2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)

Advanced SoC Virtual Prototyping for System-Level Power Planning And Validation

Fabian Mischkalla, Wolfgang Mueller
University of Paderborn, C-LAB
Fuerstenallee 11, D-33102 Paderborn, Germany
Email: {fabianm,wolfgang}@c-lab.de

***Abstract*—Today's electronic devices imply significant efforts in pre-silicon low-power design. Key techniques such as scaling of operating points, or switching power off to unused blocks play a major role and are usually managed at entire system scope. Finally, the delay paid in today's design cycles is the time needed for verification, since power-aware design starts late and, hence, relies on painful system-level simulation performances.**

In this work, we describe a SystemC-based virtual prototyping approach capturing low-power design characteristics in earlier design stages. In contrast to existing solutions, we focus on advanced modeling techniques such as blocking transactions and temporal decoupling, which are integral part of modern industrial-sized designs. As proof of concept, we use representative virtual platform models at several levels of abstraction. Based on empirical results we argue that our annotations cause reasonable overhead, but provides sufficient details to validate typical power planning scenarios.

***Keywords*—Virtual Prototyping, SystemC/TLM, IEEE 1801-2013 (UPF), Temporal Decoupling, Power Intent Validation,**

I. Introduction

System-level design requires planning the power architecture just as much as the functional architecture. It involves testing for functional correctness, emulating electrical states, and asserting accurate power-down and power-up sequences [1]. However, this has only been done in RTL-to-GDSII design stages [2]. Hence, starting at higher abstraction levels, as illustrated in Figure 1, provides earlier design time validation and may help to avoid costly redesigns.

Traditionally, a design's power architecture, denoted as power intent, is defined at RTL based on standards such as UPF[1] or CPF[2]. In both formats voltage connectivity and implementation strategies for state retention, isolation, as well as voltage level-shifting can be defined. Recent power-aware simulators add then cycle- and register accurate models and mimic the logical behavior of such power management structures. However, even quite accurate, existing solutions are limited to RTL and, hence, inefficient in managing today's entire low-power design spaces.

At system-level, SystemC-based *virtual prototypes* have been used for some years now. Since they are mainly for architectural exploration and early software development, typical of-the-shelf models are optimized for simulation speed by dropping unnecessary implementation details. A virtual prototype comprises a set of transaction-level modeling (TLM) components, which represent hardware blocks and interact via function calls in a loosely-timed (TLM-LT) simulation context. Moreover, greater performance benefits are gained by optimization techniques such as temporal decoupling [3]. It allows concurrent processes to run ahead of simulated time, and thereby reduces synchronization needs. Basically, which is equivalent with lower process scheduling efforts and hence less CPU time spent in context switching simulation.

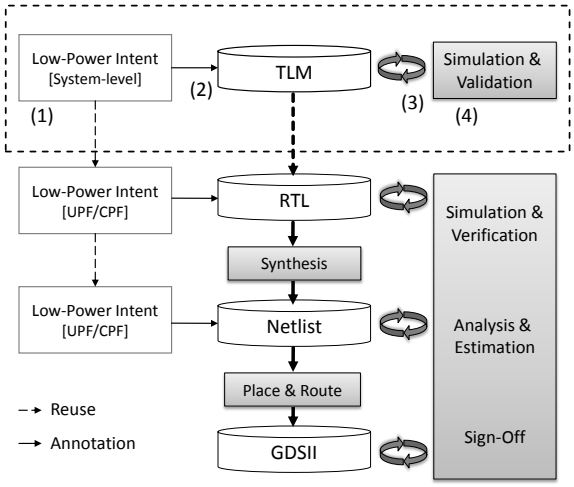

Figure 1. TLM virtual prototyping as front-end for low-power design flows

Consequently, we deal with the question whether these TLM techniques allows fast and accurate power intent simulation as well as consistent validation. Although virtual prototypes maybe initially correct, a native application might become highly inaccurate, when power architectures are inserted and electrical effects like voltage ramps take place. For this, we developed a low-power abstraction method for managing temporal decoupled and DMI-based TLM models. It works with legacy SystemC components and performs power intent overlay at standard TLM interfaces. Main steps include a generic instrumentation method as well as automatic low-power design elaboration. Finally, our approach guaranties fast and accurate simulation and provides observation of most important power management events. In detail, our contributions can be stated as follows:

1) System-level low-power abstraction concepts as extension for CPF/UPF. This includes abstract definition of voltage relationships, but also dynamic aspects such as operating conditions and TLM power state semantic.
2) Efficient annotation considering standard TLM coding styles, i.e., generic protocols and core transport interfaces.
3) Accurate power intent preserving in temporal decoupled simulation contexts.
4) An evaluation on power intent validation using representative virtual prototyping models.

[1]Unified Power Format (IEEE Std. 1801)
[2]Common Power Format (Si2)

978-1-4799-5412-4/14/$31.00 ©2014 IEEE

II. Background & Problem Definition

In general, power-aware design from a hardware perspective consists of additional power management (PM) structures and functionality enabling low power techniques such as power gating and multi-voltage scaling. As a result, required information are captured either directly in a hardware description languages (HDLs) or provided as side-files. To demonstrate this, Figure 2 outlines a power gating scenario with state retention capabilities. There are two functional blocks of which one can be power gated. Power up and down is controlled by an additional power control unit and a switching circuit driving the internal power supply net `VDDi`. As shown in Listing 1, equivalent RTL behavior can be easily accomplished and HDL pragmas (line 2,8, and 13) enable to switch between normal and power-aware simulation mode. Presuming a synchronous clocking style and an asynchronous reset, the behavior of the power gated block should be altered in the following way:

- If power gated, i.e. `VDDi` is zero, all port outputs becomes their default initial value (e.g. "X" for 4-state types) and internal states are deleted as result of insufficient power supply
- If notified, save register values and reinitialize as response to a restore request
- Keeping priorities of power, retention, reset, and clock signals to ensure sequential correctness

Today, several EDA simulators automatically integrate such overlays for functional simulation. Technically, these tools first identify the sequential elements that are inferred by RTL. Afterwards, they add models that mimic cycle-accurate behavior for data corruption as well as retention, isolation, and level-shifting strategies. Finally, they perform voltage-level simulation and power state coverage. However, as obvious power-aware RTL simulators are even slower than conventional ones and, hence, fall short at industrial-sized system-level scenarios. Since 2006, there exist also standardization efforts [4, 5]. Hereby, power intent is expressed in Tool Command Language (Tcl) syntax. Nevertheless, as these standards mainly address RTL-to-GDSII flows, their kind of low-power abstraction is not well-suited for higher-level system design stages. In particular, there is neither a mapping to loosely-timed simulation semantic nor any TLM directives for concepts like power states or retention.

In case of low-power verification, as described in [6, 7], issues such as power domain crossings and detailed power control sequences are investigated. Basically, verification engineers rely on recent RTL property checking techniques. Thus, dynamic assertions are either implemented as simulator

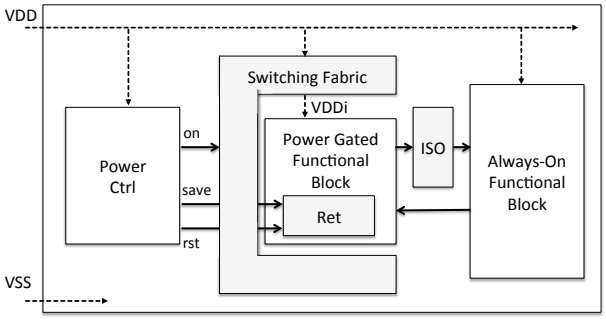

Figure 2. A power gating design with state retention support (adapted from [1])

```
2  'idef RTL_PG_EMULATE!
       wire vddi;
4      assign vddi = pon
       reg[3:0] save_current_state;
6  'endif

8  always @(posedge clk or negedge nrst
   'idef RTL_PG_EMULATE
10     or vddi or posedge save or posedge restore
   'endif
12 ) begin

14 'ifdef RTL_PG_EMULATE
       if (!vddi)
16         current_state <= 4'bXXXX;

18     // retention behavior
       else if (save)
20         save_current_state <= current_state;
       else if (restore)
22         current_state <= save_current_state;
       else
24     begin
   'endif
26
       //normal behavior
28     if (!nrst)
           current_state <= 4'b0000;
30     else
           current_state <= next_state;
32 'ifdef RTL_PG_EMULATE
       end
34 'endif
   end
```

Listing 1. RTL code extended with power-gating semantic

built-in checks or derived from user specifications in languages such as SystemVerilog or the Property Specification Language (PSL). Besides simulation approaches, there are also formal methods [8, 9, 10], but still require RTL as input.

Addressing power-aware design and verification at higher-level abstractions is certainly a very novel research area and there exist only few approaches. First SystemC based solutions continue to be limited to RTL coding styles. In [11], for instance, the authors adapt the SystemC kernel for power gating simulation via hardware data types and clocked processes. Likewise, in [12], they model pin and signal driven power structures according to UPF 1.0 and execute them in conjunction with cycle-accurate SystemC models. A first discussion aligned to TLM appears in [13]. While the author provides only general remarks, [14, 15] go a step further and propose working solutions for specific modeling scenarios. In [14] they extend a proprietary virtual platform model by means of voltage, reset-, and clock-tree integration. The authors of [15] apply UPF low-power abstraction for power estimation and exploration. However, both modeling styles still tend to overspecification and rely on detailed supply network specifications (i.e. supply ports, nets, switches) instead of abstract voltage relationships. Consequently, their power intent models imply implementation knowledge of the power supply network, a good understanding of the power control infrastructure, and general access to the entire design. Moreover, different power supply on individual ports has not been considered so far which may result in incorrect simulation outcomes. Especially, if it comes to multilevel hierarchies or integration of intellectual property (IP). Finally, no existing approach addresses a voltage-aware corruption semantic in a temporal-decoupled simulation context. Thus, accurate power management simulation remains either detailed lock-step dependent or is left to users' skills.

III. Transactional Low-Power Abstraction

Today's low-power design and verification standards [4, 5] still lack adequate semantic for higher abstraction levels than RTL. In this section we describe TLM directives which are proposed as extensions on top of these existing standards. In general, three TLM issues must be resolved: (i) lack of micro-architectural details, i.e., no registers, wires, etc., (ii) complex intellectual property (IP) adoption, and (iii) loose timings and synchronization. With respect to these handicaps, the linchpin of a TLM-driven approach is the transaction. As shown in Figure 3, transactions can be seen as a sequence of interface method calls (IMCs) from an initiator to one or multiple targets and vice versa. IMCs are called on ports and are specified by interfaces associated with the ports. A port is defined at component boundary and constitutes a TLM model in terms of the only accessible design elements. Furthermore, each method call results in two control paths of which both must be considered as individual message passing event. Messages may forward data in terms of incoming and outgoing signals grouped as payload Φ_m where m is the associated message. Additionally, each message has a simulated timestamp $\mathcal{T}_m$ with $\mathcal{T}_m \leq \mathcal{T}_{m+1}$ for messages of the same transaction and over the same port. However, partial out-of-order simulation in terms of execution order is quite permissible. Based on these characteristics, we formalize low-power abstraction by means of several views.

A. Conceptual View

Usually, power distribution specification is propagated throughout the design hierarchy. Based on a predefined scope, descendants, if not otherwise requested, are overlaid with same power intent characteristics. Since this influences functionality, TLM models should behave identically, at least at their boundaries. For this purpose, we adapt the concept of UPF *power domains* for modeling TLM component ports as power domain elements and, thus, partition the design by grouping component interfaces according to their power intent requirements. However, TLM may differ from RTL in type and number of primary in- and outputs. Thus, we assume port definitions that are continuously powered by the same supply, i.e., all underpinnings of transactions are in one power domain and does not need to be further refined. But in fact, it is the case in existing TLM models that are mainly based on memory-mapped bus communications as well as single clock, reset, and interrupt entries. Furthermore, in contrast to UPF, we map a power domain directly to one clock domain. Thus, we assume atomicity (not splittable), which guarantees a distinct mapping to functional equivalent UPF power domains, i.e., that are expected to always have the same voltages and states.

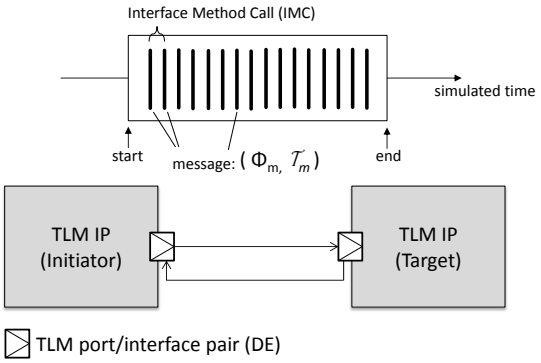

Figure 3. General TLM definition

Data Corruption

Power domains $\mathcal{PD}$s change logic behavior due to insufficient voltage. In UPF, proper semantic is described as data corruption by means of *simstates*. Each simstate defines a distinct operational level from normal functional to non-operational. As a result, signals may become unpredictable, which is defined as an instantaneous assignment of HDL default values to these signals. Design objects to which corruption applies are restricted to RTL primitives such as registers and wires as well as process triggers.

In contrast, TLM models lack these implementation details. Even more, there is rather one single process responsible for complete internal behavior which makes a straightforward adoption of simstates difficult. In particular, a TLM abstraction has to cope with corruption effects limited to component boundaries as well as data types that do not have initial default values. Moreover, transactions to or from corrupted components must be accurately discarded or answered with a power down response. Consequently, we abstract from standard simstates and distinguish two power domain states, i.e., "on" and "off". If "on", a transaction should consider clock changes by multiplying timestamps with a current domain specific frequency factor $\mathcal{F}_{PD}/\mathcal{F}_{default}$, whereas in the off state it should mimic simstate semantic on its paths. Based on these requirements, we define a TLM corruption semantic (see Equation 1) per message passing event $m = \{m_{call}, m_{return}\}$ and corruption behavior $discard(x)$ as well as $error(x)$ for Φ_m. Function $discard(x)$ prevents undesired data propagation on call paths, whereas $error(x)$ informs about insufficient power supply on return. Technically, we implemented several options based on standard command and response status modifications or ignorable transaction extensions, but other solutions might be possible.

$$corrupt(\mathcal{PD}, m, \Phi_m, \mathcal{T}_m) =$$

$$\begin{cases} discard(\Phi_m) & \text{if } state(\mathcal{PD}) = \text{off} \wedge m = m_{call}, \\ error(\Phi_m) & \text{if } state(\mathcal{PD}) = \text{off} \wedge m = m_{return}, \\ \mathcal{T}_m \cdot \mathcal{F}_{PD}/\mathcal{F}_{default} & \text{otherwise} \end{cases} \tag{1}$$

Protection & Retention

Power domains represent also physical chip areas, and even being switchable, these chip areas remain connected. In order to protect adjacent regions in case of partial power downs, isolation logic for crossing signals is necessary. Furthermore, power domains may operate at different voltage ranges. In this case, also level-shifters must be inserted for translating data values from lower to higher voltage levels and vice versa. At TLM, appropriate behavior should reflect the targeted power domain as well as its own power supply. On the one hand, this keeps data corruption semantic, on the other hand, it preserves power domain ordering dependencies. Further implementation details, however, can be ignored, since neither isolation logic nor level-shifting influences a system's functionality, but would require expensive and frequent data overwriting. For isolation,

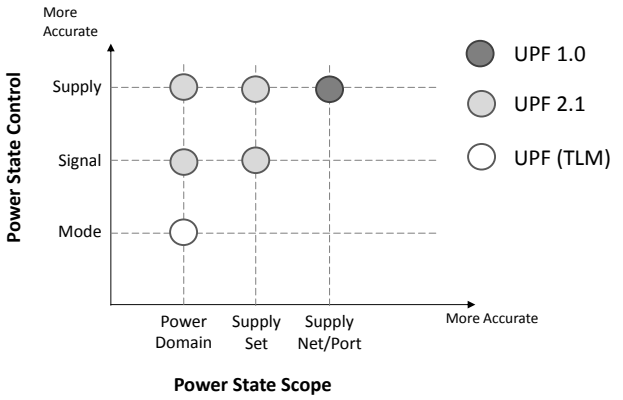

Figure 4. Power state taxonomy based on controllability and scope

we also assume automatic enabling, which is sufficient for early power management specification. However, if focus is on power control implementation, explicit isolation control might be added. In case of power gating, logic states are usually lost. For this, a retention strategy maintains state information that cannot simply be recomputed on power up. Regardless of the implementation method, i.e. either software or hardware-based, retention is typically accomplished in two ways: partial or total. A partial strategy focuses mainly on individual registers, whereas total retention addresses entire power domains. Both have in common that they require additional time. However, consistent TLM retention modeling is critical due to the lack of micro-architecture details and a limited access at component boundaries. Hence, real data preservation is not applicable without too much restrictions to coding styles. Moreover, as internal register corruption is not simulated, retention functionality seems dispensable. Naturally, this raises the question to what extent retention modeling makes sense. As a reasonable compromise, we propose retention control modeling by checking missing or overlapping save and restore control command sequences.

B. Operational View

Using active power management techniques, system portions operate at different and dynamically changing voltage and frequency levels. Usually, this state space is described in a Power State Table (PST), which defines all legal power state combinations. At system-level, today's power state specification often results in state explosion and must be captured in a hierarchical form. As shown in Figure 4, UPF power states are defined at different scopes, namely on power domains, supply sets, and supply nets. A generic TLM abstraction should rely on power domains as further power architecture details may not be available at time of modeling. In addition, state control plays an important role, which spans a 2-dimensional space for power state abstraction. Currently, there exist two modeling practices. Initially, simulation control has been accomplished at the supply network level. For this, power states are invoked on late objects such as supply nets, power-ground ports and power switches. Later on, UPF supports RTL expressions and provides power state semantic based on individual signal updates. However, both approaches do not really fit with higher-levels of design abstraction. For instance, prior to having an implementation model, a supply network may not or only partially be defined. Likewise, adequate power control state machines may not be available. In contrast, we propose a mode-based control

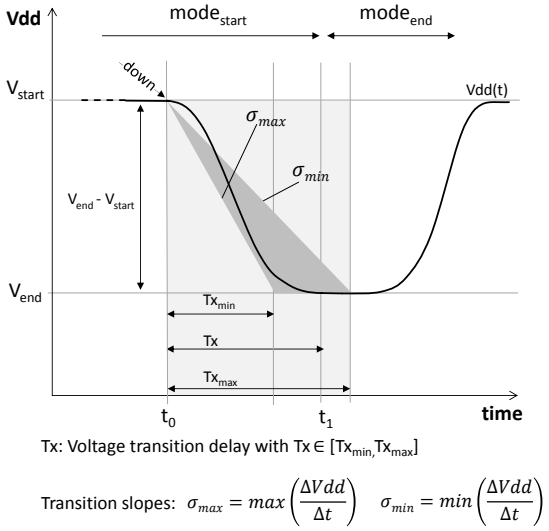

Figure 5. Voltage transition delay characterization

semantic as follows. The foundation is a set of operating conditions (OPs). Each OP identifies a tuple of electrical state characteristics, namely voltage, frequency, and the simstate, i.e., "on" or "off". Furthermore, a set of power modes (PMs) model coherent PSTs with one PM as default mode which applies unless no other is specified. PMs include a collection of power domains and associated OPs where each pair forms a power domain state. PMs in different PSTs may also be non-mutually exclusive, i.e., a mode can be associated with or span multiple other modes. Thus, a TLM design may be simultaneously in more than one power mode. Finally, a set of transition objects define valid mode transitions by their starting and end mode.

C. Physical View

Mode-based power state abstraction incorporates additional physical effects that may influence the system behavior once it is applied. When a power down events happen, power switching is usually not instantaneous according to internal capacities and the nature of silicon technology. Similarly, the process of turning on requires time that cannot be ignored. Consequently, a sufficient transition delay model must be derived, which provides power state accuracy without over-specification. Figure 5 illustrates a typical power mode switching sequence from a high to lower voltage level and outlines, even simplified, voltage progress over time $Vdd(t)$. The important timing instants are t_0 as the time of the control event and t_1 as actual voltage-level switching time. Voltage transition delay is defined as the difference between both points in time. Unfortunately, exact timings are always implementation dependent and, hence, can only be estimated at time of modeling. For this, we propose an efficient abstraction using linear approximation of voltage changes characterized by the slope $\sigma = \Delta Vdd/\Delta t$ with voltage step ΔVdd per minimal time resolution Δt. Moreover, these transition rates may be defined as restrictions of minimum σ_{min} and maximum σ_{max} boundaries for later implementation.

Based on these constraints, we define a voltage transition delay model as follows:

$$Tx_{min} = \begin{cases} \frac{|V_{end} - V_{start}|}{\sigma_{max}} & \text{if } \sigma_{max} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

If σ_{max} is defined, we denote the minimum transition delay (see Equation 2) as quotient of the absolute voltage-level difference and σ_{max}. Otherwise, transition delay is zero which we consider as an ideal voltage switching assumption. The same formula applies for the maximum transition delay if σ_{max} is replaced with σ_{min}. In the end, voltage transition delay can be modeled as range $Tx = [Tx_{min}, Tx_{max}]$ In fact, this delay model involves all possible design scenarios, i.e., unknown ($Tx_{min} = 0$), small or even longer transition times and, hence, a worst-case in terms of power mode changes at arbitrary points in time. Nevertheless, as power states (see Equation 1) have a direct impact on system functionality, a precise semantic preserving is necessary to get correct results.

IV. Temporal-Decoupled Simulation

A power-aware simulation flow incorporates several tasks; (i) power intent elaboration, (ii) original design instrumentation, and (iii) efficient simulation to reflect power related deviations in system behavior. In [16], we published a proof of concept framework based on UPF parsing and a SystemC library for system-level low-power design constructs. By this, we are able to translate UPF compliant specifications into executable power intent models. Furthermore, we introduced systematic TLM instrumentation with regard to power domains and power states [17]. As a result, we achieved transaction observation only where necessary and reduced the number of TLM design hooks significantly. However, we rely on simulation speedups that heavily depend on power intent specification. In fact, we derived synchronization needs based on given voltage transition delays. This section describes a simulation mechanism that eliminates such dependencies and addresses early design phases, in which implementation details are unknown. For this, we propose a look-ahead approach based on dynamic quantum boundaries. Our main goal is power intent preserving with reasonable simulation overhead. In essence, we achieve a precise transaction synchronization despite the use of temporal decoupling.

A. Scope & Limitations

In general, SystemC allows models at different degrees of accuracy and speed. However, raising both at the same time is typically not possible. Usually, a high performance involves less modeling details, and a higher degree of accuracy comes along with lower speed. As specified in [3], TLM distinguishes therefore two coding styles: an approximately-timed (AT) and a loosely-timed (LT) coding style. TLM-AT

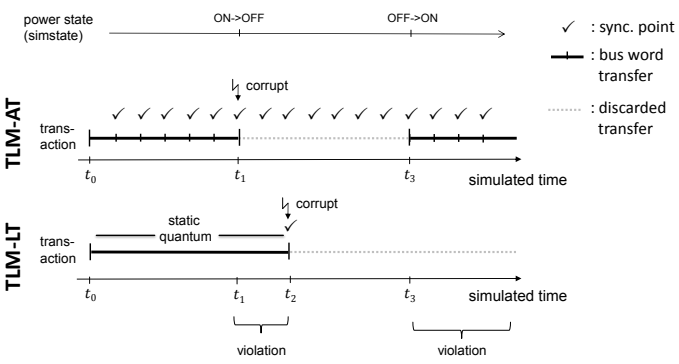

Figure 7. Power intent simulation within standard TLM

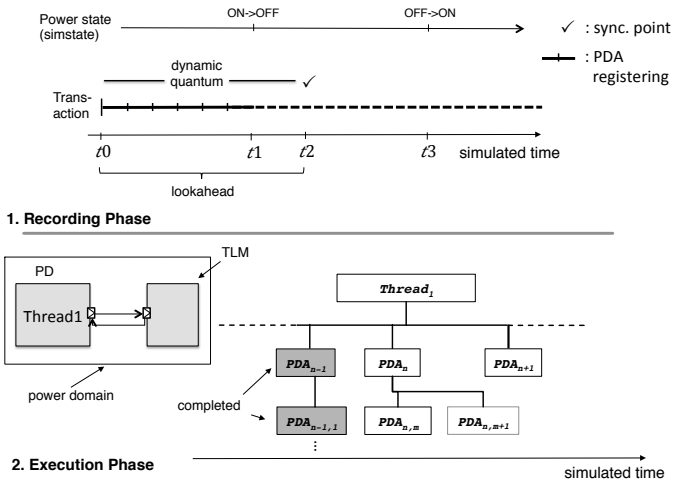

Figure 8. Power intent preserving using dynamic lookahead simulation

may break a transaction down in individual data transfer phases and implies lockstep synchronization at each phase. As a result, frequent process scheduling introduces a lot of context switching overhead. In contrast, TLM-LT uses transactions according to complete bus transfers and allows processes to run ahead of simulation time. By this way, TLM-LT is able to reduce the amount of context switches significantly. To ensure that processes synchronize, the SystemC TLM standard defines a static global time quantum which is the greatest amount of time that a process may differ from general simulation time. However, as shown in Figure 7 power state preservation can not be assured anymore. For this, TLM entities sending and receiving transaction data must be aware of ongoing power domain states. In other words, transactions have to be preempted and corrupted at right time, i.e. at time *t1*. If, for instance, a process reads from a location that is actually switched off, it will get a normal reply instead of the correct power down response. Obviously, this is not acceptable in terms of consistent functional simulation.

Recently, there exist some approaches for improving simulation accuracy together with TLM-LT. However, they focus on non-functional properties such as timing or energy. Basically, they apply resource [18, 19] or traffic models [20, 21] and postpone results to quantum boundaries. In the end, actual transactions are still executed in a non-preemptive manner, i.e., either optimistic at start or in the end. Hence, functional violations based on inaccurate power state assumptions still remain. A comparable approach has been presented in [22], however, it seems to work only in their use case, as they presume detailed knowledge and full access to individual TLM components. In contrast, our synchronization approach works also with general TLM IPs.

B. Power-Aware Synchronization Layer

Since power switching is hardly predictable in advance, today's virtual prototypes would result in SystemC processes that have to synchronize at each single message. In the end, TLM-LT simulation performance will degrade down to the level of TLM-AT. To solve this issue, we try to look ahead in reliable boundaries to determine future power state changes. As in TLM-LT, the main idea is to reduce the number of yield operations and hence the amount of SystemC scheduling. However, instead of fixed time slices, application dependent

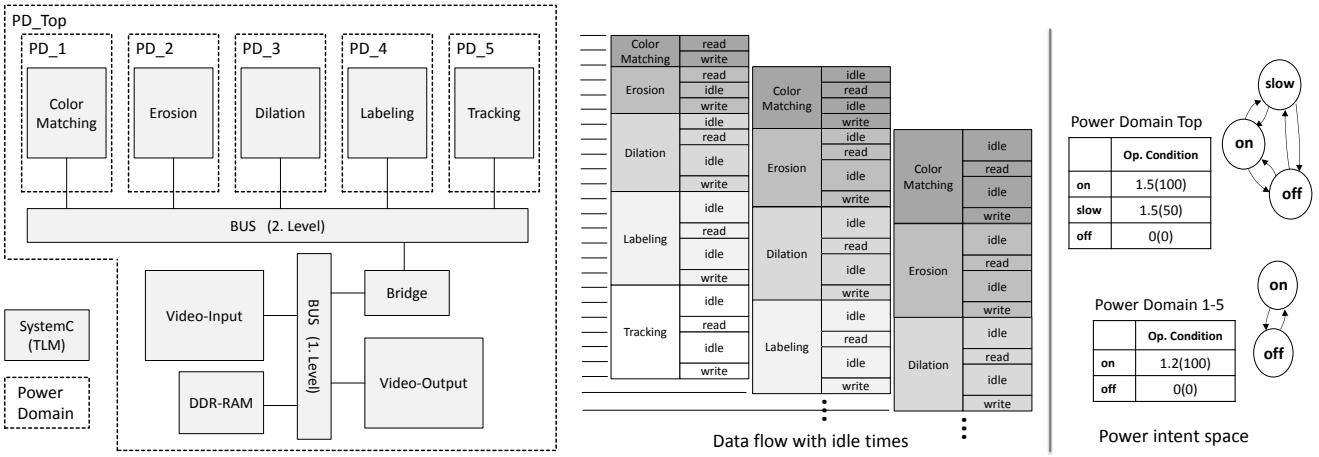

Figure 6. Embedded video processor SoC (TLM design + power intent)

synchronization is used. Existing TLM-LT models integrate so called quantum-keepers by default. It is a utility class that performs synchronization if local process times exceed the firm global quantum boundary. Nevertheless, a quantum-keeper by itself is not applicable for our variable synchronization needs. Instead, we accomplish power-aware synchronization by an additional layer of abstraction. As shown in Figure 8, it is built up on two alternating phases during normal process simulation.

In the first phase, the recording phase, TLM components simulate in their original temporal decoupled manner. Thus, all SystemC processes issue transaction messages as usual. Instead of real execution, transactions are only registered in terms of power domain accesses (PDAs) and are returned immediately. After all TLM initiators have reached a point where synchronization is explicitly demanded, the protocol switches to a subsequent execution phase. For this, temporal decoupling stops in two cases. First, if there might be data dependencies within a process' execution trace, i.e., read before write scenarios; second, if there are explicit synchronization demands, i.e., upper lookahead boundaries.

In the execution phase, power domains start to process the registered PDAs. At first, a power domain checks for any power change requests it has received in the previous recording phase. If nothing is found, transactions are completed without further changes. Otherwise, transactions are executed step-wise until the next power state change. This is mandatory, since timing budgets as assumed in [22] are not guaranteed. However, we are able to benefit from given voltage transition delay constraints (as described in Section III). This procedure is repeated either until all transactions are executed or until an original quantum period is reached. Furthermore, if PDA's trigger additional transactions these are also registered and processed in the same way. Besides corruption, transactions may also be delayed or accelerated due to switching operating conditions. Depending on this updated time, next synchronization constraints are calculated and SystemC processes are rescheduled.

Compared to TLM, the proposed method requires only context switches where they are unavoidable, i.e. if data dependencies exist or if users wish to synchronize. In general, the algorithm is more efficient than explicit synchronization due to SystemC's context switching overhead. Finally, our synchronization scheme requires no kernel modification and no insides into the TLM components.

Implementation Details

In this subsection we explain some implementation principles in more detail. A requirement not explicitly stated before is that eventual existing quantum-keeper must be deactivated. For this, we apply the standard API to set a design's global quantum to its greatest potential value. This eliminates unwanted synchronization, and allows for adjustments which are completely transparent to the involved TLM components. However, since later execution still requires global quantum information the original value is buffered.

During a recording phase, all observed transaction messages are stored in tree structures. As illustrated in Figure 8, a power domain holds a structure for each process that provokes power domain accesses (PDAs). In detail, to register a PDA means an efficient shallow copy of forwarded payload pointers plus some metadata, which prevents expensive data movements as well as intensive memory wastage. However, as already mentioned, time stamps are not meaningful, since local process times are not updated. Finally, if there are no more pending SystemC events, i.e., no remaining SystemC process, the execution phase is started. In this phase, transactions are processed with regard to updated power state properties. The procedure starts with first PDA and performs rotating execution in conformance with original SystemC order. The step size itself is given by minimal power transition delay constraints. If this triggers further transactions and hence creates additional PDAs in the tree structure, these PDAs are immediately processed in depth-first search manner. When all transactions have been executed successfully, SystemC processes are waked up at right time and next process quantums are calculated by the original global quantum and new simulation time in mind. Afterwards, it restarts with next recording phase.

V. Experimental Results

We evaluated our approach by several differently abstracted TLM designs. We extended a SystemC-based system-on-chip (SoC) design [23] that implements real-time object tracking within video sequences and was systematically refined to FPGA, hence providing representative TLM abstraction levels down to RTL. The refinement starts from a very abstract specification model and results in different computation and communication granularity including complete image transfers (TLM-LT), bus

accurate arbitration phases (TLM-AT), as well as cycle-and pin accurate data word passing (TLM-CA). As a result, it applies TLM principles intensively which permits meaningful investigations on transaction based annotations.

A. Case Study

As shown in Figure 6, the case study consists of set of TLM components arranged in a two level bus hierarchy. Main components are video in- and output, color matching, erosion, dilation, labeling, as well as object tracking. Video input grabs images either from MPEG-2 file or a camera and if required converts them into YUV422 frames. Video output displays results on the workstation screen. Object segmentation itself is performed as a sequence of image processing steps. At first, each YUV frame is analyzed by color matching. It sets pixels that are out of a given color range to black and all other pixels to white. The resulting binary image is then processed by two morphological operations, i.e., erosion and dilation, to remove noise and interference. Afterwards, objects, which remain as cohesive pixel arrays, are labeled and archived into an analyzable region list. Finally, object tracking is performed by means of numerical features such as distance and size. In the given design, all pipeline stages have been connected to a bus and share external memory (DDR-RAM) to store their computed results. While this saves internal local memory, only one read or write transaction takes place at once. Moreover, running across fixed priority bus scheduling in combination with ascending pipeline priorities, it enforces additional idle times in the segmentation process (~1µs - 30µs). Having this in mind, additional power intent is constructed as follows. It is built up on six power domains as well as power management constraints like isolation, retention, and level-shifting. In detail, there is one top-level power domain for the entire system and five switchable subdomains, one for each pipeline stage. The top-level power domain is either in an *on*, *slow*, or *off* power state. Each of the subdomains is either *on* or *off*. Due to this, the power intent space spans a total number of 51 $(2 \cdot 2^5 + 1)$ potential power state combinations. Accordingly, as proposed in [17] the TLM design instantiates transaction delegates and registers callbacks from the power intent model.

B. Performance Evaluation

For performance analysis, first we added a power-management policy that exploits the existing idle periods as power down candidates. Furthermore, we applied functional stimuli (320x240 MPEG-2 images) to measure simulation overheads at each design stage, namely TLM-LT, TLM-AT, TLM-CA, and RTL. At TLM-LT, all active components proceed temporal decoupled and access the external memory via blocking transport calls. In one case, the global quantum matches image boundaries, in another case processes synchronize at each line transfer. At TLM-AT, communication has been refined to synchronized bus phases, i.e., request and data phase. At TLM-CA, also computation has been converted into state machines to shift the individual bus words cycle-accurately. Finally, RTL design is the result of an automatic SystemC synthesis step[3]. As comparison criteria, first we perform pure functional as well as power-aware simulation at RTL. For this, we applied a commercial simulator[4] and measured performance waste

[3]Celoxica's Agility compiler was used

[4]Questa Advanced Simulator (Mentor Graphics) with output-only corruption

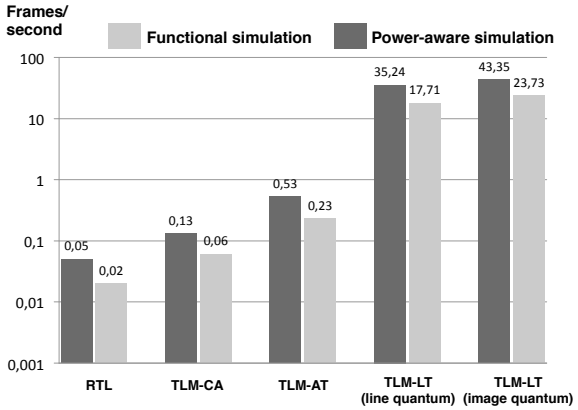

Figure 9. Simulation performance impacts due to power intent annotations (Note: There is equivalent power state accuracy at all levels)

of around a half due to additional power intent. Based on this experience, we tried to qualify our TLM-based power intent abstraction. As shown in Figure 9, overhead at TLM-CA complies with the one in power-aware RTL simulators, hence it should be an acceptable impact. Furthermore, we evaluated performance impacts on more abstract TLM scenarios, i.e., TLM-AT and TLM-LT. As one can see, we keep similar ratios. However, we maintain greater overall simulation speed while preserving the same power state simulation accuracy than at TLM-CA or RTL. Additionally, due to efficient look-ahead synchronization, we further benefit from longer quantum periods.

C. Validation Analysis

Traditionally, dynamic power intent verification relies on property checking while simulating RTL models. Therefore, commercial simulators provide built-in checks, but also understand standard assertion languages to assist the verification process. If enabled, proper guards are automatically inserted into the design and triggered by RTL signal changes. A typical example is as follows; some power-aware designs require that incoming signals do not toggle as long as power supply is down. In reality, non-gated clocks cause a great battery drain, so that eliminating unwanted clock toggling is an effective way to reduce power dissipation. For comparison, we developed a manageable TLM assertion subset and implemented proper power intent checks raising exceptions into our virtual prototyping front-end. For instance, the rule shown in Algorithm 1 helps in monitoring that a clock belonging to a specific power domain is gated as long as power supply is off. However, in contrast to explicit event-based assertion models, it must be immediately called in the program flow each time a power domain's operating condition is changed. In consequence, only properties at power state changes can be considered. For instance, verifying correct ordered isolation and state retention sequences based on control signal watching is out of scope. A survey of assertion categories covered by our framework

Algorithm 1 Clock Toggling Observation Rule

Require: Power domain $\mathcal{PD}$, triggering at each domain's operating condition change
Ensure: Raise exception in case of unwanted clock toggling

```
1: if voltage(PD) == 0 then
2:   if frequency(PD) != 0 then
3:     throw("Invalid clock toggling during power off in power domain" + name(PD))
4:   end if
5: end if
```

Table I
A SURVEY ON POWER INTENT VALIDATION SUPPORT

Category	Layer	Support
Protection (Iso./Level-Shifting)		
Missing/Incorrect	ESL	Yes
Powered	ESL	Yes
Enabled/Disabled	ESL	Yes
Race/Functionality	ESL/RTL	–
Clamp Value/Toggle	RTL	–
Retention		
Powered	ESL	Yes
Enabled/Disabled	ESL	Yes
Level/Toggle	RTL	–
Power Domain/States		
Power domain ordering	ESL	Yes
Mode/State observation	ESL	Yes
Control signal sequencing	RTL	–

is shown in table I. Hereby, we further distinguish ESL and implementation dependent (RTL) categories as latter require information either not modeled in TLM specifications or not accessible from interfaces of components. As a result, our simulation front-end covers up to 70% of the overall amount of power intent checks implemented in Mentor Graphic's Questasim and, hence, helps in reducing verification efforts at later design stages.

VI. CONCLUSION

It is commonly agreed that today's low-power design decisions can only be managed from a system-level point of view. To address this issue, we developed a virtual prototyping methodology by modeling and simulating common power intent semantic in combination with early available SystemC TLM specifications. Hence, we bridge an existing ESL design gap to validate low-power design decisions in an abstract but functional correct manner. At first, we presented low-power abstraction concepts which can be seen as system-level extensions for recent low-power design and verification standards. For this, we aligned standard UPF concepts such as power domains and power states to TLM . Second, we proposed a novel simulation approach considering both complex hard IP adoption and free choice of TLM coding style. Moreover, by using look-ahead synchronization, we obtained cycle-accurate power state simulation in spite of temporal decoupled process scheduling. Finally, we demonstrated the usability on a representative TLM case study. For quantification, we compared performance impacts at different TLM abstraction levels as well as validation goodness against state-of-the art solutions. Our experimental results have shown that a simulation overhead at cycle-accurate TLM is comparable with the overhead in available RTL simulators. At higher abstraction levels, however, simulation performances will benefit significantly, while the power intent semantic is 100% preserved. Finally, up to 70% of architectural property checks derived from a state-of-the-art RTL reference simulator could be incorporated.

ACKNOWLEDGMENTS

This work was partly funded by the German Ministry of Education and Research (BMBF) through the projects EffektiV (01S13022) and ARAMiS (01IS11035). We greatly appreciate the cooperation with the project partners.

REFERENCES

[1] M. Keating *et al.*, *Low Power Methodology Manual: For System-on-Chip Design*. Springer, 2008.
[2] S. Jadcherla *et al.*, *Verification Methodology Manual for Low Power*. Synopsys Inc. and ARM Ltd., 2009.
[3] IEEE Std 1666™-2011, "Standard SystemC Language Reference Manual", 2012.
[4] IEEE Std 1801™-2013, "Standard for Design and Verification of Low Power Integrated Circuits", 2013.
[5] Silicon Integration Initiative (Si2), *Common Power Format, Version 2.0*, 2011.
[6] F. Bembaron *et al.*, "Low Power Verification Methodology Using UPF", in *Design & Verification Conference & Exhibition (DVCon)*, 2009.
[7] A. Crone and G. Chidolue, "Functional Verification of Low Power Designs at RTL", in *PATMOS*, 2007.
[8] R. Mukherjee *et al.*, "Static and Formal Verification of Power Aware Designs at the RTL Using UPF", in *DVCon*, 2008.
[9] A. Hazra *et al.*, "Formal Verification of Architectural Power Intent", *IEEE Transactions on Very Large Scale Integration Systems*, 2013.
[10] Jasper Design Automation, "Formal Verification of Power-Aware Designs Using the JasperGold Low-Power Verification App", Tech. Rep., 2013.
[11] G. S. Silveira *et al.*, "Open SystemC Simulator with Support for Power Gating Design", in *International Journal of Reconfigurable Computing*, 2012.
[12] C. Trummer *et al.*, "Automated Simulation-based Verification of Power Requirements for Systems-on-Chips", in *International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2010.
[13] G. Delp *et al.*, "Understanding the Low Power Abstraction", in *DVCon*, 2010.
[14] A. Nohl, "Increase the battery mileage with virtual prototypes", Synopsys, Ed., 2011. [Online]. Available: http://blogs.synopsys.com/viewfromtop/page/2/.
[15] O. Mbarek *et al.*, "Using unified power format standard concepts for power-aware design and verification of systems-onchip at transaction level", *IET Circuits, Devices & Systems*, 2012.
[16] F. Mischkalla and W. Mueller, "Efficient power intent validation using loosely-timed simulation models", in *PATMOS*, 2013.
[17] F. Mischkalla and W. Mueller, "Architectural Low-Power Design Using Transaction-Based System Modeling And Simulation", in *Int. Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2014.
[18] S. Stattelmann *et al.*, "Fast and accurate resource conflict simulation for performance analysis of multi-core systems", in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, 2011.
[19] W. Ecker *et al.*, "TLM+ Modeling of Embedded HW/SW Systems", in *DATE*, 2010.
[20] T. Bouhadiba *et al.*, "System-Level Modeling of Energy in TLM for Early Validation of Power and Thermal Management", in *DATE*, 2013.
[21] D. Greaves and M. Yasin, "TLM POWER3: Power Estimation Methodology for SystemC TLM 2.0", in *Forum on specification & Design Languages (FDL)*, 2012.
[22] S. Simon Hufnagel *et al.*, "Advanced Temporal Decoupling", in *European SystemC Users' Group (ESCUG) Workshop*, 2013.
[23] GreenSoCs, "Embedded Video Detection (EmViD)". [Online]. Available: http://forge.greensocs.com/de/projects/GreenBench/EmViD.