Power Measurements and Analysis for Dynamic Circuit Specialization

Amit Kulkarni
ELIS department, Computer Systems Lab,
Ghent University,
Sint-Pietersnieuwstraat 41,
Ghent B-9000, Belgium
Email:Amit.Kulkarni@UGent.be

Robin Bonamy
LEAT, CNRS UMR 7248,
University of Nice-Sophia Antipolis,
BP 145 - 930 Route des Colles,
06903 Sophia Antipolis Cedex, France
Email: Robin.Bonamy@unice.fr

Dirk Stroobandt
ELIS department, Computer Systems Lab,
Ghent University,
Sint-Pietersnieuwstraat 41,
Ghent B-9000, Belgium
Email: Dirk.Stroobandt@UGent.be

Abstract*—Dynamic Circuit Specialization (DCS) is a technique for optimized FPGA implementation and is built on top of Partial Reconfiguration (PR). Dynamic Partial Reconfiguration (DPR) provides an opportunity to share the silicon area between different Partially Reconfigurable Modules (PRMs) and therefore results in smaller and faster designs that potentially also reduce the power consumption. In this paper, we show that energy consumption is an important factor that has to be considered while implementing a parameterized design using DCS. In order to make a good choice for implementing a parameterized design with the goal of power optimized implementation, it is important to have a good power consumption estimate of the Dynamic Circuit Specialization. In this context, our paper presents a detailed investigation of the power consumption of a parameterized design implemented using DCS on the Xilinx Zynq-SoC FPGA. We propose an energy analysis of DCS and investigate the benefits of the use of DCS in comparison with a classic static FPGA implementation. We see that the power needed for the reconfiguration is much higher than the gain in power using the reconfiguration over the static implementation. An important reason is because of the CPU involved during the reconfiguration and the interface between the AXI bus and the HWICAP. To reduce the reconfiguration power, we include a clock gating technique to the reconfiguration interface AXI-HWICAP that makes DCS more power efficient. We also relate the power gain to the size of the implementation and to the allowed time to reconfigure versus the useful run time. We conclude that for an implementation with 10 FIR filters, the reconfiguration time should not take more than 30.3% of the total time in order to remain energy efficient. Considering a specific use case with 10 FIR filters at a reconfiguration rate of 0.01, the energy consumption using DCS implementation is 20.5% lower than using the static FIR.

I. Introduction

Dynamic Partial Reconfiguration (DPR) is a technology that provides the flexibility on Programmable Logic Devices to modify some logic blocks while the rest of the logic remains active. Xilinx provides commercial tools for Partial Reconfiguration (PR) technology that have been in the market for quite a while. However, due to the reconfiguration overhead, the use of PR has not really taken off in the industry. Dynamic Circuit Specialization (DCS) is a form of DPR that is tailored to implement parameterized applications [1].

In the classical DPR approach, it is necessary to synthesize Partially Reconfigurable Module (PRM) bitstreams and store them in the memory such as BRAM or SD card beforehand. If the given situation meets a set of predefined conditions, the reconfiguration manager in the FPGA triggers the reconfiguration process and the FPGA is reconfigured with the PRM bitstreams [2]. This adds the required flexibility to reuse hardware (silicon) area and results in a reduction of power cost.

In the DCS approach, specialized bitstreams are generated on the fly depending on the values of the infrequently changing inputs (parameter values) and the FPGA is reconfigured with the specialized bitstreams. Therefore, for every change in parameterized input values, the FPGA is reconfigured via a configuration interface: the Hardware Internal Configuration Access Port (HWICAP). A detailed implementation of the DCS tool flow is described in [3].

Due to the lack of power estimation tools for the Partial Reconfiguration technique, authors in [4] proposed power consumption models for DPR. Similarly, we analyze the energy needed for Dynamic Circuit Specialization and compare this to the energy required to run the parameterized design. We also consider the static FPGA implementation of the same parameterized application and compare the power consumption by performing a power analysis. Defining the energy models for DCS and comparing the power behavior between the static and the DCS FPGA implementations, are the main contributions of this paper.

We describe the state of the art in Section II. The experimental setup to measure the FPGA core power consumption for the DCS technique is described in Section III. In Section IV, we characterize the power consumption of the parameterized FIR filter design implemented using DCS. The results of our experiments and the comparison of the power consumptions are discussed in Section V. In Section VI we further discuss results and compare the power consumptions between the conventional static implementation and the DCS implementation for a FIR filter. Finally we conclude our work in Section VII.

II. State of the Art

Dynamic Circuit Specialization enables us to use less FPGA resources (LUTs) than the conventional FPGA implementation. The reduction in the number of LUTs for a parameterized FIR filter design is about 42%. This gain also

978-1-4673-9406-2/15/$31.00 ©2015 European Union

helps to shorten the critical path of the design and thereby improves the design performance [1].

The reduction in the number of LUTs indirectly contributes to less Programmable Logic (PL) power consumption because of reduced idle power of the LUTs. However, since DCS is a tailored version of DPR for parameterized applications, it uses reconfiguration technology and therefore we need to account for the power consumed by the reconfiguration technology, specifically the CPU and the HWICAP used during the reconfiguration.

The DCS tool flow consists of two main stages: the generic stage and the specialization stage. In the generic stage, the parameterized design, described in a Hardware Description Language (HDL), is processed to yield a Partial Parameterized Configuration (PPC). The PPC contains bitstreams expressed in the form of Boolean functions of infrequently changing parameters. In [5] it is explained how a parameterized design is mapped on to the virtual Look Up Tables called Tunable Look Up Tables (TLUTs). TLUTs are intermediate representations of conventional FPGA physical LUTs that contain truth table entries expressed as Boolean functions of the parameters.

In the specialization stage, the Boolean functions are evaluated for every change in parameter values to produce the specialized bitstreams. The evaluation is performed by the Specialized Configuration Generator (SCG) and can be implemented on an embedded processor such as ARM Cortex - A9 in the Zynq-SoC. The SCG reconfigures the FPGA by swapping the specialized bitstreams into the configuration memory via the HWICAP. The bitstreams are accessed in the form of frames and a frame is defined as a minimum addressable element of an FPGA configuration.

The reconfiguration is performed using a HWICAP driver "$XHwIcap\_SetClbBits$" function [6]. The two crucial arguments of this function are:

1) Location co-ordinates of a TLUT: using this information, the frame address is generated that is used to point to the frame which contains the truth table entries of the TLUT.
2) Truth table entries: these are the specialized truth table bits that are generated after the specialization stage of the DCS.

The reconfiguration takes place in three major steps:

1) Read frames: using the frame address, the current truth table entries of the TLUT are read by fetching four consecutive frames from the configuration memory.
2) Modify frames: The current truth table entries of the TLUT are replaced by the specialized truth table bits. Therefore the modified frames contain the specialized bitstreams.
3) Write-back frames: the frames containing specialized bitstreams are written back to the configuration memory using the same frame address thus accomplishing DCS reconfiguration.

The DCS reconfiguration is a fine grained form of dynamic reconfiguration and incurs three major costs:

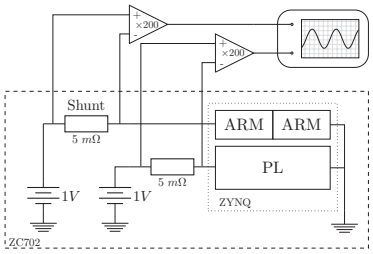

Fig. 1. Current measurement schematics on ZC702 board.

1) The *PPC memory* required to store all the Boolean functions.
2) The *evaluation time* taken to evaluate the Boolean functions upon change in parameter values.
3) The *reconfiguration time* needed to update the truth table entries of all the TLUTs.

In [7] and [8], the authors proposed various methods to reduce the major costs of DCS. However, the power consumption wasn't included as an overhead factor. In this paper, we consider power consumption as one part of the overhead of DCS and we investigate the detailed power variations of DCS.

III. Experimental Setup

A. Power Measurement

The Xilinx ZC702 board is used for the power measurements and the DCS approach is implemented on the XC7Z020 Zynq-SoC. Ten power rails are present on this platform. Each rail is equipped with a shunt resistor on which current consumption can be monitored. Two channels are more interesting for the experiment. They separately supply the ARM cores and the Programmable Logic core. An external board is designed for this purpose and two high-precision amplifiers are used to enhance the signal levels. The amplified signals are then sent to a digital oscilloscope for visualization and power trace analysis as shown in Figure 1. With this procedure, it is possible to measure power consumption variations as low as $0.1\ mW$. This accuracy is good enough for our energy analysis.

B. Zynq-SoC configuration setup

To obtain the energy models for DCS we used a clock frequency of 100MHz to drive the Programmable Logic (PL) and the same clock frequency of 100MHz for the HWICAP. The HWICAP is configured to be of the FIFO type with read and write buffer depth of 128 bytes. We used these parameters as a default project configuration in our following experiments.

The Specialized Configuration Generator (SCG) is implemented on the ARM Cortex-A9 dual core processor that operates at a clock frequency of 667 MHz. Therefore the evaluation of Boolean functions is expected to be faster than any of the tasks in the DCS.

C. Parameterized Design

We use a FIR filter with 8-bit data width and 16-taps, as a parameterized design implemented using DCS [8]. The benefits of this are discussed in [9]. The filter taps of the FIR filter are parameterized, therefore all the coefficient inputs

are the parameters and hence for each infrequent change in coefficient values, a specialized bitstream is generated and the filter taps containing constants multiplications are reconfigured accordingly.

IV. Power Characterization for Dynamic Circuit Specialization

Using the power measurement setup we were able to measure the average power values on the Zynq ZC702 Platform with the default project configuration explained in Section III-B. There are three different power consumption parts that we need to consider:

1) The FPGA Idle Power is the power consumed by the silicon area of the FPGA even if it is unused and this state of the FPGA is called the idle state.
2) The FPGA Run Power is the power consumed by the silicon area of the FPGA when the FIR filter was triggered to execute the filter function and this state of the FPGA is called the run state.
3) The FPGA Reconfiguration Power is the power consumed by the silicon area of the FPGA during DCS reconfiguration and this state of the FPGA is called the reconfiguration state.

It is to be noted that both the CPU and the PL part of the Zynq-SoC consume power in all of the above three states.

We propose an energy analysis that is based on the energy required for reconfiguring one TLUT. For this, we need to consider the time τ_{tlut} needed to reconfigure one TLUT. We measure $\tau_{tlut} = 230\mu s$.

A. Energy consumed by the reconfiguration state on top of the idle state energy:

If E_{reconf}^{tlut} denotes the energy consumed during the reconfiguration of a TLUT, on top of the idle state energy then,

$$E_{reconf}^{tlut} = (P_{reconf}^{CPU} - P_{idle}^{CPU} + P_{reconf}^{FPGA} - P_{idle}^{FPGA}) \times \tau_{tlut} \quad (1)$$

where, P_{reconf}^{CPU} is the average power consumed by the CPU during DCS reconfiguration to perform the read, modify and write-back cycles of the frames. P_{idle}^{CPU} is the power consumed by the CPU during its idle state. P_{reconf}^{FPGA} is the average FPGA reconfiguration power and P_{idle}^{FPGA} is the FPGA idle power. The idle power is defined as the power used when no reconfiguration, nor application execution is performed.

B. Relative power consumed by the reconfiguration state compared to the run state:

We also propose a relative power ratio between the reconfiguration state and the run state. If R_p denotes the relative power ratio then,

$$R_p = \frac{(P_{reconf}^{CPU} - P_{idle}^{CPU}) + (P_{reconf}^{FPGA} - P_{idle}^{FPGA})}{(P_{run}^{CPU} - P_{idle}^{CPU}) + (P_{run}^{FPGA} - P_{idle}^{FPGA})}$$
$$R_p = \frac{P_{reconf}}{P_{run}} \quad (2)$$

where P_{run} is the power consumed by the run state on top of the idle state power and depends on the size of the parameterized application. Indeed for a large parameterized design the value of P_{run} is much larger than P_{reconf} where, P_{reconf} is the power consumed by the reconfiguration state on top of the idle state.

TABLE I. Average power consumed by the CPU and the PL fabric

	Power Model	Average Power (mW)
CPU idle	P_{idle}^{CPU}	291
CPU run	P_{run}^{CPU}	384.1
CPU reconfiguration	P_{reconf}^{CPU}	390.3
FPGA idle	P_{idle}^{FPGA}	73.3 (45.6)
FPGA run	P_{run}^{FPGA}	74.7 (46.8)
FPGA reconfiguration	P_{reconf}^{FPGA}	68.7

Note: Power values after gating the HWICAP clock are mentioned between brackets.

V. Experiments and Results

From our measurements, we were able to extract the average power consumption of the Programmable Logic (PL) and the ARM Processor (CPU) of the Zynq-SoC. The average power values are tabulated in Table I.

From equation 1 the estimated average energy E_{idle}^{tlut} is 21.8 mJ. And the relative power ratio R_p (equation 2) is 4.1. Since the relative power ratio is greater than 1, the power consumption during the reconfiguration is higher than the power consumption during the execution of the FIR filter function. This is not a desired situation and we will further investigate this ratio later.

A. FPGA PL power drop during reconfiguration:

Interestingly, in Table I, the FPGA reconfiguration power is smaller than the FPGA idle power. In order to understand this phenomenon, the power curve is extracted and is shown in Figure 2. The reconfiguration happens between time units 0 and 90. Before and after that time, the system is running the FIR filter application. Clearly, the CPU power increases during the reconfiguration phase because the CPU has to perform the Boolean evaluation and the reconfiguration by swapping the specialized frames into the FPGA configuration memory via the HWICAP.

However, for the FPGA PL power we notice a significant power drop during the DCS reconfiguration phase compared to the FIR run state. An average power drop of 6.2 mW was observed. Further investigation revealed that there is a power drop only during frame read activity of the DCS reconfiguration as shown in Figure 3.

During the frame read activity, the configuration data (bitstream) is fetched from the FPGA configuration memory. The fetched bitstreams are first stored in the HWICAP read FIFO buffer. The maximum data that the read FIFO buffer can hold is a user configurable parameter and in our experiment it is set to 128 bytes. Once the read FIFO buffer is full, the ICAP has to wait until all the data in the FIFO buffer is received by the CPU via the AXI bus. This waiting state is established by turning off the ICAP's clock. Once the ICAP clock is turned off there will be no transaction of data between the ICAP port and the FPGA's configuration memory. Turning

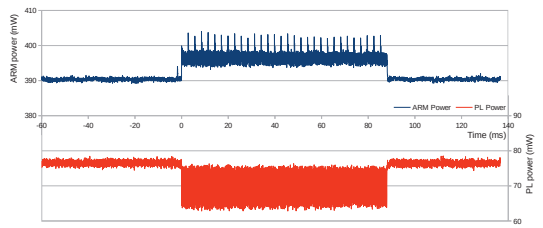

Fig. 2. Average power consumption of CPU and FPGA during run and reconfiguration state.

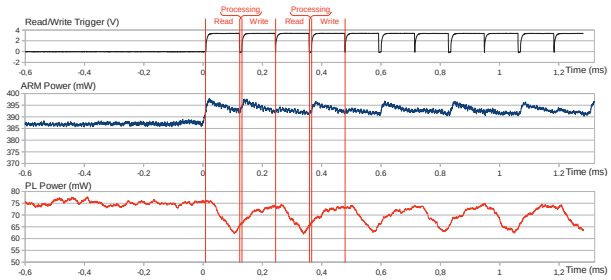

Fig. 3. Average power consumption of CPU and FPGA during Frame read and Frame write activities.

TABLE II. FPGA PL POWER GRADIENT

PL fabric AXI bus clock (MHz)	HWICAP clock (MHz)	FIFO depth	Average Power gradient (mW)
100	100	128	- 6.2
100	20	128	+ 0.07
100	100	256	- 3
100	20	256	+ 1

Note: a "+" indicates an increase in power consumption and a "-" indicates a decrease in power consumption.

off the ICAP's clock causes the significant drop in the FPGA PL power and therefore it proves to be the main reason for the power fluctuations as depicted in Figure 3. The power drop is hence due to a mismatch between the computation bandwidth and the communication bandwidth (communication bandwidth limited design).

As a communication limited design (with wait cycles for data movement) increases the total time needed for the reconfiguration, the power drop does not necessarily result in a lower energy usage as well. Indeed, the increased time in fact results in a higher energy usage.

The best solution to avoid the HWICAP to stall is to increase the FIFO depth and clock the AXI bus much faster than the HWICAP. To simulate this situation, we performed the experiments with the HWICAP clock of 20 MHz. The average power gradient for the different configurations of the HWICAP clock, FIFO depth and the PL fabric is tabulated in Table II. We observe that the AXI bus (with 100 MHz) is fast enough to receive the data from the HWICAP so the read FIFO is less likely full, therefore the HWICAP fetches the data as fast as possible. Also, the average power gradient values are halved for the experiments with the FIFO depth 256. This confirms the reason for the PL power drop during the frame read activity.

During the frame write activity, the HWICAP does not stop the ICAP's clock because the HWICAP constantly expects the ICAP's attention and makes it receive the data from the write

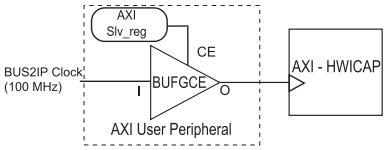

Fig. 4. Clock gating for the AXI-HWICAP.

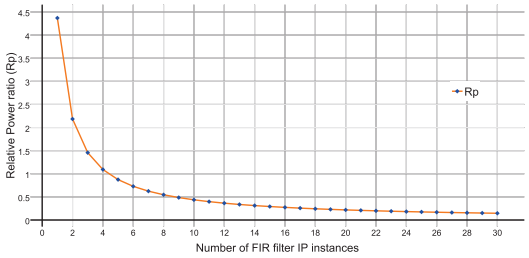

Fig. 5. Relative Power ratio as a function of the number of FIR filter IP instances.

FIFO buffer irrespective of whether the write FIFO buffer is full or not.

The Xilinx AXI-HWICAP IP consumes a huge idle power of 31.2 mW because the IP lacks clock gating and is active even if the reconfiguration process is unused. Therefore, to make DCS functional an extra power of 31.2 mW is required irrespective of whether or not the HWICAP is used for reconfiguration during the operation of the FIR filter. In order to make DCS more power efficient, we include the clock gating technique to the AXI-HWICAP IP and reduce the HWICAP idle power. The HWICAP can be replaced by a custom reconfiguration controller: MiCAP [10]. It is a lite-weight controller and consumes less FPGA resources compared to the HWICAP and therefore, the power consumption by the controller can be reduced.

B. Xilinx HWICAP with Clock gating

The clock of the AXI-HWICAP IP is gated with the help of a user AXI-lite peripheral. The required clock gating for the AXI-HWICAP is depicted in Figure 4. The CE line is controlled by a user accessible AXI slave register. The slave register is software controlled and hence we can turn ON/OFF the HWICAP clock during the power measurements. After gating the AXI-HWICAP clock, we were able to reduce the idle power of the AXI-HWICAP with 27.9 mW (31.2 - 3.3 = 27.9) ($\approx$ 90%). The HWICAP still consumes a power of 3.3 mW during its idle state as tabulated in Table IV. The corresponding FPGA idle power was reduced to 45.6 mW.

The relative power ratio of equation 2 changes after introducing the clock gating for the HWICAP, and can be expressed as a function of the number of FIR filter instances N_{FIR}.

$$R_p = \frac{P_{reconf} + P_{idle}^{HWICAP}}{P_{run} \times N_{FIR}} \tag{3}$$

As shown in Figure 5, an increase in the number of FIR IP instances decreases the magnitude of the ratio R_p.

TABLE III. FIR POWER CONSUMPTION COMPARISON STATIC VS DCS

Row no.	Project	P^{CPU}_{run} (mW)	P^{CPU}_{reconf} (mW)	P^{FPGA}_{run} (mW)	P^{FPGA}_{reconf} (mW)
1	No FIR, No HWICAP	382.8	Nil	39.5	Nil
2	Static FIR, No HWICAP	383.1	Nil	45.4	Nil
3	DCS FIR, No HWICAP	383.4	Nil	43.3	Nil
4	DCS FIR, HWICAP	384.1	390.3	74.5	68.5
5	DCS FIR, HWICAP with clock gating (Clock OFF)	384.1	390.3	46.6	68.5

TABLE IV. DIFFERENTIAL POWER RESULTS.

Extracted Power Results	CPU (mW)	PL fabric (mW)	Total Energy (μJ)
Static FIR, idle + run power (without HWICAP) (row no. 2 - row no. 1)	0.3	6.1	0.06 (1.86)
DCS FIR without HWICAP, idle + run power (row no. 3 - row no. 1)	0.6	3.8	0.044
HWICAP idle power, with Clock OFF (row no. 5 - row no. 3)	0.7	3.3	NA
HWICAP idle power, with Clock ON (row no. 4 - row no. 3)	0.7	31.2	NA
HWICAP reconf power	7.5	25.2	2838.3
DCS FIR (run) with HWICAP idle (row no. 5 - row no. 1)	1.3	7.1	0.08 (1.72)
DCS FIR with HWICAP, reconf power	7.5	29	3175.7

Note 1: The energy values for 3 FIR filter instances are mentioned between brackets.
Note 2: The row numbers mentioned in the table are the row numbers of Table III .

VI. POWER ANALYSIS

A. DCS versus static power comparison

In this section, we investigate how DCS affects the global power consumption of the system. The main objective of this experiment is to compare the global power consumption of the FIR using two different implementations:

1) FIR with static implementation: the FIR was implemented without using the reconfiguration technology. Instead, the coefficient inputs of the FIR are connected directly to the slave registers of the AXI bus and with the help of the CPU, the user can change the coefficient values at the software level. Therefore we do not make use of the HWICAP and the DCS reconfiguration technology.
2) FIR with DCS implementation: the FIR (of one IP instance) was implemented using the reconfiguration technology. As explained in Section II and Section III, we use the DCS reconfiguration technology to change the FIR coefficients by reconfiguring the multiplications of the filter taps. Therefore, the user can change the coefficient values of the FIR filter using the CPU at the hardware level.

To get a clear picture of the comparison, we measured the power consumption of the CPU and the PL for projects with different configurations given in Table III. The differential power and the energy consumption of the idle and run power combined together are tabulated in Table IV. These values are the difference in power values between different rows of Table III. For example, the static FIR (idle + run) power is obtained by the difference in corresponding power values of row no.2 and row no.1 of Table III.

The power consumed by the HWICAP during the reconfiguration process is obtained by the difference between P^{FPGA}_{reconf} of row no.4 and P^{FPGA}_{run} of row no.3 of Table III i.e. (68.5 - 43.3 = 25.2 mW). The CPU power consumption during interaction with HWICAP is obtained by the difference between P^{CPU}_{reconf} of row no.4 and P^{CPU}_{run} of row no.1 i.e. (390.3 - 382.8 = 7.5 mW). Therefore total HWICAP reconfiguration power consumption is 32.7 mW (25.2 + 7.5 = 32.7). The DCS FIR power together with HWICAP reconfiguration power is obtained by the difference P^{FPGA}_{reconf} of row no.5 and P^{FPGA}_{run} of row no.1 of Table III i.e. (68.5 - 39.5 = 29 mW). The CPU power is obtained by the difference between P^{CPU}_{reconf} of row no.5 and P^{CPU}_{run} of row no.1 i.e (390.3 - 382.8 = 7.5 mW).

To investigate the power consumption by the DCS FIR (LUTs) compared to the static FIR (LUTs), we have removed the HWICAP from the DCS implementation and measured the power consumption. The FIR with DCS implementation consumes less FPGA idle and run power without HWICAP compared to the static FIR implementation. We observe a difference of 2.3 mW (6.1 - 3.8 = 2.3) ($\approx$ 36%). This difference is because of the reduction in FPGA resources (LUTs) utilized by the FIR filter. However, there is a huge FPGA reconfiguration (PL + CPU) power difference of 30.1 mW (29-6.1 + 7.5-0.3) between the static FIR (without any reconfiguration) and the FIR with DCS which proves to be an unavoidable overhead. During the reconfiguration process, the CPU consumes a maximum of 7.5 mW and this power is considered to be an overhead.

The corresponding average energy consumption (CPU + PL) is also listed. For one FIR IP instance the DCS implementation consumes more energy (0.02μJ) than the static FIR filter implementation. The HWICAP consumes extra energy in the DCS implementation (plus 0.036μJ) and this energy is constant irrespective of the number of FIR IP instances. Therefore, we need bigger designs (more FIR filter implementations) before the energy calculations start to be in favor of reconfiguration. We investigated that DCS becomes energy efficient for 3 or more FIR IP instances and the corresponding energy values are shown within the brackets in Table IV. We observe an energy gain of 0.14μJ. More details are discussed in SubSection VI-B.

B. Power efficient DCS implementation and its reconfiguration rate

The results from the previous section show that the reconfiguration process using the HWICAP is power-hungry. However, the reconfiguration process is triggered only if the parameters (coefficients of the FIR filter) change. It is interesting to investigate the reconfiguration rate (expressed as the reconfiguration time over the total execution time), allowed under the constraint that the DCS energy is less than or equal

to the static energy as a function of number of the FIR filter IPs. On the one hand only 950 LUTs are used to implement the FIR filter with DCS, and on the other hand 2525 LUTs are used for the static FIR filter implementation.

There are two important parameters that need to be considered to evaluate the global average energy (E_{static} and E_{DCS}): the number of FIR filter IPs (N_{FIR}) and the relative amount of time spent for reconfiguration (the reconfiguration rate R_{rate}) which is $R_{rate} = \frac{T_{reconf}}{T_{reconf}+T_{run}}$, where T_{reconf} is the time taken to reconfigure all the TLUTs of the FIR filter and T_{run} is the time taken to execute the FIR filter function. Accordingly, we deduce equation 4 and equation 5 for the energy needed for the execution of the implementation for a single round of constant coefficient values.

$$E_{static} = N_{FIR} \times P_{FIR}^{static} \times T_{run}^{static} + P_{coef} \times T_{coef} \quad (4)$$

$$E_{DCS} = N_{FIR} \times P_{FIR}^{DCS} \times T_{run}^{DCS} + P_{reconf} \times T_{reconf} + P_{idle}^{HWICAP} \times (T_{reconf} + T_{run}) \quad (5)$$

where, P_{reconf} and P_{coef} are the power consumption during the change of coefficient values for the DCS and static implementation of the FIR respectively. T_{run}^{static} and T_{run}^{DCS} are the time taken to execute the filter functions for the FIR with static and DCS implementations respectively.

Assuming P_{coef} is negligible, we can solve for the variable R_{rate} for a worst case scenario [1] where $T_{run}^{static} = T_{run}^{DCS}$.

$$R_{rate} = \frac{(P_{FIR}^{static} - P_{FIR}^{DCS}) - (\frac{P_{idle}^{HWICAP}}{N_{FIR}})}{(P_{FIR}^{static} - P_{FIR}^{DCS}) + (\frac{P_{reconf}}{N_{FIR}})} \quad (6)$$

This ratio provides the reconfiguration rate as a function of the number of FIR IP instances (reconfigured) for the condition that the average energy of DCS and static implementations are equal. Accordingly, we can plot a graph shown in Figure 6. Clearly, for less than 3 FIR IP instances DCS is inefficient in energy since the reconfiguration rate is negative. The DCS reconfiguration is energy efficient for 3 or more FIR IP instances if it has reconfiguration rate within the shaded region. For example, suppose if the reconfiguration rate is 0.3, then we need to run at least 10 FIR filter IPs before the DCS reconfiguration becomes energy efficient. Vice versa, if we have 10 FIR filters, the reconfiguration time should not take more than 30% of the total time in order to remain energy efficient.

VII. Conclusion

Power consumption is one of the overhead factors of a DCS reconfiguration. At the same time, DCS can save power in running implementations more efficiently. Based on a set of experiments, this paper presents the FPGA core power consumption during the DCS reconfiguration and corresponding energy analysis for DCS. The dependency on the AXI-HWICAP clocking along with the FIFO read buffer depth influences the global average power consumption. The AXI-HWICAP lacks a clock gating solution and therefore it consumes a significant amount of power compared to all other components during its idle state. We have shown that providing a clock gating technique to AXI-HWICAP will reduce the idle power consumption by $\approx$ 90%. We have expressed the reconfiguration rate of DCS as a function of the number of FIR IP instances to investigate a case that contains multiple FIR IP instances in which the DCS is energy efficient compared to the static FIR implementation.

[1] the DCS implementation usually runs about 20% faster than the static implementation.

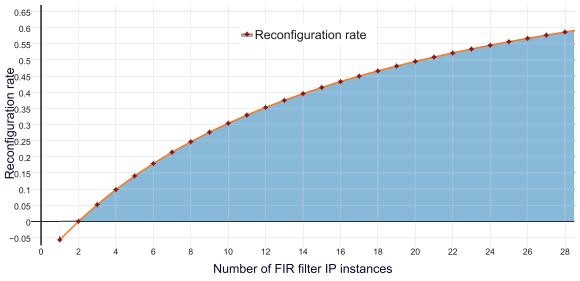

Fig. 6. Reconfiguration rate as function of number of FIR filter instances.

References

[1] K. Bruneel, W. Heirman, and D. Stroobandt, "Dynamic Data Folding with Parameterizable Configurations," *ACM Transactions on Design Automation of Electronic Systems*, vol. 16, no. 4, 2011.

[2] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Invited paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAS," in *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, Aug 2006, pp. 1–6.

[3] K. Bruneel, F. Abouelella, and D. Stroobandt, "Automatically mapping applications to a self-reconfiguring platform," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, April 2009, pp. 964–969.

[4] R. Bonamy, D. Chillet, S. Bilavarn, and O. Sentieys, "Power Consumption Model for Partial and Dynamic Reconfiguration," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, Dec 2012, pp. 1–8.

[5] K. Heyse, T. Davidson, E. Vansteenkiste, K. Bruneel, and D. Stroobandt, "Efficient implementation of Virtual Coarse Grained Reconfigurable Arrays on FPGAs," in *Proceedings of the 23rd International Conference on Field Programmable Logic and Applications*. Piscataway, NJ, USA: IEEE, 2013, pp. 1–8.

[6] A. Kulkarni, K. Heyse, T. Davidson, and D. Stroobandt, "Performance Evaluation of Dynamic Circuit Specialization on Xilinx FPGAs," in *Proceedings of the 11th FPGAworld Conference*, ser. FPGAworld '14, 2014.

[7] F. Mostafa Mohamed Ahmed Abouelella, K. Bruneel, and D. Stroobandt, "Efficiently generating FPGA configurations through a stack machine," in *Field Programmable Logic and Applications, 20th International conference, Abstracts*, Milano, Italy, 2010.

[8] A. Kulkarni, T. Davidson, K. Heyse, and D. Stroobandt, "Improving Reconfiguration speed for Dynamic Circuit Specialization using Placement Constraints," in *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, Dec 2014, pp. 1–6.

[9] F. Mostafa Mohamed Ahmed Abouelella, T. Davidson, W. Meeus, K. Bruneel, and D. Stroobandt, "How to efficiently implement Dynamic Circuit Specialization systems," *ACM TRANSACTIONS ON DESIGN AUTOMATION OF ELECTRONIC SYSTEMS*, p. 38, 2013.

[10] A. Kulkarni, V. Kizheppatt, and D. Stroobandt, "MiCAP: A custom Reconfiguration Controller for Dynamic Circuit Specialization," in *ReConFigurable Computing and FPGAs (ReConFig), 2015 International Conference on*, Dec 2015, pp. 1–6.