Natural Language based Power Domain Partitioning

David Lemma[1] Daniel Große[1,2] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{lemma,grosse,drechsle}@informatik.uni-bremen.de

***Abstract*—The increased importance of power consumption as a design factor is now undeniable. Power aware design flows are increasingly targeting high abstraction levels (e.g. ESL), where optimization gains are bigger. The designers are thus required to define the power intent already at these levels. Here the major challenge is to perform power domain partitioning. However, this is a fully manual step based on reading and understanding the system specification, and it has to be performed before the Virtual Prototype (VP) is built.**

This paper presents an approach to aid architects in specifying power intent by suggesting coarse-grained power domain partitioning schemes, as the VP is built. The approach starts with structural and behavioral information being extracted from the system specification using Natural Language Processing (NLP) techniques. Then, a semantic network map is created which depicts the hierarchical structure and the abstract block level dependencies that can be used as a foundation for the VP. Finally, a partitioning scheme is derived from the application of an extendable set of analytic rules. Experimental results on an encoding system demonstrate the applicability and efficacy of the proposed approach.

I. Introduction

Both the boom of devices running on batteries and continuous miniaturization have put power-aware design methodologies into the spotlight. Design concerns, especially those concerning static power consumption, are becoming more and more important each day and are now at the same level of relevance as functional requirements. The influence of power concerns extends to the overall design, leading designers to specify the power intent at higher levels of abstraction, where the most significant reduction of power consumption can be achieved [1], [2]. Research on power aware design for the *Electronic System Level* (ESL) has grown considerably, because of its potential to address the increasing system wide impact of power related decisions [3]–[5]. In a typical ESL top-down design flow a SystemC-based *Virtual Prototype* (VP) is created from the specification. Essentially, a VP is a software simulation model of the entire hardware platform, formed by composing *SystemC/TLM* [6] models of the individual IP blocks. This VP is then used as a reference for (early) embedded software development, hardware verification and power verification.

Designers involved in power aware design efforts need to decide the power architecture concurrently with creating the VP, since the prototype will be built conforming to that architecture. Given the nature of the work, designers in charge of the decisions effectively act as system architects. In the decision process undertaken by the system architects, a significant step is power domain partitioning. This step implies grouping components in power domains (lists of components sharing the voltage and ground levels) that will be managed through a power management logic. The goal of the management policy is to reduce the power consumption without compromising the functionality of the system.

This work was supported in part by the University of Bremen's graduate school SyDe, funded by the German Excellence Initiative, and by the German Academic Exchange Service (DAAD).

One natural way to reduce power consumption is to power off parts of a design not performing any task at a given time. The procedure (known as *power gating*) is one of most powerful and frequently used techniques to curb static and dynamic power consumption [7]. At a first glance, the simplicity of the technique is striking, but it relies on an adequate power domain partitioning scheme. When determining such a scheme, system architects are faced with trade-offs, having to make a decision at global scale. A coarse-grained partitioning scheme usually leads to reduced savings, a fine-grained one to area and verification overhead [8]. Additionally, finding a suitable power domain partition scheme is fully dependent on carefully reading and understanding the system specification.

In this paper we propose an approach to address the challenge of power domain partitioning before the VP is built. To the best of our knowledge, this is the first approach that derives a power domain partitioning scheme directly from a natural language based specification. Our approach assists the system architects in implementing the VP with a suitable power aware architecture. The proposed approach consists of four steps. They cover the analysis of the specification document using *Natural Language Processing (NLP)* techniques to extract structural and behavioral information, forming a semantic network map to represent the hierarchical structure and the abstract block level dependencies and defining and applying analytic rules to determine a coarse-grained power domain partitioning scheme.

The structure of the paper is as follows: Section II reviews related work. In Section III the used NLP techniques are explained succinctly. Section IV introduces the proposed approach. The experimental evaluation is given in Section V. Finally, the paper is concluded in Section VI.

II. Related work

Determining a suitable power domain partitioning scheme is a challenging task along the design [9]. The unavoidable trade-off involving the power reduction achieved and the added complexity becomes noticeable for non-trivial designs. Given the general intractability of the problem [10], researchers have approached it using sophisticated optimization algorithms, see e.g. [11]–[13]. Methodologies using these algorithms rely on the assumption that the designers have access to circuit-level information. However, this limits the applicability of the methodologies for architectural system-level exploration and consequent decision making [14].

Other research lines involving system-level space exploration focus on the information contained in specifications. Several works have highlighted the benefits of specification analysis [15], usually through NLP or related techniques [16]. For instance, concept mining applied to a set of specifications can help designers extract knowledge of architectural value [17]. Research and development of proper ontologies for knowledge extraction from specifications is, unfortunately, still in its infancy [18].

Singh et al. [19] have based themselves on previous work on specification analysis to peform pre-design power analysis. Their work is one of the few approaches on system level power aware design related to natural language. However, they target power estimation via a knowledge base generated through an ontology, not power domain partitioning.

Our paper focuses on how to extract knowledge from specifications (or parts of it) by focusing on the relations between components. These relations contain relevant information used by our approach to aid system architects in reaching a suitable power domain partitioning scheme. The output of the approach (the partitioning scheme) can be fed to optimization algorithms if desired.

III. Natural Language Processing Technique

In this section we describe the NLP techniques used to analyze the system specification. While the focus is on the context of a system specification, some aspects are presented from a general viewpoint.

A system specification describing the behavior (an algorithm) or the block structure of a system contains information in the form of entities. An entity can be broadly defined as a self standing notion. A conceptual entity can be taken as a notion that represents a concept in the system specification (for example, an object). To retrieve this information it is necessary to use techniques such as *Information Extraction* (IE) which we will explain in the following.

IE is the task that consists of automatically eliciting relevant data from free texts. The texts do not need to be entire documents, as a few sentences may be enough to extract the information desired. IE is composed of several subtasks, among which two are central: *entity extraction* and *relationship extraction*. The former can be described as retrieving the relevant entities and the later is characterized by obtaining the links that these entities have to each other. The output of both subtasks taken together constitutes information useful to conduct further analysis on the meaning of the text.

In order to showcase the two subtasks, we will take the following sentences from the specification of the FIR filter example shipped with the official SystemC distribution:

> *The filter is a 16 tap FIR filter(fir.cc). The test bench feeds simply ascending values into the FIR(stimulus.cc) and the output is sampled (display.cc) and displayed with print statements.*

Based on this excerpt entity extraction and relationship extraction work as follows:

TABLE I
ENTITY EXTRACTION

display.cc
fir.cc
fir
output
print
statement
stimulus.cc
testbench
value

TABLE II
RELATIONSHIP EXTRACTION

Entity 1	**Type of relationship**	**Entity 2**
print	associatedWith	statement
Fir	associatedWith	fir.cc
fir	associatedWith	stimulus.cc
Fir	hasDeterminer	the
statement	hasQuantifier	multiple
value	hasQuantifier	multiple
Fir	hasDataValue	16
fir	hasDeterminer	the
output	hasDataValue	the
Fir	hasQuality	Tap
tesbench	hasDeterminer	the

1) Entity Extraction (EE): This subtask happens after chunking (parsing of sentences using syntactic rules). Ontologies are heavily preferred if they are available, but the step can still be carried out without domain specific knowledge.

Running the FIR filter specification (see above) through an EE tool gives a list whose elements are relevant entities retrieved from the excerpt, as shown in Table I. Some of the entities in the table may refer to concepts of architectural value in the domain of the text under analysis. For instance, any designer familiar with System on Chip (SoC) design would conclude that the following pairs of entities refer to the same concept given in brackets: $fir.cc$ and fir (FIR), $output$ and $display.cc$ (Output), $value$ and $stimulus.cc$ (Input).

While $fir.cc$, $stimulus.cc$ and $display.cc$ are conceptual entities which embody the important objects of the design, the table also contains other entities that represent properties and actions ($print$, $statement$, $value$) related to the objects.

2) Relationship Extraction (RE): The list of entities extracted from a text have semantic links between them that are detected and classified in this step. Semantic links are the relationships held by the entities. As in Entity Extraction, ontologies are of great assistance in this activity, but a more shallow approach can also be taken. This usually implies lexical semantics and the use of lexical databases [20], which still gives acceptable results.

An example output applying RE to the FIR specification text is shown in Table II. Each row in the table shows how an entity (first column) is related to another entity (third column) and the type of relationship (second column). As a concrete example consider the first row: the entity *print* is associated with the entity *statement*. Another example with a different relationship can be seen in the next to last line: the entity *FIR* has a *Tap* quality. Relationships of the type $hasQuantifier$,

TABLE III
RELEVANT RELATIONSHIPS FOR ENTITIES

Entity 1	Type of relationship	Entity 2
fir.cc	associatedWith	Fir
stimulus.cc	associatedWith	fir
display	comesFrom	output
display	isRelatedTo	statement

$hasDataValue$ or $hasDeterminer$ should be generically interpreted following the pattern "[Entity 1] has a [Entity 2] property".

Some of the relationships extracted may not be useful to get structural and behavioral information of the design. For instance, the type of relationships $hasDeterminer$ and $hasQuantifier$ are generally of the grammatical type and mostly not useful for system level design understanding. Relationships of the type $hasDataValue$ and $hasQuality$, tend to refer to fixed value properties and may be useful to distinguish instances of entities of architectural value. In general, the type of relationship that gives the most valuable conceptual information is the type $associatedWith$.

As the final result of RE a list of relevant relationships between entities is produced. The produced list shows the link between conceptual entities, as well as the the links these have to other entities representing relevant properties, notions and events. A short example of this list can be found in Table III. The first two rows show that $fir.cc$ and $stimulus.cc$ relate to each other because they are associated with the same entity (fir). The last two rows shows how $display.cc$ is linked to other entities representing notions in the system ($output$ and $statement$).

Overall, the output of the described information extraction process (which combines EE and RE) is further elaborated to allow for deeper analysis. For that we use a semantic network map presenting the conceptual structure of the system. For the FIR example the respective semantic map is depicted in Fig. 1. This figure only demonstrates the principle. How we further improve the semantic network in our approach is detailed in the next section. As can be seen in Fig. 1 there are three square blocks representing the conceptual entities of architectural value, with full line arrows representing the links between them (of sequential nature in this case). In addition, there are three circular blocks representing notions and associated event flows for these entities, linked to the square blocks through dashed lines.

IV. APPROACH FOR COARSE-GRAINED POWER DOMAIN PARTITIONING

In this section the proposed approach for deriving a power domain partitioning scheme from the textual specification is presented. Our approach consists of four steps: block extraction, block relationship extraction, semantic network map creation and power domain partitioning scheme generation. The first two steps use the NLP techniques described in Section III with domain specific knowledge being applied to the results. The third step condenses the information into an

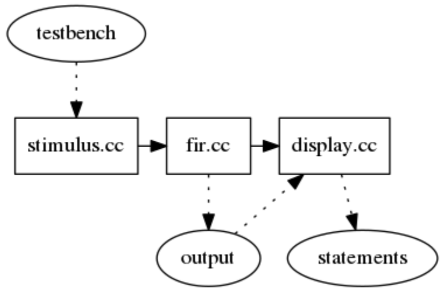

Fig. 1. Semantic network of FIR specification

architectural description of the system, whereas the fourth step involves a rule-based method to generate a suitable power domain partitioning scheme for the system.

To demonstrate the proposed approach we will use an LZW encoder design [21]. An excerpt of its specification reads:

> *The main state machine controls the data movement and control flow for all the total design including the serial port and the LZW encoder block. The input RAM is used to store the data received from the serial port of the PC. The LZW encoder block implements the LZW algorithm, both the control and the data path. The code value RAM is the Hash Table implementation, while the dictionary block implements the LZW dictionary. The output forming logic breaks and merges the 13-bit data output from the LZW data path on correct 8-bit boundaries before writing it to the output RAM. The output RAM is used to store the compressed data which is than transmitted through the serial port to the host PC to be displayed on the terminal.* [1]

The four main steps of our approach are detailed in the following subsections.

A. Block Extraction

In this step the system architect processes the specification text looking for the functional blocks in the underlying system design. To retrieve the blocks, the list of conceptual entities returned by the NLP technique is pruned to only consider those representing objects in the design. Upon doing this, it is possible to discover the overall system architecture. For small designs it is probable that each of functional block is implemented by a single module, but for larger designs this may no longer apply, as not only modules but also submodules may contain the implementation of a functional block.

In the case at hand, 7 blocks have been identified: *Main State Machine, Serial Port, Input RAM, Encoder Block, Hash Table Implementation, Dictionary Block* and *Output Block*.

B. Block Relationship Extraction

In this step the links, that is, the relationship between the identified blocks, are extracted. Some relationships reveal behavioral sequences, others give information about the way in which the blocks influence each other, including their place in the system hierarchy, all of relevance to the next step.

Relevant properties and events associated to the blocks are obtained by applying the type of analysis that was explained in Section II, for the subtask of Relationship Extraction.

[1]It is a verbatim copy used "as is", even when it contains typos.

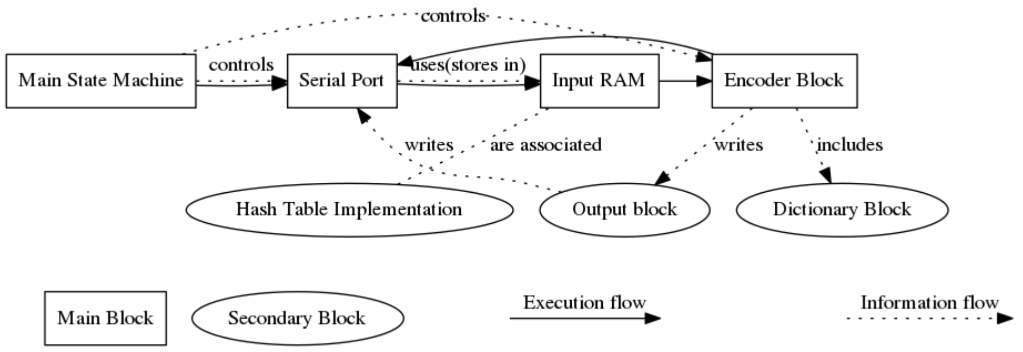

Fig. 2. LZW Encoder semantic network map

C. Creation of Semantic Network Map

In this step, the combined information of the two previous steps is condensed in graphical form within a semantic network map. This allows for easier analysis by the designers by showing the hierarchy and overall design structure of the system. Even if the semantic network is incomplete, the provided information helps decide how to partition the system.

For our design case, the 7 blocks are divided into two groups: the *main blocks* and the *secondary blocks*. The division is done according to their participation (or non-participation) in the top level information flow. The division takes into account the behavioral sequence (that is, the execution flow) and the system hierarchy that were obtained from the previous step. According to that information *Main State Machine, Serial Port, Input RAM* and *Encoder Block* are *main blocks*, whereas *Hash Table Implementation, Output Block* and *Dictionary Block* are *secondary blocks*.

The figure shows that the *Encoder Block* writes to the *Output Block* and includes the *Dictionary Block*, thereby showing why the latter are secondary blocks. Furthermore, the information flow as described in specification text between the main blocks is represented in the semantic network map as can be seen in Fig. 2. This flow represents a basic understanding of the nature of the design. Dashed lines with labels indicate the events that link the blocks. For example, the *Main State Machine* controls both the *Serial Port* and the *Encoder Block*, and the *Output Block* writes to the *Serial Port*.

In most cases, a semantic network map will be similar to a block level diagram of the design, as evidenced by comparing Fig. 2 to the actual diagram block provided by the documentation of the LZW encoder design in Fig. 3.

D. Generation of Power Domain Partitioning Scheme

This step is both the final and the core step of the approach. In this step, analytic rules are applied to the information from the semantic network map. The following set of basic rules are applied by default (but can be overridden later):

R1: Each block belongs to its own domain (this rule produces a very basic power domain partitioning scheme, disregarding hierarchy).

R2: Blocks that are linked to many others are taken to be part of a power domain that is always on, or that controls others and, therefore, has a longer active time.

R3: Any two blocks not directly linked to each other through the execution flow belong to different domains.

R4: Blocks relating to each other through hierarchical dependencies are grouped together in a single power domain.

The rule R1 produces the first estimation for an appropriate power domain partitioning scheme. The rules R1, R2 and R3 refine the initial partitioning into a solid result. The reason behind the need for a refinement is that the rule R1 neglects the associated costs of more power management logic embedded into the design. Each domain requires its own power management logic, regardless of the size or importance of the blocks it contains. The first estimation for the domain partitioning scheme is fine grained, which means that it will probably feature several domains with the same power management logic, thereby implying area/verification/power overhead.

We applied the rules to the LZW encoder to produce the final suggested power domain partitioning scheme taking into account that the blocks refered to as before can be implemented by modules and/or submodules of the system. To take into account the structure of the design we consider the following tenets:

- Hash Table Implementation has a relationship to Input RAM, but it is not a submodule of it and is not a top level functional block/module. Not being able to know its place in the hierarchy of the system, it will not count as a possible power domain.
- Main State Machine has control over both the Serial Port and the Encoder Block, leading to the existence of an always on power domain to which the Main State Machine belongs.
- Input RAM is related to Serial Port and to Encoder Block by means of the execution flow, which shows the sequential nature of the design. These three modules mostly do not share their active/inactive states, hence they are placed in three different power domains.

By following the above, the output of the approach is a power domain partitioning scheme consisting of four power domains,

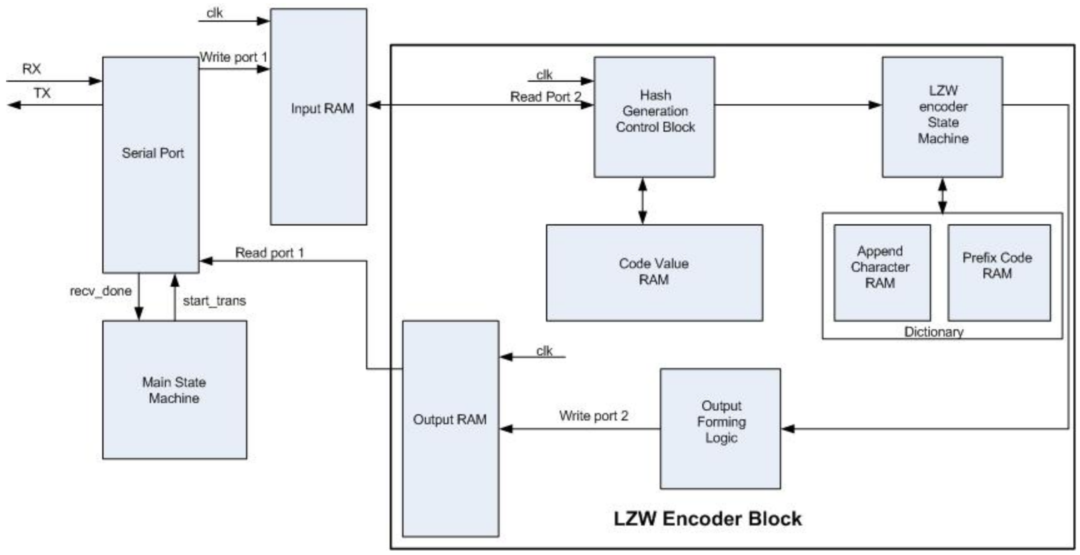

Fig. 3. Block Level Diagram of LZW encoder, based on [21]

namely: **Main State Machine**, **Serial Port**, **Input RAM** and **Encoder Block**. Each of the power domains consists of one module in the design. The validity and quality of this power partitioning scheme is evaluated in the next section.

V. Experimental Evaluation

This section presents the experimental evaluation of the proposed partitioning scheme for the LZW encoder. The scheme determined by our NLP-based approach is ready to be implemented within the VP, but before this is done it is necessary to assess its validity. While the quality (clarity) of the specification text has a direct impact on the output scheme, the effect can be partially offset by intelligently applying the rules, as we did by following the tenets in the previous section. While an intelligent application of the rules requires system architects to have basic knowledge of the expected modules for the design, this is already the case for most (if not all) designers and does not represent a significant barrier.

In order to carry on the qualitative evaluation of the suggested power partitioning scheme, we run the testbenches associated with the LZW encoder SystemC based VP. During the runs, functional checks are performed and information about the model execution is logged. This allows us, for example, to determine the time when the *Serial Port* is active as well as when certain states of the *Main State Machine* have been visited.

We extended the logging to get a complete picture, in particular to also get the time points in which the Input RAM module and the Encoder Block module are active. By analyzing the logged simulation data together with determining the signals that control the activity/inactivity of a module, we construct a basic *Power State Table* (PST) for each of the power domains. Essentially, the PST stores the times for which a power domain is on or off, according to the value of the determined control signals. In the case of the *Main State Machine*, the PST consists of a single ”always on” state, as this module controls the execution flow of the entire system.

Now, the PST is fed into the PKTool, a power estimation tool for SystemC [22]. We use the *fixed_power* model from PKTool, which calculates the energy following the equation $E = PT$, where E is the energy, P is a fixed power value (provided by the us) and T is the simulation time for each state (calculated by the tool through the PST). We adopted a value of 0.005 mW as a static power consumption value for an inactive module and 1 mW for the fixed power consumption of an active module. These values are not based on the actual consumption of the design's modules, but serve to illustrate the differences in power consumption of a power domain when active and when inactive. Accurate estimation of consumption is beyond the scope of this paper.

So as to show that the scheme determined by our approach is a solid decision, we compare the following three schemes:

- *Single domain scheme*: it consists of only one power domain that includes all four of the main blocks and is on an active state throughout the entire time. Essentially, this is equivalent to performing no power domain partitioning.
- *Four domain scheme*: it consists of four power domains (one for each of the main blocks). This is the partitioning determined by our approach.
- *Six domain scheme*: it consists of six power domains (one for each of the main blocks, one for the *Output Block* and one for the *Dictionary Block*). This somewhat corresponds to the output of the approach if only rule R1 is applied.

First we compare the single domain scheme to the four domain scheme by analyzing the results presented in Table IV. The rows in the table represent the schemes under comparison while the columns show the energy consumption for each of the modules and also the total energy consumption. As the single domain scheme simply neglects the activity profile of

TABLE IV
COMPARISON BETWEEN SINGLE DOMAIN AND FOUR DOMAIN SCHEME

	Energy consumption [nJ]				
Partitioning scheme	**Main State Machine**	**Serial Port**	**Input RAM**	**Encoder Block**	**Total**
Single domain scheme	389.23	389.23	389.23	389.23	1556.94
Four domain scheme	389.23	388.03	21.77	26.58	825.75

TABLE V
OUTPUT BLOCK, DICTIONARY BLOCK AND ENCODER BLOCK

Block	Energy consumption [nJ]
Encoder Block	26.58
Output RAM Block	22.40
Dictionary Block	20.76

the modules, the total energy consumption for it is considerably higher than that of the four domain scheme in the second row (15556.94 nJ vs 825.75 nJ, respectively). The first row shows that the consumption of each module (each representing a block) for the single domain scheme is the same, implying that all the modules belong in the same power domain (which follows the activity profile of the Main State Machine module). The second row shows that the four domain scheme considers the activity profile of the modules. Both the *Encoder Block* and the *Input RAM* have activity profiles which show that they are inactive for longer periods of time, thereby consuming less energy than the *Main State Machine*.

From the evaluation above we conclude that the four domain scheme is better than the single domain scheme. We will show that the four domain scheme also compares favorably to the six domain scheme via the comparison of the energy consumption of the *Output Block*, the *Dictionary Blocks* (two secondary blocks) and the *Encoder Block* (a main block). To do this we assume that the Output RAM module corresponds to the *Output Block*, while the Dictionary module corresponds to the *Dictionary Block*, following Fig. 3. The Output RAM module and Dictionary module are active at different periods, but the total time in which they are active is comparable to the active time of *Encoder Block*. That is why the consumption of the three blocks under comparison is similar. We present the comparable values in Table V, where the first row shows the *Encoder Block* consuming 26.58 nJ, the second row shows the *Output Block* consuming 22.40 nJ and the third row shows the *Dictionary Block* consuming 20.76 nJ.

If both the *Output Block* and the *Dictionary Block* constituted extra power domains themselves, there would be a reduction in the overall energy consumption. However, the magnitude of the possible savings does not compare favorably to the associated costs of adding power management logic to these new power domains. The result of the analysis of tradeoff between power consumption reduction and overhead costs leads us to conclude that a six domain scheme brings no real benefit over a four domain scheme. Hence, the four domain scheme is the best scheme among the three compared.

VI. CONCLUSION

We have introduced an approach to derive a power domain partitioning scheme from the textual specification of the system before a Virtual Prototype (VP) is built. Our approach uses NLP techniques within an Information Extraction framework to extract structural and behavioral information from the specification. Then, a semantic network map is created as the basis for a rule-based power domain partitioning scheme. The approach is applied to a LZW encoder VP, with the soundness of the suggested partitioning scheme shown when compared to alternative schemes.

In the future, we want to integrate more powerful information extraction techniques and more refined partitioning rules.

REFERENCES

[1] Z. Zhang, D. Chen, S. Dai, and K. Campbell, “High-level synthesis for low-power design,” *IPSJ Transactions on System LSI Design Methodology*, vol. 8, pp. 12–25, 2015.

[2] Y. Samei, “Automated power-aware system-level design with the mavo framework,” Ph.D. dissertation, University of California, Irvine, 2014.

[3] S. Ahuja, “High level power estimation and reduction techniques for power aware hardware design,” Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2010.

[4] A. Agarwal, “Use of high-level design information for enabling automation of fine-grained power gating,” Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

[5] V. Herdt, H. M. Le, D. Große, and R. Drechsler, “Towards early validation of firmware-based power management using virtual prototypes: A constrained random approach,” in *FDL*, 2017, pp. 1–8.

[6] *IEEE Standard SystemC Language Reference Manual*, IEEE Std. 1666, 2011.

[7] H. Jiang, M. Marek-Sadowska, and S. R. Nassif, “Benefits and costs of power-gating technique,” in *ICCD*, 2005, pp. 559–566.

[8] E. Sperling. How Many Power Islands Is Too Many? Semiconductor Engineering. [Online]. Available: http://semiengineering.com/how-many-power-islands-is-too-many/

[9] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low power methodology manual: for system-on-chip design*. Springer, 2007.

[10] A. Dobriyal, R. Gonnabattula, P. Dasgupta, and C. R. Mandal, “Workload driven power domain partitioning,” in *Progress in VLSI Design and Test*. Springer, 2012, pp. 147–155.

[11] N. Agarwal and N. Dimopoulos, “FSMD partitioning for low power using simulated annealing,” in *ISCAS*, 2008, pp. 1244–1247.

[12] A. Agarwal and Arvind, “Leveraging rule-based designs for automatic power domain partitioning,” in *ICCAD*, 2013, pp. 326–333.

[13] L.-Y. Chiou, Y.-S. Chen, and Y.-L. Jian, “Energy-aware partitioning for on-chip bus architecture using a multi-objective genetic algorithm,” in *VLSI-DAT*, 2011, pp. 1–4.

[14] B. Wang, Y. Xu, R. Hasholzner, C. Drewes, R. Rosales, S. Graf, J. Falk, M. Glaß, and J. Teich, “Exploration of power domain partitioning for application-specific SoCs in system-level design,” in *MBMV*, 2016.

[15] B. Singh, A. Shankar, Y. Shiyanovskii, F. Wolff, C. Papachristou, D. Weyer, S. Clay, and J. Morrison, “Knowledge-guided methodology for specification analysis,” in *ICTAI*, 2013, pp. 749–754.

[16] I. G. Harris, “Extracting design information from natural language specifications,” in *DAC*, 2012, pp. 1252–1253.

[17] A. Shankar, B. P. Singh, F. Wolff, and C. Papachristou, “NEFCIS: Neuro-fuzzy concept based inference system for specification mining,” in *ICTAI*, 2013, pp. 337–343.

[18] A. Shankar, B. Singh, F. Wolff, and C. Papachristou, “Ontology-guided conceptual analysis of design specifications,” in *DAC*, 2014, pp. 1–6.

[19] B. Singh, A. Shankar, F. Wolff, and C. Papachristou, “An expert system based tool for pre-design chip power estimation,” *DVCon*, 2014.

[20] G. A. Miller, “Wordnet: a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[21] Orahyn Ltd. An implementation of LZW encoder in SystemC and verified in FPGA. [Online]. Available: https://github.com/arshadri/lzw_systemc/

[22] G. B. Vece, M. Conti, and S. Orcioni, “Pk tool 2.0: a SystemC environment for high level power estimation,” in *ICECS*, 2005, pp. 1–4.