the `Partition1` and located at the prosopons repository, as shown in Figure 3. One prosopon implements a simple incremental counter function, $\Pi_{CNT}$, and the other implements a pulse width modulation function, $\Pi_{PWM}$.

For each one of the two prosopons, i.e., $\Pi_{CNT}$ and $\Pi_{PWM}$, we generated two different configuration mode versions. One version generated to be according to AO/M, and the other according to SC/M. This way, we could not only change the hardware functionality at runtime, through PR, but also verify the impact each configuration mode and prosopon sizes cause in the reconfiguration process in terms of power consumption and performance.

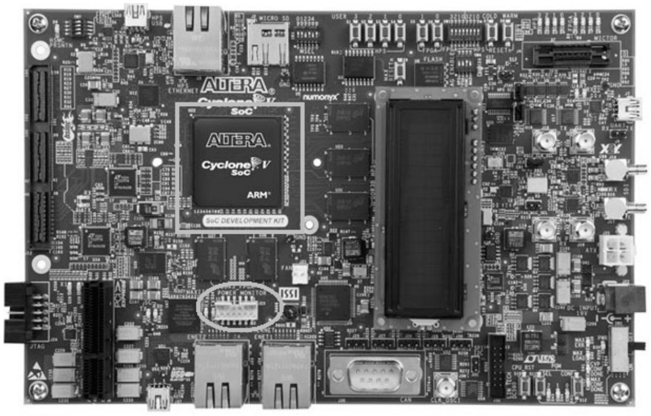
A *second experiment* was also carried out. That one considered a setup with one partition as well, however, its two prosopons implemented two different algorithms for encryption purposes: the advanced encryption standard (AES) (Daemen and Rijmen, 2002b), $\Pi_{AES}$; and the data encryption standard (DES) (Daemen and Rijmen, 2002a), $\Pi_{DES}$.

In the same way, as in the first experiment, each prosopon has also two different versions. One version generated to be according to AO/M, i.e., $\Pi^{AO/M}_{AES}$ and $\Pi^{AO/M}_{DES}$, and the other according to SC/M, i.e., $\Pi^{SC/M}_{AES}$ and $\Pi^{SC/M}_{DES}$. The second experiment considered the prosopons repository containing both versions of $\Pi_{AES}$ and $\Pi_{DES}$.

For each experiment, ProgDev commanded an one-time FR in ReconDev, so that the latter becomes aware of `Partition1`. Next, ReMan together with ReCon commanded static scheduling PRs considering the prosopons located at the repository.

A Cyclone V SoC development kit was used to implement FRunPR, as illustrated in Figure 4. This hardware kit is built with a fabric containing a dual-core ARM Cortex-A9 and a FPGA with 110 K logic elements.

**Figure 4** Hardware board used to implement the FRunPR platform design



Notes: The grey square highlights the SoC fabric. The grey ellipse under the SoC fabric highlights the power monitor connector used for onboard power consumption measuring.

Power consumption measurements were performed using a DC1613A, which is a USB-to-power management bus. The Cyclone kit is equipped with dedicated circuits allowing onboard power measuring.

Performance metrics, i.e., the time to complete a reconfiguration process, was measured with an oscilloscope.

Prosopons were written using Verilog in the Quartus software. ReCon and ReMan were implemented using C programming language and built with the GNU compiler collection (GCC)-based `arm-altera-eabi` toolchain.

Notice that as mentioned before, our models and platform design abstracts a heterogeneous hardware architecture that comprises a programmable device, such as an embedded processor or microcontroller unit, and a reconfigurable device, such as a modern partial reconfigurable FPGA. These abstractions are general enough to capture a number of COTS devices, e.g., Intel-FPGA and Xilinx. Also, these devices present all the described parameters of the models (10) and (20), and all the areas of the FPGA-based runtime partial reconfigurable platform design, making it simple to use another hardware kit besides the one shown in this paper.

## 6 Results and discussion

Table 1 shows all variable values used in the experimental measuring and in the models evaluation analyses.

### 6.1 Static data

Table 2 shows each generated prosopon with its respective size. Notice that for SC/M, prosopons have the same size, since this mode always overwrites the whole columns, up and down, considering the whole partition area.

**Table 1** Experimental and analytical variables and values

| *Variable* | *Description* | *Value* |
|---|---|---|
| $C$ | Lumped capacitance considered for the reconfiguration process | 220 pF |
| $v$ | Circuit supply voltage | 1.5 V |
| $r$ | Sense resistor | 3 mΩ |
| $\tau$ | Configuration data clock period (maximum supported by the device) | $\frac{1}{125 \times 10^6}$ s |
| $\gamma$ | Heuristic constant | $1 \times 10^{-6}$ |
| $\omega$ | Data bus width, encapsulated within the communication bus | 2 bytes |

**Table 2** Generated prosopons size

| $\pi$ | *Mode* | *Size in byte* |
|---|---|---|
| $\Pi_{CNT}$ | AO/M | 634,636 |
| $\Pi_{CNT}$ | SC/M | 514,660 |
| $\Pi_{PWM}$ | AO/M | 644,568 |
| $\Pi_{PWM}$ | SC/M | 514,660 |
| $\Pi_{AES}$ | AO/M | 3,082,040 |
| $\Pi_{AES}$ | SC/M | 1,873,812 |
| $\Pi_{DES}$ | AO/M | 3,001,156 |
| $\Pi_{DES}$ | SC/M | 1,873,812 |

### 6.2 Experimental measuring

Table 3 presents the experimental power consumption measurements for each prosopon bitstream mode. $p_{EXP}$ is described in equation (4).

**Table 3** Experimental measurements related to power consumption

| $\pi$ | *Mode* | $i_{PEAK} \times 10^{-3}$ | $p_{EXP} \times 10^{-3}$ |
|---|---|---|---|
| $\Pi_{CNT}$ | AO/M | 25.80 | 22.34 |
| $\Pi_{CNT}$ | SC/M | 30.20 | 26.15 |
| $\Pi_{PWM}$ | AO/M | 24.67 | 21.36 |
| $\Pi_{PWM}$ | SC/M | 30.40 | 26.33 |
| $\Pi_{AES}$ | AO/M | 76.80 | 66.51 |
| $\Pi_{AES}$ | SC/M | 67.00 | 58.02 |
| $\Pi_{DES}$ | AO/M | 74.00 | 64.09 |
| $\Pi_{DES}$ | SC/M | 68.27 | 59.12 |

With respect to power and considering the AO/M, only columns belonging to a partition are effectively written in the CRAM. This is basically due to the masked bitwise operations, i.e., 'AND with one' and 'OR with zero', which are intended to preserve unchanged the column regions outside the partition under PR. In this case, it means less memory written and consequently less power consumed for $\Pi_{CNT}$ and $\Pi_{PWM}$. However, when taking larger prosopons, such as $\Pi_{AES}$ or $\Pi_{DES}$, this two-pass masking operation caused an increase in power consumption compared to SC/M.

Considering the SC/M, CRAM partition column bits are always overwritten with new data and CRAM column bits located outside partition area are also overwritten with their original values. It means more memory written, and therefore more power consumed, as shown in Table 3 for SC/M $\Pi_{CNT}$ and $\Pi_{PWM}$. Nevertheless, when larger prosopons are considered, i.e., $\Pi_{AES}$ and $\Pi_{DES}$, this one-pass writing method is less power hungry than AO/M.

Table 4 shows the experimental performance measurements for each prosopon bitstream mode.

**Table 4** Experimental measurements related to performance

| $\pi$ | *Mode* | *Time* $\times$ *10*$^{-3}$ |
|---|---|---|
| $\Pi_{CNT}$ | AO/M | 2.73 |
| $\Pi_{CNT}$ | SC/M | 2.23 |
| $\Pi_{PWM}$ | AO/M | 2.76 |
| $\Pi_{PWM}$ | SC/M | 2.22 |
| $\Pi_{AES}$ | AO/M | 12.70 |
| $\Pi_{AES}$ | SC/M | 7.76 |
| $\Pi_{DES}$ | AO/M | 12.40 |
| $\Pi_{DES}$ | SC/M | 7.76 |

With respect to performance, the two-pass AO/M tends to take more time to complete when compared to the one-pass SC/M. This fact can be noticed in Table 4.

## 6.3 Model assessment

The assessment of the analytical power consumption model, described in equation (10), is presented in Table 5. Values are estimated for each prosopon bitstream mode along with the error $\epsilon_p$, and the percentage error $\%\epsilon_p$, both related to power.

Model error $\epsilon$ is computed as seen in equation (22), taking into consideration the values estimation resulted from the proposed analytical models (10) and (20), and the experimentally measured values.

$$\epsilon = \mathit{Analytical} - \mathit{Experimental} \tag{22}$$

Model error percentage $\%\epsilon$ is computed as seen in equation (23).

$$\%\epsilon = \left| \frac{\mathit{Analytical} - \mathit{Experimental}}{\mathit{Experimental}} \right| \times 100 \tag{23}$$

The analytical power consumption model *average accuracy* is 89.76%, with a minimum error of 4.59%, and a maximum error of 14.49% (Table 5).

The assessment of analytical performance model, described in equation (20), is presented in Table 6. Values are estimated for each prosopon bitstream mode along with the error $\epsilon_t$, and the percentage error $\%\epsilon_t$ both related to performance.

The analytical performance model *average accuracy* is 94.82%, with a minimum error of 2.93%, and a maximum error of 7.68% (Table 6).

Table 7 shows the energy consumption comparison between the computed energy $E_E$ based on the experimentally measured values, and the estimated energy $E_A$ based on the analytical models. It also presents the error $\epsilon_E$ and the error in percentage $\%\epsilon_E$ with respect to energy. The *average accuracy* is 88.38%.

**Table 5** Analytical power consumption model results evaluation

| $\pi$ | *Mode* | $p \times 10^{-3}$ | $\epsilon_p \times 10^{-3}$ | $\%\epsilon_p$ |
|---|---|---|---|---|
| $\Pi_{CNT}$ | AO/M | 19.11 | –3.24 | 14.49 |
| $\Pi_{CNT}$ | SC/M | 23.24 | –2.91 | 11.13 |
| $\Pi_{PWM}$ | AO/M | 19.11 | –2.26 | 10.57 |
| $\Pi_{PWM}$ | SC/M | 22.88 | –3.44 | 13.08 |
| $\Pi_{AES}$ | AO/M | 69.57 | 3.05 | 4.59 |
| $\Pi_{AES}$ | SC/M | 63.44 | 5.42 | 9.34 |
| $\Pi_{DES}$ | AO/M | 69.57 | 5.48 | 8.55 |
| $\Pi_{DES}$ | SC/M | 65.15 | 6.03 | 10.19 |

Notes: $10.57 \leq \%\epsilon_p \leq 14.49$, for $\Pi_{CNT}$ and $\Pi_{PWM}$.
$4.59 \leq \%\epsilon_p \leq 10.19$, for $\Pi_{AES}$ and $\Pi_{DES}$.

**Table 6** Analytical performance model results evaluation

| $\pi$ | *Mode* | $t \times$ *10*$^{-3}$ | $\epsilon_t \times$ *10*$^{-6}$ | $\%\epsilon_t$ |
|---|---|---|---|---|
| $\Pi_{CNT}$ | AO/M | 2.54 | –190 | 7.01 |
| $\Pi_{CNT}$ | SC/M | 2.06 | –170 | 7.68 |
| $\Pi_{PWM}$ | AO/M | 2.58 | –180 | 6.58 |
| $\Pi_{PWM}$ | SC/M | 2.06 | –160 | 7.27 |
| $\Pi_{AES}$ | AO/M | 12.33 | –370 | 2.93 |
| $\Pi_{AES}$ | SC/M | 7.50 | –260 | 3.41 |
| $\Pi_{DES}$ | AO/M | 12.00 | –400 | 3.19 |
| $\Pi_{DES}$ | SC/M | 7.50 | –260 | 3.41 |

Notes: $6.58 \leq \%\epsilon_t \leq 7.68$, for $\Pi_{CNT}$ and $\Pi_{PWM}$.
$2.93 \leq \%\epsilon_t \leq 3.41$, for $\Pi_{AES}$ and $\Pi_{DES}$.

**Table 7** Energy consumption estimation and comparisons

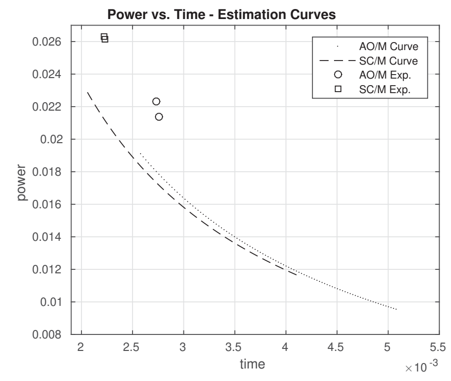
| $\pi$ | *Mode* | $E_E$ | $E_A$ | $\epsilon_E$ | $\%\epsilon_E$ |
|---|---|---|---|---|---|
| $\Pi_{CNT}$ | AO/M | 61.00 | 48.50 | –12.49 | 20.48 |
| $\Pi_{CNT}$ | SC/M | 58.32 | 47.85 | –10.48 | 17.96 |
| $\Pi_{PWM}$ | AO/M | 58.97 | 49.26 | –9.70 | 16.46 |
| $\Pi_{PWM}$ | SC/M | 58.45 | 47.11 | –11.34 | 19.40 |
| $\Pi_{AES}$ | AO/M | 844.69 | 857.61 | 12.93 | 1.53 |
| $\Pi_{AES}$ | SC/M | 450.26 | 475.51 | 25.24 | 5.61 |
| $\Pi_{DES}$ | AO/M | 794.66 | 835.10 | 40.44 | 5.09 |
| $\Pi_{DES}$ | SC/M | 458.80 | 488.32 | 29.52 | 6.44 |

Notes: $E_E, E_A$, and $\epsilon_E$ values are in $\times\ 10^{-6}$.
$16.46 \leq \%\epsilon_E \leq 20.48$, for $\Pi_{CNT}$ and $\Pi_{PWM}$.
$1.53 \leq \%\epsilon_E \leq 6.44$, for $\Pi_{AES}$ and $\Pi_{DES}$.

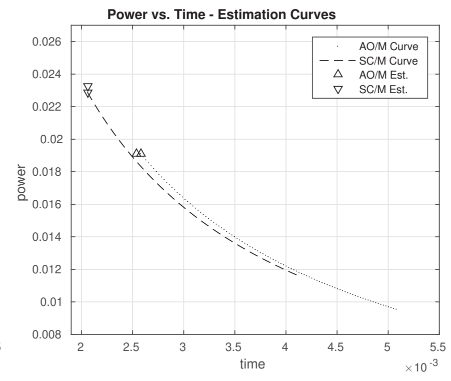
## 6.4 Power consumption vs. time

Notice that $\tau$ is the coupling variable between models (10) and (20). $\tau$ is the one which mainly decides the trade-off between power consumption and time spent in the PR process.

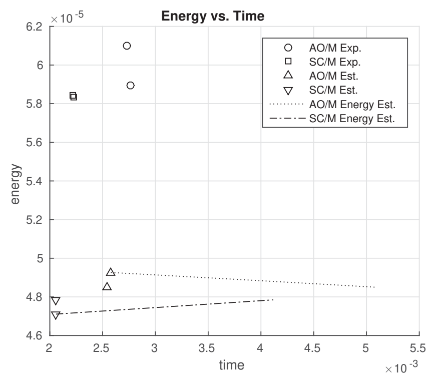
**Figure 5** (a) (b) (d) (e) Power consumption vs. time: curves were obtained based on the introduced analytical models for power consumption and performance, comparing the experimentally obtained values (exp.) to the estimated ones (est.), for both AO/M and SC/M (c) (f) Energy consumption vs. time: energy experimentally measured and estimates based on the analytical models along with two lines for both AO/M and SC/M showing the energy variation in time



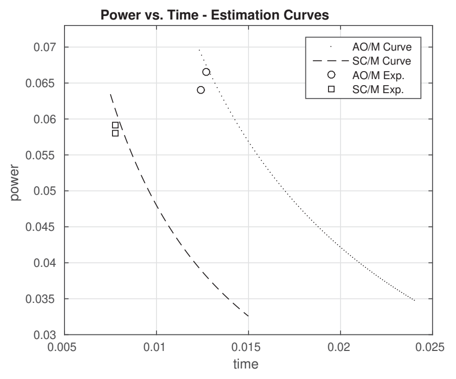
(a)



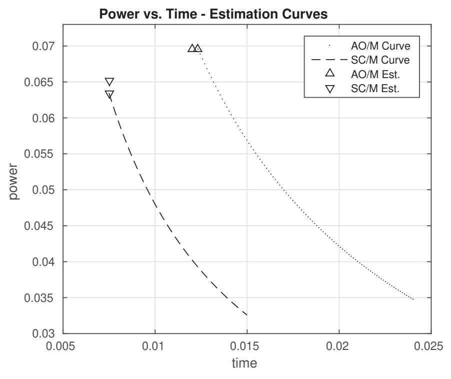
(b)



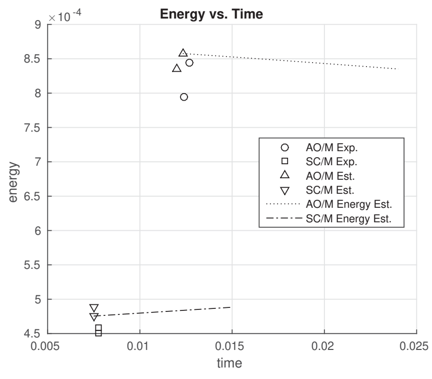
(c)



(d)



(e)



(f)

Considering this previous statement, we can obtain $p'$ from equation (10) and $t'$ from equation (20) as follows. In this sense, some terms will remain constant, held by $K_1$ and $K_2$.

$$p' = \frac{K_1 \times \pi \times \mu_m}{\tau} \tag{24}$$

where $K_1 = 0.5 \times C \times \upsilon^2 \times \gamma$.

$$t' = K_2 \times \pi \times \tau \tag{25}$$

where $K_2 = \frac{1}{\omega}$.

Given this, we drew the 'power vs. time' curve for both AO/M and SC/M by assuming $K_1$ and $K_2$ as constants and with the values as stated in Table 1. Therefore, by changing the other variables values in equations (24) and (25), these curves are obtained, as illustrated in Figure 5.

Each one of Figures 5(a) and 5(d) shows two different curves based on equations (24) and (25). One curve refers to AO/M, and the other to SC/M.

The AO/M curve considers the prosopon size variation from $\Pi_{CNT}^{AO/M}$ to $\Pi_{PWM}^{AO/M}$, illustrated in Figure 5(a), and from $\Pi_{AES}^{AO/M}$ to $\Pi_{DES}^{AO/M}$, as shown in Figure 5(d).

Notice that both reconfiguration time and energy consumption are invariant to the type of application being configured. However, they depend on the application size. Thus, the prosopon size is part of our models, as in equations (10) and (20). Because of this, the models' assessment took into consideration the prosopons sizes variations in bytes, as shown in Table 2.

In this context, the prosopons sizes are representative, to the best of our knowledge. This relates to a main PR benefit, i.e., just a small part of the system is intended to be reconfigured at a time. Other researches (Bonamy et al., 2012) also use prosopon sizes similar to the ones used here.

The SC/M curve considers no variation in prosopon size, since $\Pi_{CNT}^{SC/M}$ and $\Pi_{PWM}^{SC/M}$ have the exact same size as a characteristic of SC/M. This also holds for $\Pi_{AES}^{SC/M}$ and $\Pi_{DES}^{SC/M}$.

Both curves also take into account $\tau$ varying from $\frac{1}{62.5\times10^6}$ to $\frac{1}{125\times10^6}$. $\mu_m$ is also changing value according to equations (11) and (12). The other variables such as $C$, $\upsilon$, $\gamma$, and $\omega$ remained fixed with values as stated in Table 1.

The experimental measuring values for AO/M and SC/M are also shown in Figures 5(a) and 5(d).

Figures 5(b) and 5(e) exhibit the same estimation curves as Figures 5(a) and 5(d). Nevertheless, Figures 5(b) and 5(e) show the analytical model estimation values closer to the curves, as expected.

Notice that AO/M is more efficient in terms of power consumption, and SC/M in performance, taking into account $\Pi_{CNT}$ and $\Pi_{PWM}$. Conversely, SC/M is more efficient in both power and time regarding $\Pi_{AES}$ and $\Pi_{DES}$.

The values shown in Figure 7 are graphically displayed in Figures 5(c) and 5(f) considering the time axis. That figure also presents two lines, one for AO/M and one for SC/M, showing energy variation in time between the prosopons alike.

In the energy consumption vs. time perspective, SC/M is a better solution for both energy and time when compared to AO/M, for the accomplished experiments.

## 7 Related work

The embedded systems community is continuously seeking for low power consumption and high performance applications (Wong et al., 2011; Portilla et al., 2007; Ho et al., 2006).

The research of Bonamy et al. (2012) investigates power consumption related to the dynamic and PR approach. Those authors propose three power models with complexity and accuracy trade-offs. Their medium complexity model, which is comparable to our power model, presented an accuracy of 90.2% on average.

An estimation of power consumption when running dynamic PR is presented in Rihani et al. (2016). The authors evaluated the PR influence on the performance of the global design. However, no accuracy is mentioned in that research.

Vera et al. (2009) proposed a dynamically reconfigurable computing model applicable for video processing applications. López et al. (2014) also considered the dynamic and PR subject when developing a power-aware multi-objective evolvable FPGA-based hardware. They measured the power consumption by an instrumentation circuit involving an ADC.

According to Mu and Lysecky (2011), runtime reconfiguration has already demonstrated a number of benefits in terms of power consumption and performance. In their research, an online estimation method is presented to evaluate the power and performance metrics for runtime partial reconfigurable embedded systems based on the current system execution behaviour. The online estimation model presented an average accuracy of 82% for performance and 76% for power.

The work of Reis and Fröhlich (2020) claims for a deterministic FPGA reconfiguration mechanism that can mitigate the interference generated by other operations occurring in parallel. A deterministic reconfiguration is relevant to aid in power consumption and performance analysis.

A power estimation model, applicable to SoCs comprised of CPU and coarse-grained reconfigurable arrays, is presented by Deng et al. (2017). Those authors claimed that power modelling is a critical component in optimisation flows such as design space exploration (DSE).

A proposal for energy efficiency using PR is presented by Liu et al. (2013). That research indicates the minimisation of reconfiguration time overhead as a key element, and also points out that PR can eliminate static and dynamic power.

Wehner et al. (2016) evaluates timing and power consumption considering a Zynq FPGA. The authors discuss performance, reconfiguration process, and the undefined behaviour during reconfiguration. PR is employed as a mean to reduce system power consumption, and also to contribute to the systems upgrade.

Performance scalability in a reconfigurable processor is presented by Takano (2017, 2012). The author consider issues when dealing with instruction-level parallelism (ILP), in which partial datapath reconfiguration is comparable to instruction hardware fetching and that regular processor-based system has already reached its limit.

An analytical processor performance and power modellings are presented by den Steen et al. (2016). The authors claim that power and performance optimisation can lead to substantial energy efficient and that analytical models can be helpful for DSE tools since they can provide fast estimates and insights.

A multi-objective DPM using voltage and frequency scaling and power gating is proposed by Rahmani et al. (2017). The research addresses the need to consider characteristics in runtime to achieve better performance while keeping up with peak power upper bound.

A power budget and performance optimisation related to network-on-chip (NoC) is discussed by Wang et al. (2016). Most of the research in this area considers techniques involving frequency and voltage scaling.

A scalable DPM scheme for SoC is presented in Shafique et al. (2016) aiming to energy efficiency. The authors change applications degree of parallelism in runtime depending on the workload and performance constraints.

A reconfigurable hardware accelerator is presented by Babecki et al. (2016). The authors mention an improvement in energy efficiency, however, it is not permitted for more than one application to reuse those hardware accelerators.

Nunez-Yanez et al. (2016) explore energy optimisation in FPGA with voltage, frequency, logic scaling, and power gating. However, the latter presents significant overheads.

In summary, these research works take into account different architectures and reconfigurable hardware arrangements, as well as some power consumption and performance improvement techniques, mainly based on frequency and voltage scaling. Our paper proposes a runtime reconfigurable platform design and two models able to *estimate power consumption and performance* from a different perspective, i.e., when the system is under *runtime PR*.

Runtime PR can also be applied as a tool addressing both power consumption and performance optimisations. Depending on the runtime constraints a system has to comply with, a reconfiguration can take place to load a ‘less hungry’ power consumption algorithm, or even a better performance one to attend a hard deadline.

## 8 Conclusions

We presented in this paper a heterogeneous SoC FPGA-based reconfigurable platform design together with power consumption and performance analyses applicable to the PR process.

Two analytical models were proposed. One for power consumption and another for performance estimation. In addition, our platform design was implemented, and experimental measuring was compared to model estimates.

Results show a maximum error of 14.49% for the power consumption model, 7.68% for the performance model, and 20.48% for the energy consumption during the PR process involving two different CRAM writing modes.

According to our experiments, it can be concluded that the 'scrub' mode (SC/M) is more efficient in terms of both power consumption and performance for larger prosopons size, e.g., $\Pi_{AES}$ and $\Pi_{DES}$. However, considering smaller prosopons, such as $\Pi_{CNT}$ and $\Pi_{PWM}$, the 'and/or' mode (AO/M) and its two-pass fashion plays better for the power consumption constraint.

The introduced models represent a formal estimation tool with an average accuracy of 89.76% for power consumption, 94.82% for the performance, and 88.38% for the energy consumption, applicable to the trade-off issue involving power consumption and performance.

The proposed models can be used together with DVFS techniques to achieve an optimal power-performance runtime PR scheme. This can also contribute as a possible input to design space exploration tools when simulating a number of different possible scenarios.

## Acknowledgements

This research work was supported by the Regular Research Awards grant #2014/24855-8, São Paulo Research Foundation (FAPESP), and partially by the Altera University Program.

## References

Babecki, C., Qian, W., Paul, S., Karam, R. and Bhunia, S. (2016) 'An embedded memory-centric reconfigurable hardware accelerator for security applications', *IEEE Transactions on Computers*, Vol. 65, No. 10, pp.3196–3202.

Beck, A.C.S., Lisbôa, C.A.L. and Carro, L. (2013) *Adaptable Embedded Systems*, Springer, New York.

Bobda, C. (2007) *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*, 1st ed., Springer Publishing Company, Incorporated.

Bonamy, R., Chillet, D., Bilavarn, S. and Sentieys, O. (2012) 'Power consumption model for partial and dynamic reconfiguration', in *2012 International Conference on Reconfigurable Computing and FPGAs*, pp.1–8, Springer, The Netherlands.

Borkar, S. and Chien, A.A. (2011) 'The future of microprocessors', *Commun. ACM*, Vol. 54, No. 5, pp.67–77.

Chattopadhyay, A. (2013) 'Ingredients of adaptability: a survey of reconfigurable processors', *VLSI Design*, Article ID 683615, 18pp [online] https://doi.org/10.1155/2013/683615.

Chien, A.A. (2018) 'Computer architecture: disruption from above', *Commun. ACM*, Vol. 61, No. 9, p.5.

Daemen, J. and Rijmen, V. (2002a) *The Data Encryption Standard*, pp.81–87, Springer, Berlin, Heidelberg.

Daemen, J. and Rijmen, V. (2002b) *The Design of Rijndael: AES – The Advanced Encryption Standard*, Springer, Berlin, Heidelberg.

den Steen, S.V., Eyerman, S., Pestel, S.D., Mechri, M., Carlson, T.E., Black-Schaffer, D., Hagersten, E. and Eeckhout, L. (2016) 'Analytical processor performance and power modeling using micro-architecture independent characteristics', *IEEE Transactions on Computers*, Vol. 65, No. 12, pp.3537–3551.

Deng, C., Liu, L., Liu, Y., Yin, S. and Wei, S. (2017) 'PMCC: fast and accurate system-level power modeling for processors on heterogeneous SOC', *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 64, No. 5, pp.540–544.

Digalwar, M., Gahukar, P., Raveendran, B.K. and Mohan, S. (2017) 'Energy efficient real-time scheduling algorithm for mixed task set on multi-core processors', *International Journal of Embedded Systems*, Vol. 9, No. 6, pp.523–534.

Estrin, G. (1960) 'Organization of computer systems: the fixed plus variable structure computer', Paper presented at the *Western Joint IRE-AIEE-ACM Computer Conference, IRE-AIEE-ACM'60 (Western)*, ACM, New York, NY, USA, 3–5 May, pp.33–40.

Hennessy, J.L. and Patterson, D.A. (2006) *Computer Architecture, Fourth Edition: A Quantitative Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ho, C., Leong, P., Luk, W., Wilton, S. and Lopez-Buedo, S. (2006) 'Virtual embedded blocks: a methodology for evaluating embedded elements in PFGAS', in *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines – FCCM*, IEEE, pp.35–44.

Kim, H., Jang, J. and Park, M. (2018) 'Dynamic frequency scaling for embedded systems with memory intensive applications', *International Journal of Embedded Systems*, Vol. 10, No. 2, pp.137–147.

Koch, D. (2013) *Partial Reconfiguration on FPGAs: Architectures, Tools and Applications*, 1st ed., Springer-Verlag, New York.

Le Sueur, E. and Heiser, G. (2010) 'Dynamic voltage and frequency scaling: the laws of diminishing returns', in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems, HotPower'10*, USENIX Association, Berkeley, CA, USA, pp.1–8.

Liu, S., Pittman, R.N., Forin, A. and Gaudiot, J-L. (2013) 'Achieving energy efficiency through runtime partial reconfiguration on reconfigurable systems', *ACM Trans. Embed. Comput. Syst.*, Vol. 12, No. 3, pp.72:1–72:21.

López, B., Valverde, J., de la Torre, E. and Riesgo, T. (2014) 'Power-aware multi-objective evolvable hardware system on an FPGA', in *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp.61–68.

Loubach, D.S. (2016) 'A runtime reconfiguration design targeting avionics systems', in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sacramento, USA.

Mu, J. and Lysecky, R. (2011) 'Profile assisted online system-level performance and power estimation for dynamic reconfigurable embedded systems', in *Proceedings of the 16th Asia and South Pacific Design Automation Conference, ASPDAC'11*, IEEE Press, Piscataway, NJ, USA, pp.737–742.

Noergaard, T. (2013) 'Chapter 4 – embedded processors', in Noergaard, T. (Ed.): *Embedded Systems Architecture*, 2nd ed., pp.137 – 229, Newnes.

Nunez-Yanez, J.L., Hosseinabady, M. and Beldachi, A. (2016) 'Energy optimization in commercial fpgas with voltage, frequency and logic scaling', *IEEE Transactions on Computers*, Vol. 65, No. 5, pp.1484–1493.

Paul, A. (2014) 'Real-time power management for embedded M2M using intelligent learning methods', *ACM Trans. Embed. Comput. Syst.*, Vol. 13, No. 5s, pp.148:1–148:22.

Perera, D.G. and Li, K.F. (2019) 'A design methodology for mobile and embedded applications on FPGA-based dynamic reconfigurable hardware', *International Journal of Embedded Systems*, Vol. 11, No. 5, pp.661–677.

Portilla, J., Riesgo, T. and de Castro, A. (2007) 'A reconfigurable FPGA-based architecture for modular nodes in wireless sensor networks', in *3rd Southern Conference on Programmable Logic, SPL*, IEEE.

Rahmani, A.M., Haghbayan, M.H., Miele, A., Liljeberg, P., Jantsch, A. and Tenhunen, H. (2017) 'Reliability-aware runtime power management for many-core systems in the dark silicon era', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 25, No. 2, pp.427–440.

Reis, J.G. and Fröhlich, A.A. (2020) 'Towards deterministic FPGA reconfiguration', *International Journal of Embedded Systems*, Vol. 13, No. 2, pp.236–253.

Rihani, M.A., Nouvel, F., Prévotet, J., Mroue, M., Lorandel, J. and Mohanna, Y. (2016) 'Dynamic and partial reconfiguration power consumption runtime measurements analysis for ZYNQ SOC devices', in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp.592–596.

Shafique, M., Ivanov, A., Vogel, B. and Henkel, J. (2016) 'Scalable power management for on-chip systems with malleable applications', *IEEE Transactions on Computers*, Vol. 65, No. 11, pp.3398–3412.

Takano, S. (2012) 'Design and analysis of adaptive processor', *ACM Trans. Reconfigurable Technol. Syst.*, Vol. 5, No. 1, pp.5:1–5:34.

Takano, S. (2017) 'Performance scalability of adaptive processor architecture', *ACM Trans. Reconfigurable Technol. Syst.*, Vol. 10, No. 2, pp.16:1–16:22.

Vera, G.A., Llamocca, D., Pattichis, M.S. and Lyke, J. (2009) 'A dynamically reconfigurable computing model for video processing applications', in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pp.327–331.

Wang, X., Zhao, B., Mak, T., Yang, M., Jiang, Y. and Daneshtalab, M. (2016) 'On fine-grained runtime power budgeting for networks-on-chip systems', *IEEE Transactions on Computers*, Vol. 65, No. 9, pp.2780–2793.

Wehner, P., Rettkowski, J., Kalb, T. and GÖhringer, D. (2016) 'Simulating reconfigurable multiprocessor systems-on-chip with MPSOCSIM', *ACM Trans. Embed. Comput. Syst.*, Vol. 16, No. 1, pp.4:1–4:24.

Wong, S., Brandon, A., Anjam, F., Seedorf, R., Giorgi, R., Yu, Z., Puzovic, N., McKee, S., Sjalander, M., Carro, L. and Keramidas, G. (2011) 'Early results from era – embedded reconfigurable architectures', in *2011 9th IEEE International Conference on Industrial Informatics (INDIN)*, pp.816–822.