DESIGN & VERIFICATION DVCON 2014 CONFERENCE & EXHIBITION

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

*Bhanu Singh, Arunprasath Shankar, Francis Wolff and Christos Papachristou.*

Dept. of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY EST. 1826

# Motivation

- The International Technology roadmap for semi-conductors (ITRS) has listed power as one of the five crosscutting challenges for semiconductor industry.

- Low-power design is an important design constraint due to requirements of extended battery life in portable devices.

- It is important that rough power estimation is done for system-on-chip (SoC) designs at product conception stage.

# Motivation (Cont.)

- The pre-design phase provides maximum opportunity to optimize system architecture, select technology libraries and IPs best suited to meet chip power budget.

- 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.

# Current-state-of-the-art

- Power estimation tools are already available from commercial EDA vendors at RTL and 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 %.
- Traditionally engineers have been using spread-sheet based analytical approach for rough estimation of pre-design chip power.
- In research, various system level power modeling techniques have been proposed depending on the type of component.

# Current-state-of-the-art (cont.)

- 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.
- Power estimation techniques proposed for processors can be divided into structural and instruction based techniques.
- 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.

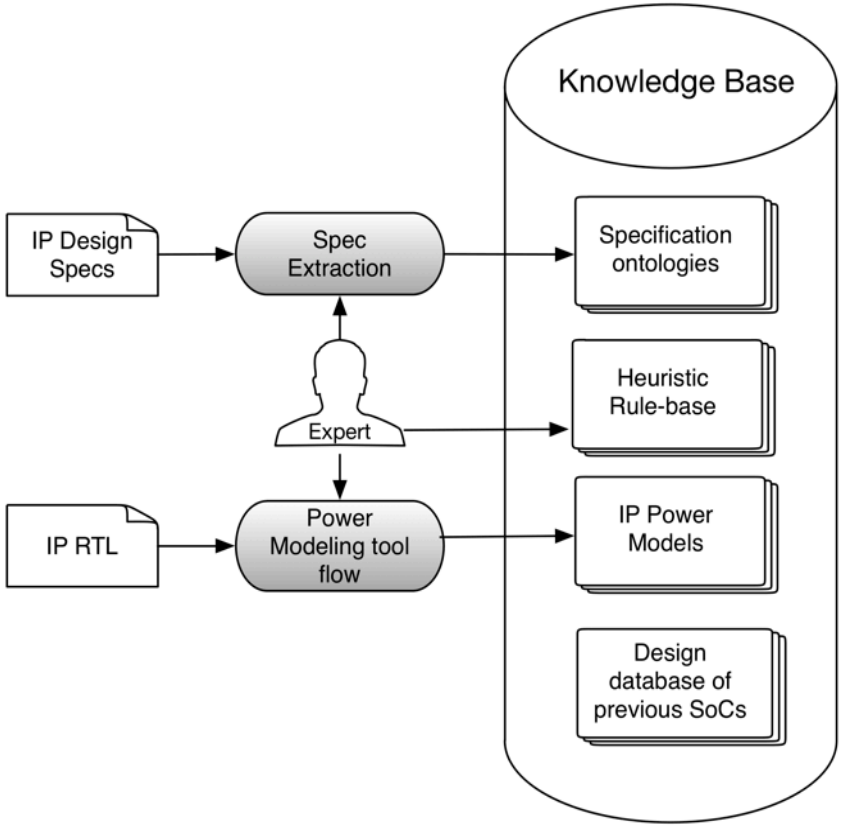
# Current-state-of-the-art (cont.)

- Power estimation tools like **Wattch** and **SimplePower** have been proposed for processor design domain.
- The memories and caches can be considered as regular implementations and various analytical models have been proposed for their power estimation such as tools like **CACTI**
- System architects also use memory compiler available from memory vendors to generate various memory configurations and associated power dissipation information
- For on-chip buses transaction level modeling (TLM) based power modeling techniques have been proposed at system level.

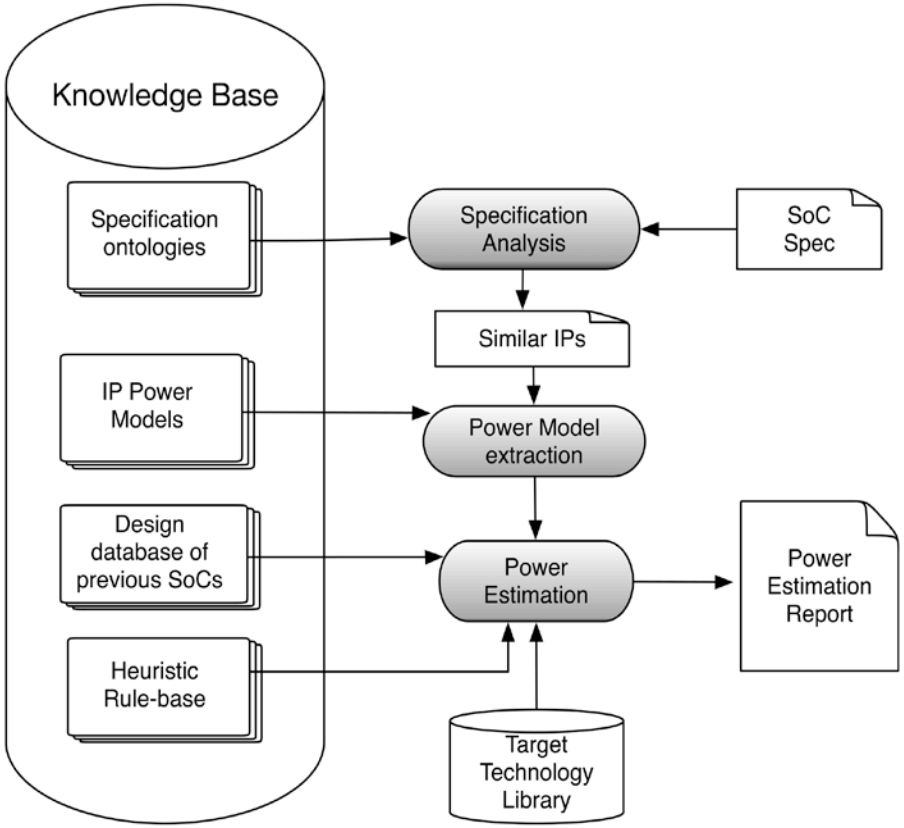
# Proposed Approach

- We propose an expert system (KB system) based tool for pre-design chip power estimation. The expert system that we have used is CLIPS, which is an inference engine originally developed by NASA.
- 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.
- Our approach characterizes a IP design for power dissipation at various functional modes.

# Expert system for Pre-design Power estimation




(a) KB Generation




(b) KB application for Power estimation

# Key elements and Assumptions

Key elements of the system are

- A KB of previous SoC designs and IPs.
- Power modeling of IPs in KB.
- Specification Analysis and KB search to provide matching IPs
- Performing power estimation using the KB.

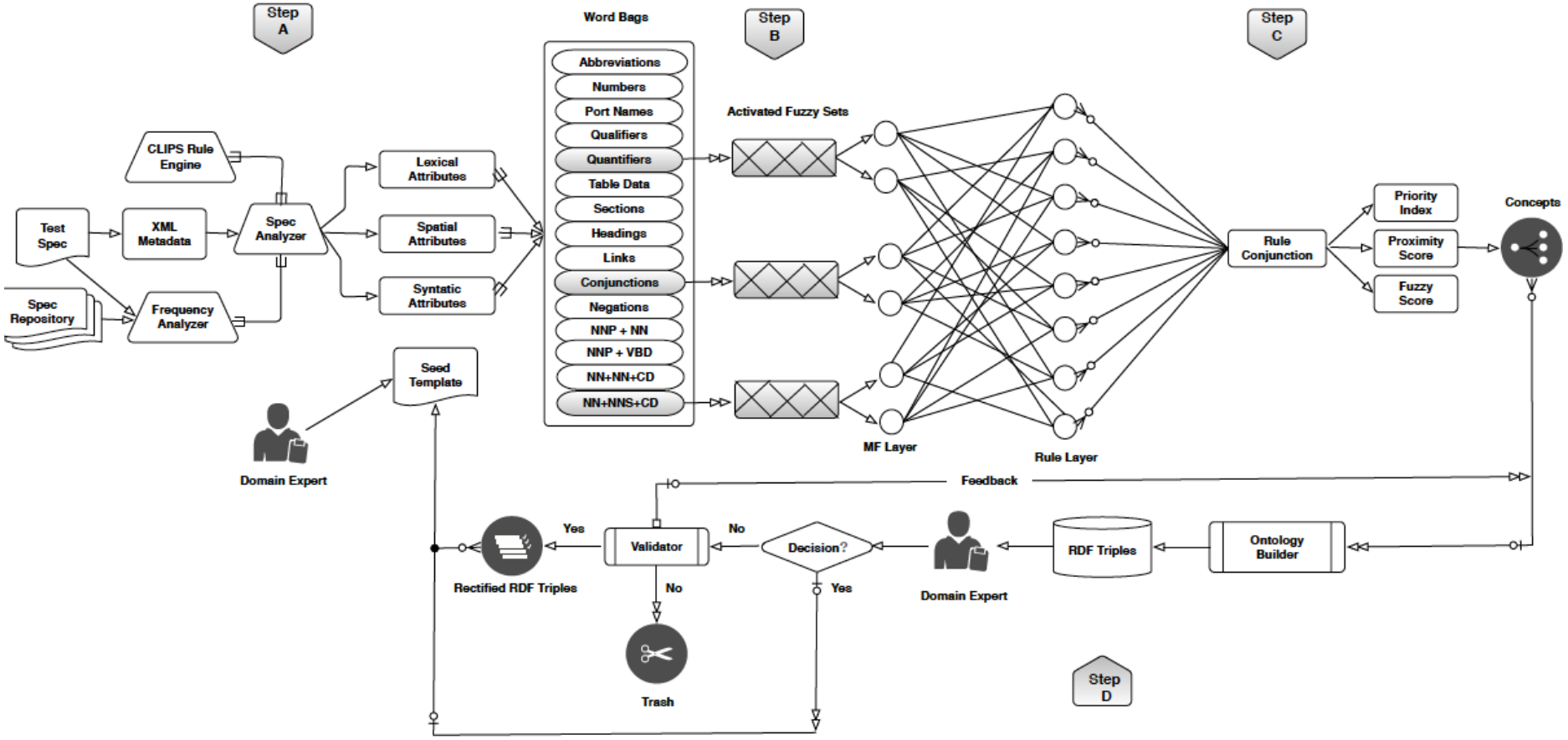
Assumptions

- Our methodology is based on the assumption that design teams practice IP reuse and platform reuse.
- New SoCs are built by mostly reusing and integrating existing IPs that are properly configured according to the specification.

# Knowledge Generation – Specification Ontology

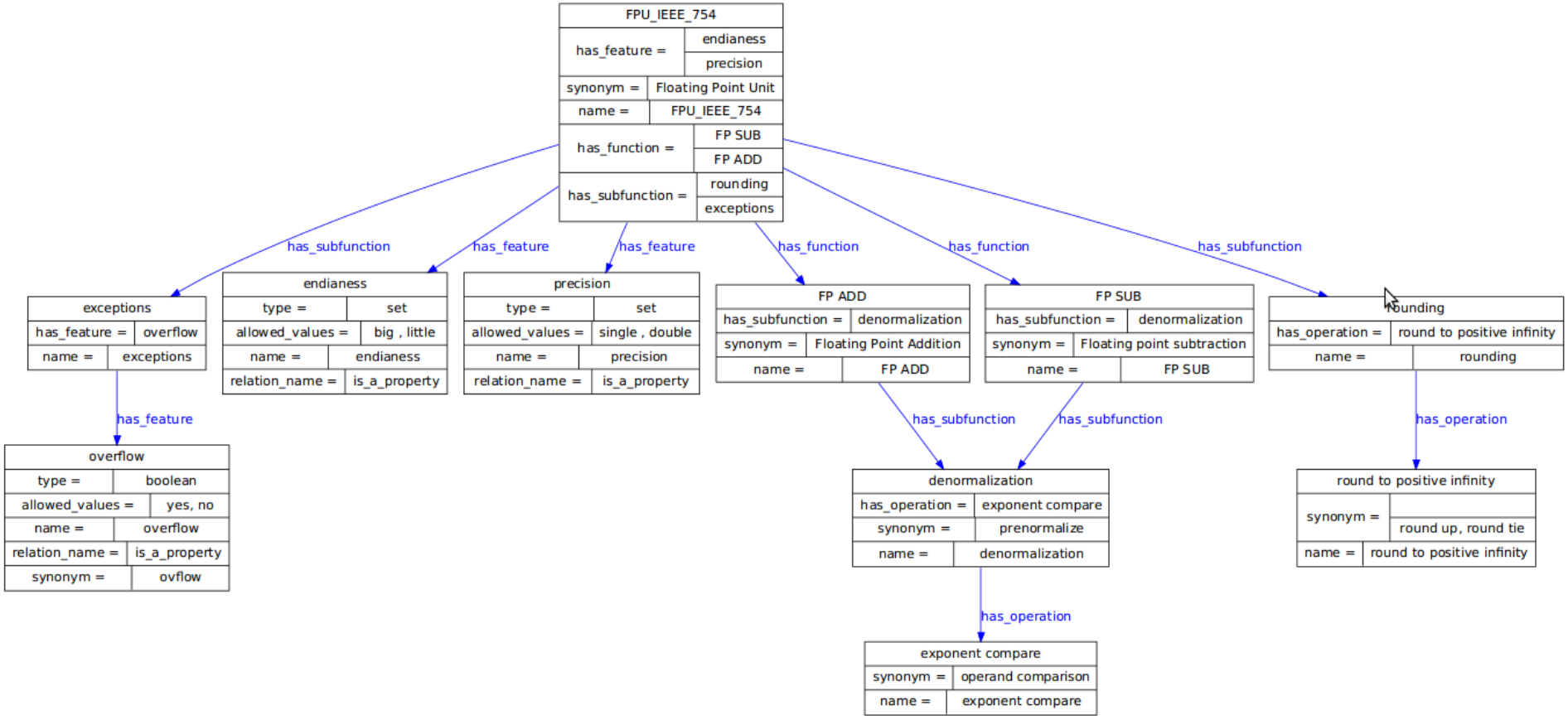
- An Ontology is defined as a knowledge representation model to explicitly represent a domain by defining its concepts and their relationships.
- A Specification document in a natural language format is generally used to describe design behavior.
- We define text-objects as words or phrases used to specify design properties such as “asynchronous” , “synchronous” and “booth encoded”
- Through analysis of different vendor specs, we observed that for a particular design domain, specs exhibit commonality of text-objects

# Ontology Miner




- *B. Singh, A. Shankar et al. " Knowledge-Guided Methodology for Specification Analysis", ICTAI 2013*
- *A. Shankar, B. Singh et al. "NEFCIS: Neuro-Fuzzy Concept based Inference System for Specification Mining", ICTAI 2013*
- *A. Shankar, B. Singh et al. " Ontology-Guided Conceptual Mining of Specification", DAC 2014 (accepted)*

# Graphical View of Ontology Fragment for Floating point Unit



# Specification Ontology

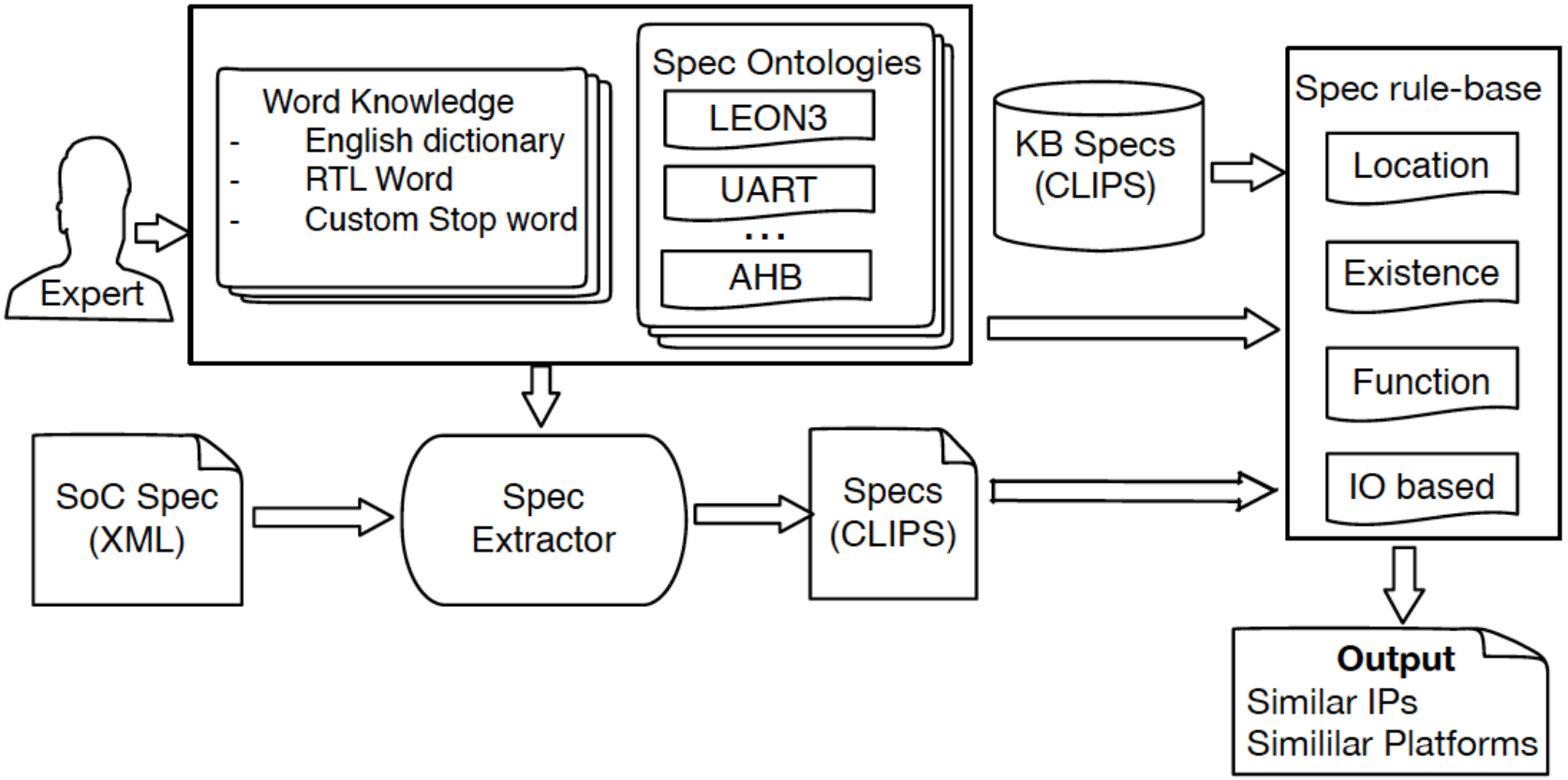
- The text objects are identified from a corpus of specs and then organized into a conceptual network, which we term as spec-ontology. An example is shown below

| OBJECT | FEATURE | SYNONYM | RELATION | TYPE | ATTRIBUTE |
|---|---|---|---|---|---|
| axi4s | axi4 stream protocol | | is-a-property | bool | yes, no |
| axi4s | data transfer | data exchange | is-a-function | bool | yes, no |
| axi4s | interconnect | | is-a-function | bool | yes, no |
| axi4s | signaling | AXI ports, AXI4S I/O | is-a-signal-list | bool | yes, no |
| … | … | … | … | … | … |
| signaling | tvalid | valid transfer | is-a-signal | bool | yes, no |
| signaling | tready | master/slave ready | is-a-signal | bool | yes, no |
| signaling | tstrb | byte qualifier data | is-a-signal | bool | yes, no |
| … | … | … | … | .. | .. |
| data transfer | slave | | is-a-property | bool | yes, no |
| data transfer | master | | is-a-property | bool | yes, no |
| data transfer | side band signaling | | is-a-property | bool | yes, no |
| data transfer | flow control | | is-a-subfunction | bool | yes, no |
| … | … | … | … | … | … |
| interconnect | arbitration | | is-a-subfunction | set | first come first serve, fixed priority |

# Specification Analysis

- We have developed a Spec-extractor tool to automate extraction of spec objects from an IP spec document.
- Each Spec gets represented in a machine readable format by populating a uniform CLIPS fact template for  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 documents author

# Specification Analysis(cont.)



# Spec Rule-base

- The KB has a spec rule-base to analyze specs.
- The spec rule-base has multiple level of inference rules. The higher level rules perform inferences by combining lower level inferences.
- Location based rules use location proximity of text-objects to associate design features with their attributes.
- Function based rules infer design functions based on occurrence of text-objects that identity low level operations and sub-functions
- 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.

# Spec Rule-base (cont.)

- 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.

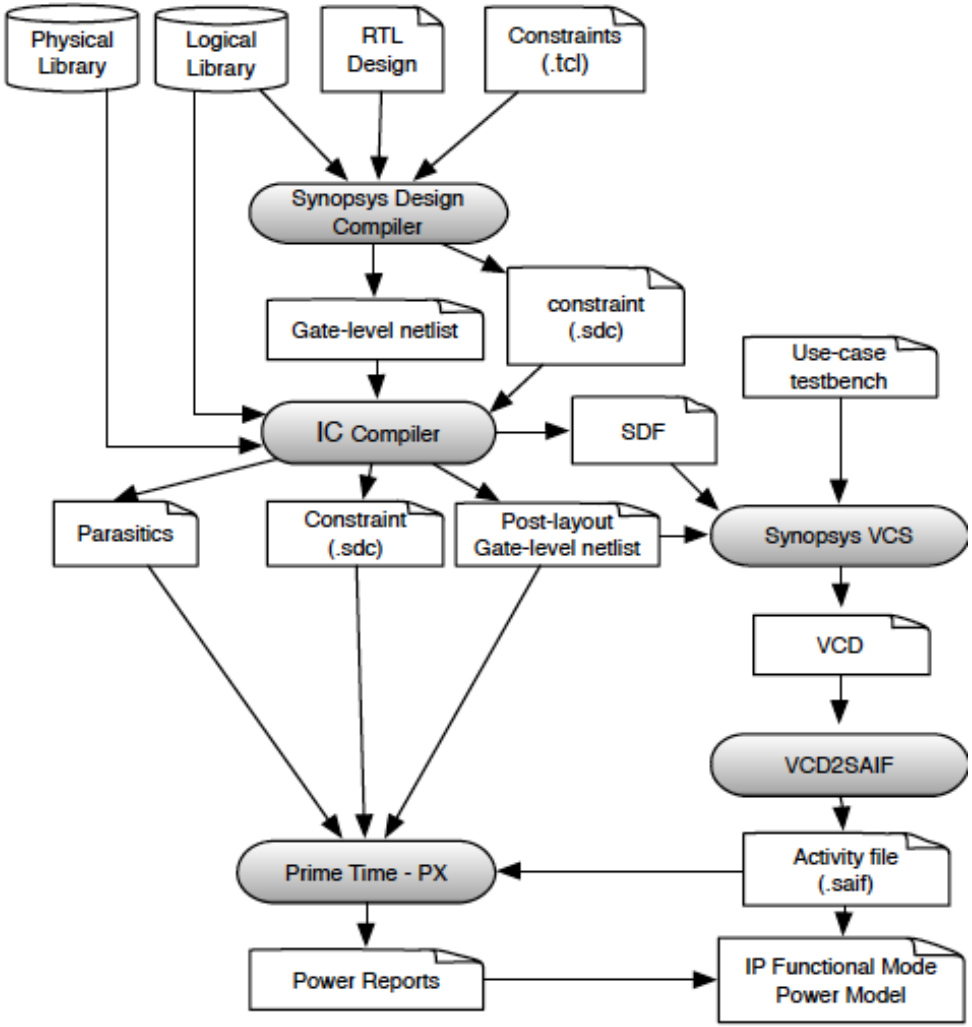
```
(defrule MAIN::find-attribute (declare (auto-focus TRUE))
(ontoFact (object ?obj)(domain fpu)(feature ?f) (relation ?
     rel)(type set)(attribute $?attr))
(wordFact (word ?word1&:(eq ?word1 (string-to-field ?f)))(
     locationId  $?loc1 ?word1loc) (cat onto))
(wordFact (word ?word2&:(member$ (str-cat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

# Power characterization

- Our approach of characterizing and building specification/power models of existing IPs is a one time exercise while building the KB. The manual effort is high initially and minimal afterwards.
- We have used Synopsys design compiler and 90nm generic technology library to perform synthesis. 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 (Cont.)

- Power characterization is performed for each IP in the KB using the tool flow shown in the figure
- We have used GRLIB IP library available from Aeroflex Gaisler.

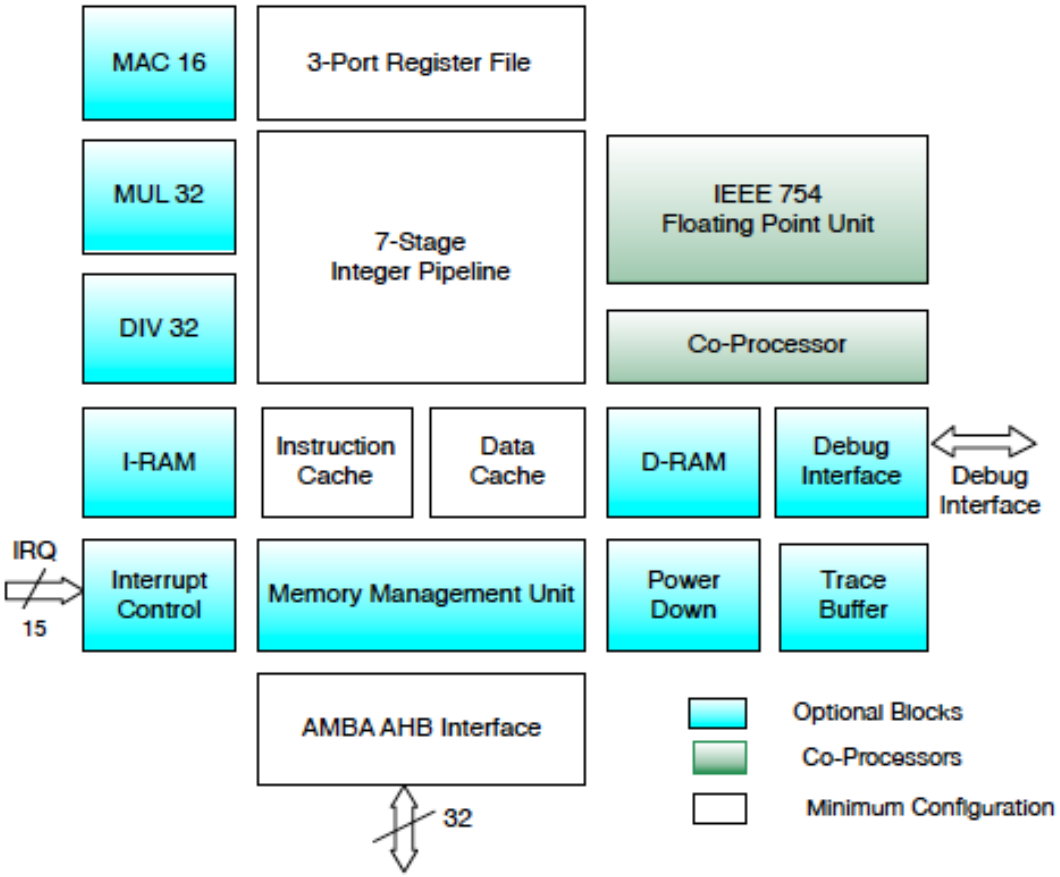



Power Modeling Flow

# Power Modeling of LEON3

- In our KB, we have characterized three LEON3 configurations, which are minimal, general purpose and high performance for power modeling.
- We have used two state approach to model processor power (a) processor core power model (b) cache model.
- Further we consider two states of the core, IDLE and ACTIVE instructions.
  - The IDLE mode corresponds to low power mode of LEON3 where only leakage power comes into picture.
  - The active state corresponds to worst case power dissipation while processor is executing *Stanford benchmark* tests.

# LEON3 Processor IP




LEON3 Processor functional block diagram

# Power Modeling of LEON3

- 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 pur-pose | 6 | 111.29 | 35331 |
| High perfor-mance | 6.74 | 128.4 | 40765 |

# Power modeling of Cache

- Cache memories are implemented using SRAM blocks from the technology library.
- We assume in the architecture file, the designers will provide the physical instance name of the memory cell and memory compiler generated power values.
- Power information for various memory configurations of LEON3 system are listed in figure below

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

# Other Soft IP blocks

- For IP blocks such as AHB controller, UART, Timer we follow the two state power model.
- We characterize each IP for IDLE and ACTIVE mode power dissipation as shown in table below.

| 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 |

# Clock tree power

- 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.
- For estimating clock tree power at pre-design stage, we use previous similar SoC platforms available in the KB.
- We use three techniques for clock-tree power estimation

# Estimating Clock-tree power

- Using per MHz per gate count power as shown below

$$P_{mg} = \frac{P_{ip}}{F_{ip} * \not{n}and_{eq}} \quad (1)$$

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)

# Estimating Clock-tree power

- Another technique is to estimate clock tree power by scaling results from a similar design in KB .
- The scaling is done through a set of heuristic rules, which capture engineering judgment 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.

# Estimating clock tree power

## *Using flip-flop count and clock-buffer fan-out*

- Let us say **F** is number of flip flops in a design and **f** is fanout of each clock buffer.
- We can calculate number of buffers in clock tree using eq 5
- We can use information from similar designs in KB to find the buffer types normally used for clock-tree synthesis.
- 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.

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.

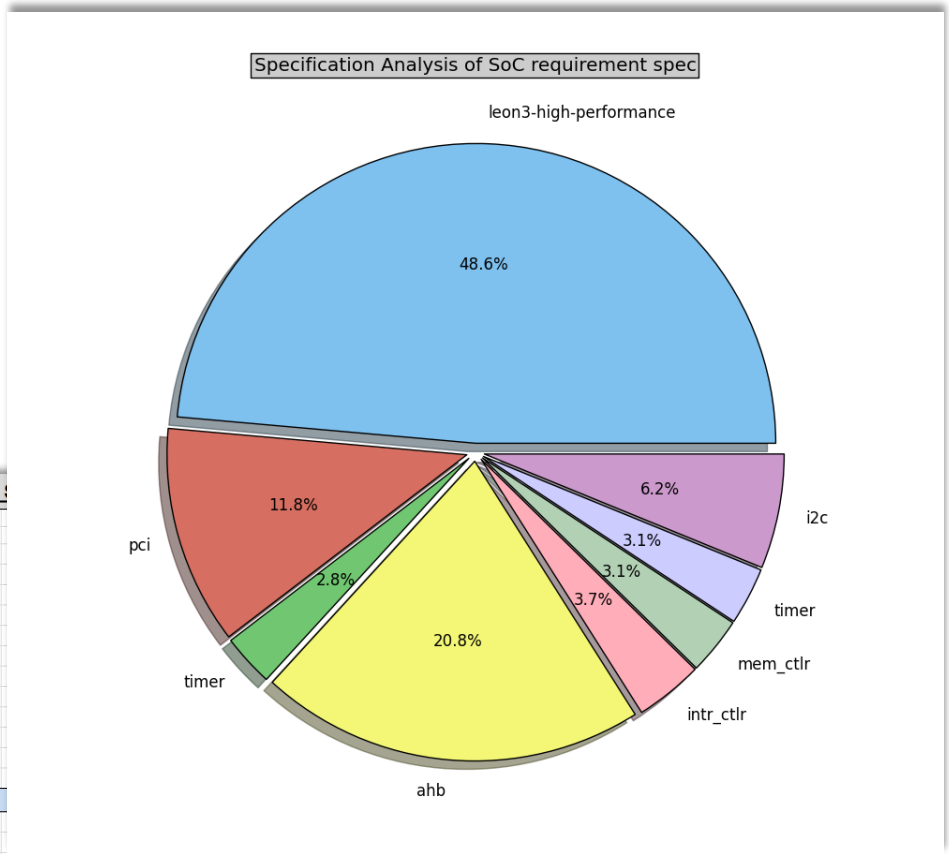
$$S_n = \frac{a(1-r^n)}{1-r}, r \neq 0 \tag{6}$$

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

# Case Study: Pre-design Power Estimation for a LEON3 SoC

- 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. IPs in an SoC spec that were identified as similar to ones in the KB are identified.
- The information of the matching IPs was listed in a preliminary design-data sheet in a csv format.
- We also generated post-layout power results for the SoC.
- The experiments show that power estimation results computed by our tool correlate with +/- 20 % accuracy with post layout power estimation results.

# Spec Analysis Results




**Chip Power Estimation**

| Chip Summary | | |
|---|---|---|
| Chip Name | | |
| Process Technology | | |
| Core : Dynamic Power | 3.762 mW | 3.135 mA |
| Core : Leakage Power | 0.013 mW | 0.011 mA |
| IO : Dynamic Power | 0.000 mW | 0.000 mA |
| IO : Leakage Power | 0.000 mW | 0.000 mA |
| Hard Macro : Dynamic Power | 1.248 mW | 1.040 mA |
| Hard Macro : Leakage Power | 0.002 mW | 0.002 mA |
| Total Leakage Power | 0.015 mW | |
| Total Dynamic Power | 5.010 mW | |
| Total Chip Power | 5.027 mW | |

Core Power

| | |
|---|---|
| Power Domain | baseband |
| Operating Voltage (Volts) | 1.2 |
| Current Consumption (mA) | 3.146 |
| Domain Powered ?(1:Y, 0:N) | 1 |

| Module/Peripheral Name | Gates (KGates) Including Flip-Flop Gates | Activity Level (0 - 1) | Clock Cut-Off ? (1:N, 0:Y) | Freq (MHz) | Memory Type | Number of such Instances | Dynamic Power Gates (mW) | Leakage Power Gates (uW) | Leakage Power Memory (uW) | Dynamic Power Memory (mW) | Total Core Leakage (uW) | Total Core Dynamic (mW) | Total Power (mW) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bt_core | 80 | 0.33 | 1 | 13 | exchg_ram9Kx32b | 1 | 0.93 | 1.28 | 1.86 | 0.24 | 3.14 | 1.16 | 1.168 |
| cvsd | 26 | 0.33 | 1 | 13 | none | | 0.30 | 0.42 | 0.00 | 0.00 | 0.42 | 0.30 | 0.302 |
| esco_gasket | 1.8 | 0.33 | 1 | 13 | none | | 0.02 | 0.03 | 0.00 | 0.00 | 0.03 | 0.02 | 0.021 |
| esco_matrix | 0.22 | 0.33 | 1 | 13 | none | | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.003 |
| pcminterface | 6.15 | 0.33 | 1 | 13 | none | | 0.07 | 0.10 | 0.00 | 0.00 | 0.10 | 0.07 | 0.071 |
| aulaw | 1.89 | 0.33 | 1 | 13 | none | | 0.02 | 0.03 | 0.00 | 0.00 | 0.03 | 0.02 | 0.022 |
| bt_rammod_0 | 0.5 | 0.33 | 1 | 13 | code_ram16Kx32b | 1 | 0.01 | 0.01 | 2.76 | 0.30 | 2.77 | 0.31 | 0.309 |
| bt_rammod_1 | 0.5 | 0.33 | 1 | 13 | main_ram9Kx32b | 1 | 0.01 | 0.01 | 1.86 | 0.24 | 1.87 | 0.24 | 0.246 |
| bt_rommod | 0.12 | 0.33 | 1 | 13 | code_rom32Kx32b | 1 | 0.00 | 0.00 | 2.24 | 0.08 | 2.25 | 0.08 | 0.081 |
| bt_mismod | 2.3 | 0.33 | 1 | 13 | none | | 0.03 | 0.04 | 0.00 | 0.00 | 0.04 | 0.03 | 0.027 |

README | Tech-Data | Power-Estimator | Revision History | Future Enhancements

# Conclusion & Future Work

- 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 as estimated activity for new design.
- We also plan to improve our prototype power estimation tool by characterizing IPs for more than two power states.

*THANK YOU*