Optimal Implementation of Power Saving Techniques in CGR Systems

Tiziana Fanni

Università degli Studi di Cagliari, DIEE
tiziana.fanni@diee.unica.it

Abstract. Coarse-Grained Reconfigurable (CGR) architectures combine high performance with flexibility, allowing the execution of a large set of applications over the same substrate. However, they are also required to be energy efficient. This work focuses on a methodology to identify which parts of a CGR architecture may benefit from the application of power saving techniques, guiding the designers towards an optimal implementation of clock gated and power gated designs.

Keywords: Reconfigurable Systems · Power Saving · Power Gating · Clock Gating · Power Modeling · Datapath Merging.

1 Motivation and Background

Nowadays small portable devices are required to efficiently execute multiple fancy functions. Reconfigurable systems [1] are a suitable solution for these colliding requirements. In particular, Coarse-Grained Reconfigurable (CGR) systems offer higher performance with a certain degree of flexibility. In CGR systems all the resources belonging to the possible configurations are instantiated in the substrate, and they are multiplexed in time. Thus CGR offer fast reconfiguration, paying the cost of the power consumed by the resources present in the substrate but not involved in active configuration.

Power consumption in digital devices is computed as in Equation 1, where: *1)* P_{lkg} is the static power dissipation caused by leakage currents, consumed even while no circuit activity is present; *2)* P_{int} is the dynamic power consumption mainly due to the cell switching activity; (3) P_{net} is again a dynamic term, related to the interconnection. With technologies below 90 nm, designers are required to minimize both static (P_{lkg}) and dynamic ($P_{int} + P_{net}$) terms.

$$P = P_{lkg} + P_{int} + P_{net} \tag{1}$$

A popular technique to minimize dynamic power consumption is the clock gating (CG). It consists on switching off the clock of resources not involved in the computation, and it is applied automatically, at gate level, by commercial synthesizers. On the other hand, power gating (PG) shuts-off the power of unused logic, thus acting also on static consumption. PG requires the instantiation of several resources (i.e. *sleep transistors* to switch on/off the power supply, *isolation cells* to avoid the transmission of spurious signals and *state retention logic* to maintain the internal state of the gated region) and the rules to manage them are quite complicated, thus commercial tools do not provide it automatically.

In a CGR architecture, considering the minimum set of disjointed *Logic Regions* (*LRs*), composed of processing elements that are always active/inactive together, it is possible to apply to all of them CG or PG techniques. However, if on one hand CG requires only one *AND* gate for each region to switch off the clock tree, PG has a much higher cost due to the additional logic, and the power overhead may easily overcome the power saved by switching off the unused resources. The research presented in this paper studied an automatic system-level analysis and implementation flow to estimate in advance the cost of CG and PG application in a CGR system, leading to the identification of those *LRs* that can benefit from the application of these power saving techniques.

2 Methods and Algorithm

The proposed model (see Equations 2 and 3) estimate P_{lkg} and P_{int} of Equation 1, when PG and CG are applied at *LRs* level. They are composed of two terms: *1)* $P_{lkg/intON}(LR_i)$ - consumption within the considered *LRs*; *2)* $Ext\_Over_{lkg/int}(LR_i)$ - consumption due to the logic inserted outside the *LR*.

$$\begin{aligned}
&P_{lkg/int}(LR_i) = P_{lkg/intON}(LR_i) + Ext\_Over_{lkg/int}(LR_i) = \\
&= \sum_{actors \in LR_i} [P_{lkg/int}(cmb) + P_{lkg/int}(RC) * \#rtn + \\
&+ P_{lkg/int}(reg) * (\#reg - \#rtn)/\#reg] * Ti_{ON} + \\
&+ [P_{lkg/int}(ISO_{ON}) * Ti_{ON} + P_{lkg/int}(ISO_{OFF}) * Ti_{OFF}] * \#iso + \\
&+ [P_{lkg/int}(Contr_{ON}) * Ti_{ON} + P_{lkg/int}(Contr_{OFF}) * Ti_{OFF}] + \\
&+ [P_{lkg/int}(CG_{ON}) * Ti_{ON} + P_{lkg/int}(CG_{OFF}) * Ti_{OFF}]
\end{aligned} \tag{2}$$

$$\begin{aligned}
&P_{lkg/int}(LR_i) = P_{lkg/intON}(LR_i) + Ext\_Over_{lkg/int}(LR_i) = \\
&= \sum_{actors \in LR_i} [P_{lkg/int}(cmb) + P_{lkg/int}(reg) * Ti_{ON}] + \\
&+ [P_{lkg/int}(CG_{ON}) * Ti_{ON} + P_{lkg/int}(CG_{OFF}) * Ti_{OFF}]
\end{aligned} \tag{3}$$

In particular, the Equations present the following contributions:

- **Combinatorial Logic** [$P_{lkg/int}(cmb) * Ti_{ON}$ and $P_{lkg/int}(cmb)$]: sum of the contributions of the combinational cells, weighted for the activation time Ti_{ON}. CG switches off only the clock tree; therefore, combinational logic is always active when CG is applied.
- **Sequential Logic** [$P_{lkg/int}(reg) * (\#reg - \#rtn)/\#reg * Ti_{ON}$ and $P_{lkg/int}(reg) * Ti_{ON}$]: this term consider only those registers ($\#reg$) inside the *LR* that are not replaced by the retention cells. CG does not have effect on the static

power, thus when it is applied only the internal contribution is multiplied by Ti_{ON} ($P_{lkg}(reg)$ and $P_{int}(reg) * Ti_{ON}$).

- **Retention Cells** $[P_{lkg/int}(RC) * \#rtn * Ti_{ON})]$: This term consider the retention cells inserted to preserve the status of some registers ($\#rtn$), when PG is applied. $P_{lkg/int}(RC)$ is the consumption of a single retention cell.
- **Isolation Cells** $[[P_{lkg/int}(ISO_{ON}) * Ti_{ON} + P_{lkg/int}(ISO_{OFF}) * Ti_{OFF}] * \#iso]$: In the PG case, isolation cells are inserted and their dissipation is proportional to their overall number.
- **Clock Gating Cells** $P_{lkg/int}(CG_{ON}) * Ti_{ON} + P_{lkg/int}(CG_{OFF}) * Ti_{OFF}]$: Clock gating cells are used in both PG and CG cases. In the PG case they are required for the proper operation of the retention cells.
- **Power Controller** $[P_{lkg/int}(Contr_{ON}) * Ti_{ON} + P_{lkg/int}(Contr_{OFF}) * Ti_{OFF}]$: inserted to properly drive the enable signals to the power saving logic.

This estimation model has been embedded in the automatic design flow for CGR systems offered by the Multi-Dataflow Composer (MDC) tool [2]. MDC handles the automatic composition and deployment of CGR systems, starting from the high-level specification of the kernels to be executed, represented as networks [3]. MDC offers also other functionalities: *1)* a *structural profiler* that performs the design space exploration of the implementable multi-functional systems to determine the optimal CGR substrate [4]; *2)* a *power manager* that partitions the multi-functional network, identifying the minimum set of disjointed *LRs* and applies to all of them either CG [5] or PG [6] (generating an ad-hoc common power format (CPF) file to specify the power intent early in the design [7]); a *rapid prototyper* to embed the CGR system into a Xilinx compliant IP [8, 9]. The MDC power manager has been extended to analyze the design, exploiting the estimation flow, to identify which regions may mostly benefit of PG and CG application [10, 11].

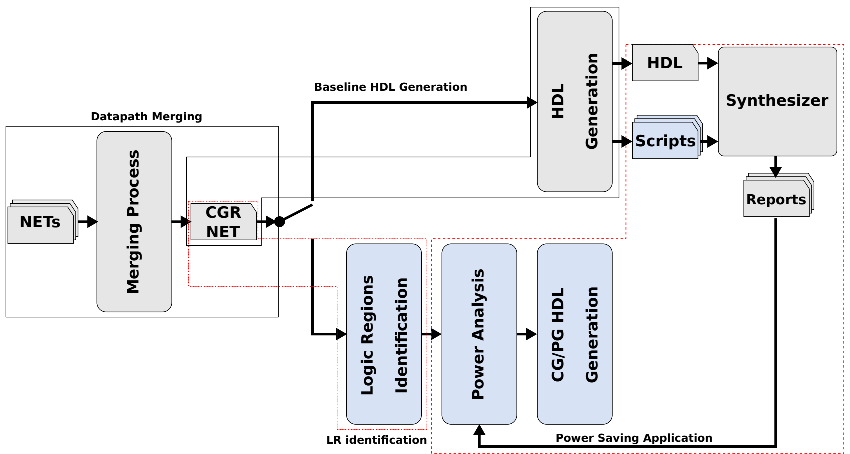

Fig. 1. Proposed analysis and implementation flow.

Fig. 1 shows the complete analysis and implementation flow. MDC derives the HDL code of the baseline CGR system and provides the scripts to perform the synthesis and all the simulations for the system back annotation. Commercial tools are used to determine the baseline system power reports, which are fed back to MDC. MDC identifies the *LRs* and estimates the number of isolation cells ($\#iso$) at dataflow level by analyzing the connections between the different *LRs*. These data, together with the information gathered by the reports of a single synthesis run of the baseline CGR system netlist, are used to estimate power consumption of the identified *LRs* (consumption of the additional power saving logic - *isolation cells* and *retention cells* is estimated by characterizing the adopted technology libraries).

The power estimation flow analyzes the *LRs* according to Algorithm 1:

– *area evaluation*: PG overhead may affect the power consumption of the smaller *LRs*. Thus, too small *LRs* are evaluated only for CG application. The threshold value is set by the user.
– *PG evaluation*: the power variations due to PG and CG application are estimated. If PG results more convenient than CG, the *LR* is candidated for the implementation of PG application. Otherwise, CG is evaluated.
– *CG evaluation*: the power variation due to CG application is estimated. If it is not able to save any power, *LR* is discarded and its logic will be included in the always on domain.

Algorithm 1: Power saving strategy selection for CGR systems.

```
PG_set is empty;
CG_set is empty;
foreach $LR_i$ in set LRs do
    evaluate_area($LR_i$, $area_{th}$)
end
function: evaluate_area($LR_i$, $area_{th}$):
calculate_$LR_i$_area;
if $area_{LR} > area_{th}$ then
    evaluate_PG($LR_i$);
else
    evaluate_CG($LR_i$);
end
PG evaluation: evaluate_PG($LR_i$):
estimate_PG_total_overhead;
if $PG\_total\_overhead < 0$ then
    estimate_CG_total_overhead;
    if $PG\_total\_overhead <$
      $CG\_total\_overhead$ then
        add $LR_i$ to PG_set;
    else
        add $LR_i$ to CG_set;
    end
else
    evaluate_CG($LR_i$);
end
evaluate_CG($LR_i$);
CG evaluation: evaluate_CG($LR_i$):
estimate_CG_total_overhead;
if $CG\_total\_overhead < 0$ then
    add $LR_i$ to CG_set;
end
```

Table 1. FFT and Zoom test-cases. *area* rows report percentage area of each *LR* wrt total area. Other rows report percentage power variation (both estimated and real) of the design when CG or PG are applied to the considered *LR*. *N.A.* refers to purely combinational *LRs*, where CG is not applied.

	LR1	LR2	LR3	LR4	LR5	LR6	LR7	LR8	LR8	LR10	LR11	LR12	LR13
FFT targeting 90 nm													
area	62.48	0.65	15.8	0.44	0.45	0.43	7.92	1.36	–	–	–	–	–
CG real	-5.64	-1.42	-1.29	N.A.	N.A.	N.A.	-0.39	-0.12	–	–	–	–	–
CG est	-5.65	-1.42	-1.29	N.A.	N.A.	N.A.	-0.39	-0.12	–	–	–	–	–
PG real	-25.37	-0.59	-6.22	-0.67	0.14	0.18	-1.92	1.78	–	–	–	–	–
PG est	-25.22	-0.54	-6.28	0.01	0.11	0.14	-1.92	1.89	–	–	–	–	–
Zoom targeting 90 nm													
area	6.39	34.39	6.44	2.00	0.65	5.01	1.86	13.13	3.40	5.18	3.95	5.58	6.92
CG real	-6.31	-23.37	-6.51	-0.96	N.A.	-0.87	-1.03	-6.98	-2.44	-5.48	-3.28	-4.27	0.65
CG est	-6.30	-23.36	-6.50	-0.95	N.A.	-0.87	-1.03	-6.98	-2.43	-5.47	-3.32	-4.26	-0.64
PG real	-6.33	-23.46	-6.54	-0.95	0.16	-0.82	-1.03	-7.05	-2.45	-5-50	-3.29	-4.28	-0.61
PG est	-6.32	-23.46	-6.53	-0.95	0.15	-0.81	-1.02	-7.06	-2.44	-5.49	-3-34	-4.27	-0.62
Zoom targeting 45 nm													
area	6.46	34.00	6.40	2.07	0.71	5.19	1.85	13.00	3.36	5.04	3.87	5.53	6.95
CG real	-5.50	-22.02	-6.26	-0.91	N.A.-	-0.83	-0.93	-6.71	-2.34	-5.26	-3.20	-4.16	-0.62
CG est	-5.50	-22.06	-6.27	-0.91	N.A.	-0.83	-0.93	-6.72	-2.35	-5.27	-3.25	-4.17	-0.62
PG real	-5.69	-22.98	-6.56	-0.94	0.21	-0.74	-0.95	-7.46	-2.48	-5.47	-3.36	-4.33	-0.58
PG est	-5.68	-22.99	-6.56	-0.94	0.19	-0.74	-0.95	-7.46	-2.47	-5.47	-3.42	-4.32	-0.58

3 Methodology Evaluation

To assess the proposed methodology, two different applications are considered, an FFT application and a computing core accelerating a zoom application. The Fast Fourier Transform (FFT) [12] is an optimised algorithm for the Discrete Fourier Transform (DFT) calculation; a DFT of size 2 (radix-2) takes the name of butterfly. The adopted use case involves a CGR radix-2 FFT of size 8, where 4 different configurations are available, FFT that uses: *1)* 12 butterflies; *2)* 4 butterflies; *3)* 2 butterflies; *4)* 1 butterfly. In this design MDC identified 8 *LRs*. The Zoom coprocessor is composed of seven computational kernels: *1)* absolute value calculation; *2)* Bilevel/grayscale block checking; *3)* Linear combination calculation; *4)* Cubic filter convolution; *5)* Median calculation; *6)* Maximum/minimum finding; *7)* Edge block checking. These kernels have been modelled as networks and implemented over a CGR hardware accelerator. In this CGR co-processor MDC identified 13 *LRs*. To validate cross-application and cross-technology effectiveness of the proposed estimation model, this section presents assessment on three implementations: FFT targeting a 90 nm CMOS technology; Zoom targeting a 90 nm CMOS technology; Zoom targeting a 45 nm CMOS technology.

These designs have been synthesized using Cadence RTL Compiler, then generated post-synthesis simulation reports have been fed back to MDC tool to estimate the power consumption of gated *LRs*, by applying Equations 2 and 3. In order to calculate the accuracy of the proposed estimation models, one power gated and one clock gated design for each identified *LR* have been implemented. Tab. 1 reports the real (extracted from the post-synthesis reports) and estimated percentage power variation for each *LR*, respectively when CG and PG strategies are applied. Comparing real power variations with estimated ones we see that the worst estimation is related to *LR4* of FFT, whose percentage power variation is so low (-0.67%) that it is impossible to estimate it accurately. For this reason, the algorithm that evaluates the *LRs* keeps in consideration also their area.

Tab. 1 also depicts the percentage area of each *LR* with respect to design area. As expected, biggest regions (*LR1* in FFT and *LR2* in Zoom) are the ones with the highest power saving, regardless of the considered strategy (PG or CG). Also the composition of the considered *LR* has an impact on the effectiveness of the applied power saving technique. *LR1* of FFT is 99% combinational, thus CG can save really little amount of power in this region. However, simply considering the *LR* size may not be sufficient to identify the best candidate for power saving; indeed it is possible to notice that *LR3* of FFT is 15.8% of total area, but switching it off only saves about 6% or total power.

The presented methodology speeds-up the evaluation of power saving application estimating in advance the effectiveness of PG or CG if applied to the *LRs* of a CGR system. For a CGR system composed of N input networks, only *one* of the baseline design without any power management, and N (one for each kernel) simulations to retrieve the real switching activity of the system, are required. Otherwise, it would be necessary to implement a design for each identified *LR* for both CG and PG applications, plus the baseline one (to evaluate the cost and benefit of the chosen power saving technique).

As a follow up of this research it is necessary to improve the model, considering in the power estimation also the contribution of the interconnection which currently is not addressed. Furthermore, power switches overhead is not considered yet; these sleep transistors are inserted only during place and route flow, and a way to estimate in advance how many switches are going to be inserted for each *LR* has not been explored yet. The number of switches has effect on the rush currents during power-up transitions, and on the power-down/up timing, thus also these issues are going to be addressed in a future work.

4 Related Works

To the best of our knowledge, literature does not treat the problem of modeling PG and CG costs in CGR designs. Shafique at al. [13] focuses on low-power techniques and power modeling for FPGAs. In [14], only CG is taken into account: different power states are defined and their consumption is characterized by low-level power analysis results. The work in [15] focuses on estimating the leakage reduction for PG and reverse body bias.

Other approaches perform an estimation that considers different components. For instance, Stokke at al. [16] propose a power modeling method for the Tegra K1 CPU, that taking into account measured rail voltages and fine-grained hardware activity predictors, expose components such as rail and core leakage currents. The FALPEM framework [17] provides power estimations at pre-register transfer level (RTL) stage, specifically targeting the power consumed by clock network and interconnection, but PG and CG costs are not defined. Finally, Li et al. [18] propose an architecture-level integrated power, area, and timing modeling framework for multi-core systems, that evaluates system building blocks (i.e., CPU, buses, etc.) for different technology nodes, providing also PG support.

References

1. R. Hartenstein, "A Decade of Reconfigurable Computing: a Visionary Retrospective," in *Proc. of Design, Automation and Test in Europe (DATE'01)*, 2001.
2. F. Palumbo *et al.*, "Power-Awarness in Coarse-Grained Reconfigurable Multi-Functional Architectures: a Dataflow Based Strategy," *Journal of Signal Processing Systems*, vol. 87, pp. 81–106, Apr 2017. doi: 10.1007/s11265-016-1106-9.
3. C. Sau *et al.*, "Automated design flow for multi-functional dataflow-based platforms," *Journal of Signal Processing Systems*, vol. 85, pp. 143–165, Oct 2016.
4. F. Palumbo *et al.*, "Power-awarness in coarse-grained reconfigurable designs: A dataflow based strategy," in *Workshop on Signal Processing Systems*, 2014.
5. F. Palumbo *et al.*, "Coarse-grained reconfiguration: dataflow-based power management," *IET Computers Digital Techniques*, vol. 9, no. 1, pp. 36–48, 2015.
6. T. Fanni *et al.*, "Automated Power Gating Methodology for Dataflow-based Reconfigurable Systems," in *Conference on Computing Frontiers*, 2015.
7. SI2., *Si2 Common Power Format Specification*^{TM} *- Version 2.1*, Dec. 2014.
8. C. Sau *et al.*, "Automatic generation of dataflow-based reconfigurable co-processing units," in *Conf. on Design and Architectures for Signal and Image Processing*, 2014.
9. C. Sau *et al.*, "Reconfigurable coprocessors synthesis in the MPEG-RVC domain," in *Conference on ReConFigurable Computing and FPGAs*, 2015.
10. T. Fanni *et al.*, "Power and Clock Gating Modelling in Coarse Grained Reconfigurable Systems," in *Conference on Computing Frontiers*, 2016.
11. F. Palumbo *et al.*, "Modelling and automated implementation of optimal power saving strategies in coarse-grained reconfigurable architectures," *Journal of Electrical and Computer Engineering*, p. 27, 2016. doi: 10.1155/2016/4237350.
12. J. Cooley and J. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series, vol. 19," *Mathematics of Computation*, 1965.
13. M. Shafique *et al.*, "Adaptive Energy Management for Dynamically Reconfigurable Processors," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 50–63, 2014.
14. J. Yi and J. Kim, "Power modeling for digital circuits with clock gating," *IEICE Electronics Express*, vol. 12, no. 24, pp. 20150817–20150817, 2015.
15. H. Xu *et al.*, "Run-time Active Leakage Reduction by Power Gating and Reverse Body Biasing: An Energy View," in *Conference on Computer Design*, 2008.
16. K. R. Stokke *et al.*, "High-Precision Power Modelling of the Tegra K1 Variable SMP Processor Architecture," in *Symposium on Embedded Multicore/Many-core Systems-on-Chip*, 2016.
17. A. Chhabra *et al.*, "FALPEM: framework for architectural-level power estimation and optimization for large memory sub-systems," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1138–1142, 2015.
18. S. Li *et al.*, "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing," *TACO*, vol. 10, no. 1, p. 5, 2013.