A Power Estimation Methodology for SystemC Transaction Level Models

Nagu Dhanwada
IBM Electronic Design Automation,
Systems and Technology Group,
Hopewell Junction, NY
nagu@us.ibm.com

Ing-Chao Lin
Dept of Computer Science,
Pennsylvania State University,
University Park, PA
ilin@cse.psu.edu

Vijay Narayanan
Dept of Computer Science,
Pennsylvania State University,
University Park, PA
vijay@cse.psu.edu

ABSTRACT

Majority of existing works on system level power estimation have focused on the processor, while there are very few that address power consumption of peripherals in a SoC. With the presence of complex cores in current day embedded system-on-chip devices, the problem of complete system level power estimation is gaining significance. Transaction level models for SoCs are gaining increasing attention with emerging architectural modeling standards like SystemC. In this paper we present a methodology for performing system power estimation for different scenarios or applications being executed on these transaction level models. We describe techniques and a setup for transaction level power characterization, and an approach to augment SystemC transaction level models to perform transaction level power estimation. We also present experimental results to validate the accuracy and speed of our approach.

Categories and Subject Descriptors: *J.6 [**Computer-aided design**]; B.7.2 [**Integrated circuits**]: Design I.6.5 [**Simulation and Modeling**]: Model Development;*

General Terms: Performance, Design, Experimentation

Keywords: SystemC, Transaction Level Models, Power Analysis, CoreConnect, PowerPC

1. INTRODUCTION

Transaction level models and simulation platforms composed of such models for IP cores are increasingly being used for SoC architecture analysis and early embedded software development. These are gaining more relevance with emerging standard architecture modeling languages like SystemC. Power consumption has emerged as an important design metric for electronic systems. Increasing design complexity is making it imperative to address power consumption at the system-level, where the benefit of performing power optimizing design changes is the greatest.

Consider the following scenario to serve as a motivation for peripheral power estimation in a system context. Let's take the case of voltage and frequency scaling algorithms executing on a wireless system-on-chip, to save dynamic power. Lets assume that the CPU is bandwidth limited and the DVFS algorithm suggests increasing clock frequency and voltage for the CPU – to recover performance (improve the bandwidth). In reality, this bandwidth saturation can be because of a peripheral that is getting some wireless data or transferring files over the network. So any increase in CPU frequency and voltage suggested by the DVFS algorithm would not improve system performance because the bottleneck is the peripheral. Therefore, in order to comprehensively evaluate the root cause of power-performance tradeoffs in a SoC, there is a clear need for having comprehensive power models for peripherals that go along with performance/functional simulation models.

SystemC and Transaction Level Modeling (TLM) [4] has been gaining significant traction and IP core providers are beginning to provide such models for the purpose of embedded software development and early architecture analysis. These models are typically at a level where they do not capture all power related aspects of the cores in order to optimize the simulation performance. To enable power estimation for a system composed of such transaction level models, we propose to incorporate power estimation techniques into a SystemC functional model designed to run embedded software. One of the basic assumptions in earlier works on instruction based estimation, was that the core performance models/functional models are developed after power related transactions or instructions have been identified. This meant that all of the power related transactions/instructions in a core are used in the SystemC functional model, thus considering a one - one mapping between all instructions identified from the detailed model and its corresponding systemC model.

In this paper, we address the problem of system level power estimation with transaction level models (TLMs), focusing on the case where the TLMs would not have been originally developed with the objective of power estimation in mind. We use an industry strength PowerPC/CoreConnect TLM simulation platform as a basis for our work. We present in detail the characterization methodology adopted for generating power models that can be used within a transaction level simulation, and a simple approach for augmenting TLMs with these transaction level power models. The characterization methodology itself has been dealt cursorily in most of the other works on system level estimation. One of the areas we focus in this paper is the setting up of a characterization methodology that considers all aspects of a detailed model in the process of generating an abstract transaction level power model.

The paper is structured as follows: the next section gives an overview of related work, and highlights our contributions. Section 3 presents an overview of the transaction level simulation platform – and discusses the organization of transaction level models in systemC. Section 4, presents the overall flow of our transaction level power modeling and estimation approach with an example. Sections 5 and 6, describe the power characterization approach and our system level transaction based power estimation technique. Finally section 7, presents experimental results followed by a section that discusses conclusions and future work.

2. RELATED WORK

Work on power estimation, modeling and optimization has been done at different levels of abstraction. In this section we limit the discussion to mainly some of the system level approaches to power estimation. Instruction based power analysis for peripheral cores were first presented in [4]. This work presented a core power evaluation technique that divided the function of the cores into instructions and performed estimation using instruction level power models. The assumption made in this approach was that the core performance simulation model would be developed based on the instructions that were identified during the characterization process. In contrast, the work being presented in this paper considers the case where there are existing legacy performance or architecture analysis models (and does not assume that the architecture models are developed based on pre-defined power related instructions) and proposes an approach for power characterization and augmenting these models to permit system level power estimation. Our work also presents a detailed transaction level analysis-friendly characterization method and data organization structures for use with SystemC transaction level models. Integration of power models for certain components of the AMBA AHB bus into a transaction-level modeling framework was explored in [2].

Some of the other works on high-level estimation include [5], where a function based power estimation method was presented for embedded software executing on microprocessors. A state based power analysis method for SoCs was presented in [6] where the interactions between various coarse grained core-states like idle, active, sleep were considered. This would not be adequate for a transaction level simulation paradigm where, a finer level of granularity would clearly be required. Another recent effort [11] develops a technique for power estimation from cycle-accurate functional descriptions. Their approach identifies the correlation between the cycle accurate functional description and the corresponding RTL implementation for power estimation. Our approach is similar in that we also create correlations, but we operate at a higher transaction boundary accurate level of abstraction as opposed to cycle accurate descriptions. To the best of the authors' knowledge this is one of the first works to address in detail the aspect of power characterization for SystemC transaction level models, and interfacing power characterization models with SystemC TLMs.

3. TRANSACTION LEVEL SIMULATION for POWERPC/CORECONNECT SYSTEMS

Transaction level modeling is a high- level approach to modeling digital systems where details of inter-module communication are separated from details of communication architecture implementation [10]. In this paradigm, a transaction is basically a mechanism for communicating between cores. E.g., bus_write(), interrupt_request(), etc. Correct amount of time is associated with each atomic transaction and added up to the total simulation time (faster than ticking at every clock). This makes it efficient, enabling execution of real software (in an embedded system) on models written at this level of abstraction.

The organization of a SystemC transaction level simulation model for a CoreConnect based system is shown in Figure 1. This is used as a platform for the techniques presented in this work.

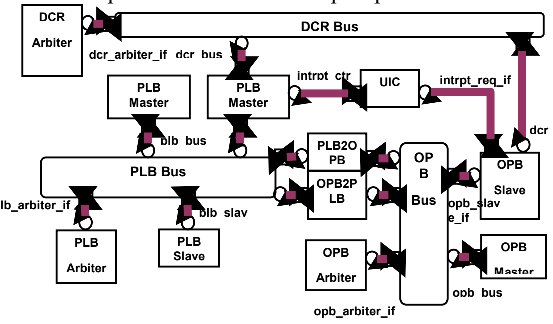

Figure 1: CoreConnect TL Simulation Platform

Elements of the CoreConnect architecture include the processor local bus (PLB), on-chip peripheral bus (OPB), a bus bridge, and a device control register (DCR) bus. High-performance peripherals connect to the high-bandwidth, low-latency PLB. Slower peripheral cores connect to the OPB, which reduces traffic on the PLB, resulting in greater overall system performance [7]. The busses in the CoreConnect complex are represented as hierarchical channels, and the masters and slaves are SystemC modules. Basic communication mechanism between SystemC modules is through ports that are bound to interfaces. Referring to the PLB subsystem in the Figure 1, a PLB master has ports of interface type plb_bus_if, through which transactions are issued to the bus. These requests are sent to the arbiter through an arbiter port that is present in the PLB bus, after arbitration when the bus is granted to the requesting master, the PLB slave port present on the bus (which is bound to the PLB slave interface), is used to issue the request to the addressed slave.

In this work, we use a transaction level model based simulation platform consisting of PowerPC processor, CoreConnect bus complex and peripheral models. These are at a level of abstraction equivalent to the Programmer's view with timing (PV+T) level as described in [9]. This is the highest level of abstraction which can include some amount of micro-architectural detail. The bus models are bus cycle accurate, with respect to transaction clock boundaries. Transactions are modeled as occurring over communication channels. The computation (inside a core) may not be modeled on a cycle-by-cycle basis, as long as the input-output delays are cycle-approximate. The transactions occurring within the system can be blocking or non-blocking. A transaction that is executing sequentially after another and needs to wait till the first transaction is complete before starting other is a blocking transaction. One that is executing concurrently with another is referred to as a non-blocking transaction. In this work here, all the communication within the TLM platform use blocking transactions.

4. OVERALL IDEA

Typical TLMs capture the functional tasks associated with the behavior of an IP Core, but would not necessarily contain a lot of the non-functional tasks related to the core. These non-functional tasks would be quite important from a power consumption point of view. Since TLMs as defined in the above section, are not developed with the view of capturing all power related tasks, this leads to a unique problem of having to map from the set of tasks (functional and non-functional) identified during the transaction level power characterization method to the set of tasks present in the current transaction level model. This is one of the unique features of the transaction level power modeling approach being explored in this work.

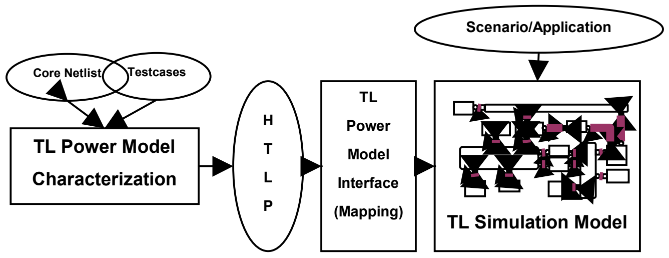

Figure 2: Overall TL Power Estimation Method

Figure 2, shows the overall flow of the transaction level power modeling flow being presented in this paper. The two main portions of this flow are the TL power characterization module, which generates a hierarchical transaction level power (HTLP) tree structure which captures transaction level power information for a particular core. This information is used by the mapping or power model interface portion of the flow in augmenting the SystemC TLM simulation platform with power information.

The impact of granularity of characterization on system level estimation is also explored in this work. Results of the characterization process is structured hierarchically – the lowest level being the address and data phases of an individual transaction, the highest being compositions of various transactions. We define a Hierarchical Transaction Level Power model tree structure (HTLP-tree) to organize the transaction level power characterization data. This structure is used by the SystemC power estimation module (through the mapping/power model interface) in conjunction with the transaction level simulation platform to generate system level power estimates for the scenarios being executed.

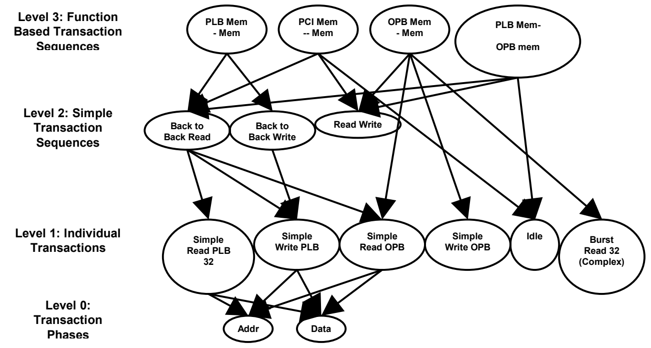

Figure 3: Sample HTLP-tree structure

Figure 3, shows the overall structure of the HTLP-tree structure. It is organized into four different levels. As we move from the lower levels to higher levels, the granularity increases helping in improved simulation speed at the cost of reduced accuracy. In this structure, the nodes represent a power model that can be used within the transaction level simulation platform, and the edges denote containment relationships between the nodes. Level 1, is the individual transaction level – where each node represents a simple transaction. These transactions have an attribute associated with them that indicates if a particular transaction is a primary one or a secondary one. A primary transaction is one that can be directly called from the transaction level simulation platform – i.e it can be seen as a state that is captured within the systemC TLM. A secondary transaction is one which does not have a direct equivalent in the SystemC TLM.

The transactions in Level 1 are reads, write, initialize, burst mode transactions which when used in a simple sequence like a back to back read, and a back to back write comprise Level 2 of this tree. The individual transactions in Level 1 can be simple or complex transactions. A complex transaction like a burst read or write consists of a set of simple individual transactions. The complex transactions are attributed to indicate whether the pattern of simple transactions that make up the complex transaction are either overlapping or non-overlapping. The non-overlapping attribute is useful for capturing communication paths that represent a single point of control. The overlapping attribute is used to represent cases where there can be multiple points of control, like a complex bus structure that is modeled by multiple finite-state machines controlling multiple communication paths.

At the highest level of the structure are function based transaction sequences which denote a more complex sequence of level 2 or

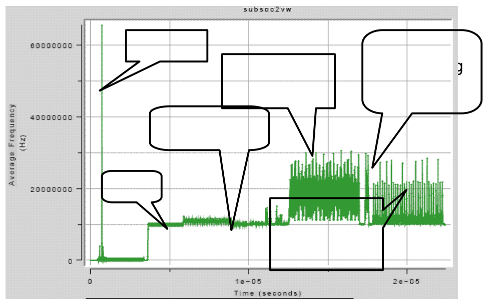

Figure 4: Power Simulation:DMA PLB to PLB

level 1 transactions that indicate a particular operation of the core. These can be a scatter-gather operation of a DMA controller, or an OPB/PLB memory-memory transfer. At the lowest level are the individual phases of a transaction – the address and data phases.

The reasons for such a hierarchical organization of the TL characterization data can be explained using the power simulation data for a high level test – that consists of a DMA memory to memory transfer through the PLB. This power simulation data is shown in Figure 3. If we are observing the entire time interval for which this high level transaction sequence is executing – and represent this entire sequence as a transaction then we can clearly see that the variation in power numbers can be quite high. This would correspond to a level 4 or even a higher node in the HTLP, where clearly the accuracy of a transaction level model using such a high level transaction sequence would clearly be dependent on the point of time within this transaction sequence when the overall power is being estimated. So, if we delve down into a particular portion of this transaction sequence, and identify individual transactions then, the variation can be contained quite a bit, which is why we even organize the HTLP-tree to contain even lowest level nodes that represent individual phases of even a single transaction. Power models are associated with each of the nodes in the HTLP-tree. The tree can also exist as a set of disjoint nodes at any particular level – i.e. there can just be level 3 of a tree for which the characterization process generates power models. The key requirement is that the SystemC TLM simulation platform should be able to make use of the available levels of the HTLP in coming up with power estimates. In this work, we have augmented the TLMs with power models at level 1 of the HTLP – with the consideration of whether the instruction is primary or secondary.

The overall methodology for transaction level power estimation can be briefly described as follows:

- Identify transactions or instructions from the core description,
- Characterize power consumption of each task or instruction from low-level implementation,
  - Generate vectors corresponding to these instructions or tasks executing on the IP core
  - Place, route and extract parasitics for these cores,
  - Use power simulation tools with the parasitics, to generate power characterization information
- Create macromodels based on various IP core parameters:
  - Parameters can be bit-width, switching activity of data, buffer size
- Augment the TLM to extract the parameters for macro-model during the transaction level simulation step and make calls to the appropriate power macro-models, thus deriving energy measures for each of the cores for that particular simulation.
  - This can be done dynamically at run-time to derive information during simulation (tradeoff between simulation accuracy and speed should be taken into account)

In the next sections, we describe the individual constituents of the power characterization methodology – the characterization and SystemC TL power estimation that uses the power model interface/mapping function.

5. POWER CHARACTERIZATION

In this section the transaction level power characterization

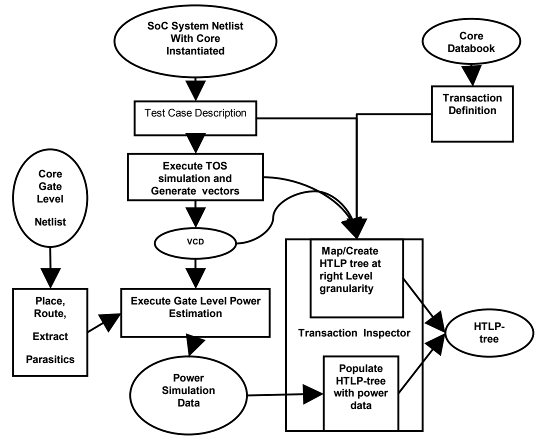
SoC System Netlist With Core Instantiated
Core Databook
Test Case Description
Transaction Definition
Execute TOS simulation and Generate vectors
Core Gate Level Netlist
VCD
Map/Create HTLP tree at right Level granularity
Place, Route, Extract Parasitics
Execute Gate Level Power Estimation
Transaction Inspector
HTLP-tree
Power Simulation Data
Populate HTLP-tree with power data

Figure 5: Power Characterization: HTLP-tree

methodology is described. This is a key component of the overall transaction level power estimation approach. In the earlier section we described the HTLP-tree structure to capture the power characterization information. The power characterization methodology described in this section, generates this structure. In the rest of this section, we describe the details of the characterization methodology being proposed in this paper. Figure 5, shows the overall power characterization approach.

The characterization process for a peripheral core has to be done considering the spatial context of the system in which it is present, which is why we use a pre-composed gate level SoC sub-system with the core instantiated as an input to the characterization process. In addition to the netlist, we use the gate level netlist of the core to obtain parasitic data – by executing the steps of placement, routing and capacitance extraction. This information is used during the gate level power simulation. In our setup we have a basic clock tree constructed till the splitter level for the core being characterized. The accuracy of the process can be increased by constructing a complete clock tree and using it in the characterization process. In the rest of this section, we describe each of the steps of the characterization approach shown in Figure 5 with the help of a DMA controller core as an example. The characterization process uses TOS, which is described below.

Test Operating System (TOS)

TOS [8] is an operating system designed to be executed on an embedded processor in a system-on-a-chip integrating several different hardware cores. TOS can exercise a number of core-specific test cases to verify the connectivity and interaction among the cores on the chip. It basically is a software based method for verifying SOC designs. TOS controls a digital simulator to complete groups of testcases in order to verify system functionality. TOS provides for both system concurrency and interconnect testing. TOS testcases use real Core IP functions to exercise system (Ethernet packets, DMA transfers and the like) functionality. Many test cases are created as part of the TOS environment for the chip under test. Each is designed and written for the particular core it is to test, and is divided into two distinct parts: the application (or task) and the device driver. TOS testing can find bugs when a core is used in a system in an unanticipated way. TOS execution is intended to emulate chip usage in a system environment.

First step is to write test cases (application) in the test operating system that would exercise various features of the core in the context of the system within which it is present. This involves exercising the core with different system level tests that exercise

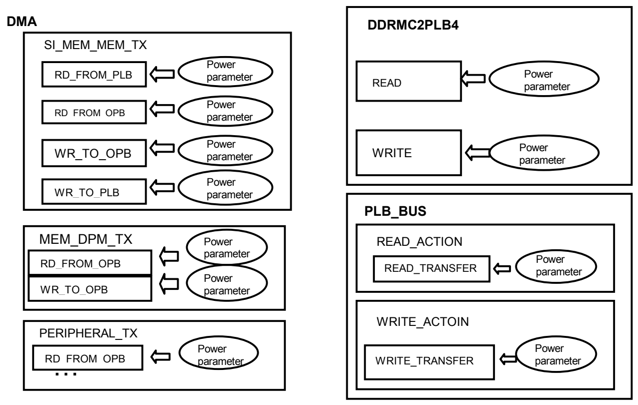

Figure 6: Power Model calls in a TLM

the core under different scenarios; these test cases are executed by setting the core to operate with different parameter settings. For example, in a DMA controller like core these can be: Interrupt Enable, Transfer type (Memory-Memory, Peripheral-Memory, etc), Destination data width (Byte, Half-Word, Word), Buffer Enable, Channel priority, Source location (PLB or OPB memory space), Destination location (PLB or OPB memory space), Prefetch enabled, Terminal count enable. In this step, a set of tests are written with different parameter settings. These tests involve software initiated memory to memory on different buses (PLB, OPB), device paced transfers, scatter gather transfers. These testcases can be decomposed into a series of transactions, which are defined from the databook. The next step is determining the structure of the HTLP tree, whether to maintain characterization data at the highest level – subsets of tests that would be carried forward to be used in the power estimation.
Once the structure is setup, the test operating system is used to execute the testcases and VCD (Value Change Dump) files are generated to be used in the power simulation. The resulting power simulation data in conjunction with the HTLP tree structure are used by the transaction inspector module to derive average power numbers, corresponding to each node in the HTLP tree. The transaction inspector module, serves two important purposes, the first is setting up of the initial HTLP-tree at the right level of granularity.

Instruction	Dyna (mW)	Total (mW)
READ (P)	32.9	41.18
WRITE (P)	28.9	37.18
BURST READ(P)	51.61	59.89
BURST WRITE (P)	46.42	54.7
Initialization (P)	29	37.28
Self Refreshing (S)	28.46	36.74
Charge all Bank (S)	28.77	37.05
Standby (S)	28.82	37.1

For this step, information from the test operating system in terms of transaction boundary points, vcd files, and testcase descriptions are used in defining the initial HTLP-tree. The parasitics from the place and route step are fed to the gate level power estimation tool that generates the power numbers. The second purpose of the transaction inspector is to populate the HTLP-tree structure with the power data derived from the gate level power simulation. The above table lists the set of instructions for the ddr memory controller; the ones tagged with P represent primary instructions and the S represent secondary instructions.

6. TL POWER ESTIMATION – MAPPING

To perform power estimation during SystemC simulation, we integrate the HTLP-tree based power data into the SystemC TLM based simulation environment. Transaction level power model calls are directly inserted into the appropriate functions of the SystemC descriptions. The HTLP-tree structure is used along with the function structures defined in the SystemC TLM. The primary instructions in level 1 would correspond to the bus API calls of the master/slave/arbiter components and would involve a call to the corresponding primary instruction power models.
In the DMA controller TLM, "dma_transfer" function is used to handle the actual transfer performed by the controller. Four kinds of operations are supported in TLM model: software_initialized memory_to_memory transfer, memory_and_device_paced, and peripheral_transfer.
For software initialization memory_to_memory, four kinds of primary instruction can be executed in this transfer: read from plb mem space, write to plb mem space, read from opb mem space, and write to opb mem space. The power estimation parameter is derived from HTLP-tree and is inserted into these TLM functions. The instruction to set up DMA controller channel is through DCR bus and is considered as secondary instruction. By using power parameter from HTLP, energy consumed by a specific transfer can be calculated. Since a core is not always active throughout the testcase execution, and its power consumption is lower when is in idle. To improve the accuracy of SystemC TLM power estimation, we take each core's idle time into consideration. Figure 7, shows the calculation of a core's idle time which is used in addition to the transaction based power models to improve the accuracy of the overall energy estimates. During simulation, active time for each core is registered with a system level monitor.
The maximum active time among these cores involved in the simulation of a scenario is calculated. A core's idle time is computed to be the difference between the overall execution time for a scenario and the core's active time, this idle time number is used to augment the core power model in addition to the transaction level power model instantiations. The following equation shows the total energy computation function

$$Energy = duration_{idle} * Power_{idle} + \sum Power_{transaction} * duration_{transaction}$$

And the average power, $Power = \frac{Energy}{duration}$

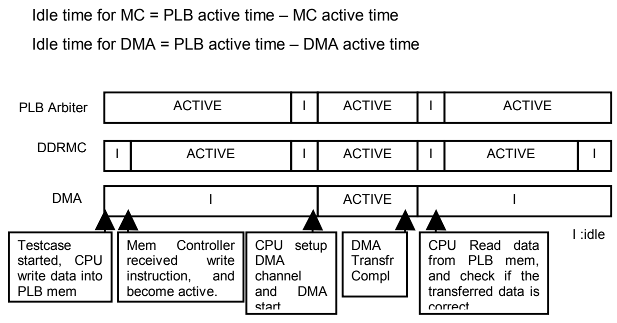
Idle time for MC = PLB active time – MC active time
Idle time for DMA = PLB active time – DMA active time
PLB Arbiter
ACTIVE
I
ACTIVE
I
ACTIVE
DDRMC
I
ACTIVE
I
ACTIVE
I
ACTIVE
I
DMA
I
ACTIVE
I
I :idle
Testcase started, CPU write data into PLB mem
Mem Controller received write instruction, and become active.
CPU setup DMA channel and DMA start
DMA Transfr Compl
CPU Read data from PLB mem, and check if the transferred data is correct

Figure 7: Idle time calculation

7. EXPERIMENTAL RESULTS

In order to analyze the effectiveness of the proposed techniques, we compared the accuracy and efficiency of the transaction level power estimation technique relative to a base case that is a gate level representation of the same SoC sub-system. Since TLMs are used for early exploration of design space, we consider various scenarios of transactions in a TLM to get an idea of their power consumption behavior. Our work is also applicable to other platforms than those considered in this paper. However, the complexity of running gate-level simulation for validation purposes for other entire platforms such as the system considered here is too time consuming to include in this work. A SoC sub-system consisting of a PLB, OPB, EBCO, PLB2OPB Bridge, DDRMC, Memory, and DMA was used. Several scenarios representing different application behaviors were executed on the SoC sub-system. The scenarios were described using a generic processor model that acts as a traffic generator for the rest of the cores on the sub-system, the total energy for each of the scenarios was collected from the power-model augmented systemC TLM simulation. The average energy numbers for the TLM platform represent the energy estimate of the sub-system represented by all the cores. The base case against which the comparison was made consists of a gate-level representation of the same SoC sub-system. The scenarios that were executed by the generic processor model on the TLM platform were executed on this gate level version of the platform using the Test Operating System setup, which converts these high level test cases representing the scenarios into vectors that drive the gate level model of the sub-system. The rows in Table 1 show the different scenarios, Column 2 shows the power estimate for the gate level model and the following column shows the run-time (in minutes) on a 1GHz dual CPU Linux workstation.

	GLM Power (mw)	GLM Run time (min)	TLM Power (mw)	TLM Run time (min)	Error
Scenario-1	57.89	22.3	56.145	0.005	-3.01%
Scenario-2	58.74	25.4	56.1194	0.01	-4.46%
Scenario-3	58.595	26.8	57.071	0.02	-2.6%
Scenario-4	22.744	35	21.975	0.02	-3.38%
Scenario-5	57	45	63.35	1	11.19%

Scenario1: PLB TO PLB: This scenario represents a software initiated memory-to-memory transfer on the PLB bus. The three main steps are 1) CPU writes data to PLB address A. 2) DMA transfers data from PLB address A to B. The number of 128 bit transfers is 32. 3) CPU reads data from PLB address B and compares data with original data. **Scenario 2: PLB TO OPB:** This is a software initiated memory-to-memory transfer between the PLB, OPB memories. OPB memory is connected to OPB BUS through External Bus Controller (EBCO). The PLBOPBBRIDGE, OPB and EBC are all required to be active in order to read or write OPB memory. This involves the following steps: 1) original data to PLB address A 2) DMA transfer data from PLB address A to OPB address B . The number of 32 bit transfers is 32, and can be adjusted if needed. 3) CPU reads data from OPB address B and compares it with original data. **Scenario 3: EBC (OPB) TO PLB** The transfer is from the memory on the OPB bus to the memory on the PLB bus. CPU writes data to OPB address A DMA transfer data from OPB address A to PLB address B. The number of 32 bit transfers is 32, and Third, CPU reads data from PLB address B and compares with original. **Scenario 4: EBC (OPB) TO EBC (OPB)**: test performs a software initiated memory-to-memory transfer between memories on the OPB. CPU first writes data to OPB address A DMA transfer data from OPB address A to B The number of 32 bit transfers is 32. Third, CPU reads data from OPB address B and compares with original.

Scenarios 1 through 4 are cases where there is a high degree of correlation between the transaction level power models used and the transactions generated by the scenarios. The scenarios are simple transaction sequences generated by the cores involved with inter-transaction limited inter-transaction interaction. Although the overall system average power number error margin is considerably low, the individual core average power estimates are slightly higher. For example, in Scenario-1 the difference between the gate level and transaction level core energy estimates for PLB, DMA, and DDRMC s is 13%, -20%, and 4% respectively. The reason for the DMA controller showing a high difference is because certain mode of operation of the controller (pre-fetch buffer enabled) was not captured in the HTLP-tree, which was being used by the scenario being executed. The individual core average power data when combined together with idle times gives acceptable accuracy for the estimation. Scenario – 5 is a more complex example with interleaved DMA transfer operations that give a higher error than the rest of the test cases because of the potential interaction between transactions.

8. CONCLUSION

In this paper, we presented a methodology for power estimation of SystemC transaction level models. The main constituents of this include a power characterization approach, a hierarchical representation of transaction level data, and a power model interface/mapping mechanism to augment TL simulation models with power information. These techniques were implemented for IBM CoreConnect based architectures. The experimental results demonstrate validity of the approach, providing a starting point for further exploration of transaction level power estimation.

9. ACKNOWLEDGMENTS

Authors would like to acknowledge Reinaldo Bergamaschi, William Dungan, William Dougherty and Paul Sulva from IBM Corporation for their suggestions.

10. REFERENCES

[1] R. A. Bergamaschi, Y. Shin, N. Dhanwada, S. Bhattacharya, W E. Dougherty, I. Nair, J. Darringer, S. Paliwal "*SEAS: A System for Early Analysis of SoCs*", Proc. CODES/ISSS 2003

[2] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi, and C. Turchetti, "System-Level Power Analysis Methodology Applied to the AMBA AHB Bus", DATE 2003.

[3] "The Open SystemC Initiative." http://www.systemc.org.

[4] T.D. Givargis, F. Vahid, and J. Henkel, "Instruction-based system-level power evaluation of system-on-a-chip peripherals", Proc. Of 13th ISSS.

[5] G. Qu, N. Kawabe, K, Usame, M. Potkonjak, Function-level power estimation methodology for microprocessors, Proceedings of the 37th DAC, 2000.

[6] R. Bergamaschi, Y.W. Jiang, "State-based power analysis for systems-on-chip", Proceedings of the 40th Design Automation Conference, 2003.

[7] "The CoreConnect Bus Architecture", available at http://www.chips.ibm.com/products/coreconnect.

[8] R. Devins, "SoC Verification Software – Test Operating System", IEEE/DATC Electronic Design Processes Workshop, April 2001

[9] M. Burton, A. Donlin, "Transaction-Level Modeling: Above RTL Design and Methodology", http://www.systemc.org.

[10] T. Grötker, S. Liao, G. Martin, S. Swan, "System Design with SystemC*",* Kluwer Academic Publishers, 2002.

[11] L. Zhong, S. Ravi, A. Raghunathan, N. K. Jha."Power Estimation for Cycle-Accurate Functional Descriptions of Hardware", ICCAD 2004.