7th RTD Framework Program

REALITY

Reliable and Variability tolerant System-on-a-chip Design in More-Moore Technologies

Contract No 216537

Deliverable D6.2

Validation and assessment of results: Validation Plan

Editor:	Giuseppe Desoli, Miguel Miranda, Paul Zuber, Stylianos Mamagkakis,Yves Laplanche, Andrea Acquaviva, Francesco Paterna, Luca Benini, Francesco Papariello, Elie Maricau
Co-author / Acknowledgement:	
Status - Version:	V1.1
Date:	06/07/2009
Confidentiality Level:	Public
ID number:	IST-216537-WP6-D6 2-v1_1.doc

The REALITY Consortium consists of:

Interuniversity Microelectronics Centre (IMEC vzw)	Prime Contractor	Belgium
STMicroelectronics S.R.L. (STM)	Contractor	Italy
Universita Di Bologna (UNIBO)	Contractor	Italy
Katholieke Universiteit Leuven (KUL)	Contractor	Belgium
ARM Limited (ARM)	Contractor	United Kingdom
University Of Glasgow (UoG)	Contractor	United Kingdom

1 Disclaimer

2 Document revision history

Date	Version	Editor/Contributor	*2.1.1.1.1 Comments*
19/06/2009	V0.1	Giuseppe Desoli, Miguel Miranda, Paul Zuber, Stylianos Mamagkakis,Yves Laplanche, Andrea Acquaviva, Francesco Paterna, Luca Benini, Francesco Papariello, Elie Maricau	Initial draft
06/07/2009	V1.1	Giuseppe Desoli, Francesco Papariello	Integrated contributions and WP6 validation tasks
07/08/2009	V2.0	Giuseppe Desoli, Francesco Papariello	Final version

3 Table of contents

4 List of Figures

5 List of Tables

6 Preface

The scope and objectives of the REALITY project are:

- Development of design techniques, methodologies and methods for real-time guaranteed, energy-efficient, robust and adaptive SoCs, including both digital and analogue macro-blocks“

The Technical Challenges are:

- To deal with increased static variability and static fault rates of devices and interconnects.
- To overcome increased time-dependent dynamic variability and dynamic fault rates.
- To build reliable systems out of unreliable technology while maintaining design productivity.
- To deploy design techniques that allow technology scalable energy efficient SoC systems while guaranteeing real-time performance constraints.

Focus Areas of this project are:

- “Analysis techniques” for exploring the design space, and analysis of the system in terms of performance, power and reliability of manufactured instances across a wide spectrum of operating conditions.

- “Solution techniques” which are design time and/or runtime techniques to mitigate impact of reliability issues of integrated circuits, at component, circuit, architecture and system (application software) design.

The REALITY project has started its activities in January 2008 and is planned to be completed after 30 months. It is led by Mr. Miguel Miranda of IMEC. The Project Coordinator is Mr Miguel Miranda. Five contractors (STM, ARM, KUL, UoG, UNIBO) participate in the project. The total budget is 2.899 k€.

7 Abstract

This document describes in details the metrics and validation strategies for critical IPs and methodologies developed by the project in the various work packages; including a description of the validation and assessment of impact of the final benchmarking activities identified during the project execution.
This document covers only the IPs and methodologies that while being developed by the partners in other tasks scattered throughout the project execution, are in the need of being deployed in a common environment, and for which sensible metrics and validation strategies that are consistent across the WPs need to be defined. Suitable figures of merit and estimation of objective functions, for example in terms of tolerance to variation of critical process parameters, as well as modeling of individual circuits and their validation are an example of those. As far as possible simulation based quantification will be the objective, but where this is not possible, qualitative conclusions and mixed models have been defined to integrate analytical and qualitative approximations into simulations platforms.
The objective of the validation plan is to clearly state how the various contributions come together and how the overall results will be evaluated and exactly what kind of metrics will be considered in drafting the requirements and assessing the impact of the results.

This document is structured in two main parts, the first defining for each relevant work package the metrics for individual selected IPs and methodologies, while the second part covers the definition of how the block IPs and system level microarchitectures defined in WP5, based on WP3,4 deliverables should be benchmarked; this involve a description of how the various blocks and circuits will be evaluated when aggregated together in the prototype platform in terms of different operating condition and margins, SW countermeasures and control policy profiles, evaluation of the SW heuristics and integrated HW/SW mechanisms that interact and interface to low level controls as defined in WP4, as well as the definition of how the libraries and control policies will be ported and integrated into run-time systems and middleware provided by industrial partners for the target platforms defined by WP5.
The plan describes also objective and qualitative metrics based on a small set of platform operating conditions profiles in accordance with the requirements of the selected applications demonstration benchmarks selected.

8 Identification of criteria and evaluation metrics

8.1 WP2 components and methodologies

8.1.1 Introduction

Once transistor parameters, such as the threshold voltage variations V_{th} become statistical, then circuit parameters, such as timing $t=h(V_{\mathrm{th},1} \,..\, V_{\mathrm{th},n})$, become statistical as well. The default method of predicting nominal values for circuit parameters is by golden reference models and tools. Today this method becomes problematic, because parameters to these models are becoming more and more random. Thus the question arises whether a tool that produces a golden reference is still needed, as the finite random sequence produced naturally during the actual manufacturing is different than the one assumed for simulation.

In fact, the new task is not to predict *one single value* for the circuit performance, since that does not make sense anymore. The new challenge is to predict another form of this circuit performance metric, which we further specify below. This document will provide guidelines to designers beyond conservative, deterministic measures, to define and use this new metric type.

Our generic guideline is to use the probability density function (PDF) of the performance metric as the most significant factor. It allows computing several characteristics such as the mean, median, mode, variance or skewness of the distribution. Above all, it also allows the computation of a parametric yield value, given the user knows a constraint value on the performance metric(s). At least on system level, this is one of the most important figures and allows predicting manufacturing efficiency for a given system specification. Also, the inverse is possible: Given a certain yield target, say 90%, we can answer the question of what is the maximum possible speed, reached by at least 90% of the processors made.

Depending on how the PDF is approximated, different sources of errors can occur. There are confidence gaps when using sampling techniques due to limited sample size:

• UoG:	Statistical Cell Characterization
• ARM:	Statistical Characterization of non-standard-cells

Other techniques of generating the PDF draw assumptions and require simplifications and result in constant errors:

• ST:	Statistical Static Timing Analysis
• IMEC:	Statistical System Characterization

8.1.2 Statistical Confidence Level

Unfortunately, with sampling based techniques, the PDF cannot be approached with unlimited accuracy, because of limited CPU simulation time and thus sample size *N* limitation. (Nevertheless, it can be shown that some areas of the PDF can be estimated with higher accuracy by giving up accuracy in other areas. In this case it is very important for the user to know which metric is of interest before deciding for which accuracy regions to focus on. See For example [Die07]).

The confidence gap is calculated from the variance on a metric derived from the PDF. The higher the confidence with which one wants to make a statement, the wider the gap, i.e. the wider apart the lower and upper bounds. Typical values for confidence are 95% and 99%, for populations lying within the 2 σ and 3 σ range, respectively, for normal distributions (where σ is a standard deviation in a normal distribution). For example, it is common in statistics to make statements that begin with "We are 95% sure that …", while in the current context such a statement could continue with "… that the parametric yield of this system will be between 94% and 99%". This confidence interval is not generally distributed symmetrically. Typically, the most likely (i.e., the mean value for a normal distribution) value is also predicted: "Most likely the yield will be 98%".

In case of symmetrically distributed metrics one can use the standard deviation to make statements such as "the read current of this memory cell is 50 +/- 10 uA".

8.1.3 Brute force confidence interval determination

Confidence intervals can be determined by a brute force repetition of the random experiments with different random sequences. A slightly different PDF with slightly different properties will result every time. This is not an exhaustive simulation, a limited number of random repetitions are used in practice. The variance of these properties is the confidence gap.

This was done and is exemplified on a standard cell gate, but it is generally possible for any Monte Carlo-based experiment, i.e. an experiment that assumes a random number sequence to sample the input domain. In this project, this is true for the contributions of UoG (Statistical Cell Characterization), and ARM (Statistical Characterization of non-standard-cells). The contributions of ST (Statistical Static Timing Analysis) and IMEC (Statistical System Characterization) provide an analytical representation of the PDF with a specific error.

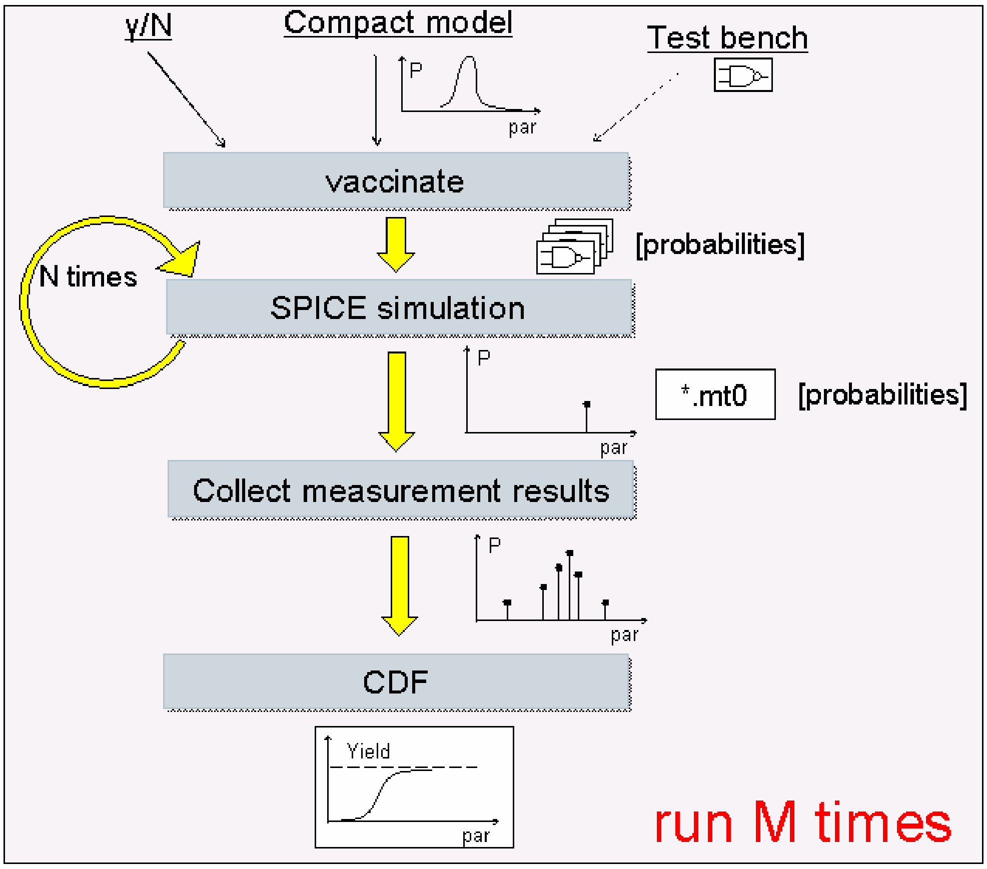

Figure 1: Analysis plan.

Figure 1 shows an overview of the analysis plan. For a given inverter, a set of *M* CDF approximations is generated. A value of *M=200* is assumed to be a reasonable choice for obtaining good accuracy of the variance predictions.

From this characteristic the standard deviation of the one-dimensional constraint at several fixed yield percentiles is extracted. In particular, one is interested in the following influence factors:

- The sample size *N*. It is the main measure of simulation speed and trades off with prediction confidence. Considering values between 10 and 10,000 seems justified, because these are typical number of Monte Carlo runs feasible on reasonable computers in reasonable run times.
- Circuits with different transistor counts and interconnect topologies of the transistors are considered.
- Setting a constraint on one of the circuit performance metrics classifies a circuit variant as yielding or not. Testing the circuit at constant yield, it is possible to amount the prediction accuracy by estimating the variance of the constraint parameter. If computed for one dimension, the yield level can be expressed in the sigma-notation, i.e. units in a probit plot Compare Figure 4 and Figure 5.
- Apart from delay, other circuit performance measures are of interest. Investigated are the rise and fall propagation delay, total power and leakage using an appropriate spice test bench.

The statistical parameter whose variance is to be estimated could be the yield at some given constraint value. However, from the previous example, it can be seen that yield is typically very asymmetrically distributed, when it comes to large or small yield numbers.

Due to the skewness of this distribution at high constraint values the calculation of a standard deviation becomes meaningless and upper and lower confidence bounds would be required, which are not straightforward to compute.

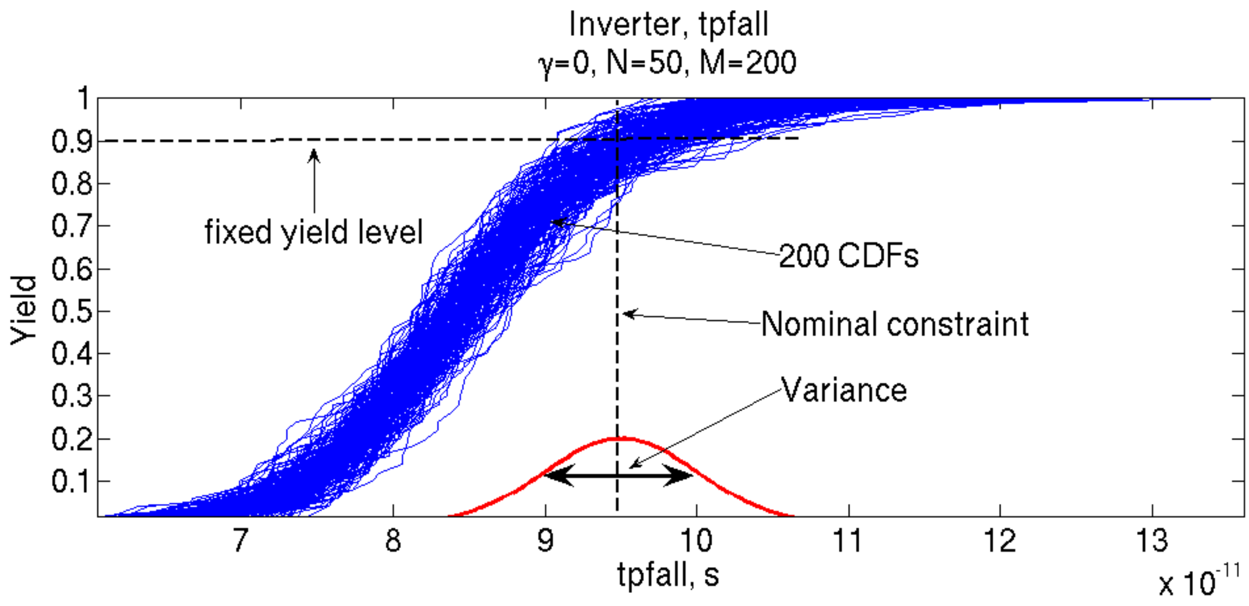

Figure 2: Repeated approximation of the CDF of an inverter delay. The variance of the delay at a fixed yield level is extracted as confidence gap.

Instead, we measure the standard deviation of the constraint at fixed yield levels as the confidence of our prediction. This is in fact a useful measure. For example, the user can learn what is the maximum speed in order to be 95% sure that the yield will be better than 99%.

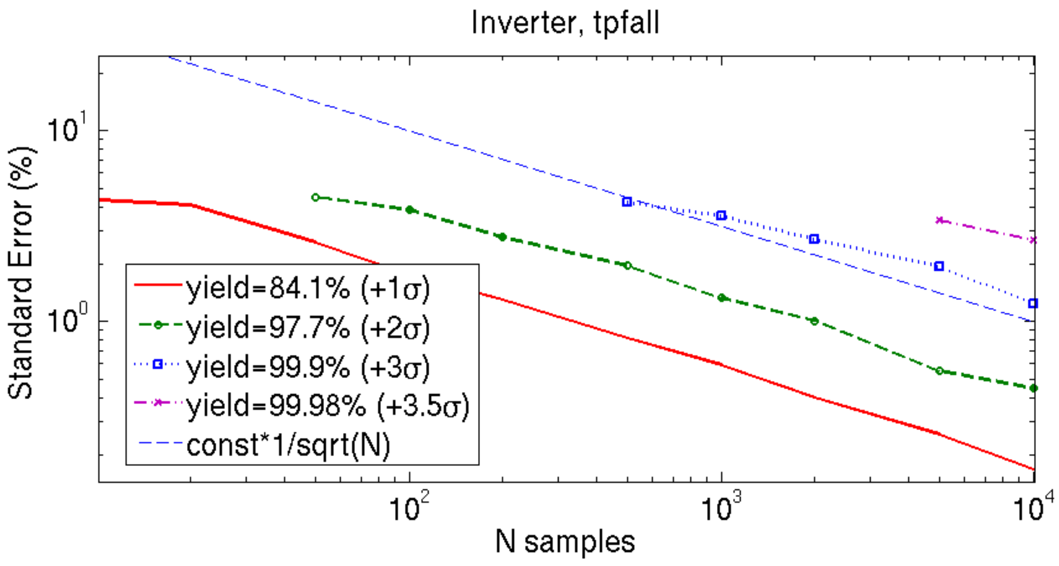

Figure 3: Results on the standard cell, showing the variance (relative to nominal constraint) over the sample size.

The results in Figure 3 show the expected decrease of standard error with increasing sample size *N*. It also shows the different yield levels which are assumed, the constant multiplier of the inverse square root changes. It is further impossible make any statements in high yield regions with too small sample sizes when using the Monte Carlo method.

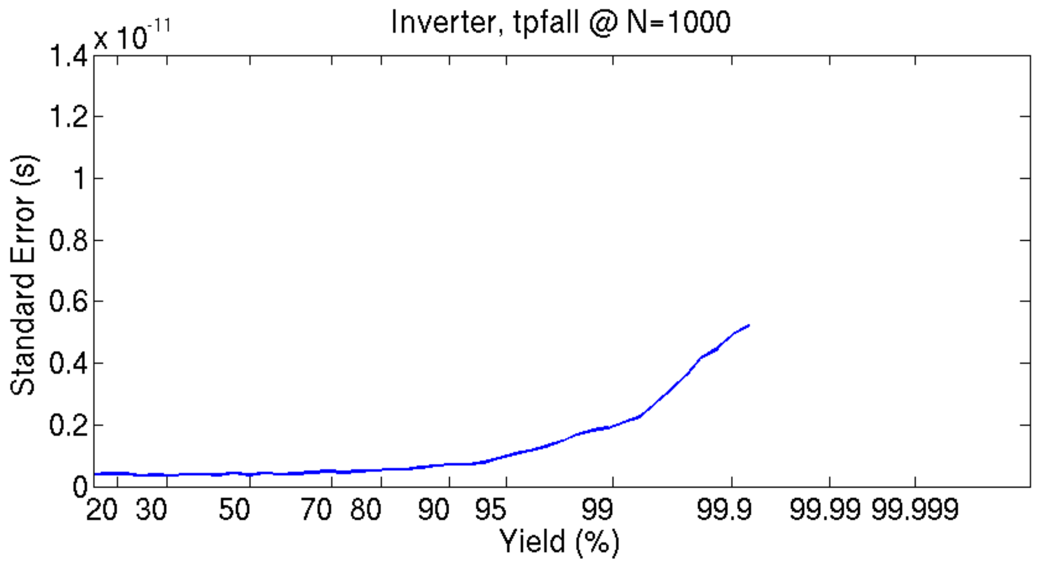

Figure 4: The standard deviation of the constraint in a fixed yield percentile.

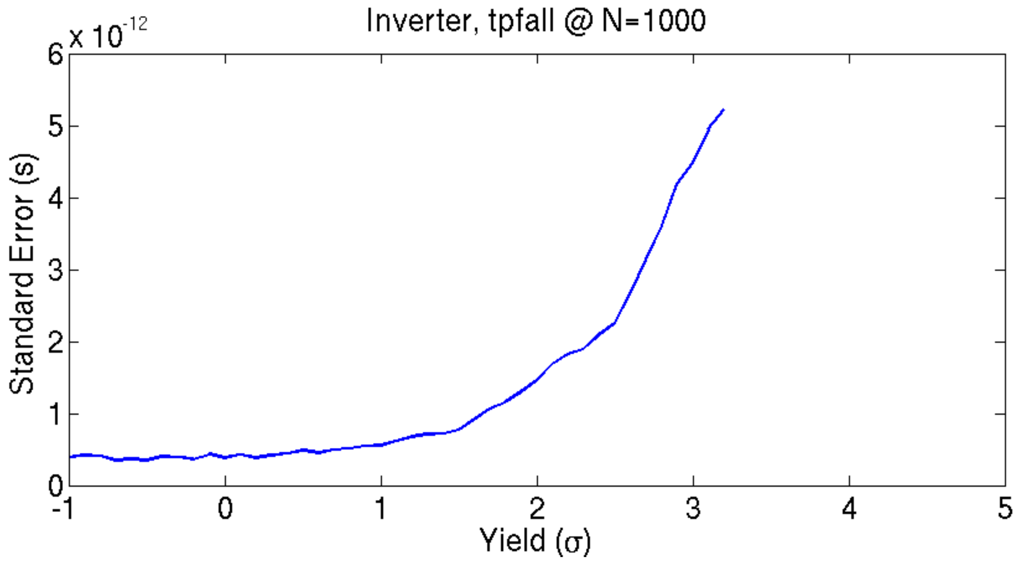

Figure 5: Standard deviation in a fixed yield percentile using the sigma notation.

8.1.4 <u>Bootstrapping</u>

For highly increased speed of confidence prediction on sample based distributions, of course at the price of decreased confidence on the confidence, one can perform bootstrapping. Bootstrapping is taking samples with sample sizes *NB*<*N* from the population of size *N* and building a bootstrap PDF from *NB* observations. This is repeated *MB* times with different, random bootstrap samples, and confidence metrics are taken from the bootstrap PDF or CDF ensemble.

8.1.5 Analytical confidence interval determination

In order to calculate the confidence level for the difference of error level (variate) based on the sample size N for different Monte Carlo runs (variable). We can use the method of univariate analysis of variance (ANOVA).

With ANOVA you can test that the difference measured between different CDFs is because of the sample size N and calculate the effect size of the sample size N. The underlying assumption is that there is a normal distribution of the error value for each of the Monte Carlo runs (i.e., each Monte Carlo run has a different number of sample size N).

On the other hand, if you want to calculate the confidence level for the difference of error level (variate) based on the yield (variable). We can use the method of Pearson's chi-square (χ^2) test.

In this particular case we assume that the error value based on yield will be PDF based on a chi-square distribution.

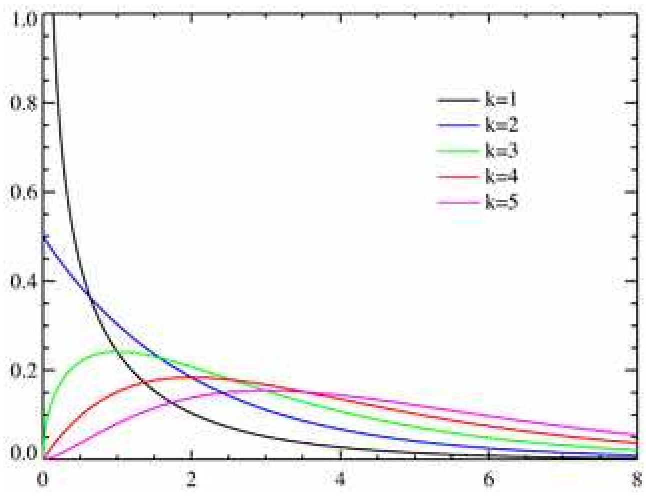

Figure 6: Chi square probability density function examples

8.1.6 Error metrics on analytical distributions

8.1.6.1 ST Statistical Static Timing Analysis

The main sources of error in the SSTA method presented are:

- The assumption of linear sensitivity of a gate performance metric to process variation.
- The assumption inherent to today’s SSTA tools that every operation ultimately results in a Gauss-shaped PDF.
- The inability of SSTA to keep correlation of system parameters.

Comparison with a Monte-Carlo based result one can see the obvious difference:

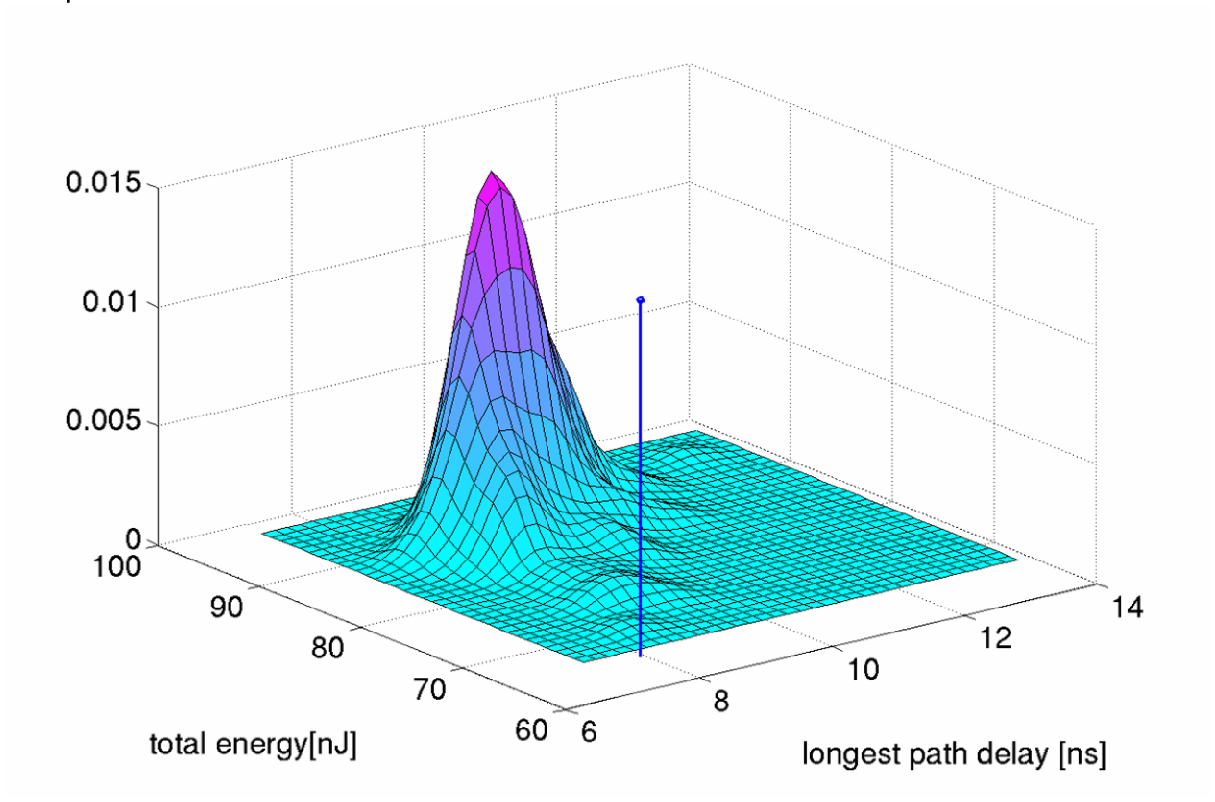

Figure 7: Digital block characteristics are non-Gauss, and correlated in general.

8.1.6.2 IMEC: Statistical System Characterization

The source of error of this contribution stems from the assumption that the electrical interaction, mainly activity, between IP blocks is negligible.

8.1.7 Propagating errors and confidence bounds

Qualitatively, the user is interested in the trade-off between parametric yield and the design constraint(s) he can set. In an iso-yield curve, for instance, one draws a curve in a plane. The axes can be delay and power consumption. The curve indicates points of equal yield. Under above described model and method errors, the expected position of the iso-yield curve moves about an expected course.

The ultimate question is, how much movement of the yield curve is generated by the sum of all error mechanisms, and how much can be tolerated?

In order to answer this question, we need a golden reference distribution of the final output PDF. The only general way to achieve it is a Monte Carlo based method over all abstraction layers, with a sample size high enough to guarantee small confidence gap of the reference PDF itself.

Then we can compare the PDF generated by WP2 with its compound error to the reference PDF. For this test again, we can use any of the above-described methods. Also, using Pearson’s chi-square test we can evaluate the error between two output samples sets.

In order to quantify the individual error components, let's assume the errors are additive (which is over pessimistic as error *squares* are additive), and constant errors such as from SSTA, and system level analysis and model inaccuracies are captured in *e1*, and one Monte Carlo process is involved with error *e3*. In order to investigate the influence of *e1*, two different points on the Monte Carlo error curve are chosen and its result tested against the reference.

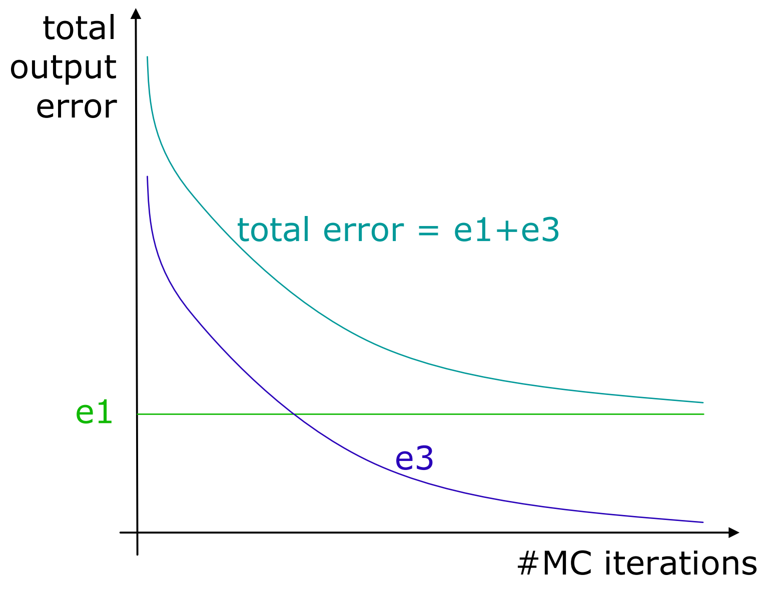

Figure 8: Error assumptions

8.2 Specific metrics and validation for WP3

8.2.1 <u>T3.2.1 Variability aware reliability simulation of digital/analog circuit under time-varying stress</u>

In this subtask a variability aware reliability simulation methodology is developed. This methodology is intended to become an essential part of the future design flow for analog or small digital circuits. It aims to give the user a fast but sufficiently accurate indication of the yield of a circuit as a function of time.
Process variability information is obtained via Monte Carlo transistor models. Ageing of different transistors in the circuit is calculated using behavioral degradation models of the most important degradation effects (e.g. hot carriers and NBTI) These degradation models form a second objective in this subtask. As a whole, this subtask is a part of the circuit level in the VAM flow where it allows characterizing basic cells more accurately.
The simulation methodology and the degradation models developed in this subtask will be validated separately:
Degradation modeling: A well chosen number of experiments, each time varying important transistor input factors (e.g. transistor width, transistor length, VDS, VGS, temperature,…) and observing relevant transistor output parameters (e.g. Ids, gm, gds), guarantees to develop a sufficiently accurate model over the entire range of all transistor input factors. Afterwards, a study of the residuals (error between the degradation model and the measurement results) reveals remaining model errors (e.g. highly non –linear factors, missing factors or missing interaction terms).

Simulation methodology: A number of small and larger typical analog and digital circuits will be stressed at elevated stress conditions (e.g. higher stress voltages). The ageing effects of relevant circuit specifications (e.g. amplifier gain and ring oscillator frequency) will be compared against simulation results using the simulation methodology developed in this subtask. A statistical hypothesis test will indicate whether the measurement results belong to the population generated by the reliability simulator.

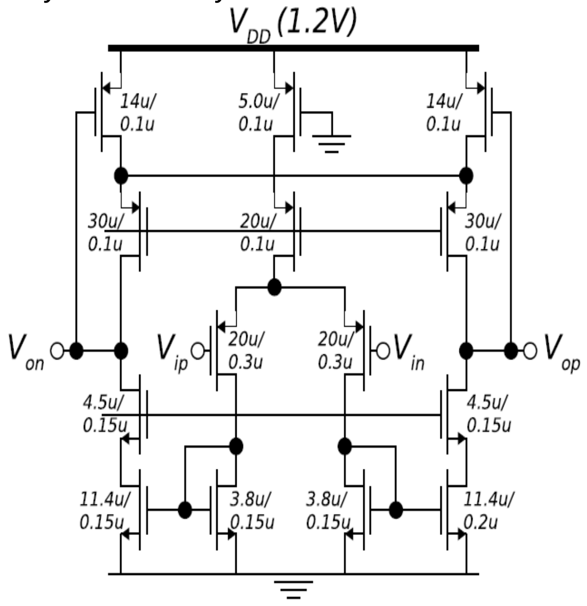

Figure 9: A fully differential OTA is part of the circuit test set to validate the simulation methodology.

8.2.2 T3.2.2 Design of variability and reliability resilient memories

The goal of T3.2.2 is to design a set of memory IP blocks for the 32nm node, taking into account the increased importance of variability and, to some extend, reliability. To prove the validity of the developed concepts and circuits, a small but complete memory will be designed. The memory will contain 1K words of 32 bits. It will have one read and one write port and a speed of 0.5GHz. The main design target will be low power (~1pJ/access) while the total yield is to be kept at 99.9% of higher.

Functionality under variation will be tested in two different ways:

During design, a failure rate is allocated to all sub blocks in order to keep the total yield at 99.9%. Monte Carlo simulations with statistic extrapolation are used during design. Importance sampling Monte Carlo simulations are used to verify the assumptions used while extrapolating statistical data. For each block, a possibility density function of all properties affecting other blocks is determined. These possibility density functions are used as input for the statistical analysis of those blocks.

The total design will be fed into the VAM platform developed by IMEC to assess the impact of variation. This way both the VAM and this design can be validated.

9 Metrics and validation scenarios for block level IPs and system level integration

WP5 is the place where the components of the project come together and will involve developing methodologies, test platforms to enable the evaluation to take place. It is responsible for assessing the severity of the un-compensated situation, and quantifying the improvements that result from the application of techniques identified in other Work Packages. It is also responsible for identifying the techniques that prove to be unsuccessful (and why), as well as to point to other techniques that might be worth investigation (beyond this project). From the beginning it was envisioned that, as far as possible, simulation based quantification will be the objective, but where this is not possible, qualitative conclusions will be offered.
The tasks that are associated to a validation activity within WP5, and to some extent, and for the assessment of the compensated situation, to WP6 are task 5.5 for the block level evaluation and integration of the building blocks from WP2, WP3, and WP4 and task 5.6 for the Integration of the building blocks from WP2, WP3, and WP4 into the two proposed evaluation platforms:

- ARM926 processor
  the evaluation will stop at RTL evaluation level of a system incorporating countermeasure mechanisms.

- xSTream platform:
  the work will also include integration of higher level mechanisms that interact and interface to low level controls.

The first deals with the application of the Variability Analysis Flow to the selected microarchitecture and its components including the statistical characterization of the library developed in WP2 T2.1.2 (deliverable D2.1.2).
The relevant metrics for this have been defined in the previous part of this document, while the validation flow for the ARM926 and the library will be described in the following.

Two types of validations must be taken into account. On the first hand we need to validate the accuracy of the methodologies we are using. On the second hand, the counter measures integrations have to be investigated.

The later will be a classical work. The special memory developed by KUL follows traditional development rules. Therefore traditional integration tests are run using the same test benches and techniques as those used to integrate any memory in the design. Concerning the WP3 hardware counter measures developed by UNIBO, the success of the integration is assessed by a successful synthesis step. At the end of the process, we get a final design with classical timing and power analysis incorporating the countermeasures. A qualitative information will also be provided to assess the ease of the integration, evaluating the amount of RTL development that eventually needs to be done to adapt the traditional design to these new techniques.

The flow evaluation is more complex. Techniques were developed in task 5.4. To evaluate the flow we have assumed the reference information will be provided by the spice models simulations. As a consequence, in terms of variability, the reference distribution will be given by time consuming Monte Carlo spice simulations. We still need to keep in mind that Monte Carlo simulations in the spice simulators are relevant as long as the number of run is not too high. As a consequence, for accuracy reasons, the comparisons will have to be performed around the mean value of the distribution. Comparing the tail of the distribution might lead to errors coming from the reference.
At the standard cell library level, the characterization of several cells using an SSTA commercial tool and the variability aware modeling tool developed by IMEC will be compared to the direct simulation of the cells. The benchmark will be performed on the figure of merits of the cells: propagation delay, leakage and dynamic power. The chosen cells will be simulated with the same conditions. The mean value and spread of the final distribution will be compared.

At the evaluation platform level, the direct simulation of the platform behavior using any spice simulation is impossible. The output of the synthesis phase will provide a critical path. The statistical evaluation of the critical path will be performed using the system level characterization methods. In parallel, a spice simulation of the critical path will be performed including statistical characterization using Monte Carlo techniques. It is plain for all to see that depending on the complexity of the critical path the simulation time can be extremely high.
The second step consists of a full characterization of the whole platform with the variability aware methodologies. Several critical paths are identified in this process. Depending on the simulation time available, the same evaluation as done for the initial critical path can be made on different paths that are identified as critical during the statistical evaluation of the full system. The output of the different characterization techniques will be compared. Only the comparison on the critical path will provide information on the accuracy of the system level characterization.

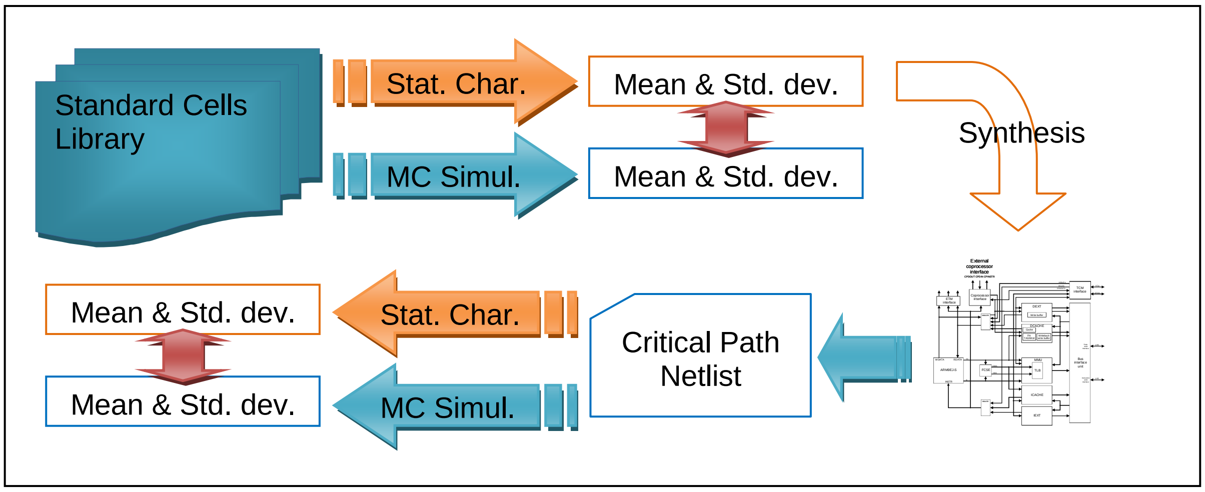

Figure 10: Flow Validation Process

A parametric analytical approach for fast estimation of variability trends of the system for different voltage and temperature combinations is deployed and its accuracy needs to be validated against the reference flow resulting from WP2.
This approach attempts to capture significant trends and to gauge the extent of potential benefits of applying the SW/HW countermeasures developed by WP2/3/4, when deployed in a representative high-end multiprocessor based embedded platform.

In the following a description of the metrics, models and scenarios, applicable to the actual definition of the benchmarking conditions and assessment of results to be carried out in WP5 and WP6.

9.1 System level platform definition and metrics

The platform HW IPs and simulation models whose simplified block diagram is shown in Figure 11, are contributed by STM and augmented with the required enhancements by both UniBo and STM.
Such a platform is highly parametric and in principle a careful design and architecture space exploration would be required in order to optimize performance, power, yield and other significant metrics for a given application domain.
Being the project focused more on the methodology and on assessing the impact and potential benefits in general rather than on obtaining the best of possible results on a specific vertical niche, the validation will be applied to a reduced set of configurations.

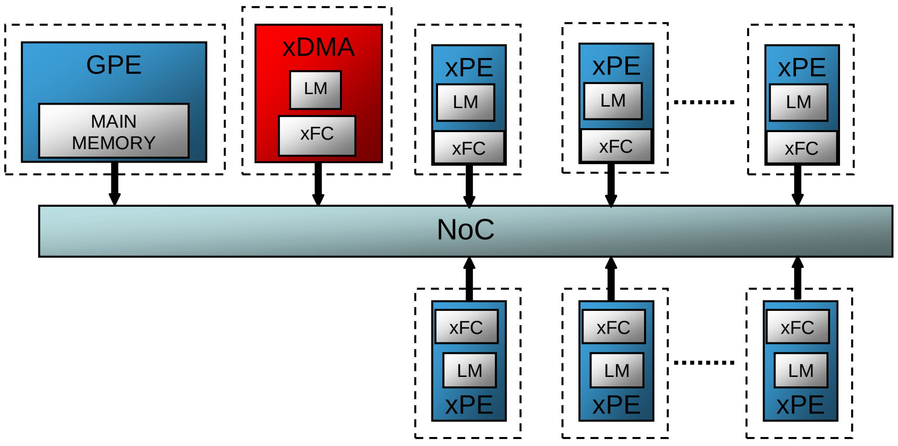

Figure 11: xSTream parametric template selected for the REALITY system platform, example of parameters are the number of accelerator cores, the width of the interconnect, the arbitration policies, the latency to read or write single words or data bursts to memory, etc.

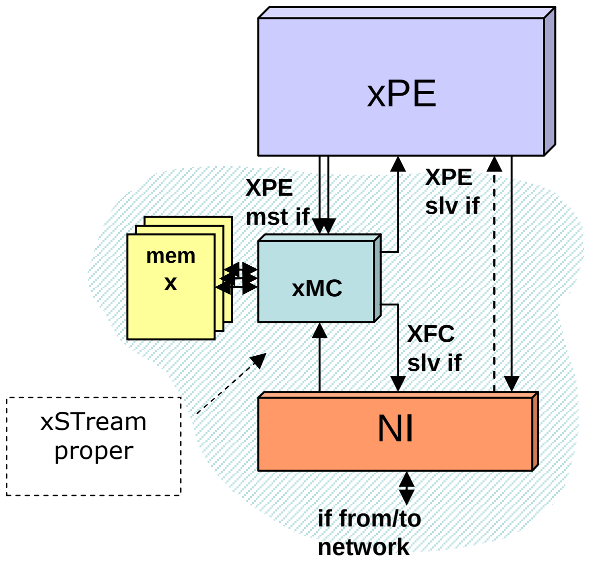

Figure 12: xSTream parametric computing node template, example of parameters are the number of threads supported per cores, the frequency of operation, the amount of local memory, etc.

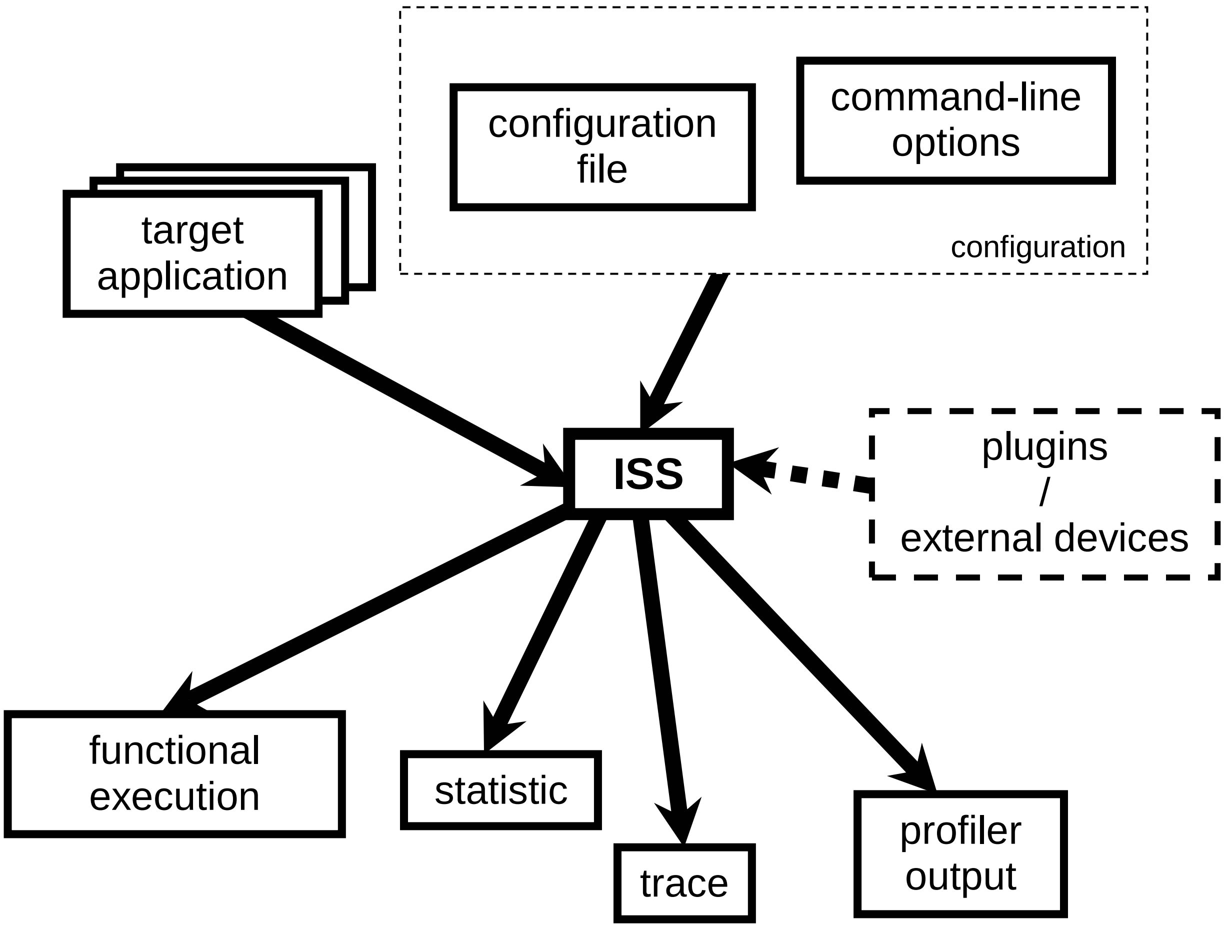

Figure 13: ISS inputs and outputs

ISS inputs

1. configuration: parameters for configuring the number and type of the cores, their devices, how they are connected, the latencies of the devices contained in the system, and how the user can interact with the simulation platform (trace, profiling, statistics, …) (more details on ISS configuration in appendix)
   the ISS configuration can be provided to the ISS using two equivalent methods:
   a. configuration file, containing a list of parameters with their values (expressions)
   b. command-line options, each specifying a parameter with its value (expression)
2. target applications: usually each core must be provided with a target application; alternatively, only the GPE can be provided with its target application, and the GPE will load the target applications of the xPEs at runtime
3. plugins/external devices: custom devices that can communicate with the other devices and the cores contained in the system and can implement custom functionalities (an example of such devices is the device for monitoring the activity of the core)

ISS outputs:

1. functional execution output, which is functionally identical (bit true) to the output produced by the real hardware with an accuracy higher than 90% ((simulated number of elapsed cycles – real number of elapsed cycles) / real number of elapsed cycles < 10%)
2. a statistics file, containing information on all the cores (number of instructions executed, elapsed cycles, branch/read/write stalls, …), on the NoC (number of read/write operations, number of bytes/words involved, occupancy of the different arbiters, …) and on the memory modules (number of read/write operations, number of bytes/words involved, …)
3. one or more trace files: those files trace instructions, frequency changes in the cores due to aging and the politics implemented, and related Vt, Vdd and static and dynamic energy
4. profiler outputs (one per HT of each core): those files illustrate how a quantity (cycles spent, power dissipation, …) are related to the pieces of code executed

9.1.1 <u>Simulation platform and associated parameters configuration</u>

The xSTreamISS is a functionally accurate timing approximate simulation platform (in short ISS) that is highly configurable. The configuration is the full set of parameters that can be tuned to configure a specific instance of the simulation platform: topological parameters (number and type of cores, devices connected to each core, latencies, frequencies, …) and user-level tracing options (e.g. if trace is enabled or not, for which cores and to which verbosity level, if dumping of statistics is enabled with level of detail, etc).
The default configuration can be dumped using the command-line option `--dump-config`. It contains all the parameters (along with their default values) for the system configuration that is dumped along with comments for each parameter. I.e., the command:

```
  xiss --ips='"[st231 gpe][xPE xpe1]"' --dump-config
```

dumps all the parameters for a system composed of an ST231 core called ‘`gpe`’ and an xPE core called ‘`xpe1`’.
Parameters are grouped in sections (a section can contain parameters and other sections) and can be assigned values and expressions of the following types: boolean, integer, unsigned integer, float and string. Expressions can contain references to other parameters, and the result will be cast to the type of the parameter before it is assigned.

The top (global) section has no name and contains the global ISS parameters. A generic section has a global identifier given by the list of local identifiers of the sections containing that section. For example, if section a contains section b, the global identifier of section b is a.b, and if section a.b contains section c, the global identifier of section c is a.b.c.
The global identifier of a parameter is given by the global identifier of the section containing that parameter, a dot and the (local) identifier of the parameter. For global parameters the global identifier is identical to the local identifier.

9.1.1.1 Global ISS parameters

The global ISS parameters belong to the global section and allow changing generic aspects of the ISS behavior and the topology of the system to some extent (number and types of the master IP models):

Parameter	**Type**	**Comment**
ips	string	List of master IP model instances, in the format '['<ip> <id>']'.
accuracy	unsigned integer	Global accuracy level (the user can selectively change the accuracy of some master IP models and/or some devices).
trace	unsigned integer	Specify if trace is enabled and which information will be traced. The value of this parameter contained in the global section is a default value that can be changed for individual IP models. The meaning of the numerical value of this parameter is IP model dependent; the following table is valid for core models: 0: do not trace instructions 1: trace instructions 2: as 1 + symbols 3: as 2 + time 4: as 3 + instructions encoding 5: as 4 + registers values > 5: as 5 + details on stages execution
trace_only_jumps	boolean	This parameter, if TRUE, specifies that, for all core models that have instruction trace enabled, not all instructions are traced but only the bundles that cause branches and the bundles target of branches. In case of exceptions and interrupts, the first bundles of the exception and interrupt handlers are traced. This parameter is only valid for core models, and the value contained in the global section is a default value, that can be changed for individual core models.
trace_only_calls	boolean	This parameter, if TRUE, specifies that, for all core models that have instruction trace enabled, not all instructions are traced but only the bundles that cause function calls and the bundles target of function calls. In case of exceptions and interrupts, the first bundles of the exception and interrupt handlers are traced. This parameter is only valid for core models, and the value contained in the global section is a default value, that can be changed for individual core models.
trace_regs	unsigned integer	This parameter allows tracing machine registers for all core models that have instruction trace enabled. The value of this parameter contained in the global section is a default value that can be changed for individual core models. The meaning of the numerical value of this parameter is: 0: do not trace registers 1: trace only changed registers 2: trace all registers >2: trace all registers with changed registers highlighted
trace_cregs	unsigned integer	Equivalent to trace_regs for tracing control registers.
trace_start_cycle	integer	In tracing mode, don't start tracing until this main clock cycle is reached.

trace_stop_cycle	integer	In tracing mode, stop tracing when this main clock cycle is reached.
dump_dir	string	Destination directory for all the dumps.
dump_name	string	Base name of all the dumps (a suffix will be appended to this name).
dump_log	boolean	If TRUE, dump the log of all the simulation to a file.
dump_perfs	boolean	If TRUE, dump ISS performance details to a file.
dump_msg_to_file	boolean	If TRUE, dump all the messages to a file.
dump_trc_to_file	boolean	If TRUE, dump the trace to files, if enabled.
multiple_trc_files	boolean	If TRUE, dump one trace file per IP model (if trace is enabled and dumped to files), if FALSE only one trace file is dumped.
dump_stats	string	Define which statistics must be dumped and the format of those dumps.

9.1.1.2 System section

The 'system' section contains just one parameter.

Parameter	**Type**	**Comment**
main_clock_freq	integer	Value of the main clock frequency in Hz. The main clock frequency is the clock frequency that is used for deriving all the frequencies of the IPs and devices contained in the system. The frequency of an IP or a device is defined as main clock frequency divided by an integer ratio.

9.1.1.3 Individual cores parameters

The parameters related to a core model are placed inside a configuration section having the same identifier as the core identifier.

Parameter	**Type**	**Comment**
target_exec	string	Target executable path (if any).
target_args	string	Space-separated list of arguments for the ELF target executable (if any).
load_files	string	Space-separated list of paths of ELF files that must be loaded (if any).
env	string	Space-separated list of environment variables (if empty, the host environment will be used).
num_of_profilers	unsigned integer	Number of profiler devices connected to this core.
trace	unsigned integer	Specify if trace is enabled and which information will be traced (see global section). The default value is the one of the 'trace' parameter contained in the global section.
trace_only_jumps	boolean	If TRUE and instruction trace is enabled, only the bundle immediately preceding and the bundle immediately following jumps are traced (see the global section). The default value is the one of the 'trace_only_jumps' parameter contained in the global section.
trace_only_calls	boolean	If TRUE and instruction trace is enabled, only the bundle immediately preceding and the bundle immediately following function calls are traced (see the global section). The default value is the one of the 'trace_only_calls' parameter contained in the global section.
trace_regs	unsigned integer	If and when instruction trace is enabled, specify if trace of machine registers is enabled and which information will be traced (see the global section). The default value is the one of the 'trace_regs' parameter contained in the global section.

trace_cregs	unsigned integer	If and when instruction trace is enabled, specify if trace of control registers is enabled and which information will be traced (see the global section). The default value is the one of the ‘trace_cregs’ parameter contained in the global section.
trace_start_cycle	integer	In tracing mode, don't start tracing until this (main clock) cycle is reached. The default value is the one of the ‘trace_start_cycle’ parameter contained in the global section.
trace_stop_cycle	integer	In tracing mode, stop tracing when this (main clock) cycle is reached. The default value is the one of the ‘trace_stop_cycle’ parameter contained in the global section.
trace_start_bundle	integer	In tracing mode, don't start tracing until this bundle is reached.
trace_stop_bundle	integer	In tracing mode, stop tracing when this bundle is reached
trace_low_pc	unsigned integer	Value of the PC starting the trace window (if tracing is enabled).
trace_high_pc	unsigned integer	Value of the PC ending the trace window (if tracing is enabled).
trace_start_pc	unsigned integer	Value of the PC where tracing must start (if enabled).
trace_stop_pc	unsigned integer	Value of the PC where tracing must be stopped (if enabled).
trace_start_pc_count	unsigned integer	Number of times the PC must be equal to the value of the 'trace_start_pc' parameter before starting tracing (if enabled).
trace_stop_pc_count	unsigned integer	Number of times the PC must be equal to the value of the 'trace_stop_pc' parameter before stopping tracing (if enabled).
trace_excs	boolean	If TRUE, trace exceptions even if instruction trace is not activated.
dump_stats_at	string	List of PC addresses and labels where statistics must be dumped (if statistics are activated).

9.1.1.4 <u>Core section of a core model</u>

Each core model has a ‘core’ subsection (global identifier is ‘<core>.core’, where <core> is the identifier of the core model), containing parameters related to the ‘core’ of the model.

Parameter	**Type**	**Comment**
index	unsigned integer	Core identifier in the system (must be unique). This value is assigned automatically.
clock_ratio	unsigned integer	Ratio between the main clock frequency and the core clock frequency. The default value is 1. The clock ration can be changed at runtime.
endianness	string	Possible values are 'little', 'big' and 'auto' (detected from target files).
branch_penalty	unsigned integer	Number of core clock cycles needed in case of branch.
num_of_hts (if the core has multiple hardware threads)	snsigned integer	Number of hardware threads (maximum value is 16).

9.1.1.5 <u>Subsystem section of a core model</u>

Each core model has a ‘subsystem’ subsection (global identifier is ‘<core>.subsystem’, where <core> is the identifier of the core model), containing parameters related to the devices directly connected to the core and composing its ‘subsystem’.

Parameter	**Type**	**Comment**
ibuff_enabled	boolean	TRUE if the instruction buffer is enabled, FALSE otherwise.
wbuff_enabled	boolean	TRUE if the write buffer is enabled, FALSE otherwise.

rambase	unsigned integer	Base address of the RAM memory module.
ramsize	unsigned integer	Size in bytes of the RAM memory module.
devices	string	Space-separated list of identifiers of the devices used by the core model.
devices_stack	string	String describing the position of the devices used by the core model.

9.1.1.6 Devices section

The 'devices' configuration section contains the following parameter:

Parameter	Type	Comment
external_classes	string	String listing the dlls implementing the custom devices and the functions for registering them in the ISS.

9.1.1.7 Section related to a device class

Each device class having identifier <class> defines a configuration class having global identifier equal to 'devices.<class>' (placed inside the 'devices' configuration section). A device class defines features common to all the instances of that class.

Parameter	Type	Comment
on_stack	boolean	TRUE if the devices of that class are memory mapped, FALSE otherwise (i.e. the profiler device). This parameter cannot be changed.
instances	string	Space-separated list of devices that must be created for the class.

9.1.1.8 Section related to a device instance

Each device instance <id> of a class having identifier <class> defines a configuration class having global identifier equal to 'devices.<class>.<id>' (placed inside the 'devices.<class>' configuration section). A device instance defines an instance of a class and can be connected to an IP and to other device instances.
The following table lists the parameters that are common to all devices.

Parameter	Type	Comment
shared	boolean	TRUE if the device is shared between two or more IPs, FALSE otherwise. This parameter cannot be changed and is automatically set to TRUE if more than one core declares to use the same device.
ips	string	Space-separated list of identifiers of the IP models owning this device.
num_of_int_in_pins (if the device defines input interrupt pins)	unsigned integer	Number of input interrupt pins. This parameter can be modified or not depending on the implementation of the device.
num_of_int_out_pins (if the device defines output interrupt pins)	unsigned integer	Number of output interrupt pins. This parameter can be modified or not depending on the implementation of the device.
int_out_pins_connections (if the device defines output interrupt pins)	string	String specifying how the output interrupt pins are connected.

Parameters specific to memory device instances

The following table lists the parameters that are defined by memory devices (class 'mem', configuration section 'devices.mem.<id>', where <id> is the identifier of the device).

Parameter	Type	Comment

clock_ratio	unsigned integer	Ratio between the main clock frequency and the device's clock frequency.
accuracy	unsigned integer	0: functional with no timing 1: functional with timing >1: cycle accurate
base	unsigned integer	Base address of the memory area the device is mapped onto.
size	unsigned integer	Size in bytes of the memory area the device is mapped onto.
log2_bus_width	unsigned integer	Log 2 of the bus width in bytes.
read_delay_first	unsigned integer	If accuracy >= 1, memory cycles needed to read the first (or only) object.
read_delay_next	unsigned integer	If accuracy >= 1, memory cycles needed to read the following objects during a burst access (see log2_bus_width).
write_delay_first	unsigned integer	If accuracy >= 1, memory cycles needed to write the first (or only) object.
write_delay_next	unsigned integer	If accuracy >= 1, memory cycles needed to write the following objects during a burst access (see log2_bus_width).
read_only	boolean	If TRUE, the device acts like a ROM, otherwise it is a RAM.
watch	string	List of address ranges that must be watched.
stats_hist_max_num_of_words	unsigned integer	Maximum number of words that must be considered for the statistics histograms (if statistics must be dumped).

9.1.1.9 Parameters specific to interconnect device instances

The following table lists the parameters that are defined by interconnect devices (class ‘icn’, configuration section ‘devices.icn.<id>’, where <id> is the identifier of the device)

Parameter	**Type**	**Comment**
clock_ratio	unsigned integer	Ratio between the main clock frequency and the device's clock frequency.
topology	string	Possible values are 'ideal', 'shared', 'crossbar' and 'custom'.
custom_topology	string	String describing how to connect arbiters and targets. This parameter accepts a string composed of one or more tokens of the following type: '['<id1> ... <idn>']' where <id1> ... <idn> are the identifiers of targets (IP model names) that must share the same arbiter. So each token specifies an arbiter and the group of targets that are connected to it. The same target can appear only once in the list. If a target is not contained in the list, it is assumed to be connected to its private arbiter, unless an empty group ('[]') is present in the list, which means that all targets that have not been specified in the list are connected to the same arbiter. Using the 'custom' topology and 'custom_topology' parameter, a shared bus is identified by the string '[]' and a crossbar is identifier by the empty string.
visible_classes	string	Space-separated list of device classes that will be seen and mapped by the interconnect.
log2_width	unsigned integer	Log 2 of the bus width in bytes.
request_cycles	unsigned	If the interconnect is not ideal, number of device clock

	integer	cycles needed to propagate a request.
response_cycles	unsigned integer	If the interconnect is not ideal, number of device clock cycles needed to propagate a (word of) response.
stats_hist_max_num_of_words	unsigned integer	Maximum number of words that must be considered for the statistics histograms (if statistics must be dumped; 0 if histograms must not be generated).

9.1.1.10 Parameters specific to profiler devices

The following table lists the parameters that are defined by profiler devices (class ‘profiler’, configuration section ‘devices.profiler.<id>’, where <id> is the identifier of the device).

Parameter	**Type**	**Comment**
mode	string	Describe the quantity that must be profiled (e.g. ‘CYCLES’ will profile the number of cycles spent executing a piece of code).
ht_index	unsigned integer	Index of the hardware thread the profiler is connected to.
pc_low	unsigned integer	Start PC of the range of PCs being profiled.
pc_high	unsigned integer	End PC of the range of PCs being profiled.
pc_granularity	unsigned integer	Constant used to subsample the address space (min = 1, max = 8).
bundles_poll_interval	unsigned integer	Number of bundles (min = 1) executed between successive invocations of the profiler.
branch_call_equivalence	boolean	If TRUE, build a branch graph instead of a classical call graph.
custom_plugin	string	Path of the dll containing the function to call for custom profiling (in this case, mode must be equal to “CUSTOM”).
custom_function	string	Name of the function to call for custom profiling.

9.1.1.11 Examples

Example 1

A configuration file containing the following line defines a system with a single ST231 core and a target application:

```
ips    “[st231 gpe]”

[gpe]
targer_exec    “example.elf”
```

This file is passed to the ISS using the command:

```
xiss --config-file=<cfg>
```

where <cfg> is the path of the configuration file.
The same results can be obtained with the following command:

```
xiss --ips=’”[st231 gpe]”’ --gpe.target_exec=’”example.elf”’
```

In general, a parameter can be passed on the command line placing a ‘--‘ before its global identifier (dot-separated list of sections containing the parameter plus a dot plus the parameter’s identifier) and a ‘=’ after. The expression must be surrounded by ‘ (primes) if it contains spaces or double quotes (strings).

Example 2
A configuration file containing the following lines defines a system with a ST231 core, an xPE core, two target applications and a crossbar with 2 cycles for propagating requests and responses:

```
ips    “[st231 gpe][xPE xpe1]”

[gpe]
target_exec    “example.gpe”

[xpe1]
target_exec    “example.xpe1”

[devices.icn.d]
topology           “crossbar”
request_cycles     2
response_cycles    2
```

Those parameters can be passed onto the command line in the following way:

```
xiss --ips='”[st231 gpe][xPE xpe1]”'
     --gpe.target_exec='”example.gpe”'
     --xpe1.target_exec='”example.xpe1”'
     --devices.icn.d.topology='”crossbar”'
     --devices.icn.d.request_cycles=2
     --devices.icn.d.response_cycles=2
```

This clearly shows that creating a configuration file is more comfortable than passing the parameters onto the command line when more than a few parameters are involved.
More than one configuration file can be passed to the ISS, and also configuration files can be mixed up with parameters on the command line. In those cases, if a parameter is specified more than once, the value of that parameter valid for simulation is the last value the ISS receives. For example:

```
xiss --config-file=<cfg> --devices.icn.d.request_cycles=3
```

if `<cfg>` is the path of the configuration file shown above, defines an interconnect having 3 request cycles and 2 response cycles., while:

```
xiss --devices.icn.d.request_cycles=3 --config-file=<cfg>
```

defines an interconnect having 2 request cycles and 2 response cycles.

9.1.2 Integration of power and aging models with the simulation platform

The monitor_core device is used for monitoring a core’s activity, power consumption and aging, and scaling its frequency accordingly. It has some memory mapped registers that can be accessed from any core in the system. It implements the modeling equations defined in WP 5.4 (see also 9.1.3).
This plug-in can compute overall statistics on (per core) power consumption and estimated lifetime, and punctual information about (per core) Vt, temperature, dynamic energy, static energy and clock frequency, computed according to the technique shown in 9.1.3.
It can be configured through the following parameters:

Parameter	Type	Description
base	unsigned integer	base address for the memory mapped registers.
Vdd	unsigned integer	Input voltage [mV].
Vt	unsigned integer	Threshold voltage [mV].
T	unsigned integer	Temperature [K].
Area	float	Area [mm * mm].
Kds	float	Proportionality constant for dynamic consumption in stall state.
Kss	float	Proportionality constant for static consumption in stall state.
Kdi	float	Proportionality constant for dynamic consumption in idle state.
Ksi	float	Proportionality constant for static consumption in idle state.
Kdpg	float	Proportionality constant for dynamic consumption in power gating.
Kspg	float	Proportionality constant for static consumption in power gating.
PdynA	float	Dynamic power in activity state [W].
PstaA	float	Static power in activity state [W].
alpha	float	Techn constant (power formula).
u	float	Parameter μ (power formula).
W	unsigned integer	Initial temperature (temperature formula) [K].
Ea	float	Parameter Ea (NBTI formula) [eV].
E0	float	Parameter E0 (NBTI formula) [V / nm].
tox	float	Parameter tox (NBTI formula) [nm].
Cox	float	Parameter Cox (NBTI formula) [F / nm^2].
Ni	float	Parameter η (NBTI formula).
Anbti	float	Parameter Anbti (NBTI formula) [(V^2 / (F^2 sec))^(1/4)].
Af	float	Forward dynamic dependency (termal model).
Am	float	Mutual dynamic dependency (termal model).
Bf	float	Forward static dependency (termal model) [K mm^2 W^-1].
Bm	float	Mutual static dependency (termal model) [K mm^2 W^-1].
Guardband	float	Guardband [0..1].

Table 1: ISS variability modelling plug-in parameters

The memory mapped registers implemented by the device are the following:

Register	Offset	Description
REG_VDD	0x00	W/R register, to set Vdd [mV]
REG_VT	0x04	R register, Vt [mV]
REG_FCK	0x08	W/R register, to set Fck
REG_T	0x0c	R register: temperature [K]
REG_EDYN	0x10	R register: dynamic energy [nJ]
REG_ESTA	0x14	R register: static energy [pJ]
REG_ETOT	0x18	R register: total energy [nJ]
REG_CyclesA	0x1c	R register: cycles passed in activity [#]
REG_CyclesI	0x20	R register: cycles passed in idle [#]
REG_CyclesS	0x24	R register: cycles passed in stall [#]
REG_CyclesPG	0x28	R register: cycles passed in power-gating [#]

Table 2: ISS measuring and control knob registers interface

9.1.3 Power and Aging Modelling

In order to estimate energy consumptions, the system platform models need extending with a sensible analytical model, the model we adopted is proposed by UniBo and summarized here, so that the proper parameters are specified for it's adoption during the WP6 benchmarking campaigns.
The method proposed defines equations to extrapolate the power/speed when $\boldsymbol{V_{dd}}$, $\boldsymbol{V_t}$, $\boldsymbol{f}$ are altered with respect to a "reference condition", and it uses fitting constants to tight up equations to available measured data. We give a detailed example on how to use the model for the case of a core processor such as the one deployed in the REALITY system level platform. Technological parameters used in the description are not final and/or realistic, while definitive ones must be provided as input by relevant partners as explained later on in the document.

Let's assume that we only know the following "measurement" data from a processor core *CoreX* in technology *TechX*.

The core is in nominal conditions $\boldsymbol{f_{nom}}$, $\boldsymbol{V_{ddnom}}$, $\boldsymbol{V_{tnom}}$, $\boldsymbol{T}$. We know:
a. Dynamic and Static Power consumed in fully active conditions (i.e. executing instructions) : $\boldsymbol{P_{dyn}}$**-A**, $\boldsymbol{P_{sta}}$**-A**;
b. Power consumed in stalled condition (i.e. waiting for a memory operation): $\boldsymbol{P_{dyn}}$**-S**, $\boldsymbol{P_{sta}}$**-S**;
c. Power consumed in idle conditions (i.e. no instructions are executing): $\boldsymbol{P_{dyn}}$**-I**, $\boldsymbol{P_{sta}}$**-I**;
d. Power consumed in power gating conditions: $\boldsymbol{P_{dyn}}$**-PG**, $\boldsymbol{P_{sta}}$**-PG**.

We use the following equations to extrapolate the behavior at different operating points:

$$T_g = \frac{1}{f} = \frac{L_f \cdot V_{dd}}{(V_{dd} - V_t)^{\alpha}}$$ delay that sets the maximum frequency of the related core (1)

$$P_{dyn} = K_d \cdot V_{dd}^2 \cdot f$$ dynamic power consumption of one core (2)

$$P_{lkg} = Z \cdot V_{dd} \cdot T^2 \cdot \mathrm{e}^{\frac{-q \cdot V_t}{K \cdot T}}$$ leakage power consumption of one core (3)

We identified and set the following parameters:

Physical-chemical constants	
K: Boltzman costant [J K-1]	**q**: electron charge [C]
1,38E-23	1,60E-19

Table 3: Physical-Chemical Constants

Technological Parameters							
Vt: threshold voltage [V]	**E0**: [V nm-1]	**α** (by Facelift):	**μ** :	**tox** [nm]:	**Cox** [F nm-2]:	**η**:	**E1** [V nm-1]:
0,50	0,20	1,30	1	0,65	4,60E-20	0,35	0,80

Table 4: Technological Parameters

Using the knowledge of on $\boldsymbol{P}$, $\boldsymbol{f}$ in nominal conditions for all states (a-d), we can determine the proportionality factors: $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$, the equations can then be used, if we change $\boldsymbol{V_t}$ (Adaptive Body Bias), $\boldsymbol{V_{dd}}$ (voltage scaling), $\boldsymbol{T}$, to determine the new $\boldsymbol{f}$, $\boldsymbol{P_{sta}}$, $\boldsymbol{P_{dyn}}$ in all states (a-d).
Of course, if we have more than one operating point (values for a-d for more than one set of $\boldsymbol{V_{dd}}$, $\boldsymbol{V_t}$, $\boldsymbol{T}$) we can improve accuracy, by using different $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$ in different regions.
Now, to model process variations, we can differentiate from the nominal core, by assigning "perturbed" $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$ to the cores that deviate from nominal.
The way to assign $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$, can be extracted by the VAM power and timing histogram.
This approach while been relatively simple can provide reliable trends for system-level estimation.

It must be noted that we are assuming here that the core has no caches, this is in line with the xPE processing engine provided by ST for the REALITY system platform. The approach works reasonably well for cores with caches as well, but it's less accurate. The best approach to model cores with caches is to use two models: one for the core and one for the caches (see memory model below 2.2). Of course this implies that we have to obtain some more profiling information if we want to estimate power (i.e. we need to know when caches are accessed).

If a core deviates from nominal with a different frequency and different power, in nominal conditions, we will have a different $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$. The same equations above (with the perturbed $\boldsymbol{L_f}$, $\boldsymbol{Z}$, $\boldsymbol{K_d}$), can be used to estimate how the perturbed core will behave when we change from nominal conditions $\boldsymbol{V_{dd}}$, $\boldsymbol{V_t}$, $\boldsymbol{T}$.

A similar approach is used for the remaining blocks in the architecture, namely memories and interconnects. For a (SRAM) memory, we need to know at least the following data in nominal operating conditions:

a. Dynamic and Static Power consumed in read/write (possibly differentiating read and write): $\boldsymbol{P_{dyn}\text{-}A}$, $\boldsymbol{P_{sta}\text{-}A}$;
b. Power consumed in idle conditions (i.e. no instructions are executing): $\boldsymbol{P_{dyn}\text{-}I}$, $\boldsymbol{P_{sta}\text{-}I}$;
c. Power consmed in power gating conditions: $\boldsymbol{P_{dyn}\text{-}PG}$, $\boldsymbol{P_{sta}\text{-}PG}$;

d. optionally, if there is a state preserving power gating condition, we need power for this too.

We also need to know the access time in nominal operating conditions.

It must be noted that the model can be used to estimate the power consumed by larger or smaller memories, using a ***N²*** factor to account for memory size. However, this is quite inaccurate and can be used in small ranges of memory. The ideal knowledge is to have the data above for each memory that we want to consider at the system level. We will be using the well-known CACTI tool to get this information. The tool makes reasonable assumptions on memory architecture, so if we are given data for only one memory size, we can use cacti and a scaling factor to get reasonable estimate of the power consumed for different memories. In order to make realistic estimates, the memories provided by ARM will be modeled as faithfully as possible.

It must be noted that the same approach described for cores can be used to deal with process variability. For a system interconnect (BUS/NOC), we need to know at least the following data in nominal conditions:

a. Dynamic and Static Power consumed in busy conditions (i.e. worst case activity expressed as number on-going transactions) : ***P_{dyn}-A***, ***P_{sta}-A***;
b. Power consumed in idle conditions (i.e. no transactions): ***P_{dyn}-I***, ***P_{sta}-I***;
c. Power consumed in power gating conditions: ***P_{dyn}-PG***, ***P_{sta}-PG***.

Here, we use again the same equations above, plus a linear interpolation between ***P_{dyn}-A***, ***P_{din}-I*** using the number of active transactions (from 0 to Nmax) as additional control variable (this can be quite easily obtained from system simulation).

9.1.4 Power consumption of cores and interconnect

We considered cores that can enter in a explicit idle state. Explicit idleness happens when a core is accessing I/O or is waiting for some other process on a synchronization point. We assume that power consumption for this state with and without the power-gating mode is a fraction of active power. We also consider an implicit idle state, which happens when a core performs idle cycles to wait for read or write into memory.
In this situation we have:

$$P_{dynimplicitidle} = 0.5 \cdot P_{dynact} \tag{4a}$$

$$P_{dynexplicitidle} = 0.1 \cdot P_{dynact} \tag{4b}$$

$$P_{dynexplicitidle-power-gating} = 0 \tag{4c}$$

$$P_{lkgidle} = P_{lkgact} \tag{5a}$$

$$P_{lkgidle-power-gating} = 0.02 \cdot P_{lkgact} \tag{5b}$$

These relations are related at cores, whilst for the interconnect we have:

$$P_{dyninterconnect} = 0.2 \cdot P_{dyncoreact} \qquad \text{(interconnect completely busy - 6a)}$$

$$P_{dyninterconnect} = 0.1 \cdot P_{dyncoreact} \qquad \text{(one core uses the interconnect - 6b)}$$

$$P_{lkginterconnect} = P_{lkgcore} \qquad (7)$$

We considered that dynamic power consumption of interconnect can vary from 10% to 20% of active core power, and it is not possible to put the interconnect in the power-gating mode.

For monitoring the changes of power consumption and maximum frequency with the supply voltage, we need a nominal values set for calculating the constant of proportionality. We used the following values as an example.

System Parameters – nominal values				
Vdd: supply voltage [V]	**T**: temperature [K]	**fck**: clock frequency [Hz]	**Pdyn act** [W]	**Plkg act** [W]
1,10	300	4,00E+08	1,00E-03	1,00E-06

Table 5: System Parameters

With these values we have the following constants of proportionality and the related table of maximum frequency and power consumption:

Constants of proportionality		
Leff: (1) [V sec]	**Kd**: (2) [sec Ω-1]	**Z**: (3) [A T-2]
1,17E-09	2,07E-12	2,47E-03

Table 6: Constants of proportionality

fck [Hz] = fck MAX												
Vdd [V]	Pdyn core act [W] (2)	Pdyn core implicit idle [W] (4a)	Pdyn core esplicit idle [W] (4b)	Pdyn core esplicit idle powergating [W] (4c)	Pdyn int act busy[W] (6a)	Pdyn int act min[W] (6b)	Psta core [W] (3)	Psta core idle [W] (5a)	Psta core idle powergating [W] (5b)	Psta int [W] (7)	Tg [sec] (1)	fck MAX [Hz] = 1/Tg
0,80	2,95E-04	1,48E-04	2,95E-05	0,00E+00	5,91E-05	2,95E-05	7,27E-07	7,27E-07	1,45E-08	7,27E-08	4,48E-09	2,23E+08
0,85	3,83E-04	1,92E-04	3,83E-05	0,00E+00	7,67E-05	3,83E-05	7,73E-07	7,73E-07	1,55E-08	7,73E-08	3,89E-09	2,57E+08
0,90	4,83E-04	2,41E-04	4,83E-05	0,00E+00	9,66E-05	4,83E-05	8,18E-07	8,18E-07	1,64E-08	8,18E-08	3,47E-09	2,89E+08
0,95	5,94E-04	2,97E-04	5,94E-05	0,00E+00	1,19E-04	5,94E-05	8,64E-07	8,64E-07	1,73E-08	8,64E-08	3,14E-09	3,19E+08
1,00	7,17E-04	3,59E-04	7,17E-05	0,00E+00	1,43E-04	7,17E-05	9,09E-07	9,09E-07	1,82E-08	9,09E-08	2,88E-09	3,47E+08
1,05	8,52E-04	4,26E-04	8,52E-05	0,00E+00	1,70E-04	8,52E-05	9,55E-07	9,55E-07	1,91E-08	9,55E-08	2,67E-09	3,74E+08
1,10	1,00E-03	5,00E-04	1,00E-04	0,00E+00	2,00E-04	1,00E-04	1,00E-06	1,00E-06	2,00E-08	1,00E-07	2,50E-09	4,00E+08
1,15	1,16E-03	5,80E-04	1,16E-04	0,00E+00	2,32E-04	1,16E-04	1,05E-06	1,05E-06	2,09E-08	1,05E-07	2,36E-09	4,25E+08
1,20	1,33E-03	6,66E-04	1,33E-04	0,00E+00	2,67E-04	1,33E-04	1,09E-06	1,09E-06	2,18E-08	1,09E-07	2,23E-09	4,48E+08
1,25	1,52E-03	7,59E-04	1,52E-04	0,00E+00	3,04E-04	1,52E-04	1,14E-06	1,14E-06	2,27E-08	1,14E-07	2,13E-09	4,70E+08
1,30	1,72E-03	8,59E-04	1,72E-04	0,00E+00	3,44E-04	1,72E-04	1,18E-06	1,18E-06	2,36E-08	1,18E-07	2,03E-09	4,92E+08
1,35	1,93E-03	9,65E-04	1,93E-04	0,00E+00	3,86E-04	1,93E-04	1,23E-06	1,23E-06	2,45E-08	1,23E-07	1,95E-09	5,13E+08
1,40	2,16E-03	1,08E-03	2,16E-04	0,00E+00	4,31E-04	2,16E-04	1,27E-06	1,27E-06	2,55E-08	1,27E-07	1,88E-09	5,32E+08

Table 7: Maximum frequency and power consumption at the fixed supply voltage. The yellow row shows the nominal case.

By scaling the frequency below the maximum we have:

Vdd [V] = 1,10						
fck [Hz]	Pdyn core act [W] (2)	Pdyn core implicit idle [W] (4a)	Pdyn core esplicit idle [W] (4b)	Pdyn core esplicit idle powergating [W] (4c)	Pdyn int act busy[W] (6a)	Pdyn int act min[W] (6b)
2,00E+08	5,00E-04	2,50E-04	5,00E-05	0,00E+00	1,00E-04	5,00E-05
2,25E+08	5,63E-04	2,81E-04	5,63E-05	0,00E+00	1,13E-04	5,63E-05
2,50E+08	6,25E-04	3,13E-04	6,25E-05	0,00E+00	1,25E-04	6,25E-05
2,75E+08	6,88E-04	3,44E-04	6,88E-05	0,00E+00	1,38E-04	6,88E-05
3,00E+08	7,50E-04	3,75E-04	7,50E-05	0,00E+00	1,50E-04	7,50E-05
3,25E+08	8,13E-04	4,06E-04	8,13E-05	0,00E+00	1,63E-04	8,13E-05
3,50E+08	8,75E-04	4,38E-04	8,75E-05	0,00E+00	1,75E-04	8,75E-05
3,75E+08	9,38E-04	4,69E-04	9,38E-05	0,00E+00	1,88E-04	9,38E-05
4,00E+08	1,00E-03	5,00E-04	1,00E-04	0,00E+00	2,00E-04	1,00E-04

Table 8: Dynamic power consumption related to frequency switching at fixed supply voltage. The yellow row shows the nominal case: frequency equal to maximum frequency for a given Vdd.

9.1.5 Power consumption of memories

The power consumption in active state can be the same as for the core (i.e. (1), (2), and (3)), but for the idle state we consider that dynamic power depends on the cell count:

$$P_{dynmemidle} = K_i \cdot N_{cell}^2 \cdot P_{dynmemact} \qquad (8)$$

In idle state the leakage power is the same that in active state like in (5a). We have a new constant of proportionality ***K**$_i$*. As nominal values of power consumption we used CACTI. Setting are shown in the appendix, while system parameters are shown in Table 2.5, constants are shown in Table 2.6 and power consumption in Table 2.7.

System Parameters – nominal values						
Vdd: supply voltage [V]	**T**: temperature [K]	**fck**: clock frequency [Hz]	**Pdyn act per read at max freq** [W]	**Plkg act** [W]	**Memory cells** [#]	**Pdyn idle at max freq** [W]
1,10	300	1,33E+08	1,12E-01	2,53E-05	8,10E+01	1,12E-02

Table 9: System Parameters, blue-gray pointed out values are taken from CACTI

Constants of proportionality			
Leff: Tg (1) [V sec]	**Kd**: Pdyn (2) [sec Ω-1]	**Z**: Psta (3) [A T-2]	**Ki** [#cells-2]
3,52E-09	6,99E-10	6,24E-02	1,53E-05

Table 10: Constants of proportionality

fck [Hz] = fck MAX						
Vdd [V]	Pdyn act per read at max freq [W] (2)	Pdyn idle at max freq [W] (8)	Psta mem [W] (3)	Psta mem idle [W] (5a)	Tg [sec] (1)	fck MAX [Hz] = 1/Tg
0,80	3,32E-02	3,32E-03	1,84E-05	1,84E-05	1,35E-08	7,43E+07
0,85	4,31E-02	4,31E-03	1,95E-05	1,95E-05	1,17E-08	8,54E+07
0,90	5,43E-02	5,43E-03	2,07E-05	2,07E-05	1,04E-08	9,60E+07
0,95	6,68E-02	6,68E-03	2,18E-05	2,18E-05	9,44E-09	1,06E+08
1,00	8,07E-02	8,07E-03	2,30E-05	2,30E-05	8,66E-09	1,15E+08
1,05	9,59E-02	9,59E-03	2,41E-05	2,41E-05	8,04E-09	1,24E+08
1,10	1,12E-01	1,12E-02	2,53E-05	2,53E-05	7,52E-09	1,33E+08
1,15	1,30E-01	1,30E-02	2,64E-05	2,64E-05	7,08E-09	1,41E+08
1,20	1,50E-01	1,50E-02	2,76E-05	2,76E-05	6,71E-09	1,49E+08
1,25	1,71E-01	1,71E-02	2,87E-05	2,87E-05	6,39E-09	1,56E+08
1,30	1,93E-01	1,93E-02	2,99E-05	2,99E-05	6,11E-09	1,64E+08
1,35	2,17E-01	2,17E-02	3,10E-05	3,10E-05	5,87E-09	1,70E+08
1,40	2,42E-01	2,42E-02	3,22E-05	3,22E-05	5,65E-09	1,77E+08

Table 11: Maximum frequency and power consumption at fixed supply voltage; the yellow row shows the nominal case.

9.1.6 Aging effects on the cores

We base our equations on the models proposed in [Abh08], and hence we considered Negative Bias Temperature Instability (NBTI) and Hot-Carrier Injection (HCI). The maximum frequency is fixed by the equation (9):

$$T_g = \frac{L_{eff} \cdot V_{dd}}{\mu \cdot (V_{dd} - V_t)^{\alpha}} \qquad (9)$$

and for NBTI we have a $\varDelta \boldsymbol{V}_{t\_stress}$, and a $\varDelta \boldsymbol{V}_{t\_recovery}$ relationship:

$$\varDelta V_{tstress} = A_{NBTI} \cdot t_{ox} \cdot \sqrt{(C_{ox} \cdot (V_{dd} - V_t))} \cdot e^{\frac{V_{dd} - V_t}{t_{ox} \cdot E_0} - \frac{E_a}{K \cdot T}} \cdot t_{stress}^{0,25} \qquad \varDelta \boldsymbol{V}_{t\_stress} \text{ NBTI}$$

(10a)

$$\varDelta V_{trecovery} = \varDelta V_{tstress} \cdot \left(1 - \sqrt{\frac{\eta \cdot t_{recovery}}{t_{recovery} + t_{stress}}}\right) \qquad \varDelta \boldsymbol{V}_{t\_recovery} \text{ NBTI (10b)}$$

whilst for HCI we have the following $\varDelta \boldsymbol{V}_t$ relationship:

$$\varDelta V_t = A_{HCI} \cdot \alpha \cdot f \cdot e^{\frac{V_{dd} - V_t}{t_{ox} \cdot E_1}} \cdot \sqrt{t} \qquad \varDelta \boldsymbol{V}_t \text{ HCI (11)}$$

Considered parameters are shown in Table 3.1, 3.2, 3.3 and 3.4. Of course all of these parameters are just examples and numbers are not meaningful in a absolute way.

Physical-chemical constants			
K: Boltzman costant [ev K-1]	**K**: Boltzman costant [J K-1]	**q**: electron charge [C]	**Ea**: [eV]
8,62E-05	1,38E-23	1,60E-19	0,13

Table 12: Physical-Chemical Constants

Technological parameters							
Vt: threshold voltage [V]	**E0**: [V nm-1]	**α** (by Facelift):	**μ**:	**tox** [nm]:	**Cox** [F nm-2]:	**η**:	**E1** [V nm-1]:
0,50	0,20	1,30	1	0,65	4,60E-20	0,35	0,80

Table 13: Technological Parameters, blue-gray pointed out values are taken from [1]

We consider here a simple example by setting 1 msec lifetime (for NBTI case the first 0,5msec in stress and the after 0,5msec in recovery).
We need to set the nominal parameters and hence the constant of proportionality, in this case we based on 1 second of lifetime (1 second of stress for NBTI).

System Parameters – nominal values						
Vdd: supply voltage [V]	**T**: temperature [K]	**fck**: [Hz]	**Δvt_stress**: NBTI [V]	**tstress**: NBTI [sec]	**Δvt**: HCI [V]	**t** HCI: [sec]
1,10	300	4,00E+08	1,00	1,00E+00	1,00	1,00

Table 14: System Parameters

Constants of proportionality		
Leff: (9) [V sec]	**Anbti** (10a):	**Ahci** (11):
1,17E-09	6,13E+09	6,07E-10

Table 15: Constants of proportionality

We have the following tables of maximum frequency at provided supply voltage after 1msec:

HCI effect after t [sec] = 1,00E-03			
Vdd [V]	fck MAX[Hz]	Δvt [V] HCI after 't sec'	fck[Hz] after 't sec'
0,80	2,23E+08	9,92E-03	2,14E+08
0,85	2,57E+08	1,26E-02	2,42E+08
0,90	2,89E+08	1,55E-02	2,74E+08
0,95	3,19E+08	1,89E-02	3,01E+08
1,00	3,47E+08	2,26E-02	3,27E+08
1,05	3,74E+08	2,69E-02	3,51E+08
1,10	4,00E+08	3,16E-02	3,73E+08
1,15	4,25E+08	3,70E-02	3,93E+08
1,20	4,48E+08	4,29E-02	4,13E+08
1,25	4,70E+08	4,96E-02	4,30E+08
1,30	4,92E+08	5,71E-02	4,47E+08
1,35	5,13E+08	6,55E-02	4,62E+08
1,40	5,32E+08	7,49E-02	4,76E+08

Table 16: Case HCI. Maximum frequency at the fixed supply voltage after one msec. The yellow row shows the nominal case.

NBTI		NBTI after t= tstress =	5,00E-04	NBTI after t= tstress+trec=	1,00E-03
Vdd [V]	fck MAX[Hz]	Δvt [V] NBTI after 't sec'	fck[Hz] after 't sec'	Δvt [V] NBTI after 't sec'	fck[Hz] after 't sec'
0,80	2,23E+08	4,60E-03	2,19E+08	2,68E-03	2,21E+08
0,85	2,57E+08	7,30E-03	2,50E+08	4,25E-03	2,53E+08
0,90	2,89E+08	1,15E-02	2,78E+08	6,67E-03	2,82E+08
0,95	3,19E+08	1,79E-02	3,02E+08	1,04E-02	3,09E+08
1,00	3,47E+08	2,77E-02	3,22E+08	1,61E-02	3,33E+08
1,05	3,74E+08	4,26E-02	3,37E+08	2,48E-02	3,52E+08
1,10	4,00E+08	6,54E-02	3,44E+08	3,81E-02	3,67E+08
1,15	4,25E+08	1,00E-01	3,42E+08	5,82E-02	3,76E+08
1,20	4,48E+08	1,53E-01	3,25E+08	8,87E-02	3,76E+08
1,25	4,70E+08	2,32E-01	2,91E+08	1,35E-01	3,64E+08
1,30	4,92E+08	3,52E-01	2,32E+08	2,05E-01	3,35E+08
1,35	5,13E+08	5,33E-01	1,42E+08	3,10E-01	2,84E+08

Table 17: Case NBTI. Maximum frequency at the fixed supply voltage after one msec (0,5ms of stress and 0,5ms of recovery). The yellow row shows the nominal case.

9.1.7 Appendix: CACTI Settings

```
RAM Parameters:
Number of banks:1
Total RAM Size (bytes):1048576
Read/Write Ports per bank:0
Read Ports per bank:1
Write Ports per bank:1
Technology Size (nm):45
Vdd:1.1

Access time (ns): 4.30331272072
Random cycle time (ns):1.00555023003
Multisubbank interleave cycle time (of data array)
(ns):2.06872869887
Total read dynamic energy per read port(nJ): 0.113092121558
Total read dynamic power per read port at max freq (W):
0.112467898848
Total standby leakage power per bank (W): 2.52855449221e-05
Refresh power (percentage of standby leakage power): 0.0
Total area (mm^2): 10.0360776519
DRAM array refresh interval (microseconds):0.0
DRAM array availability (percentage):0.0
Best number of wordline segments: 8
Best number of bitline segments: 64
Best number of sets per wordline: 32.0
Best degree of bitline muxing: 8
Best degree of sense-amp level 1 muxing: 8
Best degree of sense-amp level 2 muxing: 1

Time Components:
delay_route_to_bank (ns):0.0
addr_din_horizontal_htree (ns):0.664709172059
addr_din_vertical_htree (ns):1.05227642755
row_predecode_driver_and_block (ns):0.35174309926
row_decoder (ns):0.292477166281
bitlines (ns):0.199038017678
sense_amp (ns):0.00900144542322
subarray_output_driver (ns):0.280534148919
bit_mux_predecode_driver_and_block (ns):0.445304130227
bit_mux_decoder (ns):0.0
senseamp_mux_lev_1_predecode_driver_and_block
(ns):0.392271195631
senseamp_mux_lev_1_decoder (ns):0.0
senseamp_mux_lev_2_predecode_driver_and_block (ns):-
1.03397576569e-16
senseamp_mux_lev_2_decoder (ns):0.0
delay_dout_vertical_htree (ns):0.788824071493
delay_dout_horizontal_htree (ns):0.664709172059

Power Components:
Percentage of total dynamic power:
routing_to_bank:0.0
addr_horizontal_htree:5.42906486152
datain_horizontal_htree:-2.18201427442e-42
addr_vertical_htree:19.0526039273
row_predecoder_drivers:0.0530725622489
row_predecoder_blocks:0.579880994908
row_decoder/driver:1.70032934201
bit_mux_predecoder_drivers:0.0265362811245
bit_mux_predecoder_blocks:0.893243127269
bit_mux_decoders:0.0
senseamp_mux_lev_1_predecoder_drivers:0.0265362811245
senseamp_mux_lev_1_predecoder_blocks:0.534829127058
senseamp_mux_lev_1_decoders:0.0
senseamp_mux_lev_2_predecoder_drivers:0.0
senseamp_mux_lev_2_predecoder_blocks:0.0
senseamp_mux_lev_2_decoders:0.0
bitlines:12.2536601827
sense_amps:0.994116614341
prechg_eq_drivers:8.67985459317
subarray_output_drivers:0.987806234573
dataout_vertical_htree:29.4178758242
dyn_dataout_horizontal_htree:19.3705900465

Percentage of total leakage power:
routing_to_bank:0.0
addr_horizontal_htree:0.0304484260676
datain_horizontal_htree:0.0608968521352
addr_vertical_htree:1.2179370427
row_predecoder_drivers:0.0169181407109
row_predecoder_blocks:0.828180617333
row_decoders/drivers:12.8780807994
bit_mux_predecoder_drivers:0.00845907035545
bit_mux_predecoder_blocks:0.403047425931
bit_mux_decoders:0.0
senseamp_mux_lev_1_predecoder_drivers:0.00845907035545
senseamp_mux_lev_1_predecoder_blocks:0.403047425931
senseamp_mux_lev_1_decoders:0.0
senseamp_mux_lev_2_predecoder_drivers:0.0
senseamp_mux_lev_2_predecoder_blocks:0.0
senseamp_mux_lev_2_decoders:0.0
memory cells:80.9733170943
sense_amps:0.161973586594
prechg_eq_drivers:0.202387740816
subarray_output_drivers:0.159510000837
dataout_vertical_htree:1.67298707233
dyn_dataout_horizontal_htree:0.0608968521352

Area Components:
area (mm2):10.0360776519
area_efficiency:49.7620479435
all_banks_height (micron):4892.60747921
all_banks_width (micron):2051.27382374
bank_height (micron):4436.75178162
bank_width (micron):2051.27382374
mat_height (micron):135.003493176
mat_width (micron):432.898455935
subarray_memory_cell_area_height (micron):56.448
subarray_memory_cell_area_width (micron):199.8
routing_area_height_within_bank (micron):116.64
routing_area_width_within_bank (micron):319.68
```

9.2 Application specific metrics and parameters

The applications selected for the system level benchmarking have been described in [D6.1] , for the purpose of the WP6 validation the applications selected are, mpeg2 video decoding and ac3 audio decoding.

9.2.1 Parallel MPEG 2 video decoder

For the mpeg2 video codec table 5 from [D6.1] lists all possible main rates, the subset of levels selected for the benchmarking is reproduced here:

Level number	**Max mb per second**	**Max frame size (macrob locks)**	**Max video bit rate (VCL) for Baseline, Extended and Main Profiles**	**Examples for high resolution @ frame rate (max stored frames) in Level**
2	11880	396	2 Mbit/s	320x240@36.0 (7) 352x288@30.0 (6)
2.1	19800	792	4 Mbit/s	352x480@30.0 (7) 352x576@25.0 (6)
2.2	20250	1620	4 Mbit/s	352x480@30.7(10) 352x576@25.6 (7) 720x480@15.0 (6) 720x576@12.5 (5)
3	40500	1620	10 Mbit/s	352x480@61.4 (12) 352x576@51.1 (10) 720x480@30.0 (6) 720x576@25.0 (5)

Table 18: Selected levels for the mpeg2 codec benchmarks

Suitable test streams will be identified that fit either the PAL or NTSC profiles in the last column, the application will be considered functional from a functional yield loss perspective if can achieve the specific level/profile with the required minimum number of frames.
The Input/Output format type is only of 4:2:0 sub-sampling (no 4:2:2 & 4:4:4), a minimum of 16 and maximum of 64 frames will be required in the test streams selected to achieve a meaningful prediction of the yield loss over few seconds of decoding.

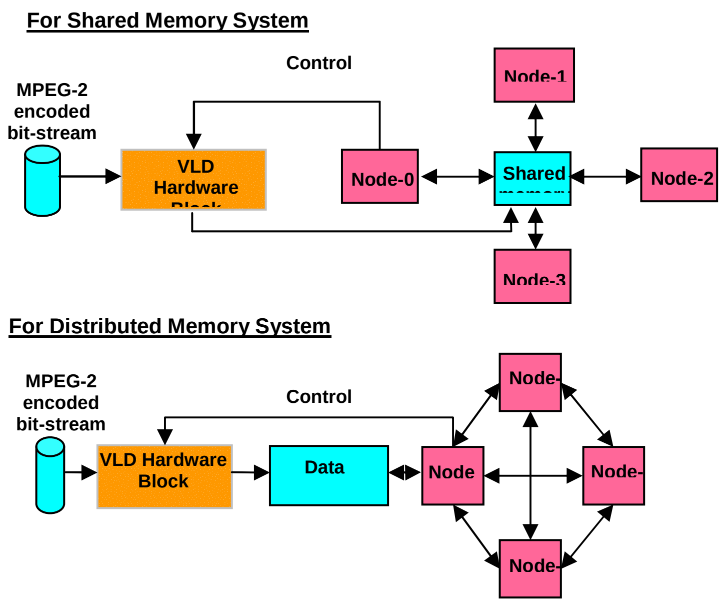

Figure 14: Shared and Distributed memory mpeg2 video decoder and encoder

The parallel mpeg2 decoder implementation will be ported to support the REALITY SW countermeasures scheduler API using a shared memory system.
The decoder using a distributed memory message passing API will only be benchmarked if time and available project resources will allow it. In the case of encoder benchmarking, the selected video modes parameters will be the same as of Table 18.

9.2.2 Parallel ac3 decoder

Dolby Digital is the marketing name for a series of lossy audio compression technologies developed by Dolby Laboratories.
Dolby Digital, or AC-3, is the common version containing up to six discrete channels of sound. The most elaborate mode in common usage involves five channels for normal-range speakers (20 Hz – 20,000 Hz) (right front, center, left front, right rear and left rear) and one channel (20 Hz – 120 Hz allotted audio) for the subwoofer driven low-frequency effects.
The benchmarking and validation task in WP6 will be performed with streams of up to 64 frames. From a functional yield perspective the application would be considered working correctly if able to decode DVD 5.1 encoded AC3 audio streams at 448 kbit/s in real time. This means that if each audio block contains 256 audio samples per channel, and 6×256 = 1536 = Audio frame size, then the decoder must be cable of producing no fewer than 781 frames/second.
The parallel ac3 decoder implementation will ported to support the REALITY SW countermeasures scheduler API using a shared memory system; no distributed message passing version is envisioned.

9.3 HW/SW control system level policies scenarios definition

9.3.1 Metrics

The system level countermeasures developed within REALITY project by UNIBO aims at minimizing one selected metric while keeping controlled or constant another one. For instance, a policy could be targeted to minimize performance impact of variability while another one could be devoted to minimize power energy consumption in presence of variability. The metrics considered will be:

1. Performance (variability impact compensation), In terms of application specific metrics, for the mpeg2 and ac2 decoder, the performance metrics are in terms of decoded video and audio frames per second.
2. Power consumption, in terms of peak mW required to decode both video and audio streams in real time according to the application metrics defined above
3. Average energy consumption is terms of mJ required to decode both video and audio stream in real time
4. Reliability (aging tolerance), in terms of functional yield loss over a time frame of 5 to 15 years of operation

Concerning reliability, it is important to note that in this project we will focus on a specific reliability aspect related to the impact of wear-out. We will not consider other types of effects impacting reliability such as radiation errors or electromagnetic noise.
Performance is to be considered as the capability of the policy to compensate for variability impact on system performance caused by core speed variations due to variations in voltage threshold or effective gate length. In a multiprocessor system, within-die variations may impact core speed depending on their location on the die. As such, variations will lead to performance unbalancing. Depending on target platform constraints, this may lead to two different situations. First, if the multiprocessor system has a single clock domain, typically the worst case critical path determines the whole system's operating frequency. Normally, a guard bound is used to compensate for aging effects. It must be noted that even if cores run at the same frequency, their dynamic and static power maybe different because of variability.
Alternatively, if multiple clock domains are allowed, each core will run at a different frequency. This implies that cores are characterized by different performance other than dynamic and static power consumption.
The REALITY simulation platform will provide the capability of running each core at a fractional frequency ratio with respect to a common clock reference for each core independently, see the simulation platform metrics for a detailed range of practical frequencies.

9.3.2 Software and Hardware knobs

In order to compensate for variability effects in terms of performance, power or energy, the policies will perform three types of software decisions:

1. Task allocation on the different cores
2. Task scheduling on the same core
3. Task migration from a core to another one.

The type of decision needed depends on the type of task model and the platform software infrastructure. For instance, scheduling is allowed only if an operating system support is present in each core. On the other side, task migration is possible if there is a software support for stopping the execution of a task on one core and resuming it on another core. UNIBO developed task migration support as part of the first deliverable for REALITY project. A port of an embedded operating system will be available.
Besides software decisions, policies may exploit hardware knobs provided in the target evaluation platform. In particular the following knobs can be considered if supported by the target platform:

1. Global frequency scaling (single domain)
2. Independent frequency scaling (independent core level domains)
3. Idleness distribution (clock gating)

Frequency scaling and idleness distribution impact performance. The power level associated depends on the specific features of the target platform. For instance, voltage scaling can be associated to frequency scaling. Concerning idleness, clock gating or power gating can be performed during idle periods.
For the WP6 benchmarking campaign, it is envisioned that global and independent (rational) frequency scaling will be available.

The simulation platform might be augmented with the ability to simulate the effects of measuring embedded HW knobs such as leakage current meters for different process and library corners, as well as temperature.
If both time and available resources allow it, the WP5/WP6 platform definition and benchmarking scenarios, will integrate a simple set of experiments exploiting one or more of these monitors, for example a leakage meter whose effects will be modeled based on the existing 45nm implementation from ST, extrapolating their behavior to 32nm technology.

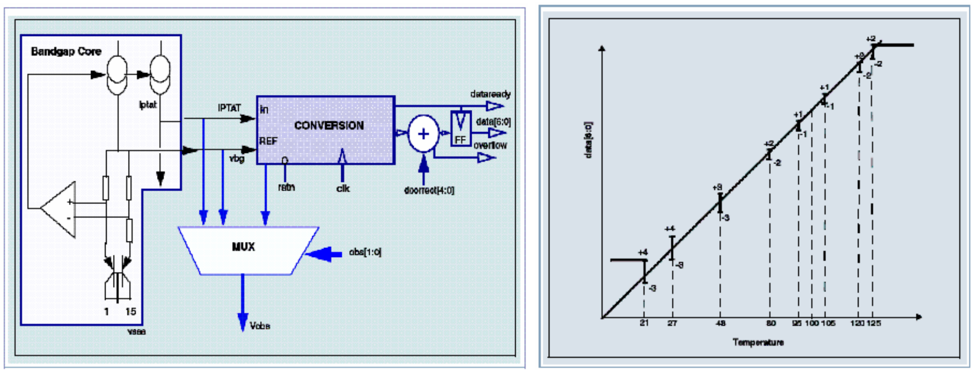

Figure 15: Thermal bandgap sensor design and associated response

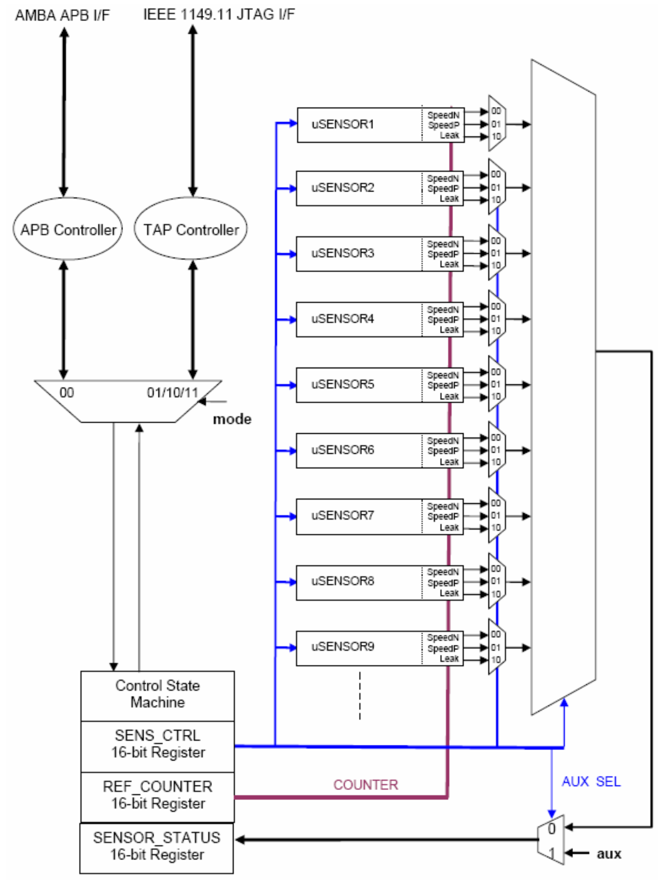

Figure 16: Leakage meter with system interfaces and muxing to allow measuring different process and Vth options (e.g. GO1, GO2, LVT,SVT,HVT)

For the sake of modeling the behavior of the leakage meters sensors, WP5 modeling tasks will resort to extrapolating an analytical model based on the 65 or 45nm measured figures as the one depicted in Table 19 and Figure 17.

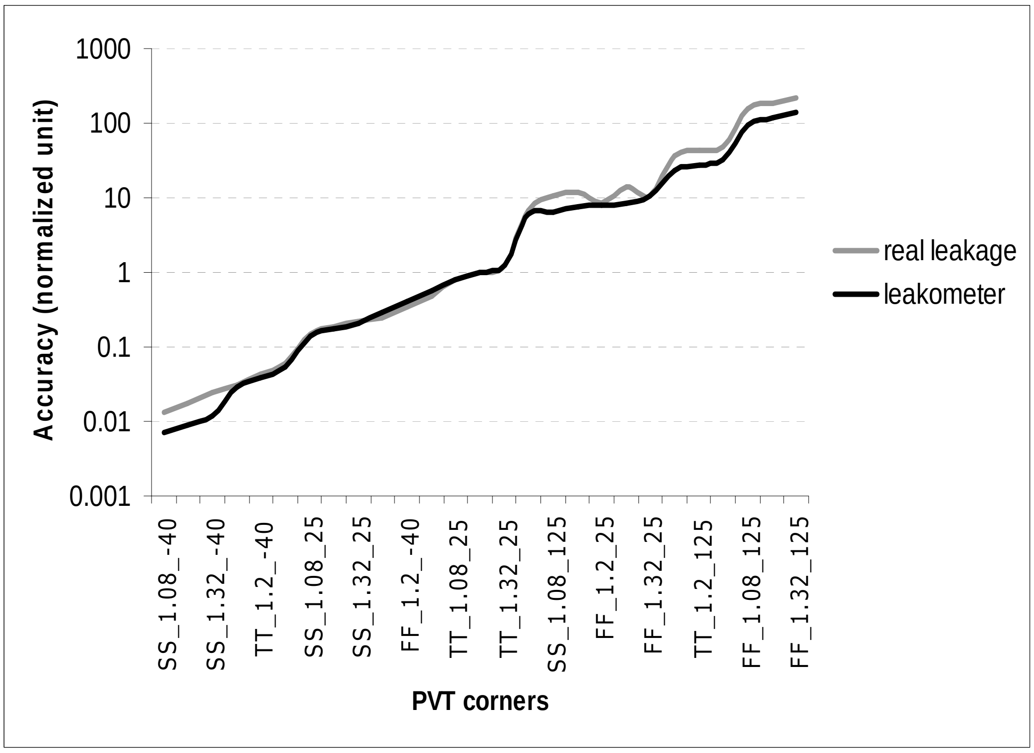

Figure 17: Measured vs. real leakage characteristic for the 65nm leakage meter cell

PVT corner	Speed
SS_1.2_25	48 kHz
TT_1.2_25	**276 kHz**
FF_1.2_25	**2.2 MHz**
SS_1.08_-40	1.9 kHz
FF_1.32_125	39 MHz

Table 19: Measured output frequencies for different corners for the 65nm leakage meter cell

9.3.3 <u>Workloads</u>

The policies studied by UNIBO in this project will be as much as possible generally applicable to a wide range of workloads. However, they will be primarily targeted to the optimization of the types of applications selected in this document. Depending on the task model, policies may be more or less complex.
For instance, an application made of independent tasks is less complex than a task graph with precedence constraints. Moreover, a key differentiator is if one or more application at a time will be considered. In particular, the following metrics are critical:

1. Independent or dependent tasks (task graph)
2. Dynamic or static (task always present or task activated by other tasks)
3. Task communication characteristics

4. Profiling information available about tasks (e.g. CPU load, memory and I/O accesses, etc)
5. Run to completion or streaming model
6. Metric: deadline miss rate, execution time or throughput
7. Mixed workload or not. If yes, we will have, for instance, a batch application such as email reading will be run at the same time as a game application.

Moreover, the number of tasks composing the benchmark as well as the total benchmark duration are critical characteristics. This is of particular importance when considering policies exploiting idleness for aging compensation or energy minimization. Indeed, the benchmark execution must be sufficient for the compensation to be possible.

For the sake of WP4 policy tuning, WP5 platform modeling and WP6 validation and benchmarking the following assumptions will be considered:

1) Mpeg2 video and audio decoding will be considered as Independent workloads, this is a simplification over a real-world scenario
2) A simple test for mixed workloads could be carried out considering both applications running concurrently
3) Only the static tasks dependencies graph associated to both applications will be considered, while dynamic task allocation and migration might still be considered if time and resources allows it.
4) Task communicated via shared memory, and exception could be considered for the distributed memory mpeg2 video decoder if this one will be deployed.
5) Run to completion will be the preferred model
6) Profiling information might be assumed to be available via simulation platform tracing and profiling support.
7) The number of tasks deployed for each benchmark depends on both the benchmark implementation, and the platform parametric definition, the simplifying assumption we make is that enough HW threads will be available at all times in the simulated platform to map the minimum number of logical threads required by the application. For the mpeg2 and ac3 decoders implemented in the REALITY WP6 tasks, this will be:
    a. Minimum of 4 threads for mpeg2 video decoder (shared memory)
    b. Minimum of 2 threads for mpeg 2 video decoder (message passing)
    c. Minimum of 6 threads for ac3 decoding

9.3.4 <u>Operating Points</u>

The policies will typically move the system into different operating points. Due to practical limitations of computing resources to apply the VAM flow to a non trivial RTL netlist, we need to select a reduced set of corners that will be applied to both the ARM9 characterization and also to extrapolate parameters for the simulation platform analytical models. Realistic corners must be selected for 32nm technology, we expect those to be not far away from the ones selected for 40nm process, the Corners at 40nm Appendix lists typical ones for STM 40nm design kits.

The following operating points will be characterized by VAM:

2 Temperatures:

Tmax 125C and Tmin -40C (typical for corner analysis).

2 Power supplies:

Vdd$_{min}$=0.9V and Vdd$_{max}$=1.1V.

2 Process:

worst and best.

This leads to the following corners:

ff32_1.10V_125C	best	1.1	125
ff32_1.10V_m40C	best	1.1	-40
ff32_0.90V_125C	best	0.9	125
ss32_1.10V_m40C	worst	1.1	-40
ss32_1.10V_125C	worst	1.1	125
ss32_0.90V_m40C	worst	0.9	-40

Table 20: Selected corners table for 32 nm

Below 0.9V there is hardly any yield for SRAMs unless a lot of redundancy is used.

A deeper analysis Min Vdd at which the SRAMs can still deliver a reasonable yield e.g., >90% could be required.

For these operating conditions, variability information will be provided by VAM for two different benchmarks. Other operating points can be extrapolated using approximate models as explained earlier.
Thanks to the information provided by WP2, two classes of policies can be developed. First, a class of static policies will be devoted at hiding as much as possible this performance, energy and power impact by playing with task allocation and scheduling strategies. This class of policies do not consider aging issues, and that is why it is called static.
A second class of policies will be developed that take aging into account. In this case, the scenario for variability/reliability compensation will be based on a what-if analysis case. Thus, assuming the variability information model coming from WP2 being applied to each of the SoC components and later evaluating the feasible range for compensation assuming a correlated drift of such variability data in the performance axis as function of lifetime. Since this drift will depend on voltage and temperature conditions, a subset of relevant operating points (i.e. temperature and voltages) will be selected as corner cases for this analysis. Other points can be extrapolated using approximated models as explained in this document in the section Power and Aging Modelling.

9.3.5 Corners at 40nm Appendix

Name	Process	Voltage	Temperature
ff40_1.05V_1.05V_105C	Best	1.05	105
ff40_1.05V_1.05V_125C	best	1.05	125
ff40_1.05V_1.05V_m40C	best	1.05	-40
ff40_1.05V_1.05V_m40C_2ey	best	1.05	-40
ff40_1.05V_1.10V_125C	best	1.05	125
ff40_1.05V_1.10V_m40C	best	1.05	-40
ff40_1.05V_1.10V_m40C_2ey	best	1.05	-40
ff40_1.05V_1.20V_105C	best	1.05	105
ff40_1.05V_1.20V_125C	best	1.05	125
ff40_1.05V_1.20V_m40C	best	1.05	-40
ff40_1.05V_1.20V_m40C_2ey	best	1.05	-40
ff40_1.05V_1.26V_105C	best	1.05	105
ff40_1.05V_1.26V_125C	best	1.05	125

ff40_1.05V_1.26V_m40C	best	1.05	-40
ff40_1.05V_1.26V_m40C_2ey	best	1.05	-40
ff40_1.05V_1.95V_105C	best	1.05	105
ff40_1.05V_1.95V_125C	best	1.05	125
ff40_1.05V_1.95V_m40C	best	1.05	-40
ff40_1.05V_1.95V_m40C_2ey	best	1.05	-40
ff40_1.05V_105C	best	1.05	105
ff40_1.05V_125C	best	1.05	125
ff40_1.05V_m40C	best	1.05	-40
ff40_1.05V_m40C_2ey	best	1.05	-40
ff40_1.10V_1.05V_125C	best	1.1	125
ff40_1.10V_1.05V_m40C	best	1.1	-40
ff40_1.10V_1.05V_m40C_2ey	best	1.1	-40
ff40_1.10V_1.10V_105C	best	1.1	105
ff40_1.10V_1.10V_125C	best	1.1	125
ff40_1.10V_1.10V_m40C	best	1.1	-40
ff40_1.10V_1.10V_m40C_2ey	best	1.1	-40
ff40_1.10V_1.95V_105C	best	1.1	105
ff40_1.10V_1.95V_125C	best	1.1	125
ff40_1.10V_1.95V_25C	best	1.1	25
ff40_1.10V_1.95V_m40C	best	1.1	-40
ff40_1.10V_1.95V_m40C_2ey	best	1.1	-40
ff40_1.10V_105C	best	1.1	105
ff40_1.10V_125C	best	1.1	125
ff40_1.10V_25C	best	1.1	25
ff40_1.10V_m40C	best	1.1	-40
ff40_1.10V_m40C_2ey	best	1.1	-40
ff40_1.20V_1.05V_105C	best	1.2	105
ff40_1.20V_1.05V_125C	best	1.2	125
ff40_1.20V_1.05V_m40C	best	1.2	-40
ff40_1.20V_1.05V_m40C_2ey	best	1.2	-40
ff40_1.20V_1.20V_105C	best	1.2	105
ff40_1.20V_1.20V_125C	best	1.2	125
ff40_1.20V_1.20V_m40C	best	1.2	-40
ff40_1.20V_1.20V_m40C_2ey	best	1.2	-40
ff40_1.20V_1.26V_105C	best	1.2	105
ff40_1.20V_1.26V_125C	best	1.2	125
ff40_1.20V_1.26V_m40C	best	1.2	-40
ff40_1.20V_1.26V_m40C_2ey	best	1.2	-40
ff40_1.20V_1.95V_105C	best	1.2	105
ff40_1.20V_1.95V_125C	best	1.2	125
ff40_1.20V_1.95V_m40C	best	1.2	-40
ff40_1.20V_1.95V_m40C_2ey	best	1.2	-40
ff40_1.20V_105C	best	1.2	105
ff40_1.20V_125C	best	1.2	125
ff40_1.20V_m40C	best	1.2	-40
ff40_1.20V_m40C_2ey	best	1.2	-40
ff40_1.26V_1.05V_105C	best	1.26	105
ff40_1.26V_1.05V_125C	best	1.26	125
ff40_1.26V_1.05V_m40C	best	1.26	-40
ff40_1.26V_1.05V_m40C_2ey	best	1.26	-40
ff40_1.26V_1.20V_105C	best	1.26	105
ff40_1.26V_1.20V_125C	best	1.26	125
ff40_1.26V_1.20V_m40C	best	1.26	-40
ff40_1.26V_1.20V_m40C_2ey	best	1.26	-40
ff40_1.26V_1.26V_105C	best	1.26	105
ff40_1.26V_1.26V_125C	best	1.26	125
ff40_1.26V_1.26V_m40C	best	1.26	-40
ff40_1.26V_1.26V_m40C_2ey	best	1.26	-40
ff40_1.26V_1.95V_105C	best	1.26	105
ff40_1.26V_1.95V_125C	best	1.26	125
ff40_1.26V_1.95V_m40C	best	1.26	-40
ff40_1.26V_1.95V_m40C_2ey	best	1.26	-40
ff40_1.26V_105C	best	1.26	105
ff40_1.26V_125C	best	1.26	125
ff40_1.26V_m40C	best	1.26	-40
ff40_1.26V_m40C_2ey	best	1.26	-40
ff40_1.95V_105C	best	1.95	105
ff40_1.95V_125C	best	1.95	125
ff40_1.95V_25C	best	1.95	25
ff40_1.95V_m40C	best	1.95	-40
ff40_1.95V_m40C_2ey	best	1.95	-40
ss40_0.90V_0.90V_125C	worst	0.9	125
ss40_0.90V_0.90V_125C_2ey	worst	0.9	125
ss40_0.90V_0.90V_m40C	worst	0.9	-40
ss40_0.90V_0.90V_m40C_2ey	worst	0.9	-40
ss40_0.90V_1.05V_125C	worst	0.9	125
ss40_0.90V_1.05V_125C_2ey	worst	0.9	125
ss40_0.90V_1.05V_m40C	worst	0.9	-40
ss40_0.90V_1.05V_m40C_2ey	worst	0.9	-40
ss40_0.90V_1.10V_125C	worst	0.9	125
ss40_0.90V_1.10V_125C_2ey	worst	0.9	125
ss40_0.90V_1.10V_m40C	worst	0.9	-40
ss40_0.90V_1.10V_m40C_2ey	worst	0.9	-40
ss40_0.90V_1.65V_105C	worst	0.9	105
ss40_0.90V_1.65V_105C_2ey	worst	0.9	105
ss40_0.90V_1.65V_125C	worst	0.9	125
ss40_0.90V_1.65V_125C_2ey	worst	0.9	125
ss40_0.90V_1.65V_m40C	worst	0.9	-40
ss40_0.90V_1.65V_m40C_2ey	worst	0.9	-40
ss40_0.90V_105C	worst	0.9	105
ss40_0.90V_105C_2ey	worst	0.9	105
ss40_0.90V_125C	worst	0.9	125
ss40_0.90V_125C_2ey	worst	0.9	125
ss40_0.90V_m40C	worst	0.9	-40
ss40_0.90V_m40C_2ey	worst	0.9	-40
ss40_0.95V_1.65V_105C	worst	0.95	105
ss40_0.95V_1.65V_105C_2ey	worst	0.95	105
ss40_0.95V_1.65V_125C	worst	0.95	125
ss40_0.95V_1.65V_125C_2ey	worst	0.95	125
ss40_0.95V_1.65V_m40C	worst	0.95	-40
ss40_0.95V_1.65V_m40C_2ey	worst	0.95	-40

ss40_0.95V_105C	worst	0.95	105
ss40_0.95V_105C_2ey	worst	0.95	105
ss40_0.95V_125C	worst	0.95	125
ss40_0.95V_125C_2ey	worst	0.95	125
ss40_0.95V_m40C	worst	0.95	-40
ss40_0.95V_m40C_2ey	worst	0.95	-40
ss40_1.05V_0.90V_125C	worst	1.05	125
ss40_1.05V_0.90V_125C_2ey	worst	1.05	125
ss40_1.05V_0.90V_m40C	worst	1.05	-40
ss40_1.05V_0.90V_m40C_2ey	worst	1.05	-40
ss40_1.05V_1.05V_125C	worst	1.05	125
ss40_1.05V_1.05V_125C_2ey	worst	1.05	125
ss40_1.05V_1.05V_m40C	worst	1.05	-40
ss40_1.05V_1.05V_m40C_2ey	worst	1.05	-40
ss40_1.05V_1.10V_125C	worst	1.05	125
ss40_1.05V_1.10V_125C_2ey	worst	1.05	125
ss40_1.05V_1.10V_m40C	worst	1.05	-40
ss40_1.05V_1.10V_m40C_2ey	worst	1.05	-40
ss40_1.05V_1.65V_105C	worst	1.05	105
ss40_1.05V_1.65V_105C_2ey	worst	1.05	105
ss40_1.05V_1.65V_125C	worst	1.05	125
ss40_1.05V_1.65V_125C_2ey	worst	1.05	125
ss40_1.05V_1.65V_m40C	worst	1.05	-40
ss40_1.05V_1.65V_m40C_2ey	worst	1.05	-40
ss40_1.05V_105C	worst	1.05	105
ss40_1.05V_105C_2ey	worst	1.05	105
ss40_1.05V_125C	worst	1.05	125
ss40_1.05V_125C_2ey	worst	1.05	125
ss40_1.05V_m40C	worst	1.05	-40
ss40_1.05V_m40C_2ey	worst	1.05	-40
ss40_1.10V_0.90V_125C	worst	1.1	125
ss40_1.10V_0.90V_125C_2ey	worst	1.1	125
ss40_1.10V_0.90V_m40C	worst	1.1	-40
ss40_1.10V_0.90V_m40C_2ey	worst	1.1	-40
ss40_1.10V_1.05V_125C	worst	1.1	125
ss40_1.10V_1.05V_125C_2ey	worst	1.1	125
ss40_1.10V_1.05V_m40C	worst	1.1	-40
ss40_1.10V_1.05V_m40C_2ey	worst	1.1	-40
ss40_1.10V_1.10V_125C	worst	1.1	125
ss40_1.10V_1.10V_125C_2ey	worst	1.1	125
ss40_1.10V_1.10V_m40C	worst	1.1	-40
ss40_1.10V_1.10V_m40C_2ey	worst	1.1	-40
ss40_1.10V_1.65V_105C	worst	1.1	105
ss40_1.10V_1.65V_105C_2ey	worst	1.1	105
ss40_1.10V_1.65V_125C	worst	1.1	125
ss40_1.10V_1.65V_125C_2ey	worst	1.1	125
ss40_1.10V_1.65V_m40C	worst	1.1	-40
ss40_1.10V_1.65V_m40C_2ey	worst	1.1	-40
ss40_1.10V_105C	worst	1.1	105
ss40_1.10V_105C_2ey	worst	1.1	105
ss40_1.10V_125C	worst	1.1	125
ss40_1.10V_125C_2ey	worst	1.1	125
ss40_1.10V_m40C	worst	1.1	-40
ss40_1.10V_m40C_2ey	worst	1.1	-40
ss40_1.65V_105C	worst	1.65	105
ss40_1.65V_105C_2ey	worst	1.65	105
ss40_1.65V_125C	worst	1.65	125
ss40_1.65V_125C_2ey	worst	1.65	125
tt40_0.90V_1.65V_125C	nom	0.9	125
tt40_0.90V_1.80V_125C	nom	0.9	125
tt40_0.90V_125C	nom	0.9	125
tt40_0.95V_0.95V_25C	nom	0.95	25
tt40_0.95V_1.10V_25C	nom	0.95	25
tt40_0.95V_1.15V_25C	nom	0.95	25
tt40_0.95V_1.65V_125C	nom	0.95	125
tt40_0.95V_1.80V_125C	nom	0.95	125
tt40_0.95V_1.80V_25C	nom	0.95	25
tt40_0.95V_125C	nom	0.95	125
tt40_0.95V_25C	nom	0.95	25
tt40_1.00V_1.80V_25C	nom	1	25
tt40_1.00V_25C	nom	1	25
tt40_1.05V_1.65V_125C	nom	1.05	125
tt40_1.05V_1.80V_125C	nom	1.05	125
tt40_1.05V_1.80V_m40C	nom	1.05	-40
tt40_1.05V_1.95V_m40C	nom	1.05	-40
tt40_1.05V_125C	nom	1.05	125
tt40_1.05V_m40C	nom	1.05	-40
tt40_1.10V_0.95V_25C	nom	1.1	25
tt40_1.10V_1.10V_25C	nom	1.1	25
tt40_1.10V_1.15V_25C	nom	1.1	25
tt40_1.10V_1.65V_125C	nom	1.1	125
tt40_1.10V_1.80V_105C	nom	1.1	105
tt40_1.10V_1.80V_125C	nom	1.1	125
tt40_1.10V_1.80V_25C	nom	1.1	25
tt40_1.10V_1.80V_m40C	nom	1.1	-40
tt40_1.10V_1.95V_m40C	nom	1.1	-40
tt40_1.10V_105C	nom	1.1	105
tt40_1.10V_125C	nom	1.1	125
tt40_1.10V_25C	nom	1.1	25
tt40_1.10V_m40C	nom	1.1	-40
tt40_1.15V_0.95V_25C	nom	1.15	25
tt40_1.15V_1.10V_25C	nom	1.15	25
tt40_1.15V_1.15V_25C	nom	1.15	25
tt40_1.15V_1.80V_25C	nom	1.15	25
tt40_1.15V_25C	nom	1.15	25
tt40_1.20V_1.80V_m40C	nom	1.2	-40
tt40_1.20V_1.95V_m40C	nom	1.2	-40
tt40_1.20V_m40C	nom	1.2	-40
tt40_1.26V_1.80V_m40C	nom	1.26	-40
tt40_1.26V_1.95V_m40C	nom	1.26	-40
tt40_1.26V_m40C	nom	1.26	-40
tt40_1.65V_125C	nom	1.65	125

tt40_1.80V_105C	nom	1.8	105	tt40_1.95V_m40C	nom	1.95	-40
tt40_1.80V_25C	nom	1.8	25				

10 IP blocks Validation

Once the IP blocks integration in the ARM926 platform has been validated an important part of the work will be to assess the interest of the IP blocks. On the RTL platform, the work is not straight forward since we need to access dynamic information on static characterization.

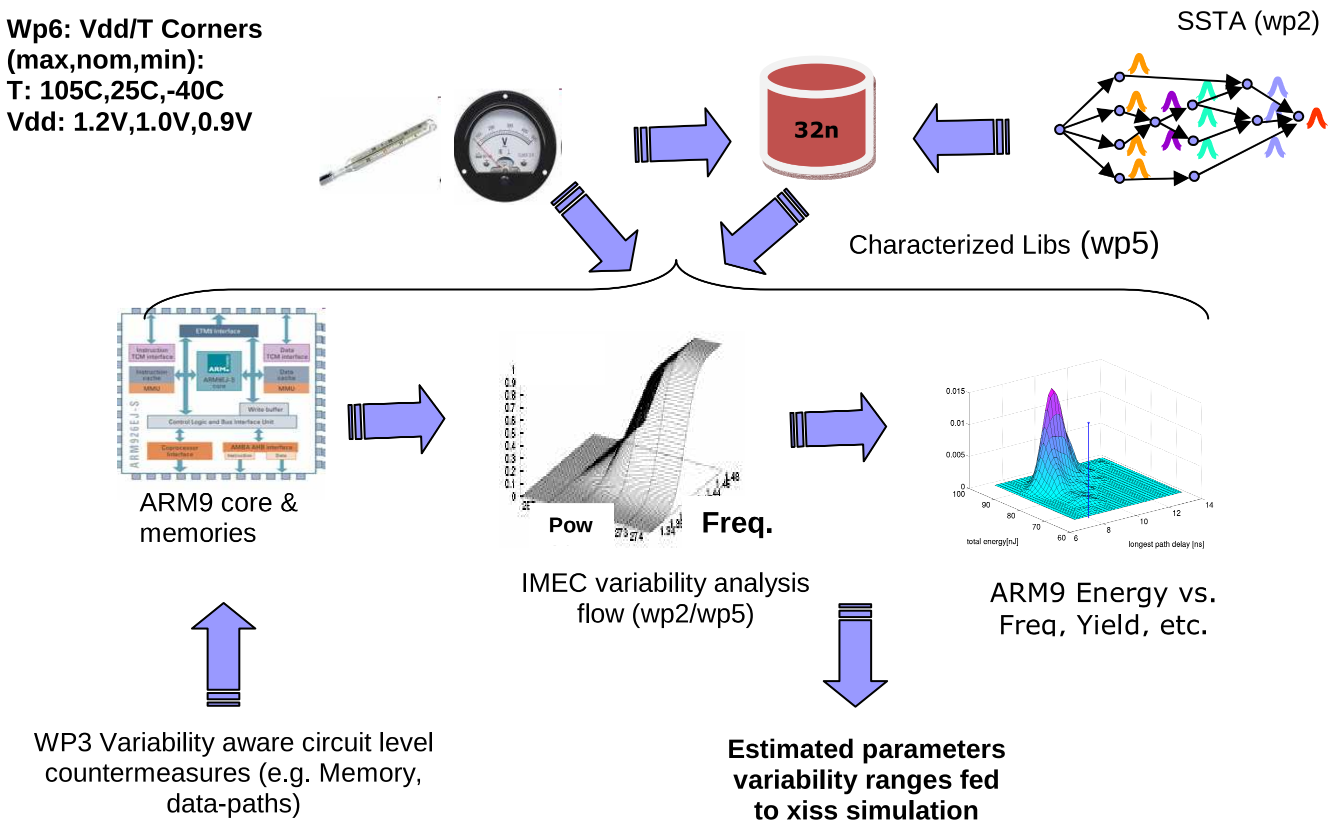
Wp6: Vdd/T Corners
(max,nom,min):
T: 105C,25C,-40C
Vdd: 1.2V,1.0V,0.9V
SSTA (wp2)
32n
Characterized Libs (wp5)
ARM9 core & memories
Pow
Freq.
IMEC variability analysis flow (wp2/wp5)
ARM9 Energy vs. Freq, Yield, etc.
total energy[nJ]
longest path delay [ns]
WP3 Variability aware circuit level countermeasures (e.g. Memory, data-paths)
Estimated parameters variability ranges fed to xiss simulation

Figure 18: Validation flow for assessing impact of variability analysis and countermeasures to the ARM9 IP block

The time dependant variability is changing the behavior of the system with the time. The countermeasures developed in WP3 allow the system to change states depending on the information they receive through the knobs. As a consequence the fundamental concepts of the work performed in WP3 are to transform the system in a reconfigurable one which continuously adapts to the current variability state of the system.

To access the relevance of the countermeasures several states of the circuits have to be evaluated. The changes in the distribution of the electrical behaviors of the system are assessed. For each state of the system, the statistical distribution of the electrical behavior is characterized and evaluated. We will therefore see how the delay, leakage and dynamic power distributions are affected by the countermeasures and how the countermeasures will put the system in the most interesting state to reach the system specifications.

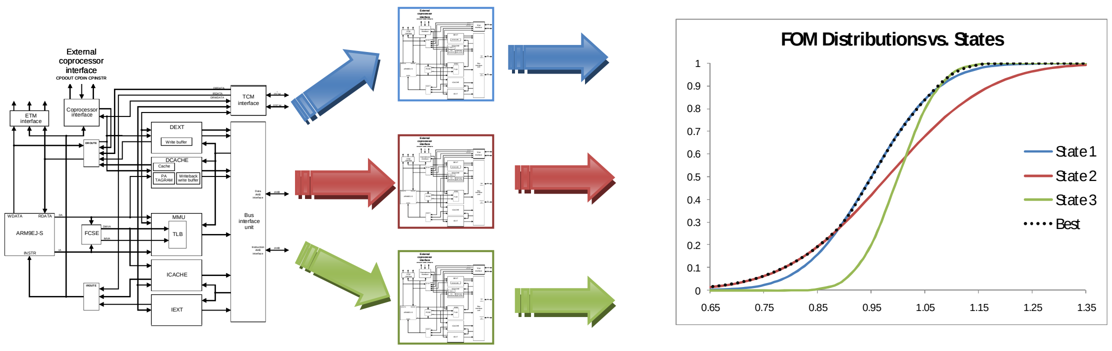

Figure 19: The incorporation of WP3 outcome leads to better performances

11 System Level Validation

The system level platform defined earlier will be used to assess the impact, in terms of high level trends, of the SW countermeasures and control policies developed in WP4, as well as to provide a qualitative indication of potential benefits of deploying certain kinds of HW knobs and monitoring circuits with respect to an uncompensated vanilla system.

The platform is defined so as to be capable of sustaining the requirements of the selected REALITY application scenarios as defined in this document in 9.2 and in [D6.1].
The ISS simulator measuring plug-in whose metrics are defined in 9.1.2 and implementing the models described in 9.1.3 , will be tuned with all of the parameters defined in Table 1 whose variability ranges will be derived from the ARM9 variability analysis flow described in Figure 18 above.
The parameters are grouped in several classes; some relates to the simplified modeling of the physical property of the 32nm process technology, an example of those parameters are given in Table 2. In practice to run the complete system level variability impact experiments, both the nominal values and their range of variability need to be computed.
The nominal values of the parameters for all the proportionality constants defined in Table 1, and for example:

Kds	float	Proportionality constant for dynamic consumption in stall state.
Kss	float	Proportionality constant for static consumption in stall state.
Kdi	float	Proportionality constant for dynamic consumption in idle state.
Ksi	float	Proportionality constant for static consumption in idle state.
Kdpg	float	Proportionality constant for dynamic consumption in power gating.
Kspg	float	Proportionality constant for static consumption in power gating.

Will be derived from a gate level back-annotated simulation of the RTL IPs being part of the REALITY system platform demonstrator as illustrated in the following Figure:

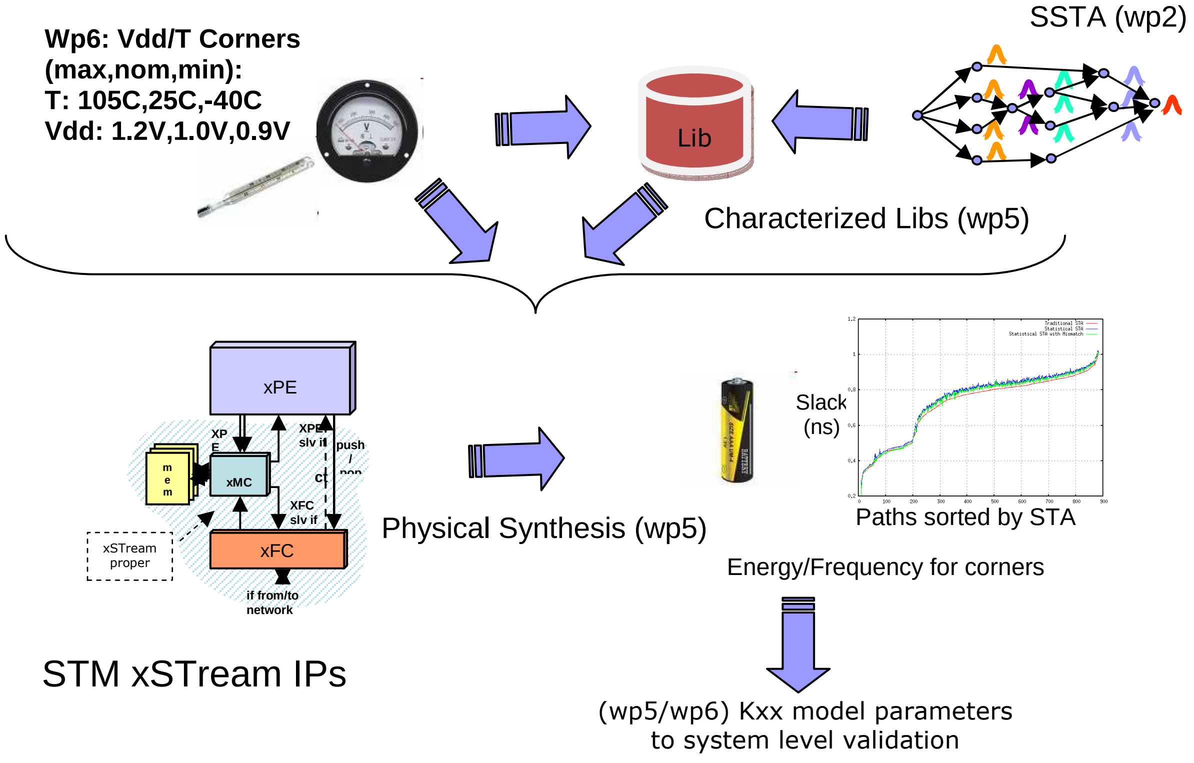

Figure 21: Extraction flow for estimating nominal values for Kxx parameters

The parameters will only be estimated for the xPE cores and not for the whole system platform; however for memories the IP block flow above will be used as reference as the same memory blocks have been defined for both the ARM9 and xPE cores.
A similar flow could be put in place for the interconnect if time and allotted resources will allow it.
Additionally, in order to fit the model to multiple corners, extraction of optimal parameters for analytical variability model by fitting of VAM power/T histograms obtained from ARM9 characterization for different corners and switching activity levels could be performed.

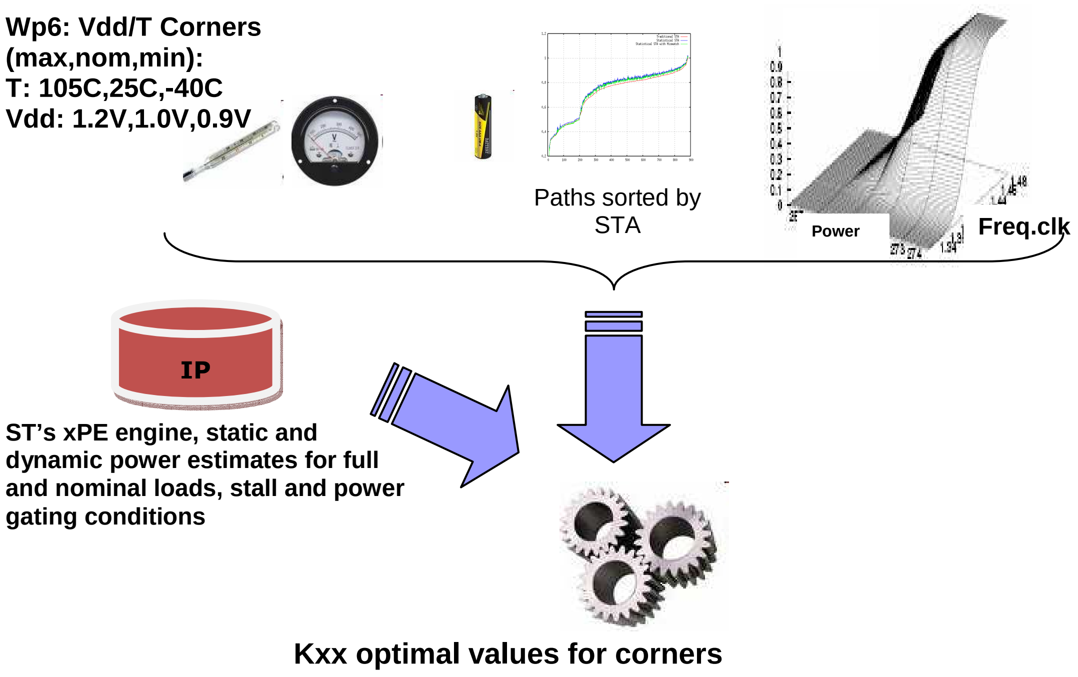
Wp6: Vdd/T Corners (max,nom,min):
T: 105C,25C,-40C
Vdd: 1.2V,1.0V,0.9V
Paths sorted by STA
Power
Freq.clk
IP
ST's xPE engine, static and dynamic power estimates for full and nominal loads, stall and power gating conditions
Kxx optimal values for corners

Figure 22: Nominal values for Kxx parameters estimated for multiple corners

1.1. System level simulation with variability injection

The final set of system level experiments will be carried out by integrating the simulation platform with all of the models and SW/HW countermeasures so far described as illustrated by Figure 23. The process involves a relatively large number of simulation of a given set of application workloads (multimedia apps) as defined in 9.2, where both the uncompensated (no SW/HW countermeasures) and the compensated test cases are executed.
The Kxx parameters variability ranges will be used to randomly generate (ala Montecarlo) a perturbation of the Kxx nominal corner baseline values estimated from the flows described above. The ISS power and variability modeling will provide the required run-time simulation of the HW measuring knobs described in 9.3.2 (e.g. temperature, frequency, leakage sensors) that will be exploited by the SW control policies developed in WP4 ([D4.2]), and adapt the run-time behavior to achieve better results in terms of objective function (e.g. power/energy consumption or aging tolerance), while simultaneously improving the reduction of estimated functional yield losses (see definition in 9.2).

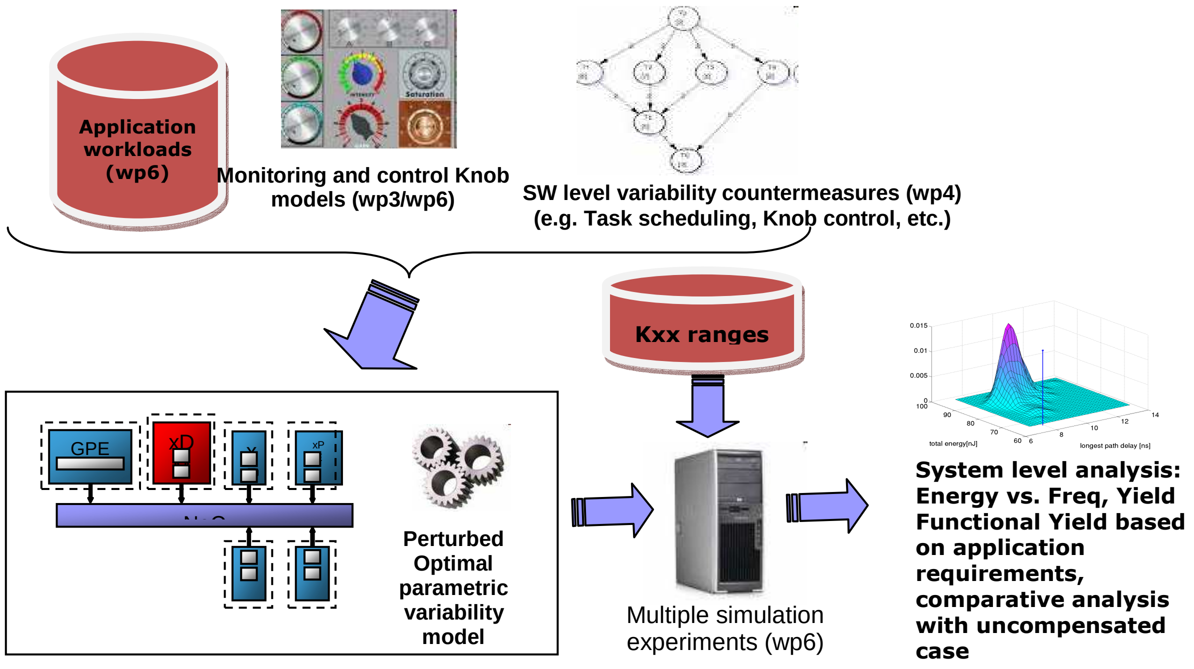
Application workloads (wp6)
Monitoring and control Knob models (wp3/wp6)
SW level variability countermeasures (wp4) (e.g. Task scheduling, Knob control, etc.)
Kxx ranges
GPE
xD
xP
Perturbed Optimal parametric variability model
Multiple simulation experiments (wp6)
System level analysis: Energy vs. Freq, Yield Functional Yield based on application requirements, comparative analysis with uncompensated case
total energy[nJ]
longest path delay [ns]

Figure 23: Final system level platform assessment of impact and variability analysis flow

For running multiple simulations the following steps will be followed:

1. identification of configuration parameters that are common to all the executions but whose values are different from the default ones
2. creation of a configuration file containing all those parameters
3. identification of configuration parameters that are different at each execution
4. regarding those parameters, identify how they will change at each execution
5. those parameters will be provided to the ISS in an additional configuration file
6. build a script launching the ISS a number of times equal to the needed number of executions, and each time provide the ISS with the configuration file containing the parameters common to all the executions (point 2) and a set of values for the parameters changing at each execution (point 5)
7. after each execution the ISS will produce a statistics file, a profiler output file for each HT of each core and a file for each core dumped by the device monitoring core activity, frequency, temperature, aging and variability

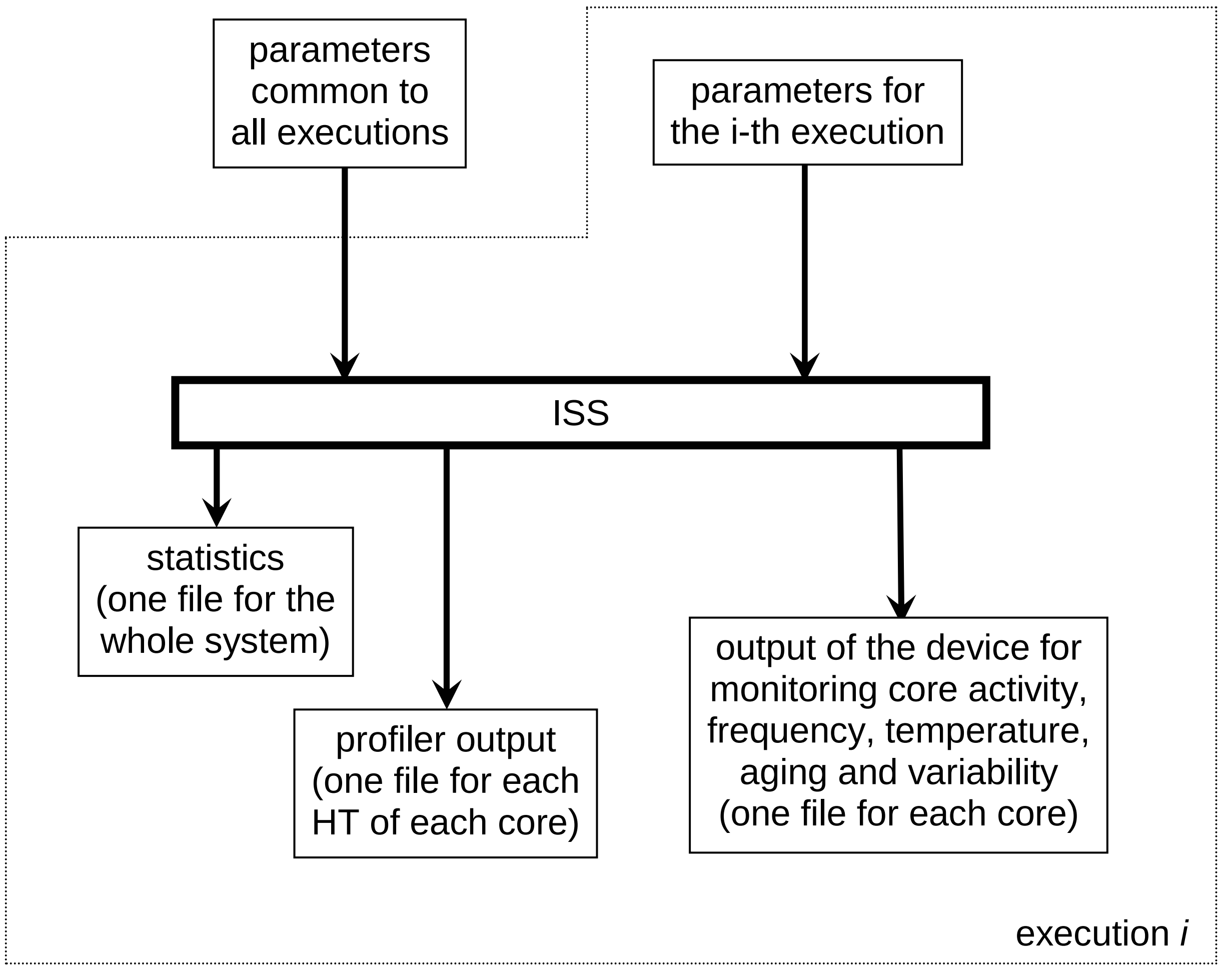
parameters common to all executions
parameters for the i-th execution
ISS
statistics (one file for the whole system)
profiler output (one file for each HT of each core)
output of the device for monitoring core activity, frequency, temperature, aging and variability (one file for each core)
execution *i*

Figure 24: One iteration with a set of variability perturbations injected in the simulated models

In order to determine the set of parameters that will change at each execution (point 3 above), the clock frequencies are quite important. As mentioned above (9.1.1), the ISS models support a main clock frequency, and the frequencies of all the cores and other models (i.e. memory modules and interconnect) are derived from that frequency dividing it by an integer ratio. If the granularity of clock scaling of cores must be fine, the main clock frequency must be higher than the one of the cores, and their initial ratio must be computed accordingly.
I.e., if the nominal frequency of all cores is 600 MHz, we can define a main clock frequency of 600 MHz and an (initial) clock ratio of 1 for all cores, but this allows us to scale the frequency only to 300 MHz (ratio=2), 200 MHz (ratio=3) and so on. If we define a main clock frequency of 6 GHz and an (initial) clock ratio of 10 for all cores, in this case we can also scale to 545 MHz (ratio=11), to 500 MHz (ratio=12), and so on.
Then, in order to have a fine tuning of core clock frequencies, we plan to use a main clock frequency that is at least 10 times higher than the nominal core clock frequencies.
So, in the following we will mention ‘main clock frequency’ and ’clock ratio’ and provide values just for illustration purposes because, in order to fine tune core clock frequencies, those values will be multiplied by 10 or more.

With these assumptions, the parameters that we will change at each execution (point 3 above) will be:

1. main clock frequency (the one originating all the frequencies of the system components): 60 (FPGA) and 600 MHz

2. clock frequency ratios for the cores:
    a. 1 for all cores
    b. 1 for the GPE and 2 for the xPEs
    c. 2 for the GPE and 1 for the xPEs
3. clock frequency ratios for the interconnect:
    a. 1, 2, 3
4. clock frequency ratios for the memory modules:
    a. 1x, 2x, 3x the xPE frequency ratios
    b. 1, 2, 3 for the main memory
5. interconnect bus width:
    a. 64-bit or 128-bit
6. interconnect latencies:
    a. 1, 2, 3 (interconnect clock) cycles for propagating a (word of) request
    b. 1, 2, 3 (interconnect clock) cycles for propagating a (word of) response
7. memory latencies:
    a. 1, 2, 3, 4, 5 (memory clock) cycles for the first (or only) word in a (read/write) access for the xPE local memories
    b. 1, 2, 3, 4, 5 (memory clock) cycles for the following words in a (read/write) burst access for the xPE local memories
    c. 1, 2, 3, 4, 5 (memory clock) cycles for the first (or only) word in a (read/write) access for the main memory
    d. 1, 2, 3, 4, 5 (memory clock) cycles for the following words in a (read/write) burst access for the main memory
8. variability modeling plug-in parameters:
    a. execution with (compensated case) and without (uncompensated case) activating the plug-in model
    b. Kss: 18%, 20%, 22% of PstaA
    c. Ksi: 8%, 10%, 12% of PstaA
    d. T: 280K, 300K
    e. Guardband: 10%, 15%

For limiting the total number of possible combinations, the following conditions will be applied:

1. all xPEs should have the same clock frequency
2. all xPE local memories should have the same clock frequency and latencies

1.2. Sanity check

In order to verify that the results obtained with the various simulation experiments have a good chance of being representative of the real working conditions, we will validate at least one or two configurations with the aid of an FPGA mapped prototype of the platform, at least one of the simulations will be matched against the hardware emulator and the discrepancy in terms of cycles to completion of the benchmark will be measured. The Figure 25 shows a block level schematics for the FPGA emulation platform replicating an instance of the REALITY system level platform with two bus nodes and up to 8 xPE processing cores.

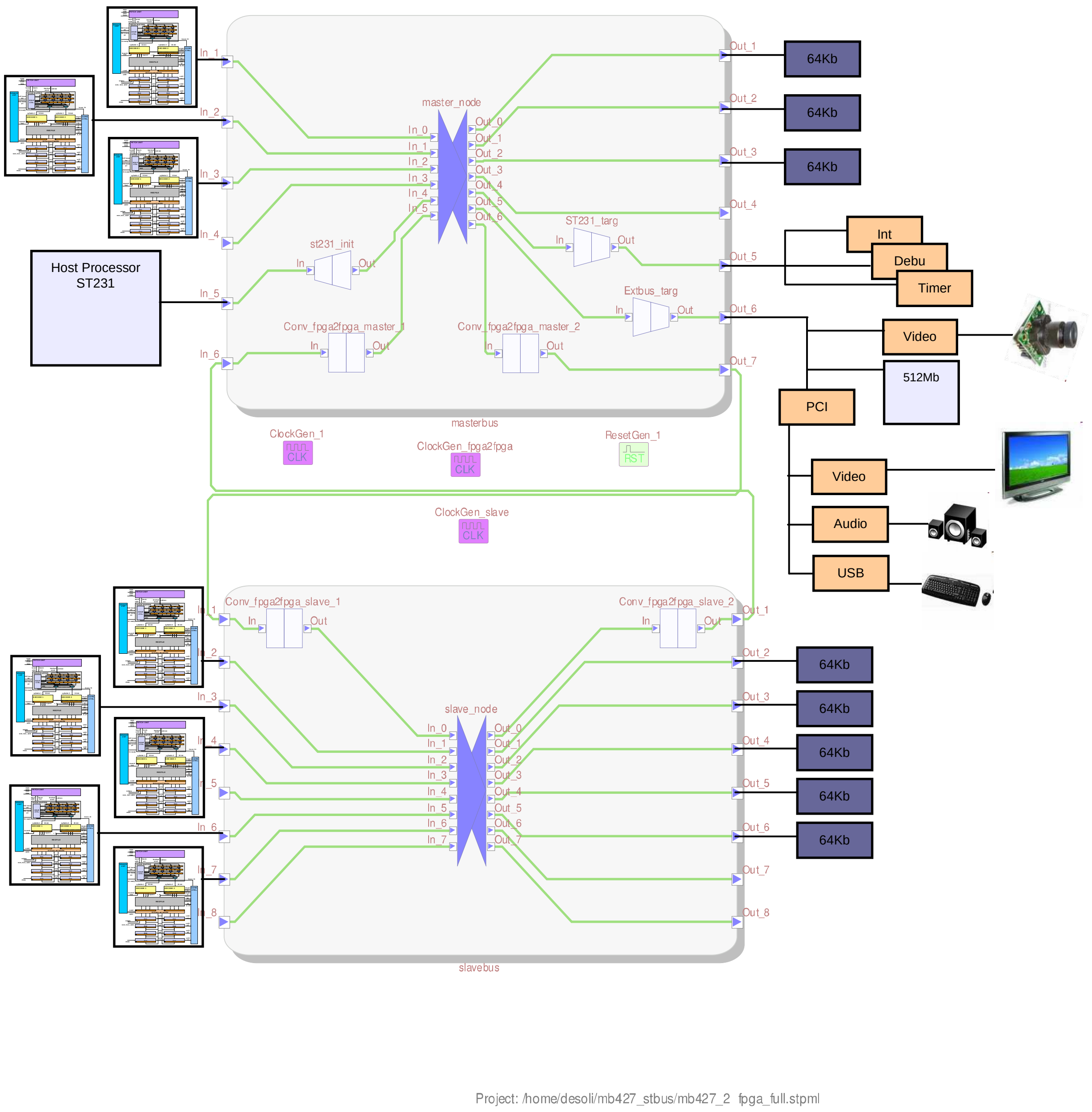

Figure 25: REALITY FPGA HW emulation platform schematics, will be used for simulation platform calibration

1.3. Analysis of results

The execution of the benchmarks on the simulation platform will give us important information (related to the particular conditions and ISS parameter values of each execution) about (see also 9.1.2):

1. the number of clock cycles consumed for executing the benchmarks at each execution
2. an estimation of the total energy consumed by the whole system, and the energy consumed by each core
3. the estimated lifetime of the SoC at each execution

The results at point 1 will be used (in the compensated and uncompensated cases) in comparison against the results at points 2 and 3 to demonstrate the effectiveness of our techniques. In particular, results 1-2 will characterize energy consumption against execution time. Results 1-3 will characterize reliability and variability against execution time.
The results at point 3 affect reliability and variability. We will estimate:

1. how much the SoC lifetime has been extended reducing the average clock frequency with how much increase in terms of execution time
2. how much the yield has been increased reducing the average clock frequency with how much increase in terms of execution time

12 References

[Hin08] Hinkelmann, Klaus and Kempthorne, Oscar (2008). Design and Analysis of Experiments. I and II (Second ed.). Wiley. ISBN 978-0-470-38551-7.

[Pla83] Plackett, R.L. (1983). "Karl Pearson and the Chi-Squared Test". International Statistical Review 51 (1): 59–72.

[Die07] B.Dierickx, et. al “Propagating Variability from Technology to System Level", Intl. Wrkshp. on The Physics of Semiconductors, Bombay, 2007

[Abh08] Abhishek Tiwari, Josep Torrellas, "Facelift: Hiding and slowing down aging in multicores," micro, pp.129-140, 2008 41st IEEE/ACM International Symposium on Microarchitecture, 2008

[D4.2] “Control algorithms for system level reliability and variability management”

[D6.1] IST-216537-WP6-D6.1.doc “WP6 Application demonstrators”

13 List of Abbreviations

REALITY	Reliable and Variability tolerant System-on-a-chip Design in More-Moore Technologies
CAD	computer aided design
ISS	Instruction Set Simulator
xPE	xSTream Processing Engine
QoS	quality of service
SoC	system on chip
SOHO	small office/home environment
SW	software