Hierarchical and Multiple-Clock Domain High-Level Synthesis for Low-Power Design on FPGA

Ghizlane Lhairech-Lebreton, Philippe Coussy and Eric Martin
Université de Bretagne-Sud / Lab-STICC
First-name.Last-name@univ-ubs.fr

***Abstract*—Power optimization has become one of the most challenging design objectives of modern digital systems. Although FPGAs are more and more used, they are however still considered as power inefficient compared to standard-cell or full-custom technologies. New dedicated design approaches are thus needed to reduce this gap. In this paper, we address low-power design on FPGA through a dedicated High-Level Synthesis (HLS) flow. The proposed approach allows to slow down the clock frequency in parts of the design, decrease the complexity of the clock-network, reduce the number of long wires and perform clock-gating. The design flow has been fully implemented and allows to automatically synthesize hierarchical and synchronous multiple-clock domain architectures. The power consumption of the architectures we generate has been investigated and compared with state-of-the-art synthesis approaches. The experiments have been realized by using a Xilinx Virtex-5 device and the power measurement results show the interest of the proposed approach.**

Keywords- hardware design, high-level synthesis, multiple-clock domain, FPGA, hierarchy, low power

I. INTRODUCTION

Nowadays, FPGAs are more and more used in order to take advantage of their programmability and lower cost in comparison with the latest ASIC technologies. This brings into focus the FPGA power inefficiency. Although power consumption optimization has been extensively addressed in the past for ASIC design (see [1][2][3][4][5][6]) few high-level and low-power dedicated design approaches for FPGA design have been proposed. Unlike ASICs, FPGAs are composed of several configurable resources such as memory blocks, I/O blocks, programmable interconnect and millions of gates grouped into programmable logic blocks *(PLB)* (including Look-Up Tables LUT and flip-flops). Like all CMOS based-technologies, FPGA power consumption is directly related to the power supply, the clock frequency and the switching activity [7]. Moreover, as shown in [8] FPGA power consumption is dominated by interconnects (45% of the total power) followed by the clock distribution network (40%). PLBs consume the remaining amount of the total power (15%). Interconnects include all of the routing resources required to connect PLBs. One can distinguish two main categories of routing resources: short and long wires. The long wires widely contribute to the power consumption even if they often constitute a small ratio of the total interconnect area. The second major contributor to the total power consumption is the clock distribution. Each PLB has flip-flops to register data and needs to be driven by a clock signal. To eliminate the clock skew, the clock is distributed through a H-tree structure [8]. However, designs that target FPGA devices are traditionally flattened and contains only one clock domain. The clock has thus to be distributed over the entire design to supply the sparse distribution of flip-flops. Since the flip-flops are scattered all over the FPGA, the H-tree covering the design is always active even though not all of the flip-flops need their clock to be active. This results in a relatively large power consumption of the distribution network [8].
A first way to design low power architecture on FGPA is thus to reduce the number of long wires. Such a goal can be achieved by splitting the design into several blocks in order to narrow the connections. A second way to optimize power consumption is to design multiple clock-domains and block-based architectures: using several clock domains allows to decrease the clock tree complexity and block-based architecture allows to use clock-gating techniques when blocks are non active. Unfortunately, due to the overhead of communication and synchronization the use of Globally Asynchronous Locally Synchronous architectures limits to large block-based architecture such as MPSoC.

In this paper we address low power design of Digital Signal Processing (DSP) applications on FPGA through both scaling the clock frequency and reducing low-level circuit complexity like long interconnection wires and clock network distribution. To do so, we use hierarchical high-level synthesis in order to automatically generate synchronous multiple-clock block-based architectures. As allowed by recent FPGA devices [9][10], a block can have its own clock frequency synchronized with the other ones avoiding to have clock skew. Block-based design allows to reduce the number of long wires, to decrease the clock tree complexity and to use clock-gating techniques. Synchronous multiple-clock domain design allows to decrease the clock tree complexity and avoid the drawbacks of GALS architectures.
The paper is organized as follows; in section II we present the related works. Next in section III, we introduce the design flow we propose with the target architecture. Finally in section IV, we compare the results we obtain by using our design flow and a Xilinx Virtex 5 FPGA device. Comparisons are made with the other several architecture possibilities which differentiate on the control (centralized vs. distributed), the communication between blocks (synchronous vs. asynchronous) and the clock domain (mono vs. multiple-clock domain).

II. RELATED WORKS

In this section, we will present the works related to (1) Hierarchical High-Level Synthesis (H-HLS), (2) multiple-clock architectures and (3) high-level approaches for FPGA low-power design which are the three main research domains our approach lies on.
Several papers on H-HLS can be found in the literature. In [11] Rosenstiel et al. present a technique that enables the sharing of unused subcomponents across different hierarchical levels of the design. The “white box” concept is introduced as a way to provide architectural information of subcomponents to a higher hierarchical level. Knapp et al. [12] introduce the behavioral templates representation to automate the use of hierarchical synthesis. Behavioral templates allow to specify local scheduling constraints of operations within the sub-component on a cycle-by-cycle basis.

1946-1488/10 $26.00 © 2010 IEEE
DOI 10.1109/FPL.2010.94

Gajski et al. [13] formalize the characterization of interface protocols by specifying a finite state machine which provides all the data and the control signals needed for a given function.
However, in all these previous works hierarchical HLS is only used to face complexity issues through the reuse of pre-designed complex functional blocks but not to optimize the power consumption. Only few automated high-level approaches have been proposed for FPGA low-power design. Jha et al [14] present a H-HLS approach for synthesizing power-optimized as well as area-optimized circuits from hierarchical data flow graphs under throughput constraints. Cheng et al. present in [15] a low power high level synthesis system, named LOPASS, for FPGA designs. Two algorithms have been proposed for power consumption reduction; a simulated annealing engine that carries out resource selection, function unit binding, scheduling, register binding, and data path generation simultaneously and an enhanced weighted bipartite matching algorithm is used to reduce the total number of multiplexers inputs. Nevertheless, neither multiple-clock design nor clock gating are considered in the proposed approach.

Works on multiple-clock architectures have only been addressed through Globally Asynchronous Locally Synchronous (GALS) systems [16]. GALS architectures have been proposed mainly in order to enhance the performance in (MP)SoC architectures targeting ASIC technology. GALS-based systems consist of several locally synchronous components that communicate with each other asynchronously. Works on GALS can be divided into three categories; partitioning, communication devices and dedicated architectures. Partitioning has been addressed by several works that will not be presented in this paper due to space limitation. Moreover, there have been many efforts to design low latency asynchronous communication devices between synchronous blocks. For example, in [17], a wrapper of synchronization optimized in area is introduced for Latency Insensitive System LIS. The wrapper is used to guaranty a latency insensitive communication between two independent blocks having the same clock frequency. To support multiple-clock domain architecture [18] proposed the design of low latency mixed-clock FIFOs, while [19] the design of point to point wrappers. Bormann et al [20], Muttersbach et al [21] realize also different asynchronous wrapper designs which support 4-phase handshaking data transfer. A few related works has focused on dedicated architectures. In [22] for instance, authors proposed a platform to implement GALS designs mixing FPGA and ASIC. Each synchronous logic block contains 24 x 20 Virtex II CLBs and 24 multiplier blocks. The logic around the synchronous block was designed by using dedicated HW (non programmable ASIC technology).

To our best knowledge, none of the existing works has proposed a fully dedicated low-power design flow considering *interconnect*, *clock network* and *hierarchy* for FPGA design by (1) scaling the clock frequency, (2) performing clock gating, (3) reducing the amount of long wires in the final design through the use of hierarchy (4) reducing the clock network complexity thanks to multiple-clock domain usage and (5) reducing the area overhead by introducing a new synchronous multiple-clock architecture instead of using GALS systems. In this scope we propose an automated high-level synthesis approach for low-power design on FPGA.

III. PROPOSED APPROACH

As mentioned in the introduction section, FPGA power consumption is widely dominated by the interconnects and the clock distribution network. The clock frequency is also a key factor for power optimization. Hence, in order to design low-power architectures on FPGA we propose to use hierarchical high-level synthesis to automatically generate synchronous multiple-clock and block-based architectures.

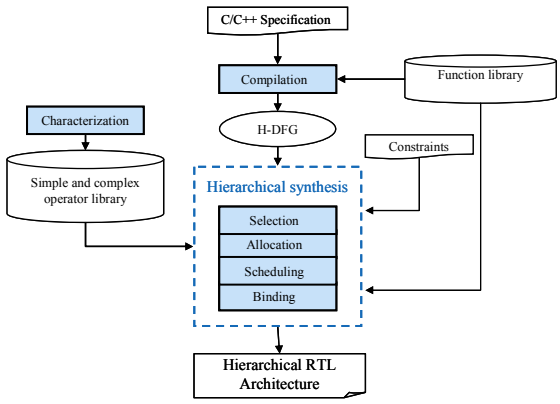


Figure 1. Hierarchical HLS Flow

Our flow is presented in Figure 1. The inputs of our HLS flow are a functional specification of the application to design, a throughput constraint, a clock period and a library of operators. The specification is a C/C++ description which can contain both elementary operations specified by using the language symbols (+,-...) and function calls. The operator library contains a set of *simple* and *complex* operators characterized in time and area. *Simple* operators implement elementary operations (registers, multiplexers and simple functional units like adders…). C*omplex operators* implement functions, are clocked and exchange control information and/or data. The throughput constraint is expressed by an initiation interval *II* (like in software pipelining) which represents the constant interval between the start of successive iterations.
We first translate the initial specification during the compilation step into a Hierarchical Data Flow Graph (H-DFG) according to the designer directives: (1) designer can in-line function calls by using standard *inline* keyword in the initial specification; (2) designer can also keep function not in-lined. When at least one function is not in-lined we perform hierarchical synthesis; the H-DFG we generate in that case contains hierarchical nodes symbolizing the function calls i.e. composite operations. The allocation step next determines the minimum number of operators that are needed to a priori meet the timing constraint requirements. This number is obtained by computing the average parallelism of each type of operation. The following equation presents the average number of operators *T(TO)* required to implement operations of type *TO* with *NbOps(TO)* represents the number of operations of type *TO*, *Tp(TO)* corresponds to the propagation time of the operator which implements operations of type *TO* and *II* is the throughput constraint.

$$T(TO) = \left\lceil NbOps(TO) \Big/ \left\lfloor \frac{II}{Tp(TO)} \right\rfloor \right\rceil$$

This first allocation is then considered as a lower bound. Thus, during the scheduling phase, supplementary resources can be allocated. During the scheduling and binding steps, operations are scheduled and bound to the allocated operators. The scheduling is a list-based heuristic in which ready operations (operations to be scheduled) are listed by priority order. The priority is fixed by the operation mobility (the difference between the as-late-as possible time and the current control-step). Binding of components (operator or register) is realized by using a Maximal Weighted Bipartite Matching Algorithm [23]. An available component with a candidate (operation or variable) has to respond to the minimization of interconnections (steering logic) between components and to the minimization of the components size. Given the set of allocated operators, our binding algorithm assigns all the scheduled

operations of the current step. Registers binding is done after the scheduling and the operator binding.

In order to optimize power consumption, we use the hierarchical structure of the data flow graph. Our approach first synthesizes each function, i.e. hierarchical node, as a complex operator and stores it in the library. The operator library contains thus at least for each non in-lined function one complex operator. The clock frequency of complex operators which have multi-cycles operations is slowed down. We thus synthesize each function by using a lower clock frequency constraint than the clock frequency initially defined by the designer. Indeed, multi-cycle operations need by definition more than one cycle to carry out its outputs. Obviously such operations do not need their outputs to be stored each cycle but each *n* cycles (with *n* the number of cycles needed by the associated operator). The multi-cycle operations can thus be down-clocked without sacrificing the timing performance of the architecture. Moreover, the clock of the neighboring operations, i.e. operations with data dependencies, can also be scaled down without increasing the area as long as they have timing slack. In our design flow, function calls are thus used to generate blocks with their own clock frequency leading to the automatic synthesis of multiple-clock and block-based architectures.

In our synthesis flow, a complex operator is composed of a datapath with its controller later referred *DP+C* in this paper. A DP+C can thus run independently from the top design (see Figure 2).Therefore, during the RTL generation step, we only generate a *start/enable* signal from the top controller to activate the complex operator during its active control steps. The same enable signal is also used to gate the clock of the module when it is inactive. Synchronous communication between a DP+C and its environment is done through registers assigned during the register binding step.
In order to generate synchronized multiple-clock architecture, we use dedicated resources provided by recent FPGA fabrics. This specific device is represented by the hash box called *DCM* (Digital Clock Management) presented in Figure 2. In this case, each DP+C runs at its own clock value so that the architecture is a *Distributed-Control Synchronous-Multiple-Clock (DC-MuC)* architecture. Architectures having only one clock domain are referred as *Distributed-Control Mono-Clock (DC-MoC)* architecture.
Hence, starting from a C/C++ specification using functions, we first synthesize each complex operator and store them in the operator library. We then automatically generate synchronous multiple-clock block-based architectures. As shown in the next section, the proposed design flow and its associated architecture allow to reduce the number of long wires, to decrease the clock trees complexity and to scale the clock frequency. Our approach allows thus to better optimize the power consumption of FPGA based component compared to other traditional architectures.

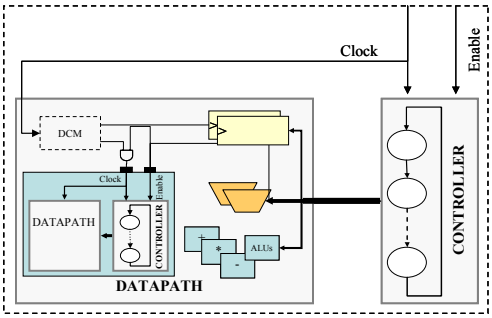


Figure 2. The proposed DC-MuC architecture

IV. EXPERIMENTS

Our experiments have been realized by using several examples from the digital signal processing domain. We have synthesized several architectures for each example in order to compare our approach to state-of-the-art: flat architecture and block-based design. A *Flat* architecture is composed of a single controller realized by a finite state machine (FSM) and a data-path containing simple operators, registers and multiplexers (see Figure 3.a). *Flat* architectures have been obtained by in-lining all the functions, leading to a non hierarchical DFG. Obviously, this means that the hierarchy of the specification was kept neither in the intermediate model nor in the final architecture. When functions are not in-lined they can be traditionally implemented by two types of operator ([11][12][13]): a simple *Datapath without controller (DP)* and *a datapath with its controller DP+C.* Contrary to the DP+C, DP can not run independently from the top design as they do not have their own controller. Thus, during the synthesis of a DP, the controller has been abstracted into a set of command words (one for each control step): the controller specification. This specification has then been stored in the operator library to provide all the information at higher level about how to control the datapath to perform the required operation. As for the DC-Muc and DC-Moc architecture, the registers have been generated to ensure a synchronous communication and the *enable* signal for the clock gating. A block-based architecture integrating only DPs will be named *Centralized-Control mono-clock CC-MoC* architecture when using the same clock signal and *Centralized-Control synchronous multi-clock CC-MuC* architecture when using multiple-clock signals (see Figure 3.b). In order to compare with the well-known LIS/GALS architectures [24]/[16], DP+C have been integrated as in LIS or GALS architectures. For this they have to be able to run autonomously and communicate through FIFOs. To make a DP+C autonomous, we have used the synchronization wrapper proposed in [17] for both the top design and the module as detailed in Figure 3.c. Each Input/Output of the allocated DP+C components is generated as an Input/Output of the top design. The scheduling and binding steps of the top specification are done regardless the state of the DP+C. Furthermore, when using an asynchronous communication scheme, the clocks can be desynchronized. Thus, bi-clock FIFOs are needed. We automatically generate bi-clock FIFOs similar to the model presented in [18] to support two distinct clocks.

Only two clock domains have been used in our experiments: high-frequency (HF) with a clock period value of 10ns and low-frequency (LF) with a clock period value of 20ns. All the mono-clock architectures have been synthesized by using the HF clock period. For the sake of comparison completeness, *flat architectures* have been synthesized twice: first by using a HF clock value as constraint and next by using the LF clock value. The *LF flat architecture* does not respect the throughput frequency but is used to show the eventual gain in power consumption one could obtain by sacrificing the required performance i.e. the throughput. Functions that did not contain multi-cycle operations have been in-lined. When designing multiple clock block-based architecture, all the functions have been synthesized using the LF clock value since they have been on multi-cycle operations.

In order to investigate only the impact of the architecture families on the power consumption, we synthesized all the architectures with the same throughput constraints. In order to generate the clock signals of each block, we used the Digital Clock Managers (DCMs) available on our Xilinx xc5vlx110 target device. A DCM consumes 44mw at 250Mhz+0.2mW/Mhz. The power consumption has been measured by using a DC power analyzer which provides programmable voltage/current and measurement features

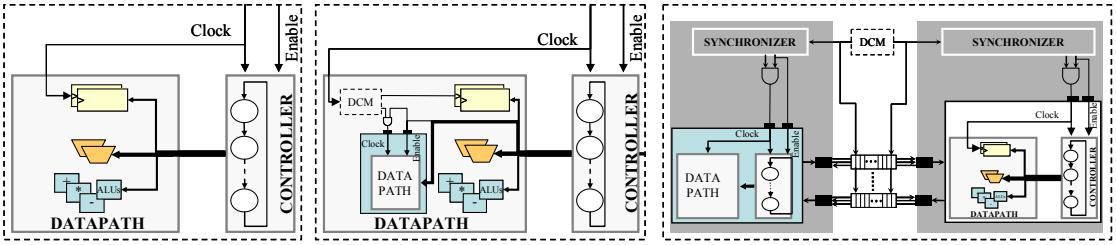


Figure 3. Bloc-based Architectures

(a) Flat architecture where operators, registers, multiplexers and controller are shared. (b) Centralized-Control architecture where only the controller is shared (c) LIS/GALS architecture: where no resource is shared and where the communication is asynchronous (by using FIFOs)

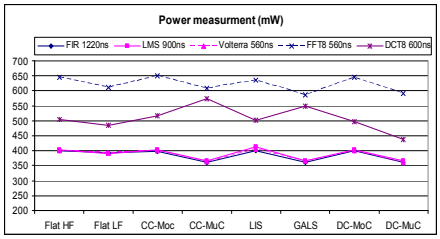


Figure 4. Power measurements for the small-complexity designs

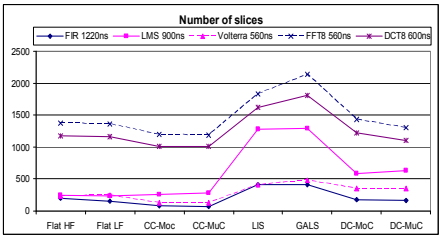
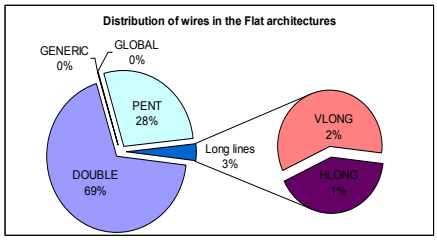


Figure 5. Area results for the small-complexity designs



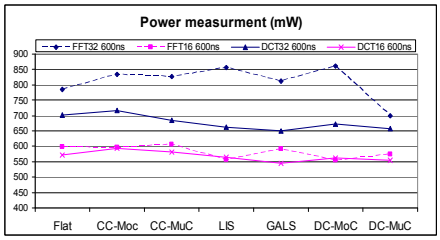


Figure 6. Power measurements for the higher-complexity designs

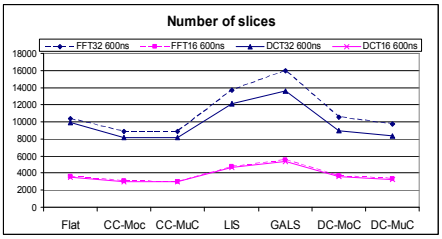


Figure 7. Area results for the higher-complexity designs

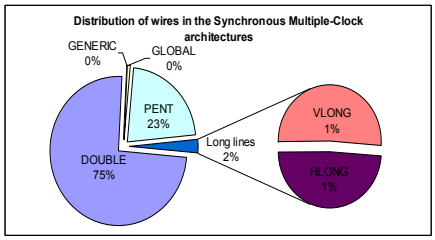


Figure 8. The distribution of wires in the flat architecture vs. the proposed architectures

(voltmeter, ammeter).The voltmeter accuracy is 0.016% + 1.5 mV and the ammeter accuracy is 0.03% + 15 µA.

The DSP applications we used for our experiments are the following: 16-tap FIR (Finite Impulse Response) filter, 16-points LMS (Least Mean Squares) Filter, 5th order Volterra, 8-points FFT (Fast Fourier Transform) and 8*8 DCT (Discrete Cosine Transform). Both, the FIR and the LMS contain multiply-accumulate (*MAC*) computations. We specified the *MAC* computation as a function and then synthesized the *MAC* component under LF clock value and 2 cycles initiation interval II (i.e. 40ns). The Volterra example consists of several multiplications. Especially, we chosen the three elements multiplication (that we called *mul3*) to be a function. Similarly, the *mul3* component was synthesized used the LF clock value and its latency was equal to 40ns. Under these conditions, the generated FIR, LMS and Volterra block-based architectures contain respectively one, five and one block(s). In the FFT, the butterfly computations have been specified as a function calls in the C functional description. The butterfly component has been synthesized with a LF clock period and a latency of 40ns. The FFT was synthesized with the HF clock period with an II equal to 560 ns. Finally, the DCT with II equal to 600 ns.

Figure 4 shows for each example the obtained results for power consumption. In the first set of experiments the average power consumption is around 400mW. The DCM power overhead in the multi-clock architectures is 40mW (10% of the total power). For the sake of fairness we remove the DCM consumption overhead in the presented graphs. Thanks to the synchronous block-based architecture, the power consumption is reduced by 13% in average. However, the area compared to the flat design is increased in most of the cases especially in the LMS (see Figure 5). The use of LIS/GALS architectures is disproportional for such example sizes since the area is more than three times the area of the flat design. Regarding area and power consumption, *Centralized-Control* architectures may present a good alternative for small designs where a centralized controller has less area overhead than many small controllers.

For low complexity applications, the gain our approach is able to achieve is hidden by the DCM overhead. We thus synthesized the FFT and the DCT with several computation complexities. In fact, we varied the DCT size up to 16*16 and 32*32 and the FFT size up to 16 and 32 points. The II constraint has been set for all the designs to 600ns. Figure 6 and Figure 7 list the obtained power and area results for each of the generated architectures. Considering the 32-points FFT, the results we have obtained show that the control distribution obtained in the *Distributed-Control* architecture enhances the power efficiency and reduces the area (5% of area reduction in comparison with the flat architecture) which is not the case in the *Centralized-Control* architecture where the control is centralized. The use of multiple-clock domains reduces the power consumption up to 11% for the *Distributed-Control* architecture.

The GALS architecture reduces the power consumption (up to 7%) but has 50% (in comparison to 9% area overhead for the *Distributed-Control*) of area overhead due to the use of FIFOs and the synchronization wrapper. One can notice that the *Centralized-Control* architecture is no more a good candidate for power consumption (only 2% power reduction in average and 16% area overhead). As mentioned in the introduction section interconnects and particularly long wires are a major component of power consumption [8]. Two main categories of routing resources exist in Xilinx FPGA: local wires which are short connections and long wires i.e. long connections. The amount of wires can be extracted from the ISE tool. All VLONG, HLONG and PENT wires can be considered as long connections and Generic, Double and Global are short connections [25]. The distribution of each category of wires is given in Figure 8. The graphs represent an average computed for all the application examples using first the flat architecture and then the DC-MuC architecture. In the DC-MuC architectures the number of PENT and LONG wires is respectively reduced by 18% and 30%.while the double wires are increased by only 8% Deeper analysis of interconnect results (Figure 9) shows that the number of long wires is reduced by using the *Distributed-Control* architecture (at most 29% for the *DC-MuC* against 22% for the *DC-MoC)*. Note that the same tendency has been observed for all the already presented examples.

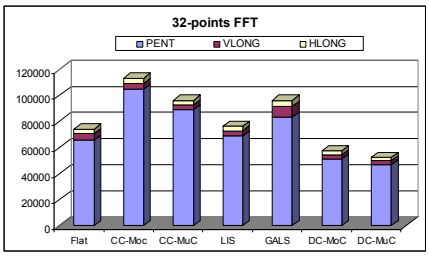


Figure 9. Number of wires in the 32-points FFT (600ns)

The experimental results presented in this paper show that using the proposed approach that relies on *Distributed-Control Synchronous-Multiple-Clock architecture* allows noticeable reduction of the power consumption. One can notice multi-clock architectures are always better in power consumption than mono-clock architectures. However, compared with GALS architectures, DC-MuC are able to further reduce the power consumption while having up to twice as small areas. Moreover, the areas of the architectures we generate are equivalent to those of the flat architectures for low complexity examples and inferior or equal for medium complexity examples. Our approach thus offers the best trade-off between area overhead and power reduction.

V. Conclusion and Future Work

We have presented in this paper an approach based on hierarchical and multiple-clock domain HLS to target low-power design on FPGA. We introduced an automated design flow and demonstrated the interest of the proposed approach through the experimental results. Our approach aims at using the FPGA structure knowledge at the high-level to drive the optimization techniques at lower levels. Experimental results showed that the proposed *approach*, using a distributed controller, synchronous communication and multiple-clocks, allows to reduce considerably the power consumption while keeping, for realistic examples, reasonable area overhead.

The block-based architectures have been generated by using function calls in the input specification. In our future work, an automatic partitioning methodology will be proposed to optimally identify the clock domains and generate the hierarchical design without the use of function calls. Furthermore, we are looking forward to investigate the coupling of our approach and a dual-Vdd/dual-V_{th} technique to further reduce the power consumption.

References

[1] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey and R.W. Brodersen, "Optimizing power using transformations," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12-31, Jan. 1995.

[2] P. Kollig and B.M. Al-Hashimi, "A new approach to simultaneous scheduling, allocation and binding in high level synthesis," *IEEE electronics Letters*, vol. 33, Aug 1997.

[3] M. Ercegovac, D. Kirovski and M. Potkonjak, "Low-power behavioral synthesis optimization using multiple precision arithmetic," *Proc. 37th Design Automation Conference,* 1999.

[4] C. Piguet "Low Power Electronics Design", *CRC Press,* ISBN 0849319412, 2005

[5] Keating, M., Flynn, D., Aitken, R., Gibbons, A., Shi, K.," Low Power Methodology Manual", 1st ed. 2007. Corr. 2nd printing, 2007, XVI, 304 p., Hardcover, ISBN: 978-0-387-71818-7

[6] Rabaey, "Low Power Design Essentials Series: Integrated Circuits and Systems", Jan 2009, XII, 288 p. 222 illus. in color. With CD-ROM., HardcoverISBN: 978-0-387-71712-8

[7] C. Piguet "Low Power Electronics Design", CRC Press, 2005

[8] Degalahal, V. and Tuan, T. 2005. "Methodology for high level estimation of FPGA power consumption". In *Proc. of the 2005 Asia and South Pacific Design Automation Conference*, ASP-DAC '05.

[9] http://www.xilinx.com/support/documentation/ip_documentation/dcm_module.pdf

[10] http://www.altera.com/support/devices/pll_clock/pll-overview.html

[11] Bringmann and W. Rosenstiel, "Resource sharing in hierarchical synthesis," *Int. Conference on Computer-Aided Design,* 1997.

[12] Ly, T., Knapp, D., Miller, R., and MacMillen, D."Scheduling using behavioral templates", In *Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference*, June 12 - 16, 1995.

[13] Fan, N., Chaiyakul, V., and Gajski, D. D., "Usage-Based characterization of complex functional blocks for reuse in behavioral synthesis". In *Proceedings of the 2000 Asia and South Pacific Design Automation Conference* (Yokohama, Japan). ASP-DAC '00.

[14] G. Lakshminarayana and N.K. Jha. High-level synthesis of power-optimized and area-optimized circuits from hierarchical data-flow intensive behaviors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(3):265–281, March 1999.

[15] Chen, D., Cong, J., and Fan, Y, "Low-power high-level synthesis for FPGA architectures", I*nternational Symposium on Low Power Electronics and Design*, 2003

[16] *Advanced Research in Asynchronous Circuits and Systems*, 2000.

[17] Bomel, P., Martin, E., and Boutillon, E. "Synchronization Processor Synthesis for Latency Insensitive Systems". In *Proceedings of the Conference on Design, Automation and Test in Europe* 2005.

[18] Chelcea, T. and Nowick, S. M. "A Low-Latency FIFO for Mixed-Clock Systems". In *Proceedings of the IEEE Computer Society Annual Workshop on VLSI*, April 27 - 28, 2000.

[19] Moore et al. "Point to Point GALS Interconnect", *IEEE ASYNC,* 2002

[20] David Bormann, Peter Y.K. Cheung. Asynchronous wrapper for heterogeneous systems. *In Proceedings of. ICCD*, 1997.

[21] J.Muttersbach, T.Villiger, W.Fichtner. Practical design of globally asynchronous locally synchronous systems. *In Proc. Int. Symp.*

[22] Jia et al. "A Novel Asynchronous FPGA Design And its Performance Evaluation", FPLA 2005, IEEE

[23] Z. Galil, "Efficient algorithms for finding maximum matching in graphs", *ACM Computing Survey*,1986

[24] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli, ''Theory of Latency-Insensitive Design,'' *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems,* 20(9):18, Sept. 2001

[25] www.xilinx.com/