Power Estimation Using Transaction Level Modeling

by

Nidhi Chandoke

Under the Supervision of
Dr. Sujay Deb
Mr. Ashish Kumar Sharma

Submitted
in partial fulfillment of the requirements for the degree of
Master of Technology in Electronics & Communication
Engineering with specialization in VLSI & Embedded Systems

to

Indraprastha Institute of Information Technology Delhi
June, 2015

Certificate

This is to certify that the thesis titled ''**Power Estimation Using Transaction Level Modeling** '' being submitted by **Nidhi Chandoke** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology (VLSI and Embedded Systems), is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

June, 2015

Dr. Sujay Deb
Department of Electronics and Communication
Indraprastha Institute of Information Technology Delhi

Mr. Ashish Kumar Sharma
Senior Manager, ST Microelectronics, Greater Noida

Mr. Maninder Singh
Group Manager, ST Microelectronics, Greater Noida

Abstract

The increasing complexity of current day Systems-on-Chip and the rising market demands for low power devices has necessitated the need to perform power analysis of the complete system-on-chip at system level. This would facilitate selection of optimal system architecture and opportunity to optimize the selected system design for reduced power consumption in the silicon as much more design optimization opportunities are available at system level in contrast to Register Transfer Level (RTL).

Transaction level model provides the conceptual implementation of the design at higher abstraction level above RTL and is available much early in the design cycle as against RTL, hence, is being used in industries for early software development and functional verification of the hardware.

In this work, we present a methodology to estimate power consumption of an IP at transaction level by capitalizing the existing transaction level model of the corresponding IP. The proposed methodology makes use of SystemC transaction level model not specifically designed with power estimation in mind.

We then validate the proposed method against RTL for accuracy and speed. The proposed approach enables estimation of dynamic power consumption and leakage power consumption with an error within 20% of RTL and total power consumption with an error within 12% with respect to RTL. In terms of speed-up, the proposed approach enables 10x faster simulation, hence, faster power estimation compared to estimating power at RTL. The accuracy and speed-up of the proposed approach are illustrated through experimental results.

To my Family - Mom, Dad, Grandmom and my Sister

Acknowledgements

My foremost expression of thanks is dedicated to the Almighty for his blessings.

I express my heartfelt gratitude to my supervisor, Dr. Sujay Deb, for his continuous guidance, support and constructive criticism that enabled the successful and timely completion of this thesis.

I am highly thankful to my mentor at ST Microelectronics, Mr. Ashish Kumar Sharma, for providing suggestions and valuable feedback in the above endeavour, which are indispensible and without which it would not have been possible to accomplish the above task effectively. I also earnestly acknowledge the support and guidance of all my team members at ST Microelectronics.

Last but not the least, I express sincere thanks to my parents, grandparents and my sister for their unconditional love, support and encouragement in all my efforts. I also thank all my friends for their support and motivation.

Contents

List of Figures

Glossary

EDA Electronic Design Automation

IP Intellectual Property

PV Programmer's View

PVT Programmer's View Plus Timing

RTL Register Transfer Level

SAIF Switching Activity Interchange Format

SoC System-on-Chip

TL Transaction Level

TLM Transaction Level Modeling

VCD Value Change Dump

VSoC Virtual System-on-Chip

1

Introduction

1.1 Context

The current market demands portable high speed devices, packed with multifarious/complex functionalities and capable of sustaining long hours of battery operation. The need to suffice the said demands has necessitated the following trends:

- Decrease in technology node to reduce device size
- Increase in frequency of operation to enable high speed devices
- Increase in density of transistors on SoC to embed multiple functionalities into single device

The current day need is to design devices with reduced power consumption so as to enable long hours of device operation but the aforesaid trends have caused the power consumption of SoCs to increase. This calls for methodologies that enable power savings in the final product. The available methodologies for power reduction are implemented late in the design cycle when RTL is available, hence, there is need to develop methodologies that could estimate power at system level itself, which would augment power savings in the final product as much more power optimization opportunities are available at system level when compared with those at RTL. The proposed methodology is a step in the said direction.

1.2 Motivation

The factors that have been the driving force and motivated me to undertake research in this area are as follows:

1. **Need to use Transaction Level Model**
   The process of designing a system involves the development of both hardware and software. The ultimate goal is to minimize the system development time, therefore, waiting for the silicon to exist before starting the development of software is not acceptable. The solution for this issue is to use synthesizable hardware as simulator and develop software on top of it. But this approach suffers from the drawback of slow simulation speed, hence, not suitable for adoption as most of the current day systems are complex systems and require fast simulators. Thus, the final acceptable solution, till date, is to model the design at higher abstraction level that not only enables faster simulation speed, but also facilitates fast prototype development, quick functional verification and early architectural exploration. The said solution is provided by TLM.

2. **Need for estimating power consumption at Transaction Level**
   The power consumption is one of the important design metrics that needs to be taken care of while designing a system. The availability of information regarding power consumption of the silicon at transaction level would enable the designer to optimize the design at transaction level for desired power performance in the final product as much more power optimization alternatives are available at transaction level compared to RTL. Hence, the designer would be able to make informed decision regarding the selection of optimal system architecture and ensure that the chosen system architecture conforms to the specified power constraints. This would ultimately lead to substantial savings in time, resources, efforts and development cost.

3. **Challenge in estimating power at Transaction Level**
   The accurateness of power estimation is governed by granularity of implementation details available at the level at which power needs to be estimated. RTL is detailed implementation of hardware whereas the transaction level model is the conceptual implementation of hardware at an abstraction level above RTL that

hides all the implementation details. Hence, accuracy of power estimated at RTL is much higher than that estimated at transaction level. Since TLM is devoid of implementation details, it makes power estimation at transaction level a challenging task.

1.3 Problem Statement

- Develop a SystemC transaction level model of an IP (VTC) at Programmer's View plus Timing (PVT) level and ensure its functional correctness by verifying the developed transaction level model against its specifications. It is required to gain proper understanding of transaction level modeling in order to be able to model a framework and develop a methodology to estimate power at transaction level.

- Develop a methodology for estimating power at transaction level to enable power aware hardware design. The knowledge of power consumption at the transaction level would enable the designer to make informed decision regarding the selection of appropriate system architecture at the architecture level itself, doing away with the need for the design to go through multiple iterations in the design cycle which are otherwise required in order to make the chosen system design conform to the specified power constraints. Hence, power estimation at transaction level enables substantial savings in terms of time, resources, efforts and cost.

1.4 Chapter Organization

The second chapter discusses related work. The third chapter describes the basics of Transaction Level Modeling and discusses the Transaction Level Model of the IP that has been developed. The fourth chapter deals with the basics of different types of power dissipation. In the fifth chapter, the power estimation methodology used for evaluating power at Register Transfer level has been described followed by the discussion of the approach proposed for estimating power at transaction level. The sixth chapter makes explicit the results obtained as a consequence of implementation of the proposed approach for power estimation at transaction level and its comparison against the power

number obtained at RT level. The last chapter concludes the dissertation and talks about the future work to extend the research to next level.

2

Literature Review

Work has been done for power estimation and optimization at different levels of abstraction. The study of the approaches discussed in the literature pertaining to estimating power consumption at abstraction levels higher than RTL brings forth as follows:

In [1], an approach is proposed to estimate the power consumption of embedded software executing on microprocessors. It is based on the construction of power bank data, consisting of energy estimation and estimated execution time, for all library functions and instructions through RTL simulation for different test cases. The power bank data contains multiple entries corresponding to each function and instruction to cover all possible scenarios, which requires hundreds of simulations at RTL. This technique is technology dependent as it makes use of power values obtained from RTL simulations.

Power modeling approach to instrument SystemC/TLM platform for estimation of power consumption is presented in [2]. The proposed framework uses external power data, obtained from IP datasheets or RTL simulations, to calculate power consumption. Different SystemC classes are defined to take into consideration the phase and mode of the cores based on which power values are extracted from power database. This approach makes use of RTL simulations for creating power database and the use of this approach requires extra modeling efforts to augment the existing transaction level models with functions defined in this approach for power estimation.

[3] describes a power estimation technique based on functional descriptions. The approach makes use of correlation between cycle accurate functional description and the corresponding RTL implementation. The power is estimated by simulating cycle-accurate functional descriptions along with power macro-models for each RTL component.The

macro-model evaluates power as a function of current and previous input vectors.
In [4], power estimation technique for SystemC transaction level model is implemented for IBM CoreConnect based architectures. It makes use of TLM based simulation platform consisting of processor, coreconnect bus complex and peripheral models at Programmers View plus Timing (PVT) abstraction level. This method includes a power characterization approach, a hierarchical representation of transaction level data and power model mapping mechanism to capture power information. The power characterization approach determines the power of different transactions from gate-level simulation and generates hierarchical representation of the power values associated with different transactions. This information is used by the mapping mechanism to augment SystemC TLM simulation platform with power information.
In [5], a power estimation framework for Multiprocessor SoC at Timed Programmers View (PVT) is presented which involves plugging of power models of different components of RTL into the PVT architectural simulator to enable the estimation of power consumption. The power models are developed based on physical measurements, extracted from RTL simulation, and analytical models. Different activities are identified and a counter is allotted for each activity. Each counter is incremented during simulation and the total activity count obtained at the end of simulation is transmitted to power models, which is utilized for calculating the total power consumption.
[6] presents an add-on library TLM Power3 on top of TLM Power2 to provide support for power estimation using SystemC modules. TLM Power3 adds functions to support estimation of dynamic energy usage in every SystemC module. It supports an external table to keep record of the power values corresponding to different phases and modes of each kind of component, which are utilised by power functions to estimate power consumption of different components. It requires extra modeling efforts to be invested and not suitable for power estimation of existing transaction level models that have not been developed using this library.
A TLM energy estimation technique is reported in[7] which proposes automatic abstraction of RTL hierarchical descriptions into corresponding TLM descriptions while preserving timing accuracy through timing annotation and keeping track of the instruction counts while running a test program. This approach extracts activity from RTL simulation and is not suitable when starting point is system level implementation.
[8] presents a framework for estimation of power at modeling level by utilizing macro-models.

The proposed approach extends the SystemC library by adding new classes that describe the computation of power characteristics of behavioral-level hardware model. In this approach, a macro-model is created for each behavioral level operation to calculate dynamic power by estimating the number of rise transitions at its input and output ports and using the following formula:

$$P_d = \sum_{m=1}^{M} (c_{in} t_{in} + c_{out} t_{out}) V_{dd}^2 \tag{2.1}$$

where, c_{in} is the input capacitance, c_{out} is the output capacitance, t_{in} is the input rise transitions, t_{out} is the output rise transitions, V_{dd} is the supply voltage and M is total number of components in the system. This approach is technology dependent as the developed macro-models are based on parameters that are technology dependent.

The aforementioned approaches make use of RTL simulation that is computation intensive and slow, therefore, the said approaches are unsuitable for evaluation of power consumption at system level where the objective is to utilize higher abstraction models of the design, such as transaction level models, to increase simulation efficiency to enable quick and easy exploration of different design alternatives and make informed decision regarding the selection of optimal system architecture.

Thus, the need arises to develop a methodology that takes advantage of the high simulation efficiency enabled by higher abstraction models and effectively utilize them to explore different design alternatives and choose the best suited one that would conform to the specified constraints.

Therefore, in this work, a differential methodology has been proposed to accomplish the aforesaid objective. The approach proposed, herein, implements equation based power model for estimation of power consumption of IPs using their SystemC transaction level models. The proposed approach utilizes transaction level simulation to extract the information required to estimate power consumption and does not employ RTL simulation to extract power values. Hence, the proposed approach increases the power prediction efficiency compared to traditional approaches and is suitable for estimation of power consumption of IPs at system level.

3

Transaction Level Modeling

3.1 Utility of Transaction Level Modeling

The classic design flow for digital electronic embedded system is shown in Fig.3.1. It can be seen that the design flow is split into system hardware and system software development activities, each executed independent of the other till a system prototype is made available. Once the system prototype is available, the system is integrated and validated. If any error is found, then design process will be iterated till a good functioning system with desired performance is obtained.

The need for raising the abstraction level from RTL arises from the bottlenecks faced at System-on chip (SoC) level, which are explained as follows:

1. Ever Increasing Complexity

   A SoC integrates multiple blocks including buses, peripheral IPs, interconnects, memories, processors etc. and is becoming more complex with time, rendering the task of maintaining reliability of SoC tough and time consuming job. Thus, the increasing complexity of SoC points out the need of a new design flow where all phases of the design cycle are linked with each other.

2. Time-to-Market pressure

   Time-to-market is the time required for conceiving an idea into real product for sales. The early availability of the product in market enables larger market share and higher revenue gain. The current complexity of the SoC products

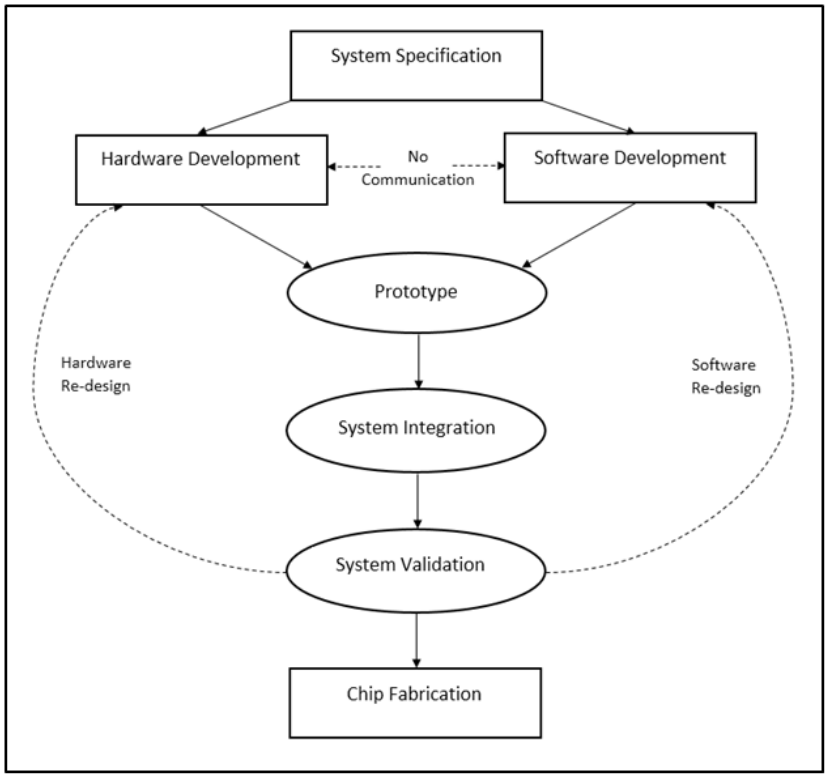

Figure 3.1: Classic design flow [9]

requires time-consuming development phases, which hinders the attempt to shrink time-to-market. Thus, the classic design flow is not suitable as it involves large waiting time for the availability of prototype and hence calls for a new efficient methodology to optimize the time management of SoC products.

3. Sky-rocketing Cost

   The cost of current SoC products is increasing due to demand for higher workforce and much costly masks, re-spins due to error in design functionality or performance, requirement of expensive tools and increasing manufacturing costs. Higher workforce is required to deal with the problems encountered in design, verification and manufacturing. Also, the Electronic Design Automation (EDA) tools required for design and verification are getting expensive. The manufacturing cost of SoC products is also increasing as the technology node is shrinking. Thus, a new design flow is required to control the increasing cost of SoC products by

eliminating the reiterations of design cycle due to errors that must be taken care of in the beginning of the design cycle itself.

In order to overcome the above mentioned SoC bottlenecks, one approach is to raise the design abstraction level above RTL. This approach is known as system level design. It extends the RTL-to-layout design flow by adding System-to-RTL on top of it, making the entry point of the SoC design reside at a higher abstraction level than RTL.
To enable SoC architecture design and verification from higher level than RTL, the right type of abstract modeling is required to be identified to accomplish system design activities for both hardware and software engineers. The Transaction Level Modeling is the suitable abstracted modeling technique for the same.
The new SoC design flow, with abstraction level raised to TLM, is shown in fig.3.2. This new SoC design flow consists of an additional system-to-RTL extension to the traditional design flow. The new design also starts from system specifications based on which the system hardware and software partitioning is performed by system architects in order to configure an optimal system architecture. Thereafter, comes the role of TLM that differentiates the new design flow from traditional design flow. The availability of TLM platform enables concurrent hardware/software development. The TLM platform finds its utility in the following domains:

- TLM platform serves as the unique reference for software teams to conduct early software development.
- TLM platform serves as the unique reference for architecture teams to perform coarse-grain architecture analysis.
- TLM platform enables the verification team to develop verification environment and associated tests that would be required for verification of the developed RTL platform.

The RTL platform, once available, can be checked for its compliance with the desired performance and other tasks such as hardware verification and low-level integration can be performed. The hardware, after thorough verification, is taped-out for chip fabrication. The introduction of TLM in new design flow enabled concurrent hardware/software development, which further made possible thorough verification of hardware and software

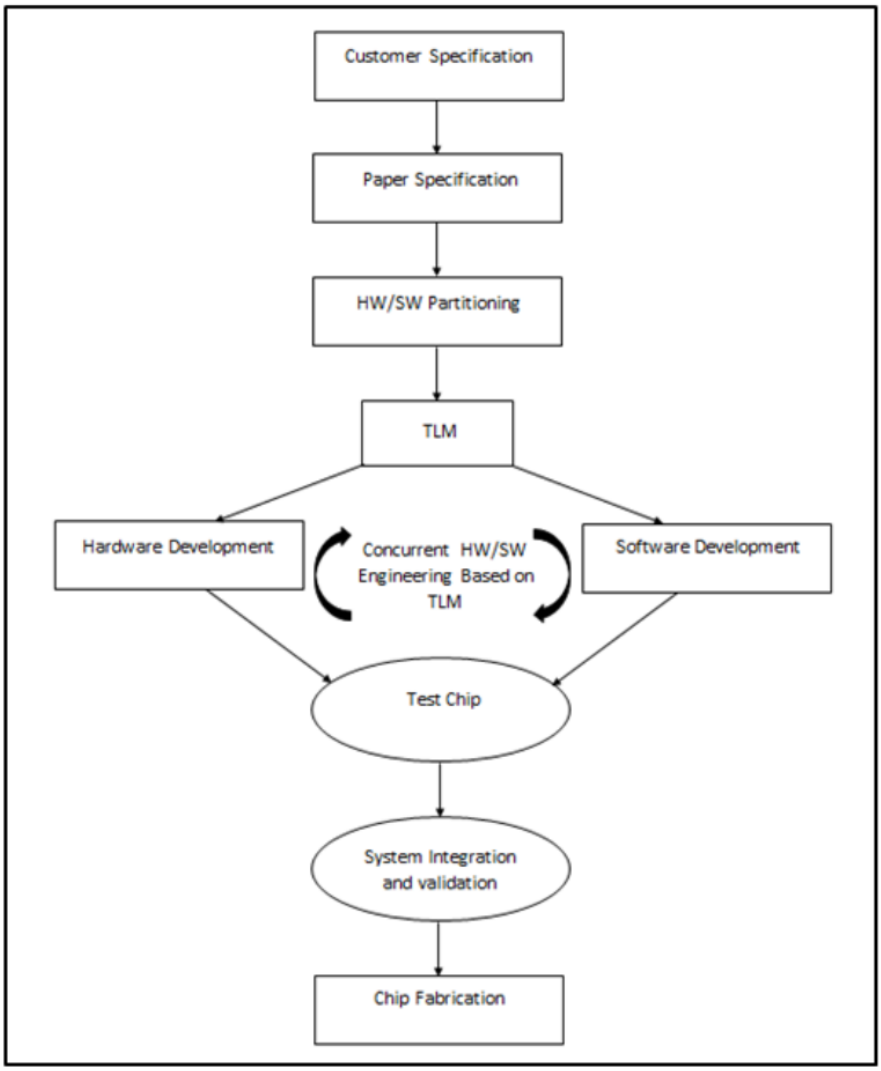

Figure 3.2: SoC design flow with TLM [9]

designs before chip fabrication, thus, increasing the probability of achieving silicon success in the first attempt.

3.2 Transaction Level Modeling: An Overview

Transaction level modeling technique is intended for designing digital electronic systems at higher abstraction level above RTL. A digital electronic system is composed of multiple components, each with a finite set of states and a series of concurrent behavior. TLM technique models each of the components as modules, represents their internal states with a set of variables defined within the scope of corresponding TLM module and models their concurrent behavior with a set of concurrent processes or threads, which can be executed in parallel. TLM modules are interconnected by means of specific TLM communication structure called channel or interconnect. Thus, TLM technique enables separation of communication from computation, while the former is implemented using channel whereas latter takes place in corresponding modules.

The binding structures used to bind TLM modules with channel are known as communication ports. Once the binding is implemented, data can be interchanged between the modules through channel to perform the desired system behavior.

The term transaction refers to the set of data being interchanged between initiator and target. Initiator, also known as master, is the module that initiates transactions in a system. Target, also known as slave, is the module that serves the transactional requests initiated by the initiator. System synchronization refers to an explicit action that is required to coordinate some behavior distributed between different modules and minimum two modules are required for this purpose. System synchronization is required to ensure proper behavior of SoC at transaction level.

The language suitable for implementing TLM should be a high-level language that could support software programming as well as development of hardware at conceptual level without actual implementation of hardware. The languages suitable for the same include SystemC, System Verilog, Hpascal, HardwareC, SpecC and the like.

In this work, SystemC is chosen for the implementation of TLM of IPs as it is widely accepted in industries for the said purpose.

3.3 Types of Transaction Level Models

The accuracy of a transaction level model is governed by the following factors:

1. Data Granularity

   The granularity refers to the fineness of data that is carried by the communication structure in a model. The granularity of data can be categorized into three levels, i.e. application packet, bus packet and bus size. The accuracy of the developed model increases on moving from application packet level towards bus size level.

2. Timing Accuracy

   Timing accuracy refers to the closeness with which a model can implement the intended timing behavior. Conceptually, it can be perceived as a scale of two extremes, i.e. untimed level and cycle-accurate level. The timing accuracy increases as one moves from untimed level towards cycle-accurate-level. Any level falling in between the two extremes is referred to as approximately-timed model.

The modeling accuracy of different approaches is shown in fig.3.3. It can be clearly understood that TLM-PVT provides optimal trade-off between simulation speed and modeling accuracy.

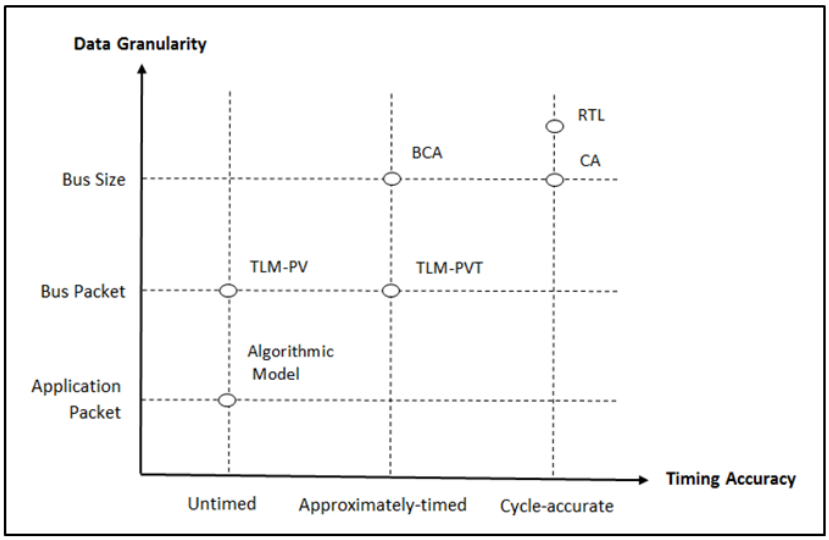

Figure 3.3: SoC design flow with TLM [9]

The transaction level models are classified into the following two classes based on the accuracy of the developed model:

1. Untimed TLM
2. Timed TLM
    - Annotated
    - Standalone

3.3.1 Untimed TLM

The untimed TLM model does not contain any information regarding the micro-architecture of component or IP under design and timing annotations are considered to be insignificant. The untimed TLM is particularly developed for software programmers and verification engineers to enable early software development and fast functional verification.
The untimed TLM consists of absolutely no timing information regarding micro-architecture, that is, there is no clock in an untimed TLM system. Since there is no clocked timing regulation, the processes are executed in a concurrent manner to access the system resource at the same instant. At the same time, the system must exhibit correct behavior during execution of concurrent processes. Thus, the computation model employed in an untimed model must ensure concurrent execution of independent processes and implementation of system synchronization to take care of causal dependencies between processes.
The explicit system synchronization between different processes in a system is required to ensure deterministic system behavior. The processes can be executed in any order in a system as long as their causal dependencies are well respected. There are two types of synchronizations- emit synchronization and receive synchronization. Emit synchronization occurs when a process sends out synchronization that may influence the behavior or state of another process. Receive synchronization occurs when a process waits for incoming event from the system that may influence its behavior or state. System synchronization ensures memory or data consistency. It prevents concurrent processes to read data at unknown state or write data at temporarily inaccessible memory area.

3.3.2 Timed TLM

The untimed TLM does not specify the complete order of execution of SoC internal events as it does not capture the timing behavior of the implementation. The timed transaction level model incorporates the microarchitecture timing details and enables complete specification implementation. Timed transaction level models are developed when the requirement is to benchmark the performance of a given micro-architecture, fine-tune the given micro-architecture and optimize the software for a given micro-architecture to meet real time constraints. The communication and computation form two important aspects whose time consumption must be taken into account while developing timed transaction level model.
The timed transaction level model can be of two types depending on the time consumption of component- Annotated model and Standalone timed model.

3.3.2.1 Annotated Timed TLM

Annotated model is the modeling approach in which timing details are inserted into untimed model. The inserted timing details correspond to the timing information of the micro-architecture level. Simple wait statements are used to insert delays that correspond to the computation time of a specific functionality. This approach is suitable when structure of algorithm is similar to that of micro-architecture.

3.3.2.2 Standalone Timed TLM

Standalone timed model is an approach where the timing behavior is modeled such that delays are computed during the execution of standalone timing model. A standalone model is a detached model incorporated with timing information. This approach is suitable when structure of algorithm is very different from the structure of micro-architecture.

3.4 Case Study: Transaction Level Modeling of Video Timing Controller (VTC)

3.4.1 Development of Transaction Level Model of VTC

The timed transaction level model, also known as Programmer's View plus Timing (PVT), was developed as per the given specifications.

- Capable of generating video timing for different video formats.
- Supports Interlaced and Progressive formats.
- Capable of generating active video periods.
- Capable of generating frame and field interrupts.
- Supports Slave and Master mode configurations.

The procedure followed to develop the transaction level model of VTC is shown in fig.3.4. The first step is to generate a Spirit File in IP-XACT format from the specifications provided for the given IP. IP-XACT is an XML format used to describe meta-data of IP designs, flows and interconnection of IP interfaces. IP-XACT was developed by SPIRIT Consortium as a standard to support automated configuration and integration through tools.

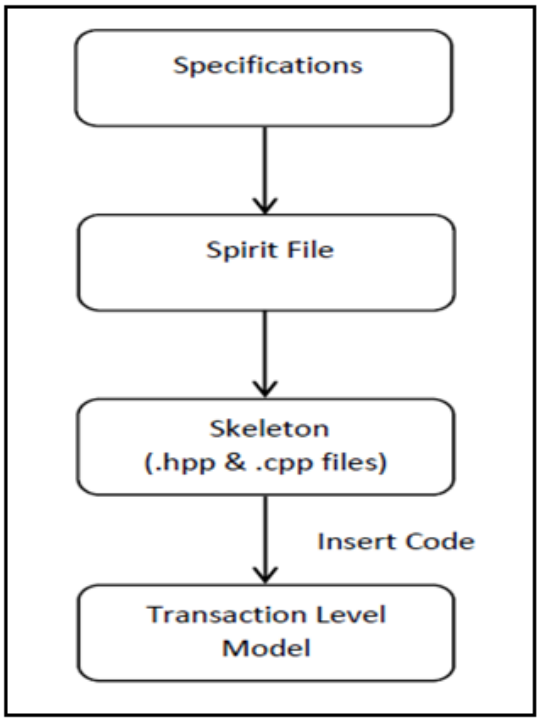

Figure 3.4: Steps involved in development of transaction level model

The second step is the generation of skeleton from the Spirit file. Skeleton refers to the basic structure of code to be developed. Skeleton generation involves generation of two folders viz. include and src. Include folder contains the header files (vtc.hpp and base_vtc.hpp) while src folder contains the source files (vtc.cpp, base_vtc.cpp and vtc_register_map.cpp).

base_vtc.hpp.file is generated, in the include folder, from the spirit file as part of skeleton generation. This file consists of all the port declarations that are listed in the spirit file. It need not be modified until and unless additional ports, apart from those mentioned in the spirit file, are required to be added to the code.

base_vtc.cpp file is created, in src folder, from the spirit file as part of skeleton generation. This file consists of port bindings of all the ports declared in the corresponding base header file, that is, base_vtc.hpp. It need not be modified, until and unless the corresponding base header file is modified.

vtc.hpp is generated, in include folder, as an empty file as a part of skeleton generation and the desired code to be inserted in this file. This file consists of all declarations including variable declarations, method declarations, event declarations and declaration of side-effect registers.

vtc.cpp is generated, in src folder, as an empty file and the desired code needs to be inserted in this file. This file contains definitions corresponding to all the declarations made in the corresponding header file, that is, vtc.hpp.

vtc_register_map.cpp is generated, in src folder, as part of skeleton generation and need not be modified. It contains the regbank, that is, it defines all register details including register name, register fields, associated bit-widths of register fields and their attributes. The registers specified in the register_map.cpp file can be accessed using the regbank object defined in the base_vtc.hpp file.

Once the skeleton is generated, the final step is to insert the code in vtc.hpp and vtc.cpp files as per the specifications of the given IP (VTC) to achieve the desired functionality of the IP. The resultant set of codes obtained is called transaction level model.

3.4.2 Functional Verification of Transaction Level Model of VTC

The functional correctness of the developed transaction level model is first ensured by comparing the functional behaviour of the transaction level model of the IP against the specifications of the corresponding IP. Thereafter, the developed transaction level

model is plugged into the Virtual System-on Chip (VSoC) environment of the SoC and supplied with an input file specifying the data to be transmitted. The data to be transmitted is encoded, processed through different IPs and then finally decoded at receiver end. The decoded file obtained at the receiver end is compared with the input file supplied at transmitter end. The correctness of IP functionality is assured by the compliance of the output file with the input file.

4

Basic Constructs of Power Dissipation

4.1 Types of Power Dissipation

The design process of SoC is primarily governed by three design metrics viz. area, speed and power. The current market demands of increased functionality and reduced power consumption of portable battery operated devices have led to increase in complexity of SoCs. In the present day scenario, the complexity of SoCs is increasing as more and more transistors are being packed into the same chip area to enhance its functionality. The increase in the packing density of the transistors has led to increase in power dissipation of the SoCs. Hence, power dissipation is one of the primary constraints that needs to be taken care of while designing current day systems. The sources of power dissipation can be broadly categorized into two categories, namely, static power dissipation and dynamic power dissipation.
The average power dissipation in a circuit can be calculated as the sum of the aforementioned components of power dissipation, which is represented by equation (4.1).

$$P_{avg} = P_{dynamic} + P_{static} \tag{4.1}$$

Where, P_{avg} is the average power dissipation, $P_{dynamic}$ is the dynamic power dissipation and P_{static} is the static power dissipation. Different components of power dissipation are depicted in fig.4.1 and are discussed in detail in the following subsections.

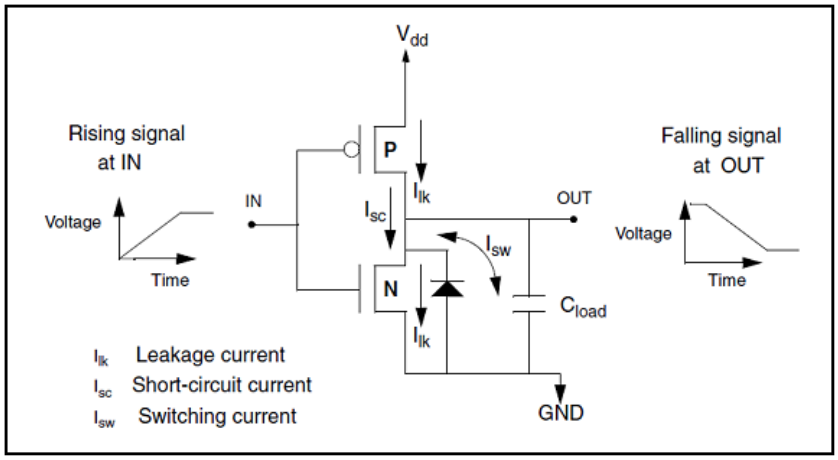

Figure 4.1: Different components of power dissipation [10]

4.1.1 Dynamic Power Dissipation

The power dissipation that occurs due to charging and discharging of capacitances in a circuit is called dynamic power dissipation. Dynamic power dissipation takes place when the circuit is active, that is, whenever a stimulus applied to the circuit causes change in the logic state of a net in the circuit. Dynamic power dissipation dominates the total power dissipation in a circuit compared to other components of power dissipation. In submicron technologies, the functionality requirement is heightening as well as the frequency of operation is increasing making dynamic power dissipation the major contributor to the total power dissipation.
The calculation of dynamic power dissipation is illustrated through an example of CMOS inverter driving a load capacitance C_L, as shown in fig.4.2(a). The load capacitance C_L is the cumulative capacitance including parasitic capacitances of PMOS and NMOS transistors, interconnect capacitance of the cell and input capacitance of the circuit driven by the inverter.

Consider that the inverter is initially in steady state and input is at logic 1. Therefore, the output will be logic 0 and the load capacitor is discharged. When the input signal makes a transition from logic 1 to logic 0, PMOS transistor turns on and NMOS transistor turns off. As a result, the current charges the capacitor C_L to supply voltage V_{dd}, as shown in fig.4.2(b). The charging process draws energy of $C_L {V_{dd}}^2$, of which half is stored in the capacitor and half is dissipated in the parasitic capacitance of the PMOS

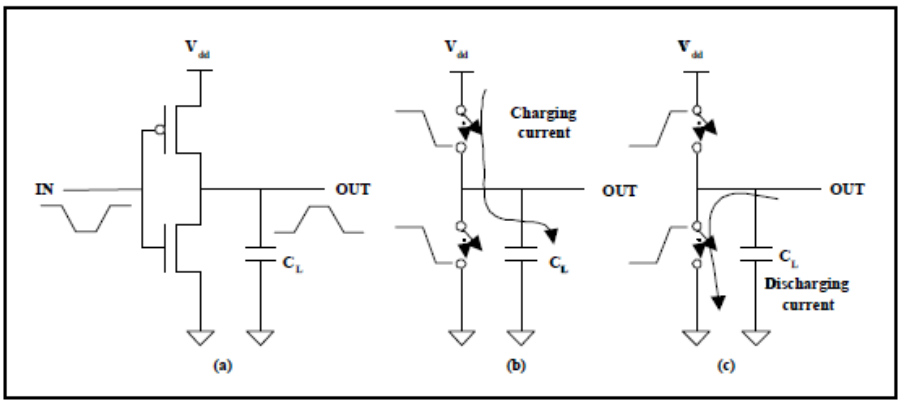

Figure 4.2: Power dissipation in CMOS inverter [11]

transistor and interconnect capacitance. When the input signal makes a transition from logic 1 to logic 0, the PMOS transistor turns off and NMOS transistor turns on. As NMOS transistor turns on, a discharge path from load capacitor C_L to ground becomes available as shown in fig.4.2(c). The energy stored in the capacitor is dissipated in the parasitic capacitance of NMOS transistor and the interconnect capacitance. Thus, the total dynamic power dissipated by the CMOS inverter in time interval [0,T] can be calculated as shown in equation (4.2).

$$P_{dynamic} = C_L V_{dd}^2 N_{0\_1}/T \tag{4.2}$$

Where, C_L is the load capacitance, V_{dd} is the supply voltage and N_{0_1} is the total number of rise transitions at the output of CMOS inverter.
Dynamic power is composed of two types of power, which are stated as follows:

- Internal Power
- Switching Power

4.1.1.1 Internal Power

The power that is dissipated within a cell is called internal power. When switching occurs, the power dissipated due to charging or discharging of the capacitances internal to the cell constitutes internal power. Internal power also includes the short circuit power dissipated due to momentary short-circuit between p-type and n-type transistors of a gate providing a conduction path from supply voltage to ground.

4.1.1.2 Switching Power

The power that is dissipated by charging or discharging of the load capacitance at the output of a cell is termed as switching power. The total load capacitance at the cell output is determined as the sum total of gate capacitance and net capacitance at the cell output. The switching power increases with increase in the number of logic transitions because increased logic transition would imply frequent charging/discharging of load capacitance and hence, higher switching power dissipation. Thus, switching power can be defined to be a function of rate of logic transitions as well as load capacitance at the output of a cell.

4.2 Static Power Dissipation

The power dissipation that occurs when the gate is not switching, that is, when the gate is static or inactive is called static power dissipation. The different sources of static power dissipation are as follows:

1. Sub-threshold leakage
   PMOS and NMOS transistors have a specific gate-source voltage at which channel inversion takes place called as the threshold voltage. In order for the transistor to conduct, gate-source voltage must be greater than the threshold voltage of the transistor. When the transistor turns off, the current flowing through the channel does not become zero instantaneously but decreases exponentially, thus making the current through the channel non-zero. This process in which current flows through the channel when gate-source voltage of the transistor decreases below its threshold voltage is called sub-threshold leakage and contributes to static power dissipation.

2. Tunneling through gate oxide
   The decrease in technology node has led to decrease in thickness of the gate oxide of the transistors. The decrease in thickness of gate oxide causes electrons to tunnel through the oxide and the resultant current is called tunneling current, which causes static power dissipation. Tunneling current becomes important at technology nodes below 130 nm with gate oxide thickness less than 20 $\mathring{A}$.

3. Leakage current through reverse-biased diodes

   Reverse bias is formed in transistors between wells and diffusion regions or between substrate and wells. The formation of reverse biased diodes causes leakage current to flow, thus causing static power dissipation. The power dissipation due to reverse biased diodes is not significant compared to that caused by sub-threshold leakage and tunnelling current, hence can be neglected.

5

Power Estimation Methodology

This chapter discusses the power estimation techniques, employed in this dissertation, for estimating power at RTL and TL. The first section describes the methodology followed for estimating power consumption at RTL. The second section characterizes the methodology proposed in this dissertation for estimating power consumption at TL.

5.1 Methodology for Estimation of Power Consumption at RTL

The power at Register Transfer Level (RTL) is calculated using the procedure shown in fig.5.1. A test case is generated to capture the maximum activity of the Intellectual Property (IP). The test case is then simulated and a Value Change Dump (VCD) file is generated. VCD is an ASCII based format to record the values of the variables during simulation by logic simulation tools. The variables whose values need to be recorded can be selectively dumped into VCD file. Each of the variables dumped into the VCD file is assigned an ASCII identifier that is used to represent changes in the value of the corresponding variable. VCD file lists the values of the variables against a time stamp, as and when the values of the specified variables change during simulation. The VCD file, thus, obtained is converted to a Switching Activity Interchange Format (SAIF) file. SAIF file specifies five values for each signal dumped into the SAIF file viz. time for which the signal had logic value '0'(T0), time for which the signal had logic value '1'(T1), time for which the signal had unknown value 'x'(TX), number of 0-1

and 1-0 transitions observed (TC) and number of 0-x-0 and 1-x-1 transitions observed (TG). The SAIF file, the netlist in Synopsys internal database format (DDC) and the technology file are, thereafter, fed into the Design Compiler to obtain the power consumption of the IP at RT level. The power report generated by the design compiler specifies the total power consumption of the IP as the combination of internal power, switching power and leakage power. The internal power and switching power make up dynamic power consumption whereas the leakage power dominates static power consumption of the IP. Thus, the total dynamic power consumption and leakage power consumption of the IP obtained at RT level are compared against the respective values estimated using SystemC transaction level model of the corresponding IP.

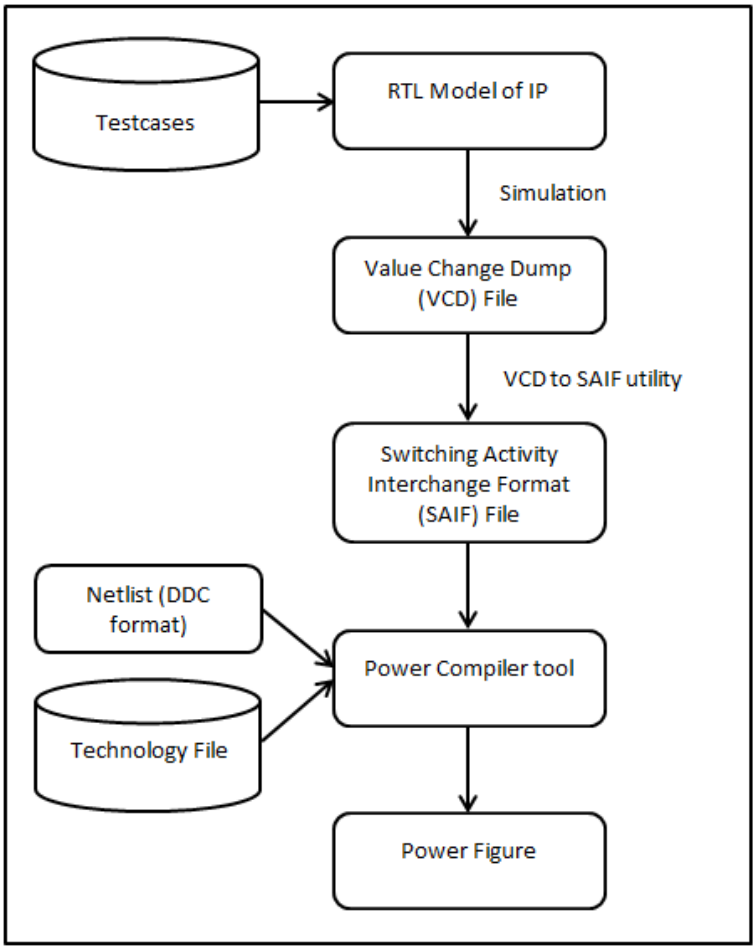

Figure 5.1: RTL power analysis flow

5.2 Proposed Methodology for Estimation of Power Consumption at Transaction Level

The proposed approach utilizes SystemC transaction level model for estimating power consumption at transaction level, which can be understood with the help of flow depicted in fig.5.2 and fig.5.3. Fig.5.2 demonstrates the activity extraction flow and fig.5.3 depicts the cell extraction flow.

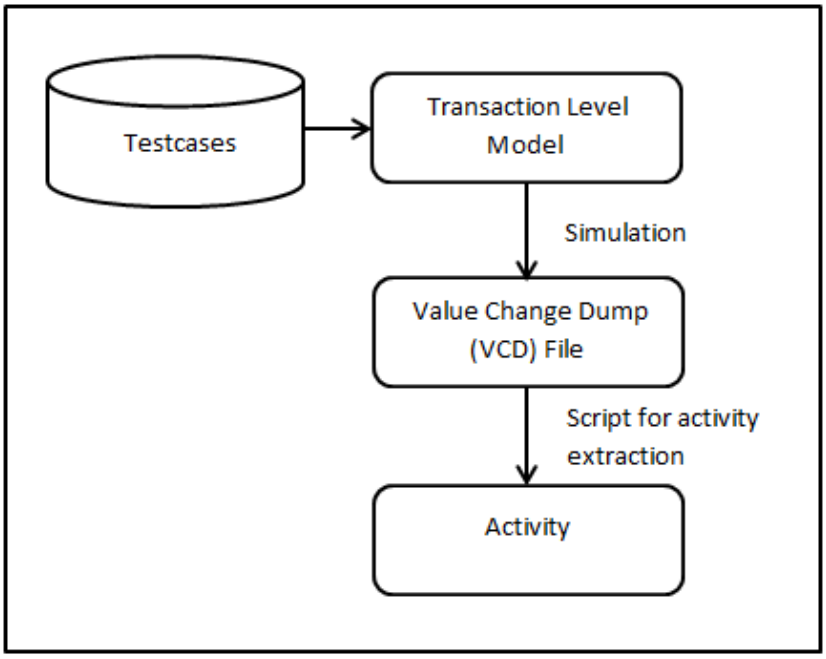

Figure 5.2: Activity extraction flow

The power consumption has two main components, namely, dynamic power consumption and static power consumption.

The estimation of dynamic power consumption at transaction level using the proposed approach requires the knowledge of activity taking place at transaction level, cells used in design implementation at transaction level, design constraints and technology file.

In order to determine activity, the first step is to generate a VCD file by applying test case, covering the scenario for which power needs to be estimated, to the transaction level model and simulating it. The ports whose activity is required to be captured by the VCD file need to be dumped into VCD file by the TLM code. The second step is to extract activity from the generated VCD file. The activity, which is taken into

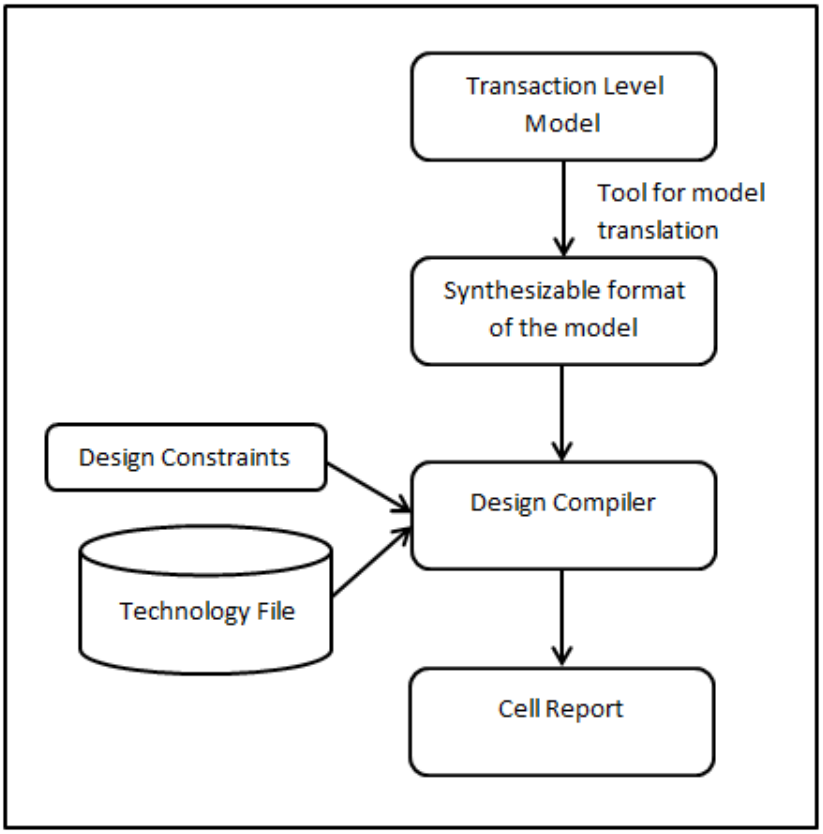

Figure 5.3: Cell extraction flow

consideration in this approach, refers to the rise transitions at bit level on the ports that are dumped into the VCD file. The activity is extracted from the generated VCD file using a script developed for the purpose.

The process followed to determine cells used in design implementation at transaction level includes conversion of transaction level model into a format that is synthesizable by Design Compiler. This is achieved by developing a tool that could extract the code implementing behavioral level operations from transaction level model and replace non-synthesizable constructs with synthesizable Verilog constructs. The generated Verilog file is fed into Design Compiler along with the design constraints and technology file. The same design constraints and technology file must be specified as the designer intends to use at RTL. The design compiler will generate a cell report consisting of optimized cells.

The dynamic power consumption is estimated using equation(5.1)

$$P_D = \sum_{i=1}^{C} n_i/N * A * RP_i \tag{5.1}$$

where,

P_D = Dynamic power consumption

C = Total number of unique cells in cell report

n_i = Number of occurrences of a particular type of cell in cell report

N = Total number of cells in cell report

A = Total activity

RP_i = Rise power of a particular cell extracted from the technology file in accordance with the design constraints specified by the designer

The estimation of leakage power consumption at transaction level using the proposed approach requires cell report and technology file. The number of occurrences of a particular type of cell is extracted from the cell report and the maximum leakage power of the corresponding cell is extracted from the technology file to calculate the total leakage power consumption. This is achieved using script developed for the purpose. The leakage power consumption is estimated using equation (5.2).

$$P_L = \sum_{i=1}^{C} n_i * LP_i \tag{5.2}$$

where,

P_L = Leakage power consumption

C = Total number of unique cells in cell report

n_i = Number of occurrences of a particular type of cell in cell report

LP_i = Maximum leakage power of a particular cell extracted from the specified technology file

The total power consumption is calculated as sum of dynamic and leakage power consumption obtained above. The comparison of dynamic, leakage and total power consumption, obtained at transaction level by utilizing the proposed methodology, with the power consumption numbers obtained at RTL is presented in the chapter 6.

6

Results

This chapter discusses the results obtained and their analysis under three heads, namely, savings in modeling effort, speed-up in simulation and accuracy of the proposed approach in estimating power consumption at transaction level with respect to RTL.

We have made an attempt to verify the proposed methodology, for estimation of power consumption at TL, by applying the proposed technique for estimation of power consumption of three IPs at transaction level using their respective transaction level models. The three IPs undertaken for verification of the proposed approach are briefly described as under:

IP1 is used to control the video timing behavior, that is, generate active video and blanking periods depending on the supplied synchronization pulses. IP1 is constituted by five major modules. The first module generates horizontal synchronization pulses for the target. The second module generates vertical synchronization pulses for the target. The third module controls the number of pixels per line and specifies when new line is to be generated. The fourth module controls the number of lines per frame and specifies when next frame is to be generated. The fifth module is used to reset all registers and counters of the IP when external reset signal is applied.

IP2 is used for the transmission of high definition digital audiovisual signals from audiovisual devices to video displays.The transmission takes place through channels that carry audio, video and auxiliary data separately on each channel. It encodes the signal to be transmitted, which is decoded at the receiver end. The frequency used for decoding at the receiver end is determined by the clock that is transmitted on a separate channel by the transmitter.

IP3 is used to read audio data, encoded bit streams or voice samples stored in the memory via a specified interface and format the data as per the requirements of the target application. The formatted data is then transferred to the specified target via a serial interface.

6.1 Comparison of Modeling Effort invested at TL with corresponding RTL for different IPs

The modeling effort refers to the amount of efforts invested by the engineer in order develop the design as per the given system specifications. The amount of efforts required to be invested depend on the size of the code to be developed by the engineer, hence, the modeling effort invested by the engineer at transaction level and RTL are compared in terms of the size of the code of the IPs developed at transaction level and RTL. The savings in modeling effort invested at TL with respect to RTL in case of IP1, IP2 and IP3 are depicted in the following subsections, which are calculated using equation (6.1)

$$\%ModelingEffortSavings = \left| \frac{CS_{RTL} - CS_{TL}}{CS_{RTL}} \right| * 100 \tag{6.1}$$

where,
% Modeling Effort Savings = Savings in modeling efforts invested at transaction level with respect to RTL,
CS_{RTL} = Size of the RTL code,
CS_{TL} = Size of the TL code.

6.1.1 Modeling Effort for IP1

The comparison of modeling effort invested at transaction level and RTL for IP1 is shown in fig. 6.1.

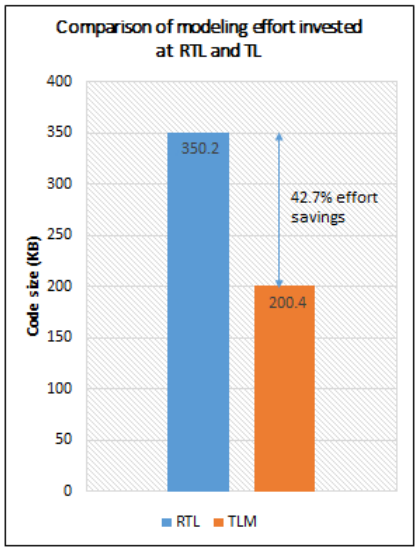

Figure 6.1: Comparison of modeling effort invested at RTL and TL for IP1

6.1.2 Modeling Effort for IP2

The comparison of modeling effort invested at transaction level and RTL for IP2 is shown in fig. 6.2.

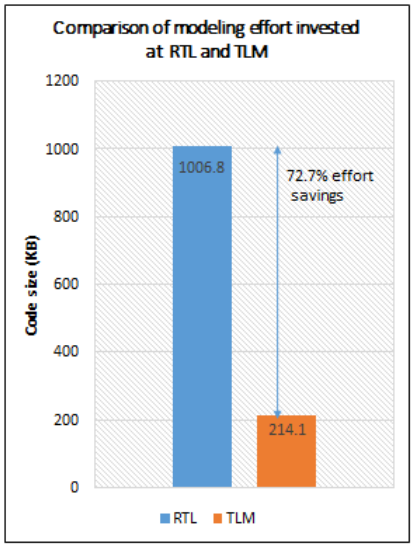

Figure 6.2: Comparison of modeling effort invested at RTL and TL for IP2

6.1.3 Modeling Effort for IP3

The comparison of modeling effort invested at transaction level and RTL for IP3 is shown in fig. 6.3.

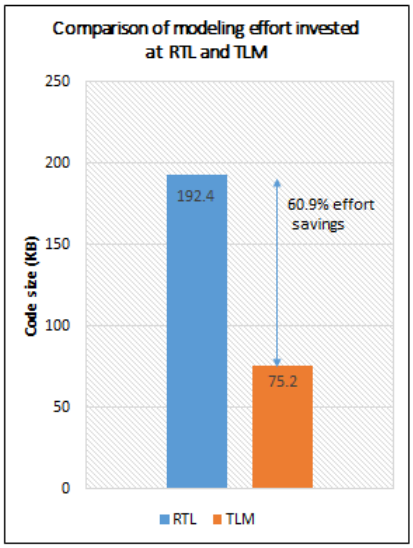

Figure 6.3: Comparison of modeling effort invested at RTL and TL for IP3

6.1.4 Analysis of Result

TLM is an abstract description of hardware as against RTL that contains detailed hardware description. Thus, modeling effort required to be invested by the designer at transaction level is much less compared to that required at RTL.

6.2 Comparison of Simulation Time required at TL with the corresponding RTL for different IPs

Simulation is required in order to verify the developed design against its specifications so as to ensure its correct functional behaviour. The time required to simulate a given code is largely dependent on the extent of details that are contained in the code to be simulated. Simulation time governs the time required to dump VCD file that is required for power estimation because VCD file is available only after completion of simulation. The speed-up in simulation time obtained at transaction level with respect

to RTL in case of IP1, IP2 and IP3 is depicted in the following subsections, which is calculated using equation (6.2).

$$\%Speed - up = \left| \frac{SimTime_{RTL} - SimTime_{TL}}{SimTime_{RTL}} \right| * 100 \tag{6.2}$$

where,

% Speed-up = Speed-up in simulation obtained at transaction level with respect to RTL,

$SimTime_{RTL}$ = Time required to simulate RTL code,

$SimTime_{TL}$ = Time required to simulate TL code.

6.2.1 Simulation Time for IP1

The speed-up in simulation obtained at transaction level with respect to RTL for IP1 is shown in fig. 6.4.

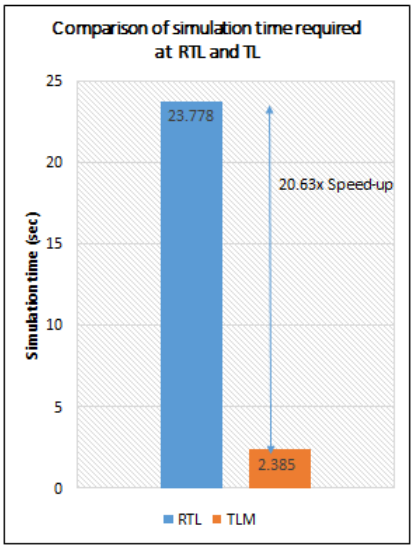

Figure 6.4: Comparison of simulation time required at RTL and TL for IP1

6.2.2 Simulation Time for IP2

The speed-up in simulation obtained at transaction level with respect to RTL for IP2 is shown in fig. 6.5.

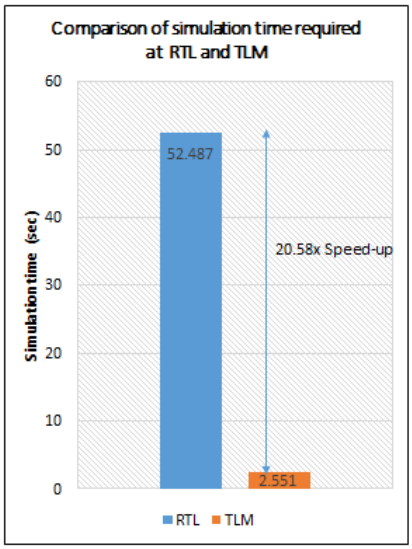

Figure 6.5: Comparison of simulation time required at RTL and TL for IP2

6.2.3 Simulation Time for IP3

The speed-up in simulation obtained at transaction level with respect to RTL for IP3 is shown in fig. 6.6.

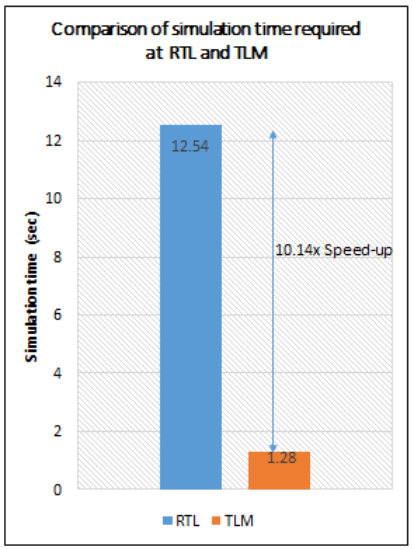

Figure 6.6: Comparison of simulation time required at RTL and TL for IP3

6.2.4 Analysis of Result

Simulation is much faster at transaction level when compared with RTL simulation because transaction level model is higher abstraction of the hardware that captures only its functional behaviour against RTL that is modelled at cycle-accurate level and possesses great fidelity to the real implementation.

6.3 Comparison of Power Numbers obtained at TL with corresponding RTL for different IPs

6.3.1 Power Consumption of IP1

The variation in power numbers obtained at TL and RTL with variation in design constraints for IP1 is studied. The rise power of a cell specified in a design cell library is governed by two constraints, namely input transition time and output net capacitance. The variation in dynamic power consumption, leakage power consumption and total power consumption is studied by altering the aforesaid constraints while keeping all other constraints constant. The following three cases depict the power consumption results obtained as a result of introduced variations in capacitance at the output of a cell and transition time at the input of a cell. The error in power consumption obtained at transaction level with respect to RTL is calculated using equation (6.3).

$$\%PowerError = \left| \frac{PC_{RTL} - PC_{TL}}{PC_{RTL}} \right| * 100 \tag{6.3}$$

where,
% Power Error= Error in power consumption estimated at transaction level with respect to power consumption obtained at RTL,
PC_{RTL} = Power consumption obtained at RTL,
PC_{TL} = Power consumption estimated at TL.

6.3.1.1 Constraint Set I

The constraint set I specifies the following constraints:
Max_capacitance = 0.01 pF, Max_transition = 0.001 ns
The comparison of dynamic power consumption, leakage power consumption and total power consumption obtained at TL and RTL for IP1 for constraint set I is shown in

fig. 6.7, fig. 6.8 and fig. 6.9 respectively. It is observed that dynamic power consumption is estimated at transaction level with an error of 17.09% with respect to RTL, leakage power consumption with an error of 18.93% and total power consumption with an error of 7.54%.

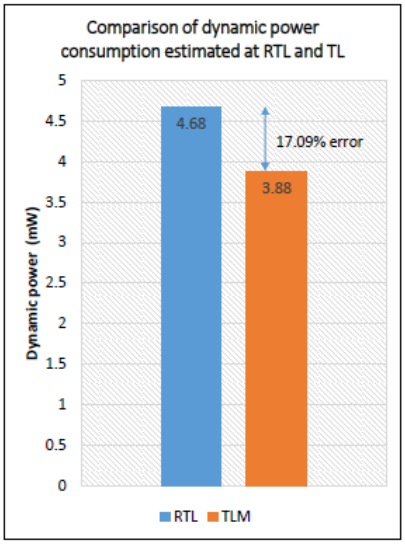

Figure 6.7: Comparison of dynamic power consumption estimated at RTL and TL for IP1 for constraint set I

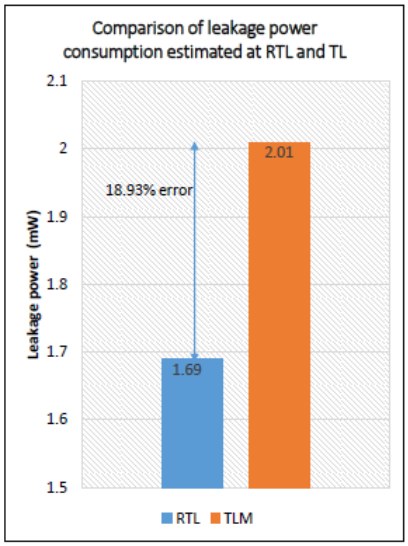

Figure 6.8: Comparison of leakage power consumption estimated at RTL and TL for IP1 for constraint set I

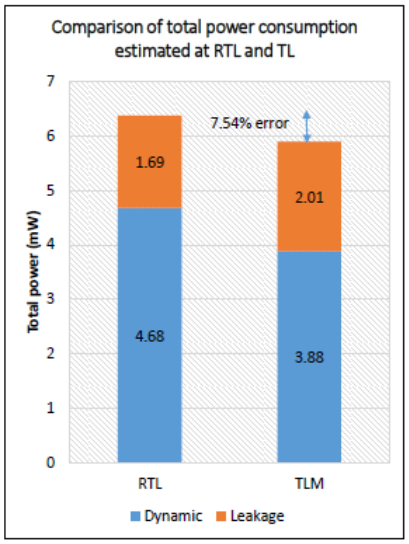

Figure 6.9: Comparison of total power consumption estimated at RTL and TL for IP1 for constraint set I

6.3.1.2 Constraint Set II

The constraint set II specifies the following constraints:
Max_capacitance = 0.14 pF , Max_transition = 0.14 ns
The comparison of dynamic power consumption, leakage power consumption and total power consumption obtained at TL and RTL for IP1 for constraint set II is shown in fig. 6.10, fig. 6.11 and fig. 6.12 respectively. It is observed that dynamic power consumption is estimated at transaction level with an error of 11.29% with respect to RTL, leakage power consumption with an error of 15.54% and total power consumption with an error of 6.04%.

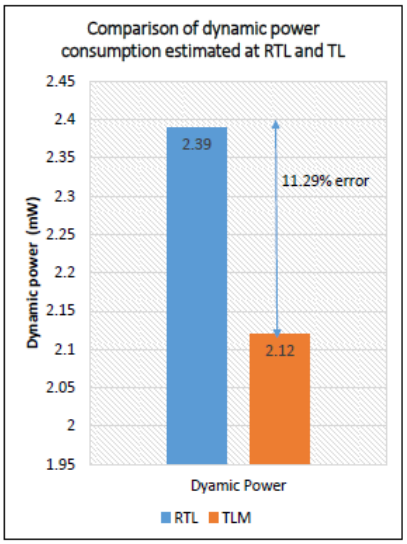

Figure 6.10: Comparison of dynamic power consumption estimated at RTL and TL for IP1 for constraint set II

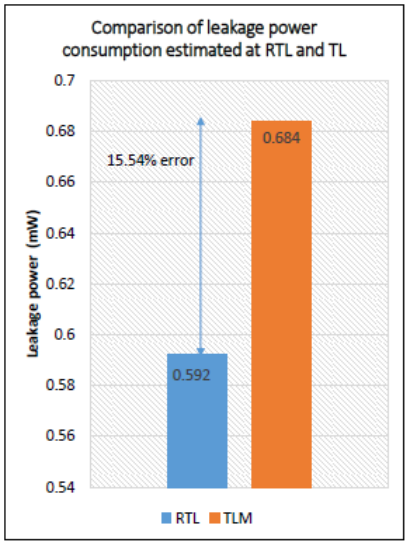

Figure 6.11: Comparison of leakage power consumption estimated at RTL and TL for IP1 for constraint set II

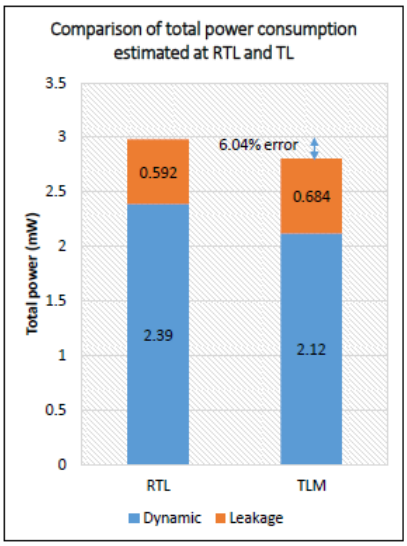

Figure 6.12: Comparison of total power consumption estimated at RTL and TL for IP1 for constraint set II

6.3.1.3 Constraint Set III

The constraint set III specifies the following constraints:
Max_capacitance = 0.1 pF , Max_transition = 1.23 ns
The comparison of dynamic power consumption, leakage power consumption and total power consumption obtained at TL and RTL for IP1 for constraint set III is shown in fig. 6.13, fig. 6.14 and fig. 6.15 respectively. It is observed that dynamic power consumption is estimated at transaction level with an error of 17.61% with respect to RTL, leakage power consumption with an error of 18.69% and total power consumption with an error of 9.88%.

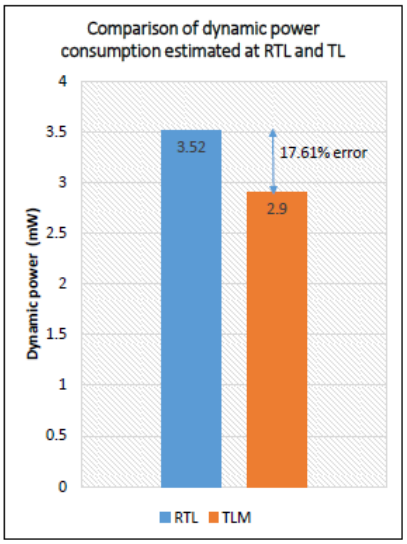

Figure 6.13: Comparison of dynamic power consumption estimated at RTL and TL for IP1 for constraint set III

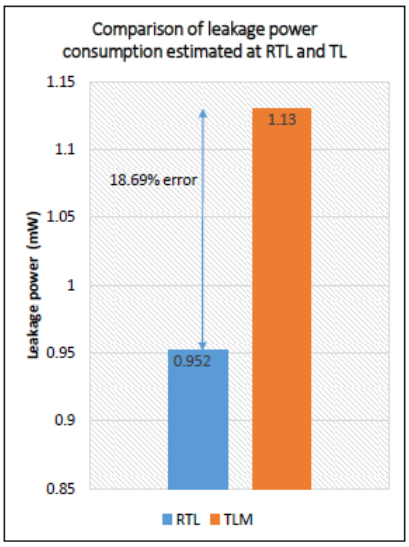

Figure 6.14: Comparison of leakage power consumption estimated at RTL and TL for IP1 for constraint set III

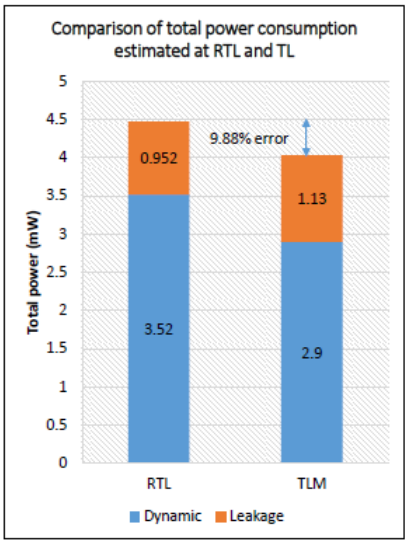

Figure 6.15: Comparison of total power consumption estimated at RTL and TL for IP1 for constraint set III

6.3.2 Power Consumption of IP2

The comparison of dynamic power consumption, leakage power consumption and total power consumption obtained at TL and RTL for IP2 is shown in fig. 6.16, fig. 6.17 and fig. 6.18 respectively. It is observed that dynamic power consumption is estimated at transaction level with an error of 16.43% with respect to RTL, leakage power consumption with an error of 18.51% and total power consumption with an error of 12.03%.

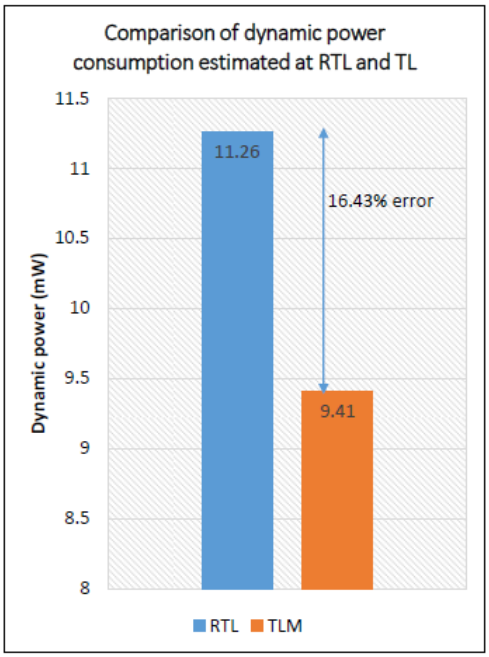

Figure 6.16: Comparison of dynamic power consumption estimated at RTL and TL for IP2

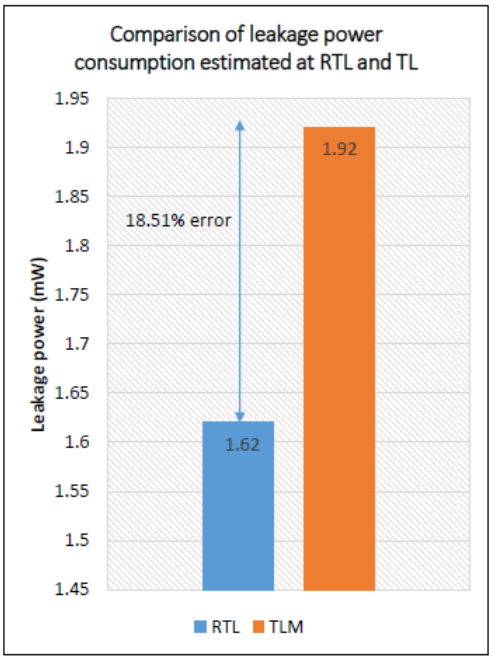

Figure 6.17: Comparison of leakage power consumption estimated at RTL and TL for IP2

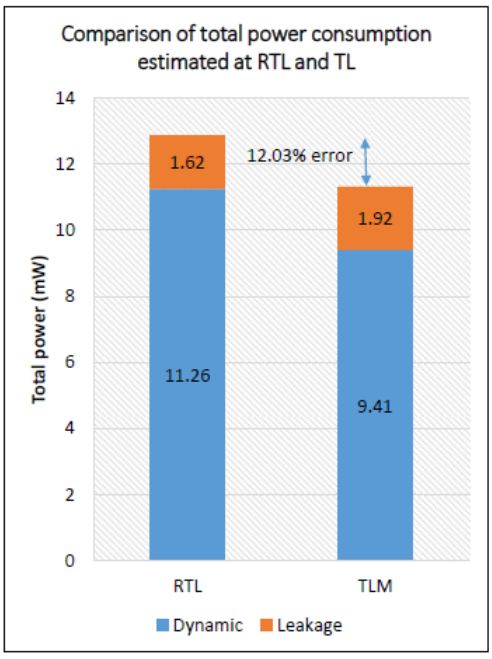

Figure 6.18: Comparison of total power consumption estimated at RTL and TL for IP2

6.3.3 Power Consumption of IP3

The comparison of dynamic power consumption, leakage power consumption and total power consumption obtained at TL and RTL for IP3 is shown in fig. 6.19, fig. 6.20 and fig. 6.21 respectively. It is observed that dynamic power consumption is estimated at transaction level with an error of 16.18% with respect to RTL, leakage power consumption with an error of 18.97% and total power consumption with an error of 8.13%.

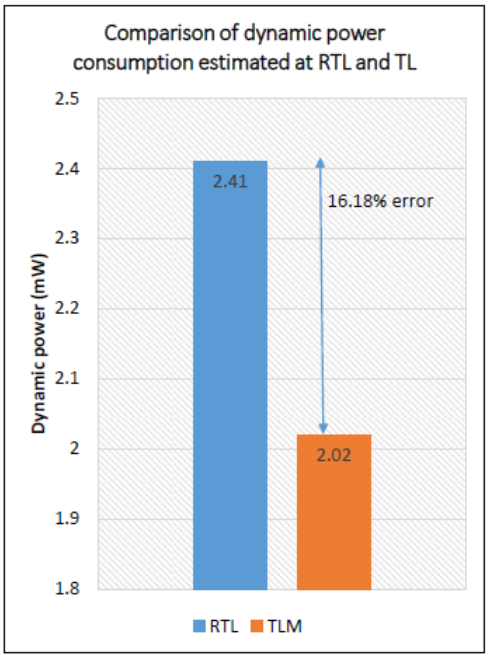

Figure 6.19: Comparison of dynamic power consumption estimated at RTL and TL for IP3

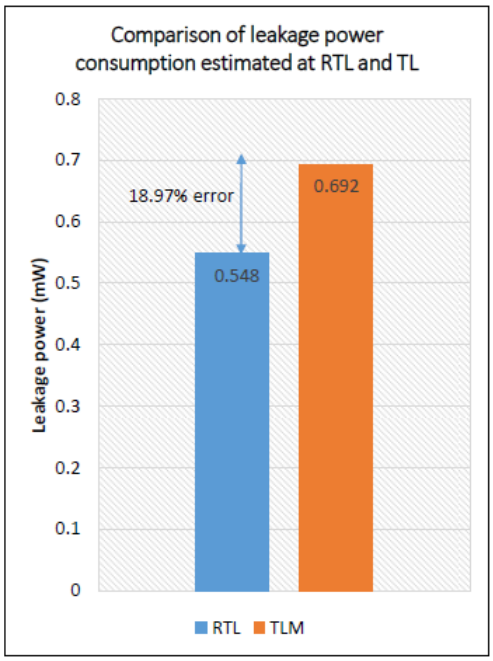

Figure 6.20: Comparison of leakage power consumption estimated at RTL and TL for IP3

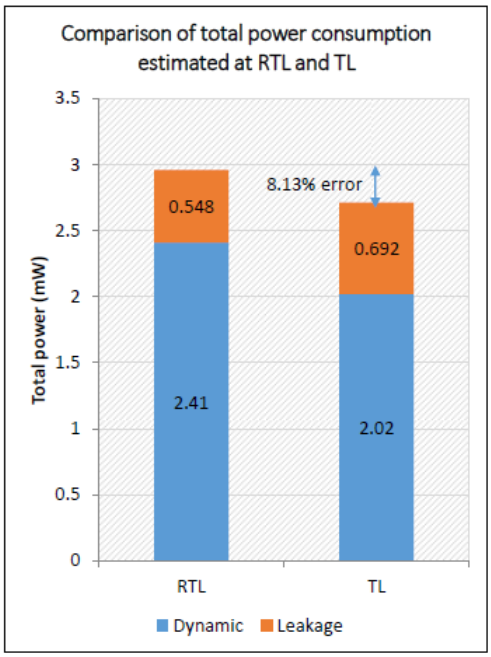

Figure 6.21: Comparison of total power consumption estimated at RTL and TL for IP3

6.3.4 Analysis of Result

6.3.4.1 Dynamic Power Consumption

The dynamic power consumption estimated at transaction level using the proposed approach is less than that computed at RTL because the dynamic power obtained at transaction level does not take into account the power dissipated during switching of nets in the design. The power dissipation due to net switching cannot be accounted for at transaction level as information regarding interconnection of different components is available only after completion of synthesis step.

6.3.4.2 Leakage Power Consumption

The leakage power consumption estimated at transaction level using the proposed approach is higher than that computed at RTL. This is because the proposed approach extracts the maximum leakage power from the technology library for each cell irrespective of the state of the cell, which is not the case at RTL.

6.3.4.3 Trends of Error in Power Consumption

The trends of error in power consumption, as inferred from the plots depicted in section 6.3, indicate that the error in dynamic power consumption and leakage power consumption for different IPs is almost constant and within 20% with respect to RTL while error in total power consumption is within 12% of RTL. The said trends, thus, tend to vouch for the efficacy of the proposed approach for estimation of power consumption of different IPs at transaction level.

6.3.5 Comparison of Proposed Approach with Existing Approaches

[2] presents comparison of power figures obtained using Prime Power and their proposed methodology for power estimation at TL for Network-on-Chip and Fast Hamdard Transform. The relative error in two cases is found to be within 10%.
[4] presents comparison of power figures obtained at gate level and those estimated at transaction level. The relative error between the power consumption values obtained at two levels is stated to be within 12%. The speed-up has been mentioned only with respect to gate level simulation.
In [8], the error in power consumption values for 4,8,16 bit adders is within 7% with

respect to exact power model.

The proposed approach estimates dynamic and leakage power consumption value with an error within 20% of RTL and total power consumption within 12% with respect to RTL. Thus, apparently it may appear to be less accurate vis-a-vis the data referred above, but at the same time, one needs to take into consideration that the proposed approach, in contradistinction to the traditional approaches, takes advantage of simulation at transaction level rather than at RT level to extract parameters required for power estimation at transaction level. Considering the aforesaid and the fact that relative values are more important for design space exploration at transaction level rather than absolute values, the proposed approach offers multiple advantages over other approaches proposed in literature for estimation of power consumption of IPs at transaction level. Firstly, the simulation at transaction level is much faster compared to simulation at RT level, hence, enables huge savings in time. Secondly, the proposed approach releases the resources, for utilization to perform other tasks, at a faster rate compared to traditional approaches, which occupy the resources for longer periods of time due to their slow simulation speed. Therefore, the proposed approach enables efficient utilization of resources. Thirdly, the proposed approach, unlike traditional approaches, makes use of existing transaction level models without any major alterations in the existing code, thus, enables huge savings in modeling efforts. Conclusively, it may not be inept to state that the proposed approach offers significant advantages over the traditional approaches.

7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, an approach has been presented to estimate power consumption of an IP at system level based on the SystemC transaction level model of the specific IP. The accuracy of the proposed approach is within 88% of RTL and enables speed-up of 10x compared to RTL. This approach can be used for getting an idea of the power consumption of the SoC at system level with respect to RT level. The knowledge of power consumption of silicon at system level would enable the designer to effectively explore the design space and optimize the design to reduce power consumption in the final product, thus, enabling the designer to make an informed decision regarding the choice of system architecture at system level itself. This prevents the need to go to the RT level to obtain information regarding power consumption of the silicon and re-designing/re-spin in case the specified power constraints are not met. Hence, the utility of the proposed approach is borne out by the fact that this approach, if implemented, has great potential for bringing about substantial savings in terms of time, efforts and resources.

The following have been achieved in this dissertation:

1. Transaction level model of IP1 has been developed and functionally verified against its specifications.

2. A methodology has been proposed to estimate total power consumption, composed of dynamic and leakage power consumption, of an IP using SystemC transaction level model of the corresponding IP.

3. The effectiveness of the proposed methodology for estimation of power consumption at transaction level has been shown by estimating dynamic, leakage and total power consumption of an IP and comparing them against the corresponding power figures obtained at RTL of the specific IP.

4. The proposed methodology has been verified by comparing power results for three different IPs.

5. The accuracy of the proposed methodology in estimating power consumption at transaction level is within 88% of RTL.

7.2 Future Work

The future work encompasses application of the proposed approach to other IPs and augmentation of the proposed approach with power models of processor, memory and bus, which would enhance the predictability and reliability of the proposed model.

References

[1] G. Qu, N. Kawabe, K. Usarni, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," in *Design Automation Conference, 2000. Proceedings 2000*, pp. 810–813, 2000. 5

[2] H. Lebreton and P. Vivet, "Power modeling in systemc at transaction level, application to a dvfs architecture.," in *ISVLSI*, pp. 463–466, IEEE Computer Society, 2008. 5, 46

[3] L. Zhong, S. Ravi, A. Raghunathan, and N. Jha, "Rtl-aware cycle-accurate functional power estimation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, pp. 2103–2117, Oct 2006. 5

[4] V. Narayanan, I.-C. Lin, and N. Dhanwada, "A power estimation methodology for systemc transaction level models," in *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, pp. 142–147, Sept 2005. 6, 46

[5] R. Ben Atitallah, S. Niar, and J. Dekeyser, "Mpsoc power estimation framework at transaction level modeling," in *Microelectronics, 2007. ICM 2007. Internatonal Conference on*, pp. 245–248, Dec 2007. 6

[6] D. Greaves and M. Yasin, "Tlm power3: Power estimation methodology for systemc tlm 2.0," in *Specification and Design Languages (FDL), 2012 Forum on*, pp. 106–111, Sept 2012. 6

[7] N. Bombieri, F. Fummi, V. Guarnieri, and A. Acquaviva, "Energy aware tlm platform simulation via rtl abstraction," in *High Level Design Validation and Test Workshop (HLDVT), 2012 IEEE International*, pp. 156–163, Nov 2012. 6

[8] R. Damasevicius and V. Stuikys, "Estimation of power consumption at behavioral modeling level using systemc.," *EURASIP J. Emb. Sys.*, 2007. 6, 46

[9] F. Ghenassia, *Transaction-level modelling with SystemC: TLM concepts and applications for embedded systems*. Springer, 2005. vi, 9, 11, 13

[10] "Power compiler user guide." http://faculty.kfupm.edu.sa/coe/aimane/coe561/Design vi, 20

[11] "Low power circuit design." vi, 21