An Expert System Based Tool for Pre-design Chip Power Estimation

Bhanu Singh, Arunprasath Shankar, Francis Wolff, Christos Papachristou
Dept. of EECS, Case Western Reserve University, Cleveland, USA

Abstract—**The power specification of a chip is generally developed during the product conception stage. Engineers select technology libraries, IPs, and make architectural decisions by weighing them against the impact on the power budget. In this paper, we present a knowledge-base (KB) system for pre-design chip power estimation, which uses post-layout power information from previous designs. KB rules are then used to perform specification analysis of a new design, against the KB, to estimate its power. We demonstrate our power estimation technique on a LEON3 System-On-Chip (SoC) design.**

Keywords-**Knowledge-Base System, Specification Analysis, Power modeling, Pre-design chip power**

I. Introduction

The international technology roadmap for semiconductors (ITRS) has listed power as one of the five cross-cutting challenges for the semiconductor industry. Low-power design has become an important design constraint due to requirements of extended battery life in portable devices. For semiconductor companies, while trying to improve performance, there is also a sufficient motivation to minimize power consumption under all real workload conditions. In a system, most of the power dissipated gets converted into heat, which adversely impacts the reliability of the device. A SoC, which consumes less power in comparison to other vendor chips, will need a smaller battery for same workload. This enables consumer products, which have smaller form factor and are optimized in terms of power, performance, and price.

The improvements in process technology and manufacturing capability are enabling integration of digital, analog and RF IPs on a single silicon die. It is important that rough power estimation is done for such complex system-on-chip (SoC) designs at product conception stage. In later stages of design cycle, the turn around time is very high and any issue with design severely impacts time to market. The pre-design phase provides maximum opportunity to optimize system architecture, select technology libraries and IPs best suited to meet chip power budget. The analysis can also reveal chip power under various power management scenarios such as supply gating and clock gating.

We consider a common scenario, where marketing team comes up with an initial SoC product specification feeding power numbers as benchmark against other chip vendors. The engineering team then has to do a pre-design power estimation for the proposed chip to make sure that power budget specified by marketing team can be met under different work-load conditions.

In this paper we propose an expert system (KB system) based tool for pre-design chip power estimation. Our approach uses a knowledge base of IP designs characterized for power dissipation at various functional modes and a design database of previous SoC designs. In particular, the paper makes the following key contributions

- It provides a methodology to perform specification analysis of the chip architectural spec document and KB search to provide matching IPs and similar platform based SoCs.
- It provides a methodology to estimate power at full chip level, taking into consideration architectural decision about supply gating, dynamic clock gating, type of IO pads, technology libraries, memory configuration setting, and estimates clock-tree power analytically.

The remainder of the paper is organized as follows. Section 2 provides background and related work done on power estimation. Section 3 introduces our methodology of applying KB system for pre-design chip power estimation. Section 4 discusses IP power characterization of LEON3 processor, IPs, Memories. Section 5 provides a case study of pre-design power estimation performed for a LEON3 SoC. We conclude and provide future directions in section 6.

II. Back ground and Related Work

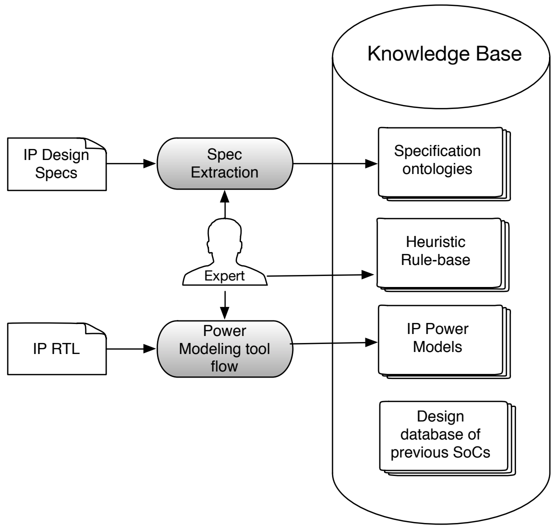
Low power design has been an important research topic and in past solutions have been proposed for power estimation at system level, register-transfer level and gate-level [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]. Power estimation tools are already available from commercial EDA vendors at implementation level (RTL) and circuit level (gate-level). These tools can estimate power consumption very accurately. For gate-level power estimation, a 10 % deviation from real silicon can be reached and for RTL power estimation the deviation is 15-20 % [12]. The highest accuracy in estimating power is achieved by performing power estimation at post-layout gate level netlist but at this stage the power saving opportunities with greatest impact have passed. For system architects, accurate power estimation at early stages of design cycle aids in exploring design space with corresponding impact on power. Early power estimation avoids the lengthy iterations caused by redesigns, required to meet the power budget during later stages of design cycle. We define pre-design as chip project planning stage, when RTL is not available for all the blocks and the architecture is still being defined. EDA power estimation tools can not be easily used in this case as they work on well defined designs, generally at RTL and gate level. At the pre-design phase, the engineers perform design space exploration and chose technology libraries, memory configuration, on-chip bus protocol, IO pads, and IPs to meet system power specification. Traditionally engineers have been using spread-sheet based analytical approach for rough estimation of pre-design chip power. The spread-sheet based approach is very time consuming and error-prone as power estimation is based on engineering judgement of an individual engineer and for complex SoC, all the operating scenarios with various power management techniques may not be covered. The approach is also very ad hoc and does not provide a methodology to capture knowledge obtained from previous designs.

Our work can be roughly related to research done in pre-design chip power estimation at system level. A system-on-chip design is an integration of various heterogeneous components with varying power characteristics (e.g. processors, on-chip buses, memories, caches, custom IP blocks). Therefore suitable power modeling techniques need to be followed depending on the type of component. Power estimation techniques proposed for processors can be divided into structural and instruction based techniques [8] [9] [10] [11]. Structural models use the micro-architecture of the processor to estimate power for each sub-block. In instruction level power modeling, every instruction in instruction-set of a processor is characterized for power dissipated during its execution. Power estimation tools like Wattch and SimplePower have been proposed for processor design domain [13] [14]. The memories and caches can be considered as regular implementations and various analytical models have been proposed for their power estimation [15]. System architects also use memory compiler available from memory vendors to generate various memory configurations and associated power dissipation information [1]. Tools like CACTI have been proposed for analytically estimating power for various memory/cache configurations used in a system [16]. For on-chip buses transaction level modeling (TLM) based power modeling techniques have been proposed at system level [17]. For custom IP blocks and third-party IP blocks, techniques for power estimation have been proposed from cycle accurate or behavioral model of the component [4] [7].

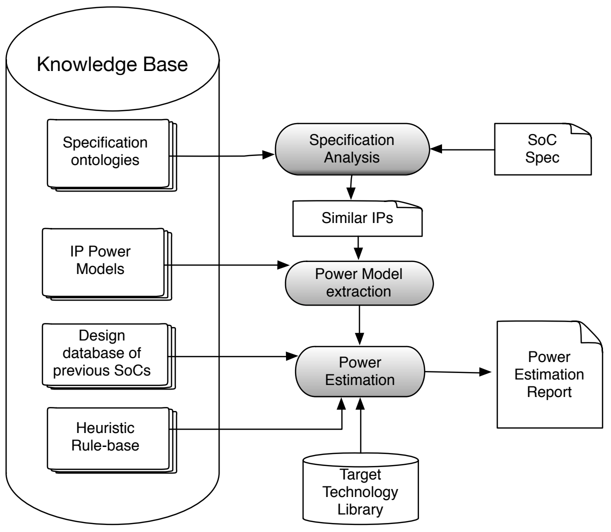
In comparison to existing power estimation techniques, we do not perform any system level cycle accurate simulations for the new design and as such do not assume that system level models are available for each IP component. Our methodology builds a KB system using knowledge from previous designs and experience of design experts captured in a rule-base. The KB is then used for power estimation. KB systems have been proposed in the past for hardware-software partitioning and random stimuli generation for hardware verification [18] [19]. The application of our work is in pre-design chip power estimation, which is different from the above proposed KB systems.

III. Methodology Overview

The main elements of our technique are : (a) a KB of previous SoC designs and IPs (b) Power Modeling of IPs in KB (c) Specification Analysis (d) Performing power estimation of a new design. Figure 1 provides an overview of our approach. We have used CLIPS expert system shell for developing our KB system [20]. A program written in CLIPS consists of facts which act as data and a rule-base to analyze data via the inference engine. For power estimation, this data is power related information in technology library, data-sheets/specs, and functional mode power analysis results of designs in KB. The generation of KB is an ongoing process, as KB is enhanced with each addition of new IPs. The methodology assumes that a functional specification of the SoC design is available. The KB then performs specification analysis and through a rule-based search process determines similar IPs in the KB to the ones in the SoC. The power models of the similar IPs are then used to perform predesign chip power estimation using



(a) KB generation



(b) KB application for Power estimation

Fig. 1. KB system for pre-design chip power estimation

a heuristic rule-base.
Our methodology is based on the assumption that SoC design teams typically practice IP reuse [21]. We assume that IP library and design database of previous SoC's is available in a company. New SoCs are built mostly by reusing and integrating existing IPs that are properly configured according to the specification. Therefore our approach of characterizing and building specification/power models of existing IPs is one time exercise while building the KB. The manual effort is high initially and is minimal afterwards. The reuse of KB across multiple projects justifies the initial effort.

The characterization effort accounts for various IP configurations possible and stores corresponding power model in terms of look up table, mathematical equations. In addition, we also consider platform based SoC, which is a family of similar chips that differ for one or more components but that are based on the same microprocessor. The post layout results for a platform based SoC can be used to estimate clock tree power. We have generated KB from GRLIB IP library available from AEROFLEX GAISLER [22]. In the following sections, we describe each element of our technique.

IV. Knowledge Base Generation

We have developed knowledge base for limited number of IPs, e.g. LEON3, FPU, AHB/APB controller, UART, TIMER, Interrupt Controller etc. The KB generation involves specification ontology generation and power modeling tasks, which are described below.

A. *Specification Ontology*

A specification document in a natural language format is generally used to describe design behavior. It has variety of notations that include text, diagrams, and tables. We define text-objects as words or phrases used to specify design properties. For example, "synchronous", "asynchronous", "booth encoded" are text-objects. Through analysis of different vendor specs, we observed that for a particular design domain, specs exhibit commonality of text-objects. These text-objects are identified from a corpus of specs and then organized into a conceptual network, which we term as "spec ontology". An ontology is defined as a knowledge representation model to explicitly represent a domain by defining its concepts and their relationships [23].

TABLE I
Fragment of spec ontology for AXI Bus

Object	Feature	Synonym	Relation	Type	Attribute
axi4s	axi stream		is-a-property	bool	yes,no
axi4s	data transfer	data exchange	is-a-function	bool	yes,no
axis	interconnect		is-a-function	bool	yes,no
axis	signaling	axi I/O	is-a-io-list	bool	yes,no
...	...	...	...	...	...
signaling	tvalid	valid transfer	is-a-signal	bool	yes,no
...	...	...	...	...	...
data transfer	flow control		is-a-property	bool	yes, no
...	...	...	...	...	...
interconnect	arbitration		is-a-subfunction	set	fixed priority, ..

A spec ontology consists of objects, features and their attributes organized in a hierarchy using relationships. Table I provides an overview of the ontology fragment for the AXI Bus. Each high level concept in the ontology is defined in terms of lower level concepts using relationships, for example “has-function", “has-subfunction", “has-operation" and “has-feature". The relationships create a conceptual network, which corresponds to a specification model for the design. For each IP, such as Leon3 processor, uart, timer etc., specification ontology is super set of concepts for that domain. Each SoC specification, which uses a particular configuration of the IP uses subset of concepts from the IP domain ontology. For example, LEON3 processor in one configurations does not include debug support unit (DSU) & floating point unit (FPU) and in another configuration uses both DSU and FPU. The ontologies that we have developed describe the design using four-level-deep functional decomposition. The first level is soft IP design along with its listed features. The second level has function objects. The third level consists of subfunction objects, which are further decomposed in terms of leaf level operations. For example, in Table I data-transfer is categorized as a AXI function and “flow-control” is its subfunction. We have developed tools to assist experts in ontology creation [24]. We first filter common words such as pronouns, conjunctions etc. by using a custom stop word dictionary. We then perform word frequency analysis and location analysis on a corpus of specs to provide input to an expert about lists of important words for a design domain.

We have also developed a “Spec-Extractor" tool to automate the extraction of text-objects from an IP spec document [25]. A spec document is analysed by first converting it from PDF into XML format using Adobe tools. The spec-extractor then parses the XML and generates a hierarchical tree based on the location of each word in a particular section, subsection, paragraph and sentence of the document. Since each word is a leaf node in the tree, the key idea is the path back to the root. A unique location vector is generated for every word derived from corresponding XML tags for document structure. Spec words are further clustered under categories as, (i) Spec ontology words, (ii) RTL symbols (port and module names) (iii) acronym/abbreviations, (iv) numbers, (v) custom stop words, (vi) English dictionary words, and (vii) unknowns. This captured data is outputted as CLIPS facts, which then gets analysed by the spec rule base. Each spec gets represented in a machine readable format by populating a uniform CLIPS fact template with text-objects existing in that particular spec. This representation enables us to compare the spec of a new design with existing designs in the KB and is independent of the writing style of the document's author. Figure 2 provides an overview of our spec analysis technique. Each IP specification used in previous SoC designs is stored in the KB in a CLIPS fact format. The requirements specification for a new SoC is also converted into CLIPS fact format and is then compared against existing specs in the KB to find matching IPs and SoC platforms.

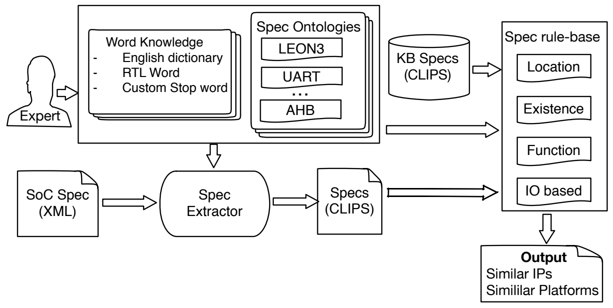


Fig. 2. Specification Analysis flow

The spec rule-base has multiple level of inference rules. The higher level rules perform inferences by combining lower level inferences. There are various category of rules in the rule-base each with a specific purpose. Location based rules use location proximity of text-objects to associate design features with their attributes. For example, for FPU designs, precision is a feature and it has a attribute value of "single" or "double". Function based rules infer design functions based on occurrence of text-objects that identity low level operations and sub-functions listed in the spec ontology. Existence rules check if any key design feature required to perform an inference about a high level design function is missing in the provided specification. For example, rounding feature for floating point unit design. IO based rules check the IO names provided in the specification to infer the bus interface type. Following is an example of a spec rule which associates features with attributes using spec ontologies.

```
(defrule MAIN::find-attribute (declare (auto-focus TRUE))
(ontoFact (object ?obj)(domain fpu)(feature ?f) (relation ?
     rel)(type set)(attribute $?attr))
(wordFact (word ?word1&:(eq ?word1 (string-to-field ?f)))(
     locationId  $?loc1 ?word1loc) (cat onto))
(wordFact (word ?word2&:(member$ (str-cat ?word2) (create$
     $?attr)))(locationId $?loc1 ?word2loc)(cat onto))
=>
(assert (specFact(specId ?*spec*) (object ?obj)(feature ?f)
     (attribute ?word2)(relation ?rel)))
```

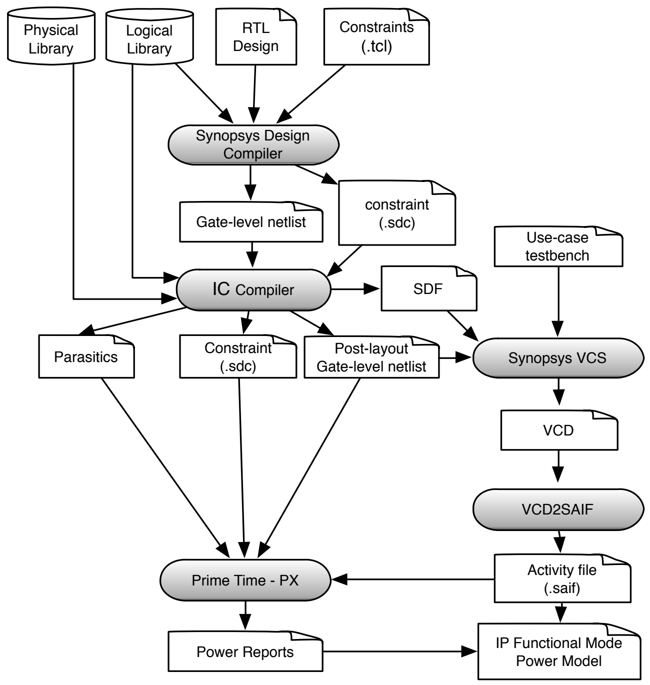


Fig. 3. Power Modeling tool Flow

B. Power Modeling

We have used synopsys design compiler and 90nm generic technology library available from synopsys to perform synthesis [26]. The IC compiler was used to perform place & route. The gate level netlist was simulated using VCS simulator by annotating SDF (standard delay format). Power characterization is performed for each IP in the KB using the tool flow shown in Figure 3 and is described below

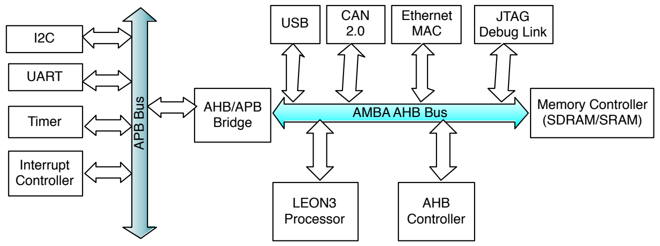


Fig. 4. Example Leon3 System

1) LEON3 Processor: LEON3 is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture and the full source code is available under the GNU GPL license [22] [27]. A typical LEON3 SoC platform consists of a LEON3 processor connected to peripherals over the AHB and APB buses as shown in Figure 4. LEON3 can be customized using the configuration capability provided with the source code by choosing the optional blocks and co-processors as shown in Figure 5. In our KB, we have characterized three configurations, which are minimal, general purpose and high performance for power modeling. We have used the approach proposed by Lee et al. in [7] to separate the processor power model into two parts a) processor core power model b) cache model. Further we consider two states of the core, IDLE and ACTIVE. The IDLE mode corresponds to low power mode of LEON3 as specified in LEON3 specification [22], where clock can be gated to LEON3 at module level and only leakage power comes into picture. The active state corresponds to worst case power dissipation while processor is executing instructions. Table II gives two state power information associated with various LEON3 configuration.

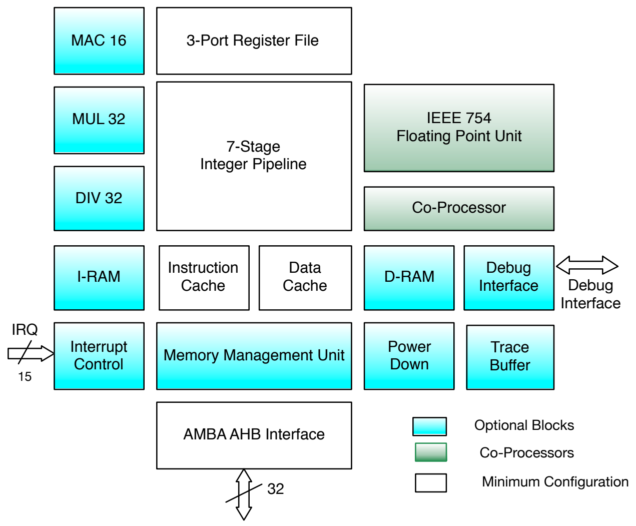


Fig. 5. Leon3 Functional block diagram

TABLE II
POWER INFORMATION FOR THREE CONFIGURATIONS OF LEON3 PROCESSOR

Configuration	Active power (mw)	Idle power (uw)	Number of gates (nand2 equivalent)
Minimum	2.63	87.2	27676
General purpose	6	111.29	35331
High performance	6.74	128.4	40765

2) Cache and Memories: LEON3 has a configurable cache system, consisting of a separate instruction and data cache. Both caches can be configured with 1 - 4 ways, 1 - 256 kbyte/way, 16 or 32 bytes per line [22]. Cache memories are implemented using SRAM blocks from the technology library. Previous works in cache power estimation have used circuit-level information to generate cache power model [15] [16]. In industry, memory compiler from vendors is typically used to generate various memory configurations and the power values associated with read and write access for each configuration [1]. We assume in the architecture file, the designers will provide the physical instance name of the memory cell and memory compiler generated power values. If the functional requirements is to have 128KB memory and it is realized using two physical instances of 64x8, then this information should be entered in the architecture file. Table III gives power information associated with various memory cell configurations.

TABLE III
EXAMPLE - POWER INFORMATION FOR VARIOUS MEMORY CONFIGURATION

Memory configuration	Dynamic Power (uw)	Leakage Power (nw)
SRAM32x256_1rw	36	3.7
SRAM32x64_max	124	66
SRAM39x32_max	71	3.7

3) Other IP blocks: For IP blocks like AHB controller, UART, Timer we follow the two state power model. We characterize each IP for IDLE and ACTIVE mode power dissipation. For active mode, we consider the worst case power dissipation use-case. For example, incase of UART the worst case dynamic power dissipation occurs when it is either in receive or transmit state. Table IV is an example of power information associated with idle and active mode of listed IPs.

TABLE IV
EXAMPLE- POWER CHARACTERIZATION OF SOME IP BLOCKS

Configuration	Active power (uw)	Idle power (uw)	Number of gates (nand2 equivalent)
AHB controller	410	19.98	6424
Timer	228	11.78	3788
UART	160	7.248	2332

4) Clock tree: Clock tree power is typically 30-35% of total dynamic power in the design. The design requirements, floor plan, clock tree building strategy all impact clock tree power. We assume a clock tree building strategy with a very tight local skew and flexible global skew, as in this case less buffers are required to balance clock skew.

Clock tree power estimation can be best performed using a post layout gate level net-list. For estimating clock tree power at pre-design stage, we use previous similar SoC platforms available in the KB. We describe three techniques for performing clock tree power estimation.

a) Using per MHz per gate count power: We use design information of previous similar IPs, to find out the total gate count and total power, then calculate per MHz per gate count power as shown in equation 1. This include both net power and cell power. Clock tree power can then be calculated as shown in equation 2.

$$P_{mg} = \frac{P_{ip}}{F_{ip} * nand_{eq}} \qquad (1)$$

where, P_{mg} = per MHz per gate power,
P_{ip} = Power of IP in previous design
F_{ip} = Frequency at which power was estimated
$nand_{eq}$ = NAND2 equivalent area

clock tree power =P_{mg} * (gate count) * (toggle rate) (2)

b) Using clock power per flip-flop: Another technique is to estimate clock tree power by scaling results from a similar design in KB [1]. The scaling is done through a set of heuristic rules, which capture engineering judgement of designers. For example, a clock-tree in previous design is analyzed to estimate clock power per flip-flop. For a new design, using information from designs in KB, an estimation is performed for clock-tree load counts. To account for library and process changes, flip flop load scaling factor is estimated by comparing clock-pin input capacitance of flip-flops in previous design with similar flip-flops in the new design. Finally estimation of power per clock in the new design is performed by scaling frequency and load counts.

c) Using flip-flop count and clock-buffer fan-out: Alternatively, we can also analytically calculate clock tree power. In a tree structure, typically one buffer will drive three or four sinks (buffers) and each of the three will further drive three. Let us say **F** is number of flip flops in a design and **f** is fanout of each clock buffer. Therefore, at leaf (flop) level the number of clock buffers are

$$\frac{F}{f}$$

Further at leaf level minus one, the number of buffers are

$$\frac{F}{f^2}$$

The number of levels **n** in clock-tree can be calculated using the condition listed in equation 3.

$$\frac{F}{f^n} \geq f \tag{3}$$

$$n = \frac{log(F)}{log(f)} - 1 \tag{4}$$

The total number of clock buffers can be calculated using the following equation

$$N_{buf} = \frac{F}{f} + \frac{F}{f^2} + \frac{F}{f^3} + ... + \frac{F}{f^n} \tag{5}$$

where, n is given by Eq 4.

Equation 5 is a geometric series and following formula can be used to find sum of first n terms of the series.

$$S_n = \frac{a(1 - r^n)}{1 - r}, r \neq 0 \tag{6}$$

where in our case, $a = \frac{F}{f}$, $r = \frac{1}{f}$, n is calculated from equation 4

If we have data about number of flop flops in a design and assuming certain fan-out number, we can calculate number of clock tree buffers in a design using above equations. Now based on the buffer size and power of selected buffer, we can estimate the total power of all the buffers in the clock tree. We can use information from similar designs in KB to find the buffer types normally used for clock-tree synthesis. Normally, buffers of middle strength are used for building clock tree. They will drive sufficient number of buffers without taking too much power or area. The selection of higher drive strength buffers lead to electromigration issues and is avoided.

5) IO PAD: IO pad selection is based upon functional, electrical and physical aspects. Pads are selected based on required functionality of being input/output/bi-directional, requiring weak/nominal pull up/down and depending on maximum frequency. Electrically, we need to see IO voltage support (1.8/2.5/3.3v), drive strength, slew rate, and max load. Physically, we need to see linear/staggered pads (depends upon die size + total IO count). The system designers choose the IO Pads and list them in the architecture specification file. The rule-base uses the power table listed for the corresponding IO pad in the .lib technology file to use dynamic and leakage power numbers for the pad.

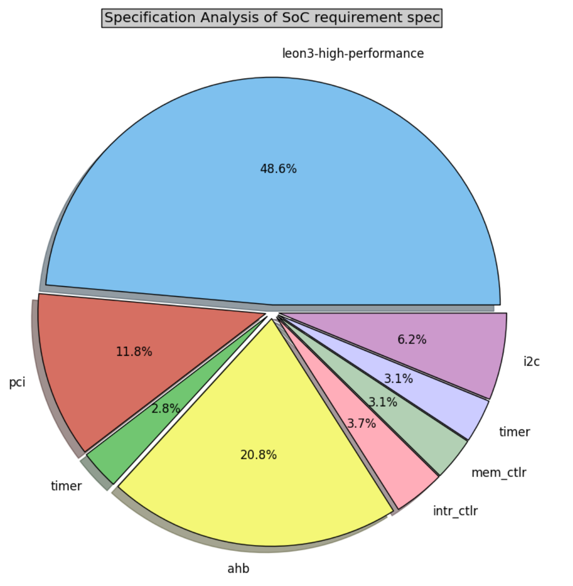


Fig. 6. SoC Spec mapping with various IP domain ontologies

V. Results

We have built a prototype expert system tool using IP/-SoC designs available in grlib-gpl-1.3.1-b4135 version of GRLIB IP library from AEROFLEX GAISLER [22]. To calculate power for any design, certain information is needed from the process technology to which the design is being targeted and data specific to the design like number of gates, types of IOs, power domain voltages, etc is required. The input to the tool is provided using **tech-data** and **design-data** entry (architecture specification) spread-sheets. The technology specific data required for power calculation is extracted from standard cell and IO data-sheets provided by the vendor and from the technology .lib files. For memories, power information in vendor data-sheets or power estimated by memory compiler for a particular configuration are entered into tech-data sheet.

As a case study, we created architecture specification for a reference SoC design from GRLIB library. We then performed specification analysis on the SoC spec. In Figure 6, IPs in an SoC spec that were identified as similar to ones in the KB are shown. The figure also shows the percentage of concepts that mapped to corresponding IP domain ontology. The information of the matching IPs was listed in a preliminary design-data sheet in a csv format. We observed that certain IPs were not identified as they were not in the KB and inputs were manually provided for those IPs in the design-data sheet.

We partitioned SoC design architecturally into various power domains. We assumed clock gating cells for each module are available at root level. We worked out modes of the design like active, clock-gated, and supply-gated. We also generated post-layout power results using the tool flow shown in Fig. 3. The experiments show that power estimation results computed by our tool correlate with +/- 20 % accuracy with post layout power estimation results obtained from Synopsys primetime tool. The inaccuracies in the system are currently due to following reasons.

a) IP or its Configuration not existing in KB: The rule based matchings are only successful if the IP is in the KB. In case, the IP power model is not in the KB, then we ask user to provide the equivalent NAND2 gate count for the IP and calculate IP power by multiplying gate count with static/dynamic power listed for NAND2 gate in the tech library. Estimation error can also be due to configuration differences between IP in the KB and the IP used in the SoC.

b) Switching activity: Currently we have not built an architectural level SystemC model for LEON3 platform. We are not doing any use case simulation at architecture level, and as such do not have switching activity information for the SoC, which results in an estimation error for dynamic power.

c) Clock tree power: We use previous designs to analytically estimate clock tree power for a new design. Two designs may have similar type of IPs but different architectures and as such floor plan. There is a variation in actual clock tree buffer count, interconnect capacitance values and other layout dependent parameters due to different clock tree synthesis results. This also results in estimation error.

VI. Conclusion

We have presented an expert system based technique for pre-design chip power estimation. The system is still under development and in future, we plan to build systemC based architectural level model of LEON3 SoC platform. The architectural level model will enable us to get real work load information while running benchmark applications and use this information to estimate activity in new design. We also plan to improve our prototype power estimation tool by characterizing IPs for more than two power states.

References

[1] B. Eisenstadt, *Making ASIC power estimates before the design*, www.edn.com

[2] S. Penolazzi, et al. *A General Approach to High-Level Energy and Performance Estimation in SoCs*, International VLSI Design Conference 2009

[3] Y. Park, et al.*System Level Power Estimation Methodology with H.264 Decoder Prediction IP Case Study*, ICCD 2007

[4] N. Bansal, et al. *Power Monitors: a framework for system level power estimation using heterogeneous power models*, 18th International VLSI design Conf. 2005.

[5] M. Onouchi, et al. *A system level power-estimation methodology based on IP-level modeling, power-level adjustment and power accumulation*, ASP-DAC 2006.

[6] E. Maccii, et al. *High Level power modeling, estimation and optimization*, DAC 1997

[7] I. Lee, et al. *Power ViP: SoC power estimation framework at transaction level*, ASP-DAC 2006

[8] M. Sami, et al., *Instruction-level power estimation for embedded VLIW cores* in Proc. CODES 2000

[9] N. Kavvadias, et al. *Measurement analysis of the software-related power consumption in microprocessors*, in Proc. IMTC 2003

[10] J. Laurent, et al. *Functional level power analysis: an efficient approach for modeling the power consumption of complex processors* , DATE 2004

[11] S. Penolazzi, et al. *Energy and Performance Model of a SPARC Leon3 Processor*, 2009 Euromicro Conference on Digital System Design/ Architecture, Methods and Tools

[12] P. Soulard, et al. *Accurate System Level Power Estimation through Fast Gate-Level Power Characterization*, www.design-reuse.com

[13] D. Brooks, et al. *Wattch: a framework for architectural-level power analysis and optimizations*, in Proc. ISCA 2000

[14] W. Ye, et al. *The design and use of SimplePower: a cycle accurate energy estimation tool*, DAC 2000

[15] M. Mamidipaka, et al. *Analytical Models for Leakage Power estimation of Memory Array Structures*, CODES 2004

[16] N. Muralimanohar, et al. *CACTI 6.0: A Tool to Model Large Caches*, Technical Report HPL-2009-85, HP laboratories

[17] M. Caladari, et al. *System-Level Power Analysis Methodology Applied to the AMBA AHB Bus*, DATE 2003

[18] M. L. Lopez, C. A. Iglesias, J. C. Lopez, *A knowledge-based system for hardware-software partitioning*, DATE 1998.

[19] Y. Naveh, M. Rimon, I. Jaeger, et al., *Constraint-Based Random Stimuli Generation for Hardware Verification*, AI Magazine, Vol 28, 2007

[20] J. C. Giarratano, G. D. Riley, *Expert Systems: Principles and Programming*, 4th ed.: Thomson Course Technology, 2005.

[21] M. Keating and P. Bricaud, *Reuse Methodology Manual: For System-on-a-chip Designs*, 3rd ed. Boston, MA:Kluwer, 2002.

[22] AEROFLEX GAISLER, *htttp://www.gaisler.com*

[23] Z. Li, K. Ramani, *Ontology-based Design Information Extraction and Retrieval*, Journal of AI for Eng. Design, Analysis and Manufacturing, 2007.

[24] A. Shankar, B. Singh, et al. *NEFCIS: Neuro-Fuzzy Concept based Inference System for Specification Mining*, ICTAI 2013.

[25] B. Singh, A. Shankar, et al. *Knowledge-Guided Methodology for Specification Analysis*, ICTAI 2013

[26] Synopsys, *http://www.synopsys.com/Community/UniversityProgram/Pages/Library.aspx*

[27] *The Sparc Architecture Manual, Version 8*, http://www.sparc.com/