2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Mixed Electronic System Level Power/Performance Estimation

using SystemC/TLM2.0 Modeling and PwClkARCH Library

Antonio GENOV, UCA & NXP SC, Sophia-Antipolis/Mougins, France (*antonio.genov@nxp.com*)

Loic LECONTE, NXP SC, Mougins, France (*loic.leconte@nxp.com*)

François VERDIER, UCA, CNRS, LEAT, Sophia-Antipolis, France (*francois.verdier@univ-cotedazur.fr*)

***Abstract*— The Hardware/Software (HW/SW) architectural exploration has become a key component of System on Chip (SoC) design modeling. The insufficient power and timing analysis capabilities at early stages of the design flow limit the optimized modeling. Thus, pushed by the need to improve that shortage and inspired by the numerous studies on Electronic System Level (ESL) modeling, we introduce a novel ESL methodology that combines power and performance estimation in one unified framework.**

In this paper, we present our new approach applied and tested on an NXP proprietary switch matrix/interconnect system used in i.MX8 series of SoCs. Our model is based on SystemC-TLM2.0 and make use of PwClkARCH library for power management. This framework allows us to develop a SoC transaction level model (TLM) written exclusively in C++/SystemC-TLM2.0 and to extract power consumption and performance metrics after the simulation. This modeling approach allows a strong separation between the functional SystemC/TLM model and its power intent description. Only a few pieces of code must be added to performance model to hook power model to it. Moreover, it makes the code easier to debug and maintain.

Keywords*— *Dynamic approach, Power estimation, Power efficiency, Electronic System Level, System design, Performances estimation

I. INTRODUCTION

The complexity and capabilities of modern high-performance devices are growing exponentially with continuously reducing the transistors size and successfully increasing the integration density. Thanks to these advances, the number of Intellectual Properties (IPs) on a single chip and the capabilities of systems are constantly growing. As a result of increasing complexity, the power consumption of System-On-Chip (SoC) are also increasing. Consequently, the trade-off between performance and power efficiency has become a real challenge. Therefore, SoC manufacturers and engineers are devoting considerable effort to researching new development strategies based on higher level of abstraction. Many power reduction techniques [1] have been developed to reduce the current leakage. The application of these techniques starts at the design specification stage and is refined at each step of the development flow [2], but it is unfortunately difficult to quantify their impact at the architectural definition stage. We will see in the following that at correctly chosen stages, we can have a real impact on the energy consumption.

In the next section [Section II], we will present a brief description of the ESL level modeling and current issues. Next, we will present some existing approaches and related work, and then in section III we will introduce our modeling choices. In section IV, we will present a solution, developed to address the discrepancies of existing techniques by allowing the application of different power reduction techniques and a dynamic approach for power estimation. Therefore, in sections V and VI we will give an example and its simulation results.

II. ELECTRONIC SYSTEM LEVEL (ESL) MODELING

A. Architecture and Design definition

The flow starts with the customer/marketing requirements, which lead to the definition of the specifications. At this level, architecture teams work on these specifications and explore component selection options. Almost in parallel is the Electronic System Level (ESL) initiation. The ESL refers to a high abstraction level design modeling and architectural exploration. This means that architecture team defines a Virtual Prototype (VP) of a system architecture starting only with the unavoidable behavioral parts of the Design Under Test (DUT) and makes some timing approximations instead of cycle accurate models. A widely used and well-established approach to early performance estimation is to create functional models using languages such as C, C++ and SystemC. With each

refinement of the functional model, we can perform the necessary performance analysis by some simple monitoring or by using existing EDA.
On the other hand, the power consumption is almost neglected at this high level of abstraction. This is due to the limited information at the beginning of the design flow and the difficulty of specify and testing power management strategies. A commonly used approach at this level is a static approach consisting of taking characteristics of the technology for leakage estimation, different IP power numbers (given by manufacturers) for the worst/best case dynamic consumption and components area information. Then, the static and dynamic power consumption is calculated using Excel tables or automated tools and by applying power equations. This approach works well for static or pseudo-static use cases but cannot be applied to more complex dynamic ones. Therefore, the need for a method to dynamically estimate complex SoC use cases energy consumption has led to multiple studies on power estimation at ESL level. IEEE Unified Power Format (UPF) standard was developed [3]. The main UPF contribution is the ability to include the power intent definition, through which we can divide the system into multiple power domains, define voltage levels, add isolation cells and level shifters, and apply techniques such as power gating and Dynamic Voltage Scaling (DVS). Prior to UPF 3.0 [4], released in 2015, the UPF required that an RTL model be developed in order to be applied for power estimation. UPF 3.0 allows to move to a higher abstraction level and nowadays, the power model can be started at the ESL level, only with power/logical states, and then refined to voltage and supply nets. However, at our level of abstraction, UPF 3.0 is not sufficient for an easy detailed power and performance design exploration, where power management needs to be defined and the trade-off between power and performance are to be investigated.

B. Related works

Previously, many power estimation methodologies have been studied or developed, mostly based on static approaches. In the following approach [5], the authors use IP activity during TLM simulation and extract it in a VCD file. Then, they synthesize the TLM model and compile the design using design constraints. The power consumption is estimated based on cell reports extracted from the design.
In another study [6], the approach also relies on IP activity. Different activity types are defined, and counters are added for each one of them. Each counter increments when the corresponding activity is executed during simulation. The corresponding power consumption is applied to each type, allowing the entire module activity to be calculated. Each type of IP has its own modelling approach i.e. memory, processor, crossbar etc.
In [7], the power information is added to the SystemC functional model. In this work, it is assumed that power related transactions are developed before the functional transactions. Each corresponding IP transactions are identified, and their consumption are characterized. Macro-models of IPs with multiple parameters are created and the TLM functional models are enriched in order to dynamically call these macro-models during simulation. All these methods need a significant amount of time and coding effort in order to extract power estimations. In addition to that, there is no information about the application of any power reduction technique.
In [8] there is an approximately timed SystemC/TLM approach applied to NoC architecture with Dynamic Voltage and Frequency Scaling (DVFS). The authors mention that they do not use pure functional models and describe IP specific power model for each IP. In their approach, the DVFS modelling is independent to the actual power modelling.
In [9], [10], the authors present a well-structured and interesting approach, which is inspired by many studies. They use SystemC functional models augmented with their PMS library to create UPF-like power intent. Once the SystemC/PMS model is sufficiently accurate and detailed, they proceed to High-Level Synthesis (HLS) to generate RTL code and UPF file. The main Power intent verification is done with the generated RTL and UPF. Like their solution is provided for HLS, they use SystemC interfaces for the communication. Thus, it remains a highly time-consuming approach for modeling complex and protocol-based systems.
Besides, there are many industrialized methods and tools that are being used for power estimation. We can cite tools like Synopsys Platform Architect [11] which uses UPF-based high-level state machine description. A power consumption is associated to each state and at each state triggered by given events, the virtual prototype detects these events and updates the total power consumption. This is a good approach for power estimation. However, the lack of easy way to describe clock/power domains and to apply power reduction techniques make it unsuitable for power management investigation.
Another industrial solution is Intel Docea Power Simulator (IDPS) [12] used to create power and thermal models of SoC at system level and to apply post-processing power analysis. Intel`s simulator is also primarily based on dynamic approach for power estimation. For each component, it needs a description of power state transition events, translated into Power State Machine (PSM). For each power state, the user needs to provide the corresponding power profile. The extraction of specific data from SystemC-TLM functional model to VCD files enables us to co-simulate the power model with these VCD files. Thus, it is an effective method for power and

temperature estimation. However, the IDPS does not provide a tool for power optimization strategies description. The only way to do it is to integrate the entire description into the functional model. There is not a direct power/functional models' co-simulation, thus it may be more time consuming and the functional/power models relation may become less stable.
Mentor Graphics presents another solution at TLM level called Mentor Graphics Vista [13], but similar to the previously cited industrial solutions, it is not suitable for power management strategies investigation.
The performance estimation approach we use is based on transaction level monitoring. It allows us to start with a higher functional abstraction and refine our behavioral model all the way down. The proposed solution for the power modelling is called PwClkARCH. It is a library of C++/SystemC-TLM classes, inspired by the UPF standard, allowing a SystemC-TLM architecture to be structured in power and clock domains and power management strategies to be applied accordingly. In this way, the impact on power consumption and on performance can be observed and dynamically evaluated for any IP/SoC executing any application. This library does not rely on state machines, but on relatively precise surface information and IP activity reports. The power estimation is extracted from the sum of these information and is more accurate than the power states-based ones.

In the next section we will introduce our functional model choices and a quick overview of the performance estimation key points that are needed to be observed.

III. MODELING CHOICES

A. Behavioral model choices

There are different techniques, tools and languages used to create functional/behavioral models, but the most commonly used is the SystemC library, which is based on C/C++ languages. For the communication part, SystemC provides interfaces, channels and ports. On the other hand, the TLM2.0 transaction level modeling methodology presents a standardized abstraction approach, which combines generic transactions information into payloads and transfer these payloads via interfaces called sockets. In this way, the multitude of pin connections is replaced by a single socket that transports the entire block of information via function callbacks. Thanks to this higher-level description, we can successfully generate fast simulating functional un-timed or timed models with reduced coding effort. TLM2.0 allows the usage of two main coding styles.

- **Loosely Timed style** - uses transactions with only two timing points i.e. at the beginning and at the end of the transaction. It is more suited to software fast simulating verification.
- **Approximately Timed style** - employs breaking down transactions into multiple timing points (four phases by default) and therefore allowing to model particular hardware communication protocols and approximate time accuracy. It is useful for architectural exploration.

We use SystemC to describe the behavior of hardware systems and TLM2.0 to simplify the communication between different IP SystemC models by using function callbacks instead of pin wiggling. We are interested in the definition of specific protocols, and more precisely in some AXI protocol capabilities. As a first step, instead of recreating the entire AXI protocol, we prefer to model only the points we need in order to create a simpler sufficiently accurate model.

B. Performance estimation

The accuracy of the performance estimation depends on the functional model abstraction. The more the functional model respects the actual behavior of the hardware and its timing, the more our accurate our performance estimation will be. As performance estimation is already a well-established and developed solution, we will focus our attention on power estimation part. This section is therefore devoted to some key points in estimating the performance of a memory-based system. In order to perform this estimation, we need to be able to configure several points for each IP. Some of the most significant configurable parameters are READ/WRITE access delays, MAX/MIN arbitration delays, number of ports and systems, ports width and data length. The most important performance criteria for performance evaluation are latency, deadline respect for real-time systems, the influence of configurations on performance and of course the bandwidth. Some of the metrics used for accuracy quantification are:

- **Transaction duration** - Time between the start and the end of each transaction.

$$T_{duration} = t_{EndTime} - t_{StartTime} \quad \text{Eq. 1}$$

- **Throughput** - Number of transactions during the total simulation time or during a given time frame.

$$TP = \frac{\sum_{n=0}^{TransNumber} Data}{T_{TotalTime}} \quad \text{Eq. 2}$$

- **Transaction reordering** - Important for modules receiving/transmitting multiple simultaneous transactions. The accurate modeling of transactions order can have a significant impact on performance estimation accuracy. The main reordering types are *Port Transaction ordering* - ordering on each port and *Global Transaction* ordering - global view on system.

If we have the RTL model, we can also find the Error rate for each metric.

C. *Power estimation and Power management*

The classic equation for power *P(t)* is the product of the voltage *V(t)* and the current *I(t)* turning in the circuit.

$$P(t) = V(t) * I(t) \quad \text{Eq. 3}$$

In many cases *V(t)* remains almost constant, so the only variable that needs to be calculated is the current *I(t)*. The energy is calculated as the integral over the time of the instantaneous power:

$$E = \int P(t)dt \quad \text{Eq. 4}$$

The power dissipated by an SoC can be divided in two general parts:

- **Static Power** - due to transistors leakage current

$$P_{static} = V_{dd} * I_{leakage} = \frac{V_{dd}^2}{R_{leakage}} \quad \text{Eq. 5}$$

- **Dynamic Power** - due to the switching activity of transistors. The short-circuit power is the power dissipated by an instantaneous short-circuit connection at the time the gate switches state. In our case we find short-circuit power to be negligibly small, hence ignored. Theoretically, if all transistors switch at the same time and at each cycle of a clock, then the dynamic power is the product of the load capacity of the transistors (C_{load}), the supply voltage of the transistors (V_{dd}) squared and the frequency (f_{clk}). In reality, we will never have all the transistors switching at the same time, so for each IP we can add an activity factor (α) with value between 0 and 1, which models the average activity based on switching transistors number.

$$\boldsymbol{P_{dynamic} = P_{short-circuit} + P_{switching} = \alpha * C_{load} * V_{dd}^2 * f_{clk}} \quad \text{Eq. 6}$$

The total power consumption is simply the sum of these two parts.

$$P_{total} = P_{static} + P_{dynamic} \quad \text{Eq. 7}$$

Equations *[Eq. 5], [Eq. 6], [Eq. 7]* results in the total power equation:

$$P_{total} = \left(V_{dd} * I_{leakage}\right) + (\alpha * C_{load} * V_{dd}^2 * f_{clk}) \quad \text{Eq. 8}$$

IV. PwClkARCH

PwClkARCH is a C++ library developed for ESL power estimation. Many previous studies have been conducted and collected in order to optimize the library [14], [15], [16], [17]. Most of the semantics included in this library are UPF based and abstracted at the TLM level. As in UPF, there is a list of components called Design Elements (DEs). DEs are defined as references or pointers to their corresponding SystemC-TLM design model. Then, a power and clock specifications are built over these DEs to define the power intent. In this way, the separation between the hardware design and the power/clock intent is reinforced (Figure 1).

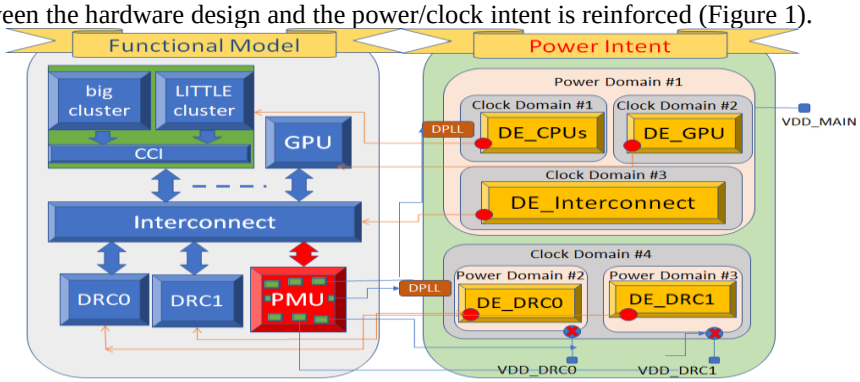

Figure 1. PwClkARCH semantics

Each DE aims to apply the generic static [Eq. 5] and dynamic [Eq. 6] power equations, taking into account the component specific capacitance (C_{load}) and the leakage resistance ($R_{leakage}$). Using the procedure for defining

DEs in the library, we can divide DEs (and thus the IPs) into different power and clock domains. Moreover, it is possible to add DPLLs, switches, power state tables, clock state tables and OPP tables. We can test different strategies by changing power/clock domain structures, applying power/clock gating, auto clock gating, testing different clock frequencies and observing the behavior of the functional model.

In Figure 1 we can see a block called Power Management Unit (PMU). This block is a SystemC - *sc_module* and it is the only mandatory power-oriented block that must be connected to the functional model. The PMU acts as an interface between the functional and the power model. It serves as a relay for the implementation of power management strategies based on the exploitation of clock/power domains states. Switching from one power mode to another corresponds to a power management request (TLM transaction) sent by an initiator from the SystemC-TLM model to the PMU. This initiator can be, for example, the CPU or another module. In summary, the initiators send orders to PMU when power state change is to occur, and the PMU applies all necessary changes and configurations to ensure compliance with the defined power strategy. It is important to mention that when we use auto-clock gating, it is the DEs that give the orders to the PMU and not the initiators. The approach has been tested on Intel`s proprietary pre-silicon simulation environment to define clock gating strategy on L2 Copro - hardware accelerators-based block [14]. Another test was performed on DRAM memory using a simple functional model and the DRAMPower tool [16]. This study shows that PwClkARCH can be connected to DRAMPower, and thus extract an accurate power consumption estimation of a given memory technology used in an ESL functional model. Furthermore, it also includes a DVFS application using PwClkARCH`s OPP tables.
Our study introduces a test on a complex interconnection module. Moreover, this is the first PwClkARCH application on approximately timed (AT) model. Power gating, clock gating and DVFS are used in use cases.

V. USED TESTBENCH AND DESIGN UNDER TEST

Our targeted Design Under Test (DUT) is a complex system serving as hierarchical memory mapped bus. Its main tasks are to manage the communication between all sub-systems integrated in the SoC, to apply transaction reordering and scheduling, and to manage the Quality of Service (QoS) of the entire SoC. In order to model it, we use AT coding style and some protocol-specific extensions for the transaction scheduling. For our first methodology test, we defined a testbench containing simple traffic initiators to mimic memory access traffic and simple targets to mimic memory blocks, allowing us to test the capabilities of our interconnect module.

A. *Testbench - Traffic Initiators and Targets*

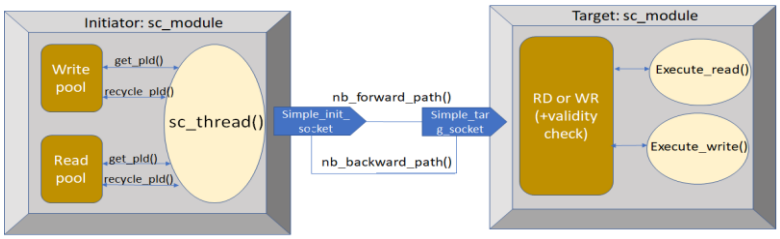

Figure 2. Simplified Initiator and Target blocks

The **Initiator** modules (Figure 2) are SystemC based traffic generators designed to recreate simplified accesses to external memory. For transaction memory management, we considered two pools - one for read transactions and one for write transactions. Their buffer size can be defined externally. We can also define different traffic configurations using parameters such as maximum outstanding READ/WRITE transactions and maximum number of simultaneous transactions. Each transaction generated by the initiators is complemented by ignorable TLM extensions to mimic signals such as transaction ID and QOS signal, representing AxID and QOS from AXI protocol. The **Target** modules (Figure 2) are basic models of DRAM controllers (DRC) + DRAM memories that can execute different memory accesses, such as READ and WRITE. Inside we consider fixed **request acceptance delay, read execution delay** and **write execution delay**. We can also manipulate parameters such as **memory width/length**.

B. *Design Under Test - Interconnection module*

As mentioned earlier, our DUT (Figure 3) is a hierarchical interconnection system with multiple subsystems and levels. The main features of this module are the programmable course interleaving, system address decoding and transaction routing to the memory mapped SoC components via a cross-linked NIC fabrics. In addition, Quality of Service (QOS) algorithms are applied in this module to manage scheduling and reordering of transactions according to their priority. QoS traffic arbitration is performed at several levels. Each subsystem of this interconnect can be used separately, allowing us to increase the interoperability of our model. Thus, thanks to

this very advantage of our approach, we can reconfigure multiple interconnection structures. Due to confidentiality restrictions, we are not able to provide more information about the architecture and functionality of the component.

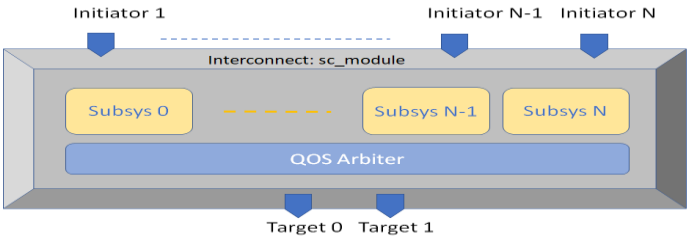

Figure 3. Simplified Interconnect block

C. Functional analysis of our architecture

In one of our first testbench, we consider multiple transaction initiators and two DRAM memories as shown in Figure 4. The initiators are connected to the two DRAM memory targets via the interconnect block.

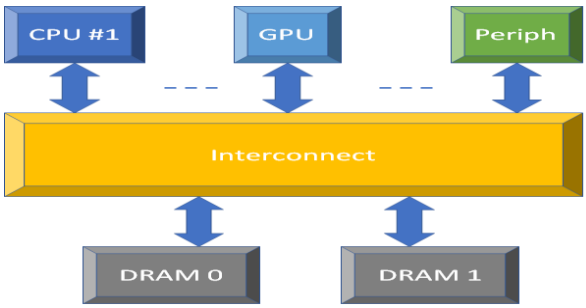

Figure 4. Testbench example

The tested use case consists of 4 functional phases (Figure 5). In the first phase, all data traffic generators send 512 READ/WRITE transactions to recreate full activity of the Interconnect block. In the second phase, there is a short inactive period that is used to test the clock gating. In the third phase, only half of our traffic generators remain active for another 512 READ/WRITE transactions, allowing us to test the auto clock gating. In the fourth and final phase, when there is no activity, we test the power gating. The reason we chose this use case is that we have a real silicon power measurement extracted from existing chip with similar to this type of use case.

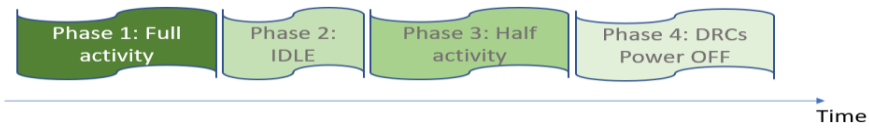

Figure 5. Functional Use Case

D. Power Management strategy

In order to maintain the separation between functional and power models, we have added another SystemC block called ***"Master".*** That block observes the system and activates the power management strategies. The Master module is responsible for transmitting transactions to the PMU requesting power state changes. In the simulation, the Master starts by setting an OPP to the structure (supply voltage, clock domains frequencies). Once the PMU has received this OPP transition, the OPP state is found in the table and the voltage and frequency values are applied. There is an assertion mechanism in each DE model, which indicates when a discrepancy between functional and power descriptions occurs. The Master then waits for the completion of all transactions. The Master module observes all the initiators but only communicates with the PMU in case of an OPP state change. In this way, both modules have no direct TLM communication with any other functional model (Figure 6). An DE has also been associated with the Master in the power intent because this unit receives an event when the power manager state transition is completed (from PwClkARCH) and the Master is ready to request another OPP change. To avoid showing the power consumed by the Master, which has no direct correspondence in the functional model, we have set its capacitance (C_{load}) to 0.0 uF and its leakage resistance ($R_{leakage}$) to 10^6 Ω. In this way, there is no significant power consumption coming from it. Figure 6 shows the power and clock intent in our case study architecture.

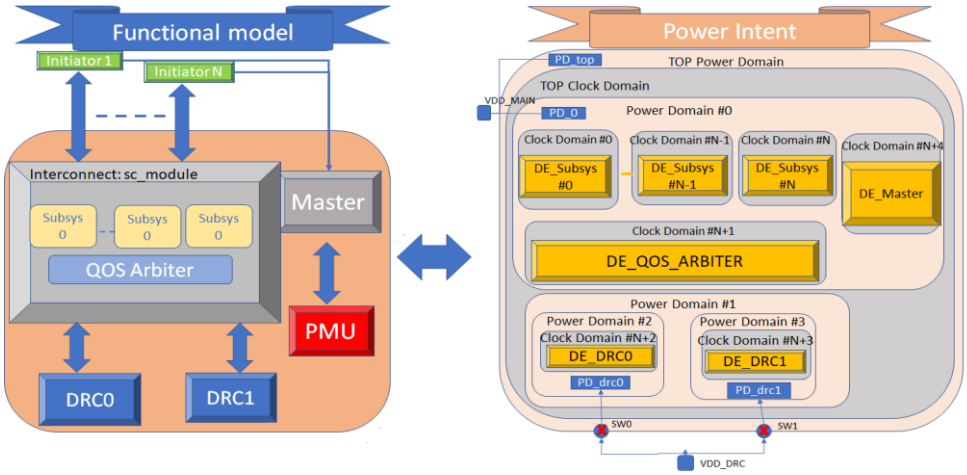

Figure 6. Power Intent

Each subsystem within the DUT interconnection system is in a separate clock domain. There is a power domain for the entire interconnect. We have also added a power and clock domains for both DRAM memories. In order to execute the power gating, we added two power switches. A DPLL is instantiated in the PMU for each clock domain. We also define the possible Power and Clock State Tables (PST and CST) and the OPP Table. These tables are used by the PMU to apply the power reduction techniques. In this paper, we present a power management strategy that includes power and clock gating tests and a simple DVFS. We define 3 OPPs as shown in Figure 7. The first OPP (Boot OPP) acts like a boot. We change the frequency from reference clock to a so-called bypass mode, a fast one-cycle operation before traffic generation to reset the process/memories to a default value. The second OPP (OPP1) indicates a start of the activity and thus the auto clock gating is activated. This is the main OPP and it is kept running for the duration of the traffic activity. Once the traffic activity is over, the third OPP (OPP2) is activated. The third OPP applies clock and power gating to both memories.

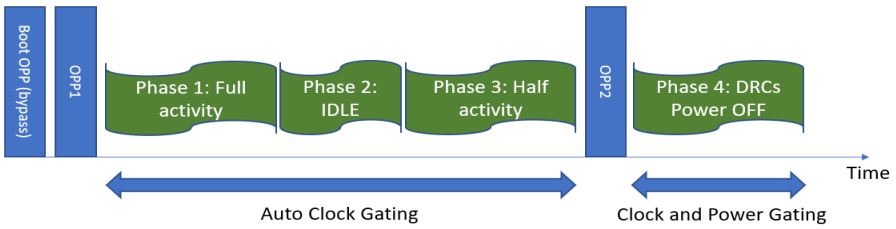

Figure 7. Simple Power management strategy

VI. RESULTS

In this section, we present part of our experiment results. In Figure 8 we can observe the different use case phases and the effect of the applied power management strategy. In the figure we illustrate the total power consumed by the design (green curve) and the static power consumption (yellow curve).

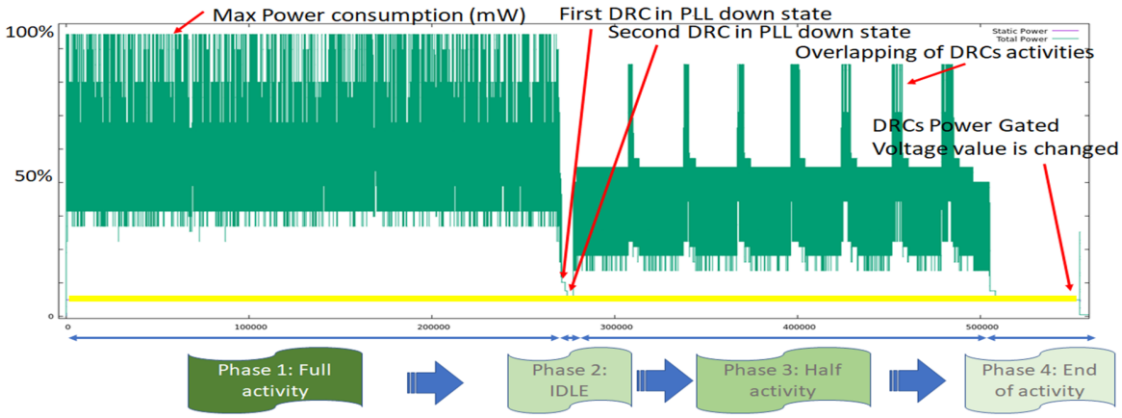

Figure 8. Overall power consumption profile in test case [mW]

In the first phase all traffic generators are active, so all the subsystems are also active. Half of these initiators start with transactions for DRC0 and the other half for DRC1. This way both memories are active. During the second phase, the dynamic power is switched off by the auto clock gating, the I/Os and PLLs of DRCs are in OFF state and the only power consumption is static. The change of DRCs state is sequential, because the execution of the transactions is completed sequentially. This can be explained by the fact that the execution delay of one transaction is longer than the execution time of the other (i.e. read execution slower than write execution), or transactions execution started at different moments. The change in PLL state is also visible thanks to the fixed delay/penalty. In the third phase, only half of the initiators have been activated. The consumption is almost halved. The reason for this is that there is only one active memory at a time. The consumption peaks are due to a short overlap between two pipelined memories, i.e. one transaction started for DRC0 and another ending for DRC1. In the fourth phase, the traffic is terminated, the clock gating is applied and the voltage value changes from 1V to 0V (normalized values) which means that the DRCs are power gated. The total consumption is switched off at the end.

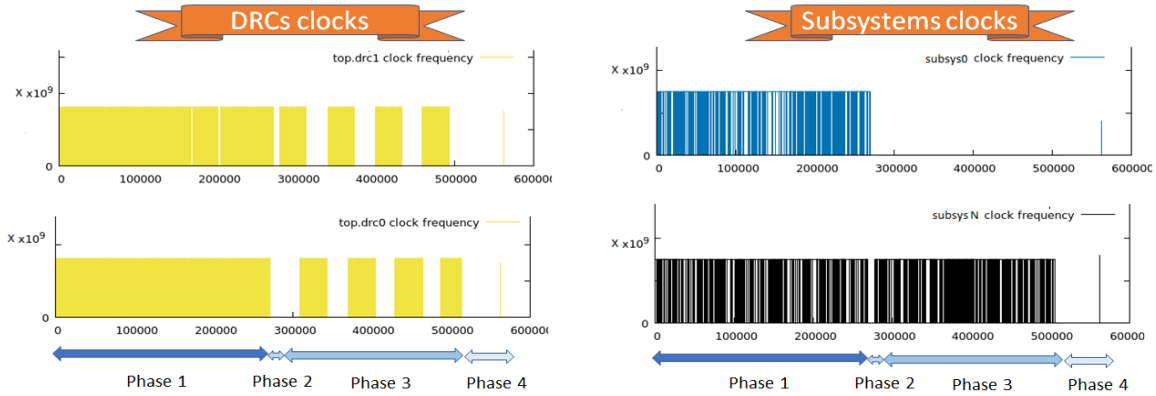

Figure 9. Clock Frequency variation during simulation

In Figure 9 we can observe the variation of the clock frequency. Both DRCs are active during the first phase, inactive during the second and interleaved during the third. We can also observe the Interconnect subsystems and the impact of clock gating. During the first phase all subsystems are active, then during the third phase, half of them are clock gated (we illustrate only two subsystems clocks with different behavior). Total energy consumption in [*mJ*] is shown in Figure 10. The absence of activity (phase 2) is visible. Similarly, the slope of the curve of the third phase is lower than that of the first phase since half of the modules are in clock gated mode.

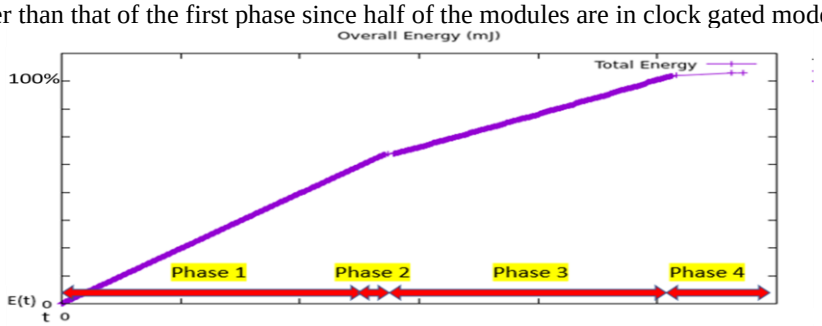

Figure 10. Overall energy consumption profile in test case [mJ]

VII. Conclusion & Future Works

In this paper, we presented a method for estimating power, showing that by using a pure functional model, we can easily define its power-related intent and extract measurements from it. Numerous tests have been performed using this test case, based on *"what-if"* study, such as memory acceptance time variation, read and write processing delays, frequencies, power intent and power management. However, we have seen that modifying model parameters has a direct impact on performance and power consumption and that the relationship between these two effects can be easily analyzed. The methodology applied in this study allowed us to test different power management strategies, to compare their impacts on energy consumption and to choose an optimal power intent. Moreover, we were able to establish a silicon correlation with the simulation results obtained and we had very good accuracy (about 90%-99%) for the maximum and average power consumption. Nevertheless, in this case, we used generic traffic generators, so we must perform some more complex use case simulations to verify the accuracy. Future work includes the development of more accurate testbenches and Virtual IPs (VIPs) models in order to have more accurate estimates and more generic initiators that can be easily reconfigured. On the

PwClkARCH side, LEAT are working on several areas to improve the tool experience. One of their work focuses on distributed PMU to reduce the complexity of internal tables (PST, CST, OPP). Therefore, we will test the interoperability of our functional framework, applying the same methodology on another chip of the same family and evaluate the effort and the accuracy. Once this has been done, we hope to apply the methodology on our next-generation products.

VIII. REFERENCES

[1] P. D. L. C. V. d. Santos, "Low Power Techniques for SoC Design: basic concepts and techniques," Embedded Systems - INE 5439 Federal University of Santa Catarina, 2014.

[2] P.-A. Hsiung, "SoC Design Flow & Tools," Dept of Computer Science & Info. Engineering National Chung Cheng University Chiayi, Taiwan, [Online]. Available: https://www.cs.ccu.edu.tw/~pahsiung/courses/soc/notes/soc01.pdf. [Accessed 21 September 2020].

[3] IEEE, "IEEE Standard for Design and Verification of Low-Power Integrated Circuits," *IEEE Std 1801-2013 (Revision of IEEE Std 1801-2009)*, no. 10.1109/IEEESTD.2013.6521327, pp. 1-736, 29 may 2013.

[4] IEEE, "IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems," *IEEE Std 1801-2018*, no. 10.1109/IEEESTD.2019.8686430., pp. 1-548, 29 March 2019.

[5] N. Chandoke and A. K. Sharma, "A novel approach to estimate power consumption using SystemC transaction level modelling," *2015 Annual IEEE India Conference (INDICON)*, no. 10.1109/INDICON.2015.7443519, p. 1–6, 2015.

[6] N. Dhanwada, I.-C. Lin and V. Narayanan, "A power estimation methodology for systemC transaction level models," in *Proc. Third IEEE/ACM/IFIP Conf. on Hardaware/Software Codesign and System Synthesis (CODES + ISSS)*, Jersey City, NJ, September 2005, pp. 142–147.

[7] R. B. Atitallah, S. Niar and J. L. Dekeyser, "MPSOC power estimation framework at transaction level modeling," in *Proc. 19th Int. Conf. on Microelectronics (ICM)*, Egypt, 2007, pp. 245– 248.

[8] H. Lebreton and P. Vivet, "Power modeling in systemC at transaction level, application to a DVFS architecture," in *Proc. IEEE Computer Society Annual Symp. on VLSI*, France, 2008, pp. 463– 466.

[9] D. Macko, K. Jelemenska and P. Cicak, "Power-management specification in SystemC," in *Proceedings of the 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, Serbia, 2015, pp. 259–262.

[10] D. Macko, K. Jelemenska and P. Cicak, "Verification of power-management specification at early stages of power-constrained systems design," in *IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, Slovakia, 2016.

[11] Synopsys, "Platform Architect MCO," [Online]. Available: https://www.synopsys.com/verification/virtualprototyping/platform-architect.html. [Accessed 21 September 2020].

[12] Intel, "Intel Docea Power Simulator," [Online]. Available: https://www.intel.com/content/www/us/en/system-modeling-and-simulation/docea/powersimulator.html. [Accessed 21 September 2020].

[13] Mentor Graphics, "Underscores Low-Power Strategy with Vista Architecture-Level Power Solution," [Online]. Available: https://www.mentor.com/company/news/esl-vista-low-power. [Accessed 21 September 2020].

[14] A. B. Ameur, D. Martinot, P. Guitton-Ouhamou, V. Frascolla, F. Verdier and M. Auguin, "Power and Performance aware Electronic System Level Design," in *SIES 2017*, Toulouse, France, 2017.

[15] A. B. Mrad, M. Augin, F. Verdier and A. B. Ameur, "A framework for System Level Low Power Design Space Exploration," in *24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Batumi, Georgia, September 2017.

[16] A. B. Ameur, M. Auguin, F. Verdier and V. Frascolla, "Mobile Terminals System-level Memory Exploration for Power and Performance Optimization," in *28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Costa Brava, 2018.

[17] V. Frascolla, R. Hasholzner, J. A. Sue, A. B. Ameur, M. M. Ayub, K. Miesniak and J. Englisch, "Cross layer optimization in terminals," in *26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, September 2018.