2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Timing-Aware high level power estimation of industrial interconnect module

Amal Ben Ameur, UCA ,LEAT, Sophia Antipolis, France(*Amal.BEN-AMEUR@univ-cotedazur.fr*)

Antonio Genov, UCA, NXP SEMICONDUCTORS, Mougins, France (*antonio.genov@nxp.com*)

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

Loic Leconte, NXP SEMICONDUCTORS, Mougins, France (*loic.leconte@nxp.com*)

***Abstract*—The semiconductor industry is developing smaller transistors and succeeding in increasing their on-chip integration density. Therefore, the computing power of modern Integrated Circuits (IC) is constantly increasing and their application domains are becoming countless. However, the increasing complexity leads to higher power consumption and more challenging designs. In order to address these issues and to differentiate themselves in the market, manufacturers and System-On-Chip (SoC) engineers are devoting tremendous effort to researching new development strategies. Numerous studies have shown that one of the essential steps to be taken is to review the early stages of the design flow and in particular to integrate simulation-based modeling and verification at higher level of abstraction.**

In this paper we address this gap and present a proof of concept of an academic power estimation and management methodology, called PwClkARCH, on an NXP intellectual property (IP). Memory power estimation has been improved using DRAMPower. The results prove that with PwClkARCH, we are able to perform mixed performance/power/energy modeling on Approximately Timed (AT) SystemC models, which are widely used for architecture exploration and optimization. Our methodology allows to dynamically extract power metrics and allows to apply power management and reduction strategies, while considering the functional model activity, the power management and reduction strategies and the memory systems consumption.

Keywords—power modeling; virtual prototyping; intellectual property; Approximately-Timed TLM models; PwCLKARCH

I. INTRODUCTION

The problem of CMOS power consumption has been discussed for decades. Designing devices that consume a minimum amount of power was and remains an important effort to consider. It is not enough to know how to calculate power consumption, it is also necessary to understand the impact of each factor, such as the clock frequency and the capacitance, on the device power consumption. There are several tools and methodologies for power estimation and management. The common abstraction level of these tools is the Register Transfer Level (RTL) since the standard Unified Power Format (UPF) already exist for RTL power-aware verification [1]. Nevertheless, RTL is no longer suitable for efficient design space explorations for complex systems. RTL simulations are very time consuming due to the multitude of blocks and signals to be considered in a SoC. However, it is very important to maintain a short time-to-market and to develop competitive products. The UPF committee has followed the trend to start designing SoCs from abstract models above RTL and has provided an updated version 3.0 that can be used at Electronic System Level (ESL). However, the standard is still missing for clock tree modeling and control or what we call the Clock Intent Specification which is mandatory for power optimization. Therefore, the UPF does not cover all the needs to support easy performance/power system design space exploration where power/performance tradeoffs need to be investigated, and where overall power management strategies need to be defined and validated. Synopsys, one of the companies involved in the implementation of UPF standard, has integrated UPF 3.0 in their tool Platform Architect MCO and Ultra [2]. Other companies prefer to use their internal tools and methodologies to solve power issues. Industrialized ESL solutions focus on power estimation capabilities and not on the implementation of power management mechanisms.

The solution presented in this paper is in line with the idea of integrating hardware power management at the ESL level as it allows exploring low-power design optimizations and their impact on different architectural options earlier in the design flow.

II. PwClkARCH

A. *Library overview*

PwClkARCH is a C++/SystemC-TLM library [3] [4] enabling the early stage Integrated Circuits (IC) power consumption estimation. The methodology of PwClkARCH is based on the co-simulation between a power model description and a SystemC-TLM based virtual prototype [5]. Thus, it dynamically extracts power metrics, asserts power/functional coherency and makes it possible to apply power management strategies and observe their impact on energy consumption. This library is inspired from the UPF standard in terms of power domains decomposition, power states tables, operating points tables, supply nets and other features. In addition, it includes clock tree description to support a clock domains management strategy [6]. The concept of Design Element (DE) has been adopted in PwClkARCH from the UPF standard. Design Elements can be compared to "shadows" of the IPs in the functional model. Each IP instantiated in the platform and included in a Clock/Power domain is associated with a DE. The parameters necessary for power computation, such as capacitance and leakage current, are specified during the instantiation of the Design Element (Figure 1).

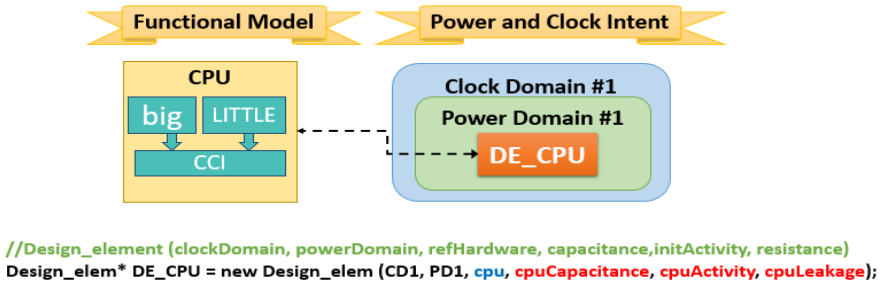

Figure 1. IP and counterpart DE

DE observes the activity of its functional counterpart hardware module using a reference or a pointer. Each DE class dynamically compute some generic power equations based on the component activity and physical data. These equations are based on the sum of static power ($P_{stat} = V * I_{leakage}$) and dynamic power ($P_{dyn} = \alpha CV^2F$) consumptions. A Power Management Unit (PMU) module is used to establish the communication between the functional model and the power model. The PMU dynamically controls all clock/power state changes and applies the power management after checking if all the following configurations are valid.

B. *Methodology behind PwClkARCH*

The approach of PwClkARCH is represented in the Figure 2.

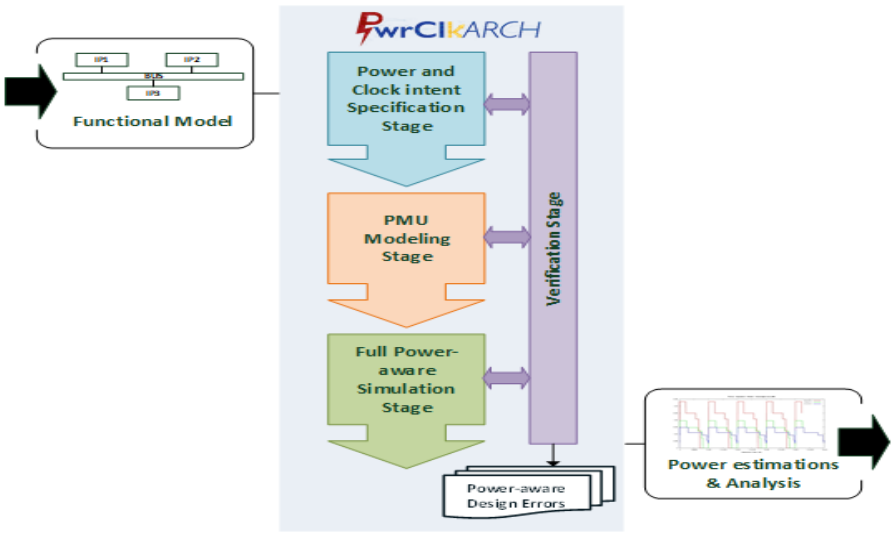

Figure 2. Overall approach of PwClkARCH based methodology

Three sequential steps are required to implement the power model. In the first step, the designers analyze how data are exchanged between the components of the functional model in order to understand when and how often each component has been activated in different use cases. Next, they organize the components into different power and clock domains. This is done by instantiating the appropriate objects from the PwClkARCH library: clock sources, generated clocks, power switches, and supply nets. At this stage, power management strategies can be defined using clock/power state tables and operating performance point (OPP) table [6]. Then, the PMU that interfaces between the functional and power models is implemented in SystemC-TLM. The PMU module is the only power-related component that has to be added to the functional model. However, this component already exists in the PwClkARCH library, so we just need to configure it following the structure of our model. The third step is the simulation of the complete architecture, including the power model and the generation of power results. At the end of this step, the values of static and dynamic power consumption and energy can be plotted in different configurations (static power per hardware module, total static power, static power per power domain, etc.). The approach includes a verification throughout the previous steps to check the consistency between the power management properties and the functional behavior.

The power models of the cited tools (see Table I), except for PwClkARCH, are mainly based on Power State Machines (PSM). With these tools, the only way to apply power management techniques (Clock/Power gating, DVFS, Voltage clustering, etc.) is to integrate them into the functional models, thus breaking the functional/power separation of concerns.

Table I. Comparison with industrial ESL-based tools

Tool	Simulation model	Power estimation	Power Management
Platform Architect MCO and Ultra [7]	SystemC-TLM	YES	NO
Intel CoFluent Studio [8]	SystemC-TLM	NO	NO
Intel Docea Power Simulator [9]	Internal simulator with possible connection with vcd files extracted from SystemC-TLM functional models	YES	NO
Mentor Graphics Vista [10]	SystemC-TLM	YES	NO
PwClkARCH	SystemC-TLM	YES	YES

C. Comparison with previous PwClkARCH based works

In this paper, we focus on four new areas of application of the PwClkARCH methodology, in comparison with previous works [5] [11] [6]. First, we have tripled the complexity and size of the functional model and we have considerably increased the number of design elements and clock domains. This broadens the size of the tables used to control the power and clock states and proves that PwClkARCH is suitable for larger and more complex designs. The second area of novelty is that the architecture used is a Approximately-Timed (AT) Transaction Level Model (TLM) of industrialized hardware, as opposed to the Loosely-Timed (LT) models used for prior application of the PwClkARCH library. This proves that PwClkARCH is suitable for architectural exploration on high-level platforms. The third area of novelty is that in addition to the PwClkARCH/DRAMPower LT co-simulation module, we have created a new one allowing the connection between PwClkARCH/DRAMPower and a functional AT model. The fourth and main focus of this publication is that we present an important case-study on an industrialized interconnection module with one generic multi-tasking and two more complex single-tasking use cases.

III. APPLICATION

To evaluate our methodology, we use NXP i.MX8QM SoC [12] which is a member of the i.MX8 series [13] developed for automotive and industrial applications. This SoC contains a variety of digital and analog modules, such as Core processing unit (CPU) clusters, Graphics processing units (GPU) and Video processing units (VPU), Display Controllers (DC), and other advanced features. For the sake of simplicity, we will refer to all these features under the common name – subsystems or IPs. Each subsystem is composed of multiple blocks and interconnects in order to match a concrete structure and application. There is one subsystem, which we will call Switch Matrix (SM) and which will be the main object of this study. The SM subsystem provides the connection and the communication between more than 15 subsystems and 2 external Dynamic Random-Access Memories (DRAM).

A. Switch Matrix (SM) module description

SM is a highly granular interconnection module with multiple QoS and routing algorithms. Like the other subsystems, SM also contains multiple blocks (Figure 3) and each block serves to translate, treat, isolate or schedule specific transaction or entire bursts of transactions. The internal application of several QoS algorithms, enables the prioritization of certain subsystems traffic and contributes to the optimization of performance and memory usage. The entire SM has its own power domain and more than 20 clock domains. The clock can be turned off for inactive blocks and kept only for the active blocks. In order to apply this dynamic power reduction technique, we use hardware auto clock gating. In addition, when the whole subsystem is inactive, we can apply the power gating and cut the static power consumption.

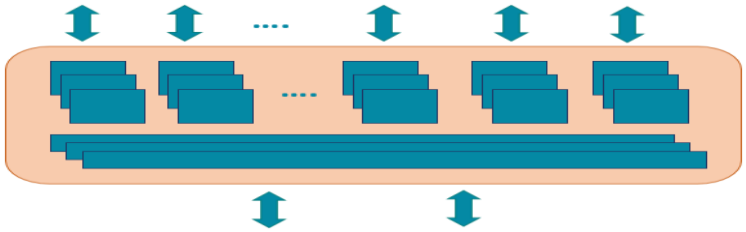

Figure 3. Switch Matrix simplified structure, Blocks` names are hidden for confidentiality concerns

B. Switch Matrix (SM) functional model

1. Communication protocol

The protocol modeling can be error prone and time consuming if we code it separately in each IP. For this reason, we separated the communication part from the behavioral part and created two reusable interface modules (Figure 4).

- **AXI initiator** – IP internal module attached to its corresponding initiator TLM socket. It uses TLM initiator socket to communicate with other IPs and callback functions to communicate with its parent IP.
- **AXI target** – IP internal module attached to its corresponding target TLM socket. It uses TLM target socket to communicate with other IPs and callback functions to communicate with its parent IP.

At each transaction transmission/reception through the socket, the protocol interface triggers a callback process (*CbProcess()*) that "wakes up" the behavioral model for treatment. In reality, these two AXI [14] reusable modules are based on generic protocol initiator and target skeletons. We instantiate the generic protocol modules within the IP functional model and define the specific communication protocol when we instantiate the IP in question in the platform under simulation. The subsystem clock frequency is also passed as construction parameter and is used by the interface modules and the behavioral SystemC models. We can use these skeletons and follow the same approach to create and test other communication protocols without significant changes on the functional prototype.

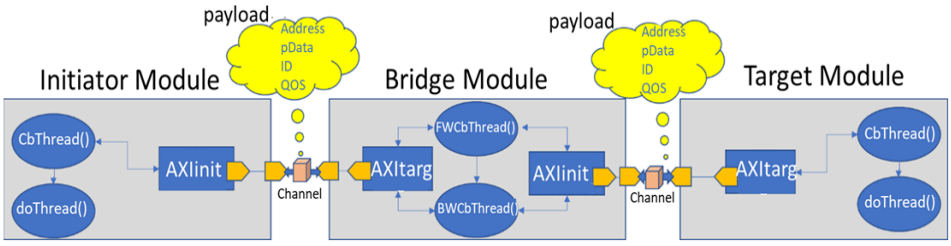

Figure 4. AXI interface modules

2. Behavior

In behavioral model it is very important to model the entire transactions treatment process in order to maintain functional accuracy and to respect the timing and deadlines. From a developers' point of view, it is also important, if possible, to maintain a fixed common structure for all IP components and add additional processing threads if necessary. With this in mind, we have created a skeleton that can be reused when coding a new IP. Each IP model contains at least two ***Accept[Read/Write]PEQ*** payload event queues (PEQ) which are FIFO-based queues containing a pair of transaction and its receive time stamp. These PEQs are notified when a read/write transaction

is received. PEQs are used in two ***Accept[Read/Write]Thread()*** processes of type *SC_THREAD* that execute the IP specific request processing. These threads await notification from the ***Accept[Read/Write]PEQ*** and execute their IP-specific request acceptance processing. Once accepted, these threads wake up their corresponding transfer methods using the ***forward[Rd/Wr]Event*** events. The timing and number of these events notifications depends on the specific IP functional model. The forwarding methods ***Forward[Read/Write]Method()*** checks whether the forward AXI channel is free and if so, they transfer the scheduled read/write transaction. Intermediate processes can be easily added, which guarantees the interoperability of the behavioral skeleton. All SM blocks, including the bridges and AXI interconnections, are based on this skeleton.

C. Testbench

During the simulation, the platform or IP under test, must be stimulated by transactions to be activated. The stimuli come from Traffic Generators (TG), which are regular SystemC initiator modules, initiating transactions in a specific way. We have developed multiple generic configurable TGs that can be used to simulate simple use cases and some application-specific ones. In this study, we analyze memory accesses through an intermediate interconnection module. Therefore, we also need an accurate memory module that communicates with the TGs through the SM module. Figure 5 shows a very simplified overview of our testbench containing several TGs, our Design Under Test (DUT) and two DRAM memory models. Using PwClkARCH components [5] we build the power model of our DUT and connect it to the functional model via a PMU block. The detailed power intent implementation is given in [15]. Since we are not involved in the creation of an accurate functional memory model, we prefer to use existing solutions.

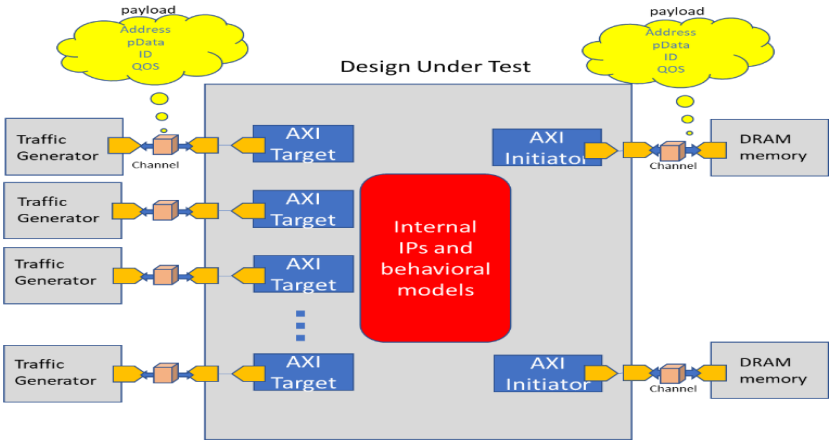

Figure 5. Testbench structure

Academic and industrial tools are available for performance and power estimation of DRAM memory. It has already been proven that PwClkARCH and DRAMPower [16] can be used in a single framework in to obtain an accurate estimate of the power consumption of a given DRAM memory taking into account all memory operations [11]. Considering that the accuracy of the memory model has an important impact on the interconnect power estimation accuracy, we made two observations at two different levels of abstraction for the memory system and compared the results. The first one was performed with simple DRAM blocks accepting and executing READ/WRITE commands and initiating transaction phase changes. The configurable parameters in these models are memory READ/WRITE accept and response delays, memory size and width and the power contribution for different states, like clock gated, IO down and PLL down states. We automate the states evolution by using a simple module activity monitoring and hysteresis counting. The second observation was made using DRAMPower. DRAMPower performs high-precision modeling of the power consumption of different DRAM operations, state transitions and power-saving modes at ESL level. The connection between DRAMPower and PwClkARCH has been established in such a way that we obtain the most accurate power consumption values while maintaining optimal simulation speed. For this purpose, DRAMPower is invoked by the SystemC-TLM functional model only once per read/write transactions window (in our case we use windows of 10 transactions). It then returns the average

power consumed by the memory for this window and updates the overall power consumption computed by PwClkARCH (Figure 6). We have also enhanced the DRAMPower tool to calculate the average response time per transaction for each window in order to improve the functional model timing accuracy.

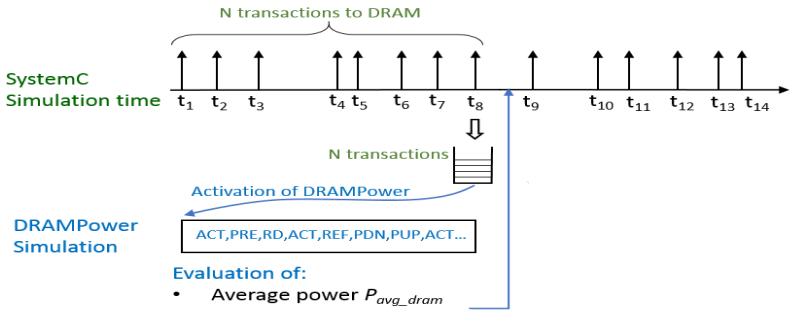

Figure 6. Connection of DRAMPower with a SystemC-TLM model augmented with PwClkARCH

IV. SIMULATIONS, RESULTS AND CORRELATION

We have applied the clock/power domains distribution and the defined power management strategy on our model, and we have tested multiple use cases starting with generic ones and moving to more application-specific ones. Our power model essentially contains more than 25 DEs and clock domains and about 5 power domains. In order to replicate the clock management used in real hardware, we have enhanced the PwClkARCH library to support hardware auto clock gating capabilities. We consider each IP clock to be activated when transactions are received/treated/kept/initiated and deactivated if there is no traffic for a period of 32 cycles. In this section we will give a quick overview of some of the tested use cases and the total power consumption (SM+DRAM). Axis scales are masked for confidentiality reasons.

A. Generic simulations with multiple active traffic generators (Figure 7)

These generic-level simulations were extremely useful during the functional model development, as the power related curves gave us a clear view of each module activity and simplified the debugging and analysis stage. Moreover, we had our first power estimates and promising silicon correlation. This can be a good first step when developing new platforms and we do not have IP models for traffic generation (like CPU, GPU, Display Controllers….). In Figure 7, we illustrate a comparison between identical simulations with the only difference that for the first one we use a simple memory model and for the second one we use DRAMPower with MICRON-16Gb-LPDDR3-1600_32bit specification.

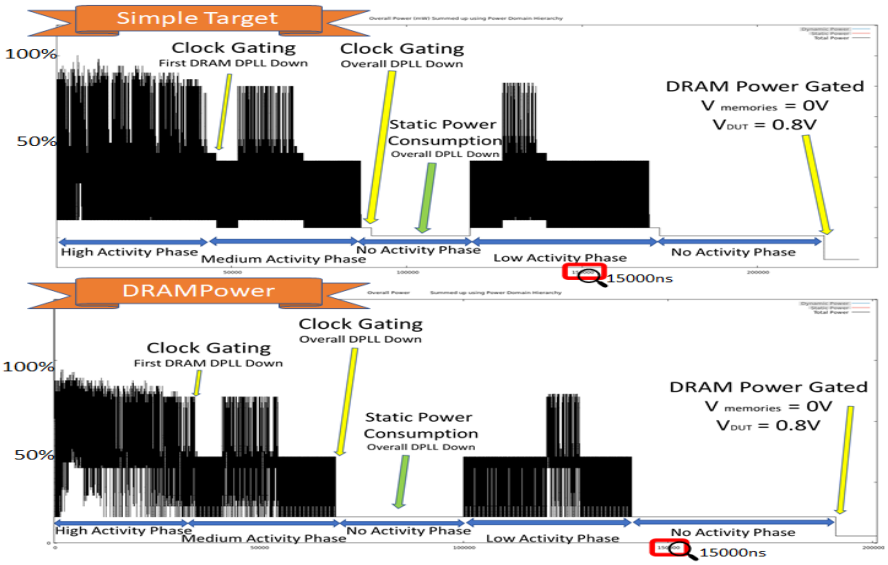

Figure 7. Overall Power consumption without/with DRAMPower

During the high activity phase, all traffic generators and both DRAMs are activated, thus the maximum power consumption is reached. During the medium and low activity phases, we activate several (but not all) traffic generators, but most of them access only one memory. There are small fractions of time when both memories are activated simultaneously. The non-activity phase is a rest phase added to test the clock gating mechanisms. Silicon correlation has shown us that the maximum and average power consumptions are 99% accurate. The dynamically obtained simulation results have been successfully correlated with the extrapolation of silicon measurements. We can clearly observe the differences between the two simulations. The PwClkARCH simulation with DRAMPower uses more realistic and variable response timing values (by considering refresh rates, banks and ranks interleaving and more), while the simple memory module uses constant values inferred and approximated from previous hardware measurements. The red rectangle shows a chosen time stamp in the simulation and highlights the timing error introduced by the constant values methodology. Thus, the combination PwClkARCH/DRAMPower improves our functional model and therefore it increases the power and performance estimation accuracy. In Figure 8, we illustrate the overall energy consumed during both simulations.

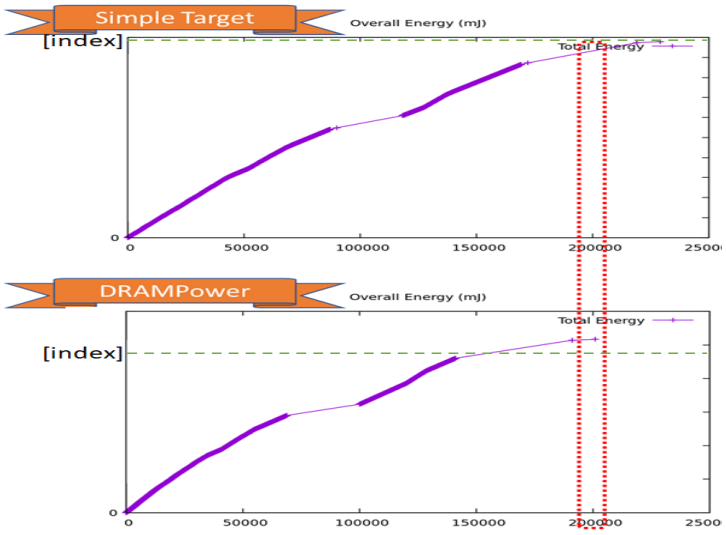

Figure 8. Overall energy consumption comparison without/with DRAMPower

The **[index]** represents a fixed value used to compare the two energy consumptions. It can be distinguished that the simulation using DRAMPower executes the same number of transactions in less time than the simple memory one. As a result, the slope of its curve is steeper. At this level, with these generic traffic generators we can have a good correlation for the maximum, average and minimum power consumption (with or without DRAMPower). However, if we want a real use case simulation, we need more application-specific traffic generators.

B. 256KB Memcopy use case (Figure 9)

In this use case, we have only one active application-specific traffic generator that executes 256KB DRAM read accesses performed by a CPU cluster. Each data access is 16 bytes and we consider sequential memory accesses. In order to optimize the memory usage, we apply 4K interleaving between our two DRAM memories.

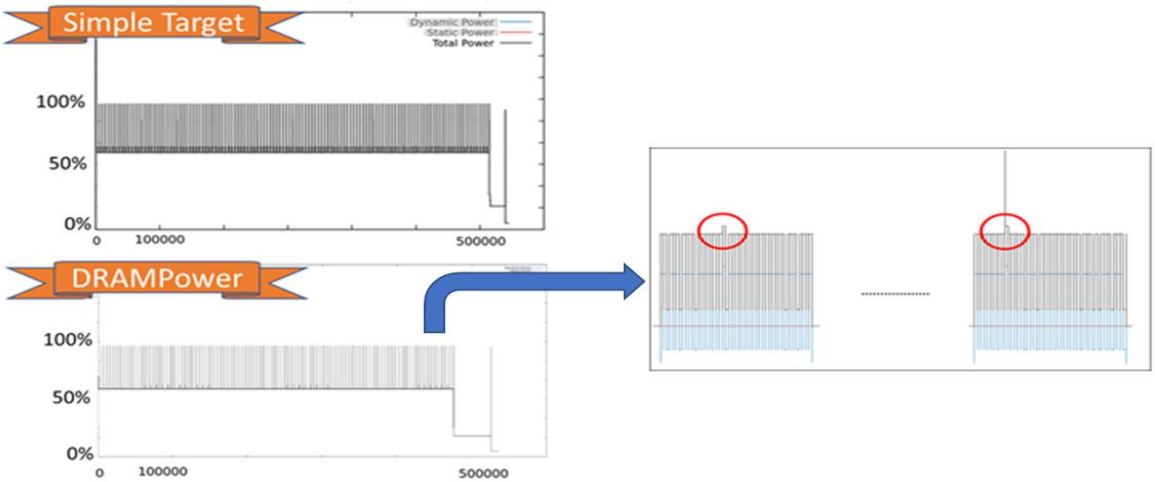

Figure 9. 256KB MEMCOPY - Total Power w/o DRAMPower (+ zoom on spikes)

On Figure 9 we can observe the constant activity due to the continuous memory accesses. The spikes come from the moments of interleaving between the two memories. The simulation with DRAMPower ignorew some of the spikes due to the interleaving, because we use windows of multiple transactions and the memory power consumption is normalized (Figure 9) for the whole window. The correlation with silicon power related measurements is approximately 95% accurate.

C. Display Refresh HD1080-1920x1080@60-32b use case (Figure 10)

In this use case we consider a HD display with 1920x1080 screen resolution and we execute one frame of transactions. The data paths are 256 bits and each pixel is 32 bits, so we have 8 pixels per transaction.

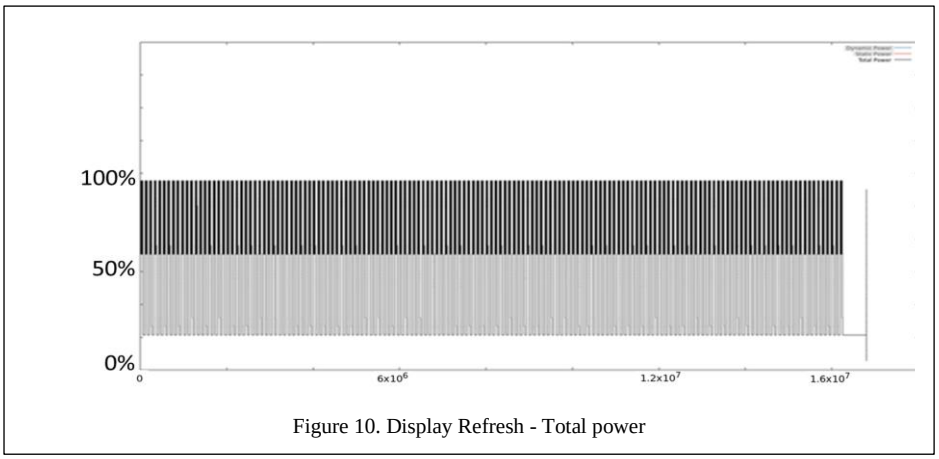

Figure 10. Display Refresh - Total power

An active display data line is 1920 pixels, which means 240 transactions. We have a total of 1080 lines, so a total of 259200 transactions for one frame. We consider a multiple lines prefetching mechanism, the horizontal and vertical blanking periods, pixel frequency and refresh rate. Each data access is 32 bytes and we consider sequential memory accesses (prev_addr+32bytes). In Figure 10 we can observe the activity variation. During the display traffic, the generator periodically sends bursts of transactions (catching lines) and there are certain moments of inactivity between these bursts. Once again, the spikes are due to the interleaving between the two memories. The silicon correlation is about 90% accurate. We are also able to extract the static and dynamic power consumption, clock activity and operating points for each component of our platform. It is important to mention that the simulation time for these use cases takes between 10 seconds to 10 minutes, which is between 10-1000 times faster than RTL simulation.

V. CONCLUSION

The purpose of this study is to present our proof of concept of a high-level timing-aware power estimation library, called PwClkARCH, on an AT SystemC-TLM2.0 model with an accurate DRAM model provided by DRAMPower. We are also contributing to an interesting and significant case study on a complex interconnection system. The extracted power metric has a good correlation with silicon measurements and tends to be reliable for more complex use cases, such as the display refresh one. The next step is to generate a GPU-based use case to test the model accuracy on a more complex workload. We are currently working on the functional model interoperability test and we have been able to easily reuse the SM functional model for another SoC of the NXP i.MX8 family. We are also reusing the power intent definition with minor changes. The effort required to integrate PwClkARCH into the functional model is minor if we have already defined some of the architectural options and corresponding power management strategies that we want to test. In order to connect the power model and the functional one, we simply need to define the design elements for each IP and associate them with clock/power domains, clock state tables, power state tables and operating points, which is done outside the functional model. The Hardware auto clock gating implementation is the only part that had to be added in the functional code, but it is not intrusive at all. We have a power observer instantiated in the constructor of the IP model and we use two SystemC events to indicate the module activity. The effort required to create the IP functional models using our skeletons is significantly reduced. We need to enrich our library of reusable IPs with SystemC models from IP vendors. One way to do this is to support the integration of PwClkARCH into an industrial EDA tool or to further develop our internal models and framework.

VI. REFERENCES

[1] *IEEE Standard for Design and Verification of Low-power, Energy-aware Electronic Systems. IEEE Std 1801–2018.*, 2018.

[2] P. Sheridan, "Time For A DDR Background Check. How background power consumption impacts the energy efficiency of an SoC," *semiconductor engineering*, July 28th. 2016.

[3] *IEEE Standard for Standard SystemC® Language Reference Manual*, IEEE Std 1666–2011, 2012.

[4] *OSCI TLM-2.0 LANGUAGE REFERENCE MANUAL*, Open SystemC Initiative, June 2008.

[5] A. Ben Ameur, D. Martinot, P. Guitton-Ouhamou, V. Frascolla, F. Verdier and M. Auguin , "Power and performance aware electronic system level design," in *12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, Toulouse, 2017.

[6] H. Affes, M. Auguin, F. Verdier and A. Pegatoquet, "A methodology for inserting clock-management strategies in transaction-level models of systemon- chips," in *Forum on Specification and Design Languages (FDL)*, Barcelona, Spain, 14-16 Sept. 2015.

[7] "Platform Architect MCO," [Online]. Available: https://www.synopsys.com/verification/virtual-prototyping/platform-architect.html.

[8] "Intel® CoFluent™ technology," [Online]. Available: https://www.intel.com/content/www/us/en/cofluent/overview.html.

[9] "Intel® Docea™ Power Simulator," [Online]. Available: https://www.intel.com/content/www/us/en/system-modeling-and-simulation/docea/power-simulator.html.

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

[11] A. Ben 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)*, Platja d'Aro, 2018.

[12] N. DataSheet, *i.MX8QuadMax Automotive and Infotainment Applications Processors*, Document Number: IMX8QMAEC , 10/2019.

[13] "i.MX 8 Series Applications Processors," [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-8-processors:IMX8-SERIES.

[14] A. I. 0022E, *ARM AMBA AXI and ACE Protocol Specification*, Patent ID022613.

[15] A. Genov, L. Leconte and F. Verdier, "Mixed Electronic System Level Power/Performance Estimation using SystemC/TLM2.0 Modeling and PwClkARCH Library," *in press DVCon EUROPE* 2020.

[16] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn and K. Goossens, "DRAMPower: Open-source DRAM Power & Energy Estimation Tool," [Online]. Available: http://www.drampower.info.