FP7 – 216693 - MULTICUBE Project

MULTI-OBJECTIVE DESIGN SPACE EXPLORATION OF MULTI-PROCESSOR SOC ARCHITECTURES FOR EMBEDDED MULTIMEDIA APPLICATIONS

Deliverable D2.3.1: Initial Multi-Objective Evaluation Metrics

Revision [2]

Delivery due date: M18 (June 2009)
Actual submission date: July 24th, 2009
Lead beneficiary: POLIMI

Dissemination Level of Deliverable		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	
Nature of Deliverable		
R	Report	X
P	Prototype	
D	Demonstrator	
O	Other	

Author(s):	*Cristina Silvano, Vittorio Zaccaria, Gianluca Palermo* (POLIMI)		
Reviewer(s):	*Carlos Kavka, Enrico Rigoni (ESTECO), Giovanni Mariani* (ALaRI) *and Prabhat Avasare (IMEC)*		
WP/Task No:	2.3	**Number of pages:**	24
Identifier:	D2.3.1_POLIMI_06.2009	**Dissemination level:**	Public
Issue Date:	June 2009		

Keywords: Analytical Modeling, Response Surface Modeling, Exploration

Abstract: This deliverable (issued at M18) presents the results of the activity carried on from M7 to M18 in Task 2.3 (High-level evaluation metrics) under the leadership of POLIMI, with contributions from POLIMI and reviewed by ESTECO, ALaRI and IMEC.

Based on the metrics defined in Task 1.2, the main goal of this deliverable is to present the analytical techniques (also called Response Surface Methods, RSMs) adopted to model the system metrics in the design space without requiring the simulation of all the possible architectural configurations.

The analysis presented in this deliverable will be extended in the next months to produce the final deliverable for the Task 2.3 (D2.3.2 – Refined and extended multi-objective evaluation metrics).

This report contains:

- A summary of the motivations to use analytical techniques when facing the problem of design space exploration;
- The description of the analytical techniques analyzed in Task 2.3 and implemented up to M18 with some validation results;
- A summary of implementation details and preliminary validation results regarding the software development of the analytical techniques analyzed in Task 2.3.

The Project Coordinator

Approved by the Project Coordinator: 	**Date: July 24th, 2009**

Table of Contents

I. Executive summary

This deliverable presents the use of some analytical techniques to support the design space exploration. In particular, the deliverable outlines the need of an alternative way to evaluate system metrics instead of simulations, presenting both interpolative and regressive techniques.

The analytical techniques considered for supporting the design space exploration and described in this deliverable are the following:

- **Shepard Interpolation**
  It is an interpolative technique that evaluates the response function in unknown points as the sum of the value of the response function in known points weighted with the inverse of the distance.

- **Radial Basis Functions**
  It is an interpolative technique that evaluates the response function in unknown points by the sum of a set of radial functions each one centered in known points.

- **Linear Regressions**
  It is a regressive technique that evaluates the response function in unknown points by modeling a linear relationship between the dependent response function and the independent design variables (or combinations of them)

- **Artificial Neural Networks**
  It is a non-linear regressive technique that evaluates the response function in unknown points by modeling the system with a set on interconnected processing elements (neurons), inspired by the way biological nervous systems work.

Currently, the analytical techniques are implemented as software modules but they are still not integrated within current prototype version of the open-source M3Explorer exploration framework. The analysis presented in this deliverable will be extended to produce the final deliverable for Task 2.3 (D2.3.2), where the integration and usage into the M3Explorer exploration framework will also be described.

This report is structured as follows. Section II outlines why there is the need of analytical techniques in the design space exploration problems faced by MULTICUBE and some problems related to the analytical modeling. Section III to VI present the interpolative and regressive techniques developed up to M18, while Section VII presents some validation results. Finally, Section VIII outlines the interfaces defined for the development of the analytical models in software modules.

II. Analytical techniques for fast evaluation of the system metrics

Nowadays, Multi-Processor Systems-on-Chip (MPSoCs) and Chip-Multi-Processors (CMPs) [1] represent the de facto standard for both embedded and general-purpose architectures. In particular, customizable MPSoCs supported by parallel programming have become the dominant computing paradigm for application-specific processors. In fact, they represent the best compromise in terms of a stable hardware platform that is software programmable, thus customizable, upgradable and extensible. In this sense, the MPSoC paradigm minimizes the risk of missing the time-to-market deadline while ensuring greater efficiency due to architecture customization and software compilation techniques.

In this context where the Design Space Exploration covers a central role, it is important to underline the problem of the efficiency of this design phase.

In fact, in computer architecture research and development, simulation still represents the main tool to predict performance of alternative architectural design points. If we consider cycle-accurate system-level simulation, it requires a lot of simulation time and the exploration of the design alternatives can exceed practical limits. Furthermore, the growing trend towards chip multi-processor architectures amplifies this problem because the simulation speed linearly decreases by increasing the number of cores to be simulated (as shown in Figure 1).

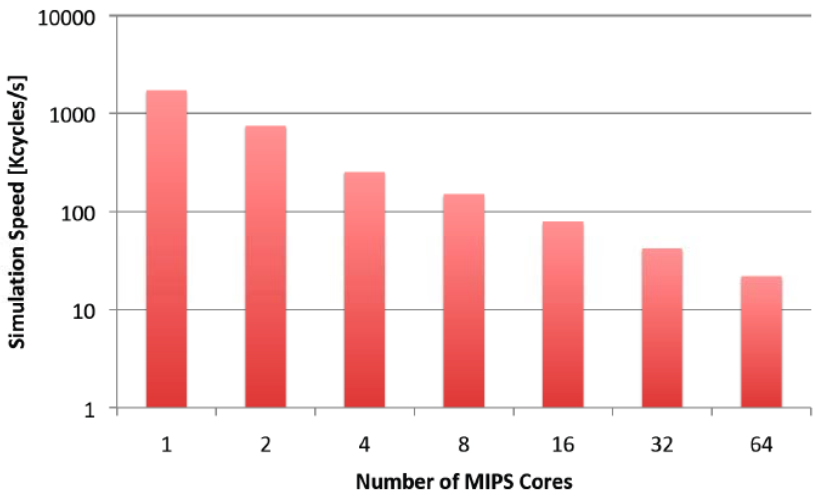

Figure 1 - Simulation speed of the SESC multiprocessor simulator [12] when executing the FFT kernel by varying the number of cores from 1 to 64 (Host machine: two Intel Xeons quad-core at 3GHz).

While from the simulation point of view, the use of statistical sampling techniques seems to represent the best solution (e.g. SIMPOINT [2]), from the design space exploration point of view efficient solutions that reduce the number of architectural alternatives to be analyzed are only recently under analysis.

To face the problem of an efficient design space exploration within MULTICUBE, we decided to try to adopt techniques coming from the fields of statistic and machine learning, to model in an analytical way (Response Surface Methods) the behavior of the system metrics into design space.

II.1. Advantages of analytical techniques

Response Surface Modeling (RSM) techniques allow determining an analytical dependence between several design parameters and one or more response variables (what we called system metrics). The working principle of RSMs is to use a set of simulations either generated by a Design of Experiment (DoE) [3] or obtained during an exploration strategy.

A typical RSM flow involves two main phases: a training phase and a prediction phase. While in the training phase, known data (also known as training set) is used to identify the RSM configuration, in the prediction phase, the RSM is used to forecast unknown system response. RSMs are an effective tool for analytically predicting the behavior of the system platform without resorting to a system simulation.

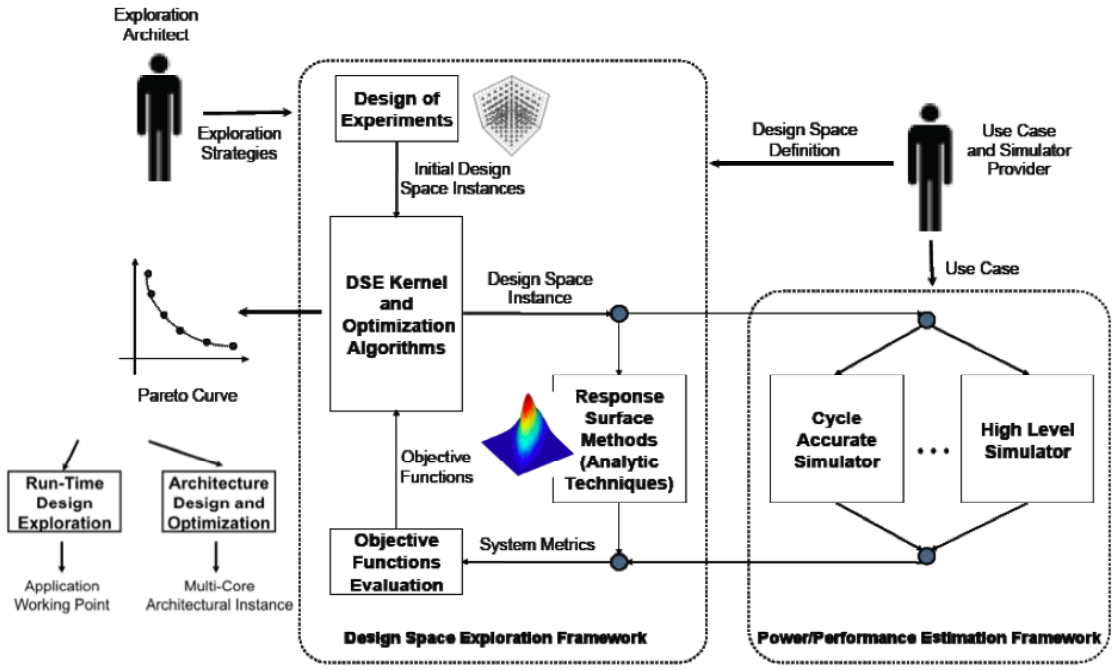

Figure 2 - Detailed view of the MULTICUBE design flow

Figure 2 shows the MULTICUBE design flow (with details regarding the design space exploration framework) where it is easy to identify the analytical techniques as an alternative to the actual simulations. In this flow, the exploration strategy can selectively use simulation-based evaluation or analytical estimation.

II.2. Considerations on the analytical techniques

The main advantage obtained by using analytical techniques instead of actual simulation is to speed up the design space exploration process. In fact, it is possible to predict the values of the system metrics without requiring simulations, or more precisely, requiring a small number of simulations to be used in the training phase.

On the other hand, the main problem when using analytical techniques is related to the accuracy of the prediction. In fact, if the system behavior predicted by the analytical techniques is not enough accurate, all the exploration and analysis done by using RSM are less effective. By using analytical techniques, we have to take care of the fact that, during the prediction phase, it is not possible to understand the accuracy of the predictions and so how effective are the analysis that we are done based on that predictions.

II.2.1. Design of Experiments

The training data distribution is one of the most important problems for the prediction of an analytical model especially when the number of training data is limited.

The term Design of Experiments (DoE) [4] is used to identify the planning of an information-gathering experimentation campaign where a set of variable parameters can be tuned. Design of experiments is a discipline that has very broad application across natural and social sciences and encompasses a set of techniques whose main goal is the screening and analysis of the system behavior with a small number of simulations. Each DoE plan differs in terms of the layout of the selected design points in the design space. Several design of experiments have been proposed in the literature so far. Here in the following we will cite only three among the most used, showing different characteristics:

- *Random*. In this case, design space configurations are picked up randomly by following a Probability Density Function (PDF).

- *Full factorial*. In statistics, a factorial experiment is an experiment whose design consists of two or more parameters, each with discrete possible values or ”levels”, and whose experimental units take on all possible combinations of these levels across all such parameters. Such an experiment allows studying the effects of each parameter on the response variable, as well as the effects of interactions between parameters on the response variable. The most important full-factorial DoE is called *2-level full factorial*, where the only levels considered are the minimum and maximum for each parameter.

- *Central composite design*. A Central Composite Design is an experimental design specifically targeted to the construction of response surfaces of the second order (quadratic) without requiring a three-level factorial.

It is important to note that, once the design space has been defined, while factorial and central composite DoE layouts require a fixed number of points, for the Random DoE the number of points is one of the parameter of the DoE.

II.2.2. Over-fitting

One of the most critical issues in developing analytical models is the generalization of the models, and in particular, for models trained by a set of known data, the problem is how well the model will be able to make predictions for cases that are not in the initial training set?

Especially regressive methods and artificial neural networks can suffer of the problem defined as “over-fitting”. In fact, over-fitting generally occurs when the model is excessively complex in relation to the amount of data available. A model, which has been over-fit, will generally

have poor predictive performance, as it can exaggerate minor fluctuations in the training data. Figure 3 shows a typical example of the over-fitting risk where the model learned by using the training data instead of the real system behaviour. The left part of Figure 3 shows the training phase while the right part shows the prediction phase.

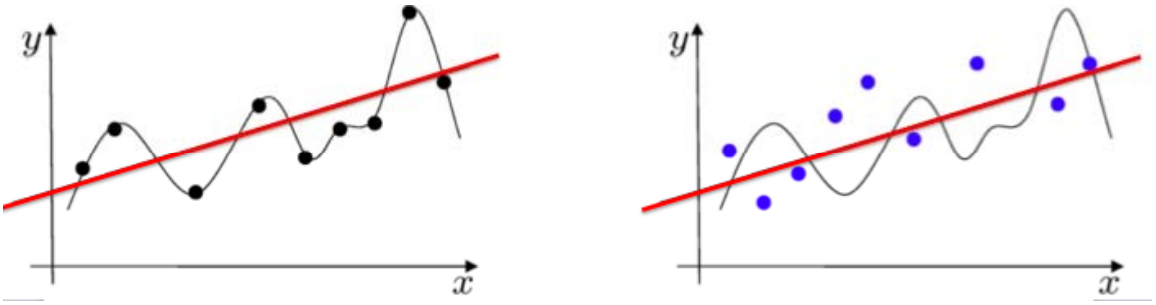

Figure 3 - Simple example to show the over-fitting problem

In the left part of Figure 3, some training data (black dots) have been generated by adding a random noise to a linear dependence between the design variable **x** and the response variable **y** (shown by the red line). By using those training data, a complex model has been trained obtaining a predicted relationship between the design variable and the response variable caracterized by the black line. In the right figure, other data (blue dots) have been produced with the same technique than before. It is easy to see that the prediction capability of the model obtained during the training phase is very low since the model has been over-fitted the training data.

There are several methods to avoid the overfitting risk, but two are the most used techniques that are also adopted in the development of the analytical models used in MULTICUBE: the simplest one suggests using simple models in order to reduce the over-fitting risk, while the second one is called early stopping criterion. The early stopping criterion suggests to split the training data into two sets: a training set and a validation set. The train phase uses only the training set while the validation set is used to verify the prediction capability of the trained model. As the validation set is independent of the training set, the prediction capability done on the validation set is a good measure of generalization for the model.

II.2.3. Pre-processing and scaling data

Data used to derive analytical models, also if originated from the same source/modeled architecture due to different positions in the design space, can have different values distribution and order of magnitudes. Especially when the case is the latter, to create better prediction it is better to *preprocess* and/or *scale* the data.

Data transformation is very important because of in most of the cases the analytical models used to predict the data would prefer when data distribution follows some rules. As an example, if we consider an analytical model that uses the standard deviation of the training data to predict unknown data, this standard deviation values can be very high if the data distribution is skewed. In this case, it is highly recommended to first transform the data to approach a better symmetry and then to perform the model training and the related prediction.

Box-Cox power transformation. A powerful transformation adopted in the above-mentioned cases is called Box-Cox power transformation [5]. The Box-Cox power transformation is a useful data pre-processing technique used to reduce data variation, make the data more normal distribution-like and improve the correlation between variables. The power transformation is defined as a continuously varying function, with respect to the power parameter λ:

$$y_k^{(\lambda)} = \begin{cases} (y_k^{\lambda} - 1)/\lambda, & \text{if } \lambda \neq 0 \\ \log y_k, & \text{if } \lambda = 0 \end{cases}$$

In the validation results of the models that we adopted in MULTICUBE, we considered a family of transformations as potential candidates λ {1, 0.5, 0,−1}. All the Box-Cox power transformations are only defined with positive values. In case of negative values, a constant value has to be added in order to make them positive.
We remark that, to keep the prediction consistent with the actual objective functions of the target problem, an inverse Box-Cox transformation has been applied on the predicted data.

Centering and scaling data. Another pre-processing step that is usually applied to the data after the data transformation is the centering and scaling step. The goal of this step is to remove the bias from the input data (mean equal to zero) and to standardize the variance (standard deviation equal to 1). This transformation is also called *autoscaling*. When the autoscaling transformation is applied to a set of data, from each value the mean value is removed and it is scaled by the standard deviation: $y_{autoscaled} = (y_{original} - \mu_y) / \sigma_y$.

III. Shepard Interpolation

The Shepard Interpolation technique [6] is a well-known method for multivariate interpolation. This technique is also called inverse distance weighting (IDW) method because the value of the response function in unknown points is the sum of the value of the response function in known points weighted with the inverse of the distance.

III.1. General Description

One of the most commonly used techniques for interpolation of scatter points is inverse distance weighted (IDW) interpolation. Inverse distance weighted methods are based on the assumption that the interpolating surface should be influenced most by the nearby points and less by the more distant points. The interpolating surface is a weighted average of the scatter points and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases. The Shepard Interpolation technique is one of the most used methods for inverse distance weighted (IDW) interpolation.
In particular, the value of a response function r(**x**) for an unknown design **x** is computed by using N known observations y_k as follows:

$$r(\boldsymbol{x}) = \frac{\sum_{k=1}^{N} w_k(\boldsymbol{x}) y_k}{\sum_{k=1}^{N} w_k(\boldsymbol{x})}$$

where:

$$w_k(\boldsymbol{x}) = \frac{1}{\mu(\boldsymbol{x}, \boldsymbol{x}_k)^p}$$

is the weighting function defined by Shepard, p is the power of the model and μ is the distance between the known point x_k and the unknown point x. It is important to note that the previous formulas are related only for unknown points, if x is one of the known point x_k the function r(**x**) is equal to the observation y_k.

III.2. Model Selection

For the case of the Shepard interpolation, the model selection phase is not so complex as for other models. In fact, in this case the only model tuning possibility that we have is the value of the exponent p that is referred in literature as the *power of the model*. Increasing the value of p decreases the influence of known points that are far from the point to be estimated, while decreasing the value of p increases that impact.

IV. Radial Basis Function

Radial basis functions (RBF) [7] represent a widely used interpolation/approximation model for multivariate problems. In particular, Radial functions are special classes of functions that have as main characteristic feature the fact that (in most of the cases) their response decreases (or increases) monotonically with distance from a central point.

IV.1. General Description

Radial basis functions (RBF) represent a widely used interpolation/approximation model. The interpolation function is built on a set of training configurations x_k as follows:

$$r(\boldsymbol{x}) = \sum_{k=1}^{N} \lambda_k \, \phi(\|\boldsymbol{x} - \boldsymbol{x}_k\|)$$

where ϕ is a scalar distance function, λ_k are the weights of the RBF and N is the number of samples in the training set. In MULTICUBE, we consider the following definitions for ϕ[1]

$$\phi(z) = \begin{cases} z & \text{linear} \\ z^2 \log z & \text{thin plate spline} \\ (1+z^2)^{1/2} & \text{multiquadric} \\ (1+z^2)^{-1/2} & \text{inverse multiquadric} \\ e^{-z^2} & \text{gaussian} \end{cases}$$

The weights λ_k are the solution of a matrix equation which is determined by the training set of configurations x_k and the associated observations y_k:

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

where

$$A_{jk} = \phi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|), \qquad j,k = 1,2,\dots,N,$$

IV.2. Model Selection

As can be easily noted by the general description section, the prediction capability on independent test data of the Radial Basis Functions is strongly related on the type of function ϕ used as radial function. For the case of RBF, the model selection phase have to decide what is the most suitable radial function ϕ for the target problem. Linear, thin plate spline, multiquadric, inverse multiquadric and Gaussian are the possible functions we considered in MULTICUBE.

[1] The thin plate spline function is usually defined only for bi-dimension problems. In our analysis we adopted that ϕ function also for other problem sizes.

V. Linear Regression

Linear regression [5] is a regression method that models a linear relationship between a dependent response function f and some independent variables x_i, i = 1…p plus a random term ε. The linear regression is the most common used analytical technique for understanding the relationship among system metrics and system parameters.

V.1. General Description

Linear regression is a technique for building and tuning an analytic model r(x) as a linear combination of x's parameters in order to minimize the prediction residual ε.
We apply regression by taking into account also the interaction between the parameters and the quadratic behavior with respect to a single parameter. We thus consider the following general model:

$$r(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j^2 + \sum_{l=1}^{n} \sum_{j=1, j \neq l}^{n} \beta_{l,j} x_l x_j + \sum_{j=1}^{n} \gamma_j x_j$$

where x_j is the level associated with the j-th parameter of the system configuration. Least squares analysis can be used to determine a suitable approximation of the parameters. The least squares technique determines the values of unknown quantities in a statistical model by minimizing the sum of the squared residuals (the difference between the approximated and observed values).

V.2. Model Selection

A well known measure of the quality of fit associated with the resulting model obtained by applying the linear regression is called *coefficient of determination* and it is defined as follows:

$$R^2 = 1 - \frac{\sum_k (r_k - y_k)^2}{\sum_k (y_k - \bar{y})^2}$$

where y_k is the k-th observation, $\bar{y}$ is the average of the observations, and r_k is the prediction for the y_k observation. R^2 corresponds to the ratio of variability in a data set that is accounted for by the statistical model. Usually, the higher R^2 the better is the quality of fit (with $0 \leq R^2 \leq 1$). Although adding parameters to the model can improve the R^2, there is a risk of exceeding the actual information content of the data, leading to arbitrariness in the final (fit) model parameters (over-fitting). This reduces the capability of the model to generalize beyond the fitting data, while giving very good result on training-data.

To this purpose, an "adjusted" definition of the R^2 has been introduced in the past. This term adjusts for the number of explanatory terms in a model; it increases only if the terms of the model improve it more than expected by chance and will always be less than or equal to R^2; it is defined as:

$$1 - (1 - R^2)\frac{N - 1}{N - p}$$

where p is the total number of terms in the linear model (i.e., the set of coefficients α, β, γ), while N is sample set size. Adjusted R^2 is particularly useful in the model selection stage of model building.

In order to understand the optimal number of terms of the linear model and the corresponding model order, we analyzed the behavior of the RSM cross-validation error and adjusted R^2.
In fact, the equation of the adjusted R^2 represents an improved measure of the overall quality of fit of the linear regression: it is inversely proportional to the model's degrees of freedom (i.e., $N - p$) which, in turn, depend on the order of the chosen polynomial r(x). As a matter of fact, higher degrees of freedom increase the chance of reduced variance of the model coefficients thus improving model stability while avoiding over-fitting.

For this reason, we limited the set of considered models to the following configurations:

1) First order model, without any interaction between parameters
2) First order model, with interaction between parameters
3) Second order model, without any interaction between parameters

Considering this, the model selection phase for the linear regression have the goal to select the more suitable model among the three models listed before.

VI. Artificial Neural Networks

An Artificial Neural Network (ANN) [8] is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the structure of the information processing system that is composed of a large number of highly interconnected processing elements (neurons) working in together. The learning process of ANNs, as in biological systems, involves adjustments to the synaptic connections that exist between the neurons. In particular, ANNs learn by example.

VI.1. General Description

Artificial Neural Networks (ANNs) are machine-learning models that automatically learn to approximate a target function (e.g. application performance, power consumption) based on a set of inputs (e.g. architectural parameters such as cache size or associativity). Figure 4 shows an example of ANN consisting of three input neurons, four hidden neurons, and one output.

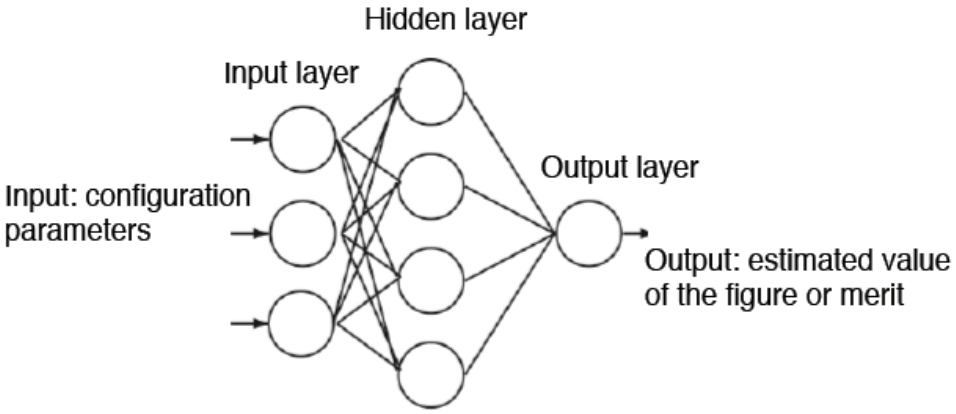

Figure 4 - Example of three layers ANN

In a fully connected feed-forward ANN, an input unit passes the data presented to it to all hidden units via a set of weighted edges. Hidden units operate on this data to generate the inputs to the output unit, which in turn calculates ANN predictions. Hidden and output units form their results by first taking a weighted sum of their inputs based on edge weights, and by passing this sum through a non-linear activation function (see following figure representing a generic neuron).

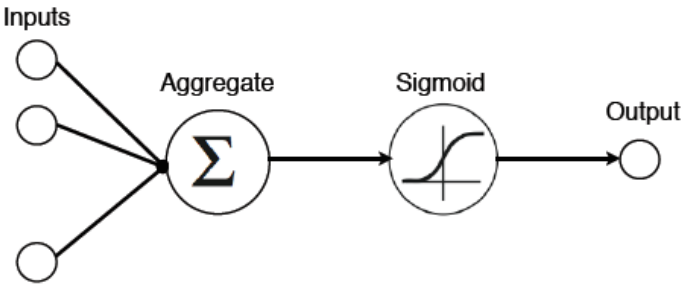

Figure 5 - Graphical representation of a single neuron adopting the sigmoid activation function

Increasing the number of hidden units in an ANN leads to better representational power and the ability to model more complex functions, but increases the amount of training data and time required to arrive at accurate models. ANNs represent one of the most powerful machine learning models for non-linear regression; their representational power is high enough to model multi-dimensional functions involving complex relationships among variables as for the cases faced in the MULTICUBE project.

VI.2. Model Selection

One of the key concepts for using the Artificial Neural Network is the model selection. In ANN we refer to model selection when we talk about the identification of the number of hidden layers and number of neurons for each hidden layers.
In MULTICUBE, we focus our attention on the Cascade 2 [9] algorithm for identifying a suitable topology for our ANN. The Cascade 2 algorithm is an iterative algorithm that selects candidate neurons to be added to the ANN by populating the hidden layers of the network and training the network weights. In particular, the topology built by Cascade 2 is composed by N hidden layers with only one neuron for layer. By using this algorithm, the number N of hidden layer is the only variable to be tuned during the model selection of the ANN.
To perform the model selection for the ANN avoiding the well-known problem known as over-fitting, the training set is partitioned into two subsets: a learning set and a validation set. While the learning set is used for identifying suitable candidate neurons, deciding its number and updating the network weights, the validation set is used only for computing the estimate prediction error of the neural network. In such way, a training stop (early stop) condition to avoid the over-fitting problem can be identified whenever the validation error starts to increase.

VII. Validation of the analytical techniques

This section shows the prediction capabilities of the previous presented analytical techniques. The results will be presented showing the performance of the analytical techniques by varying:

- The model parameters as presented for each technique in the model selection
- The Box-Cox transformations used to pre-process the training data

The performance metric of the analytical techniques used as prediction capability is the average normalized error by varying the number of points used as training set.

VII.1. Experimental Setup

Since at this point of the project we are still not able to make a "deep exploration process" of the MULTICUBE use cases, for the validation of the analytical techniques we used the results obtained by using the widely used SESC open-source multiprocessor simulator [12]. With the term "deep exploration process" we would like to identify an exploration process based on a large set of data for which the analytical model can be effectively trained and used to predict the system performance. In the D2.3.2 (Refined and extended multi-objective evaluation metrics) to be issued at M24, the results of the application of analytical techniques to the MULTICUBE use-cases will be provided.

For the evaluation of the system metrics (mainly execution time and energy consumption) during the validation phase, the open-source SESC [12] simulator has been used. SESC is a fast simulator for chip-multiprocessor architectures with out-of-order processors that can provide energy and performance values for a given application. The evaluation of the energy consumption of the memory hierarchy is supported by CACTI [13], while the energy consumption computation due to the core logic is based on the WATTCH models [14]. The SESC simulator has been chosen because of it offers a good trade-off between simulation speed and modeling accuracy (less than 5% with respect to an actual implementation of the MIPS R10K architecture) and because of it is widely adopted by the scientific community for multiprocessor design space modeling and exploration.

The analytical models described in this deliverable have been used to predict the execution time and energy consumption of an MPSoC architecture. The target system architecture is composed of a shared-memory multiprocessor with private L2 caches. In the target architecture, a MESI snoopy-based coherence protocol acts directly among L2 caches, requiring additional invalidates/writes between L1 and L2 caches to ensure coherence of the data. Cache inclusion is maintained explicitly by using the mechanism adopted for propagating the coherence events in the cache hierarchy [10].

To provide a comprehensive yet affordable validation of the proposed methodology, we identified a design space composed of a set of 9 independent platform parameters presented in Table I.

Table I - Design space used for the validation phase

Parameter	Min.	Max.
# Processors	2	16
Processor issue width	1	8
L1 instruction cache size	2KB	16KB
L1 data cache size	2KB	16KB
L2 private cache size	32KB	256KB
L1 instruction cache assoc.	1w	8w
L1 data cache assoc.	1w	8w
L2 private cache assoc.	1w	8w
I/D/L2 block size	16	32

The corresponding design space size consists of $|X| = 2^{17}$ configurations (131 072). If we assume a simulation time for each configuration of 10 minutes, then we will have more that 900 days of simulation in a single-core machine for the full search exploration. This underlines the need of alternative ways to explore the design space.

The selected target applications have been derived from the SPLASH-2 [11] parallel benchmark suite (U = {FFT, OCEAN, LU, RADIX}). For each application, we considered 3 different input data-sets, thus resulting into 12 application-data-set scenarios.

In the following validation, results are provided showing the average normalized error computed over a set of 15.000 randomly selected configurations of the design space. The results will be presented by varying a number of random configurations used as a training-set (known points) starting from 200 to 1800 configurations.

VII.2. Shepard Interpolation

Figure 6 shows the validation results for the Shepard Interpolations. In the figure, the different lines represent different values of the power variable in the Shepard Interpolator and different Box-Cox transformations.

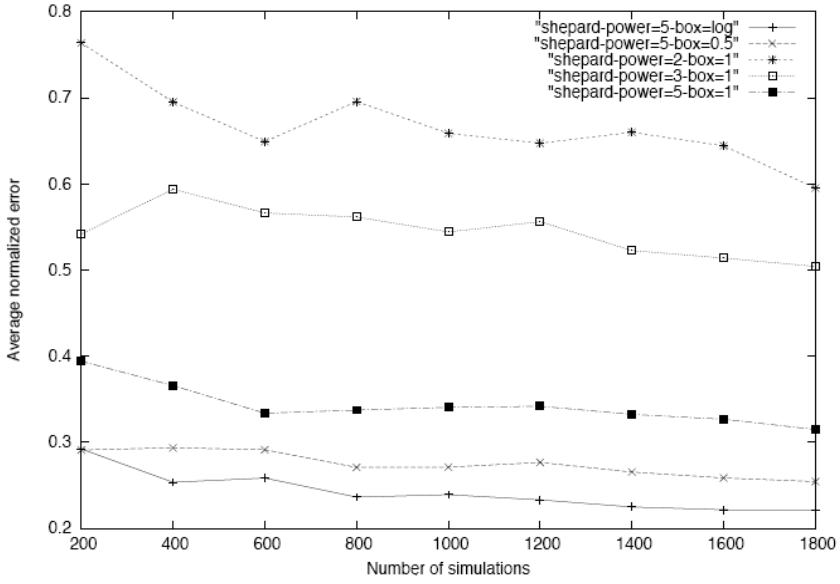

Figure 6 - Average normalized error for the Shepard Interpolation

In Figure 6, only the most significant power coefficients and preprocessing transformations have been shown. The prediction accuracy of all the selected models slightly increases (the average normalized error decreases) with the increment of the number of evaluated point in the training set. The model with a power value of 5 and $\lambda = 0$ provides the best performance in terms of mean error (less than 25%).

VII.3. Radial Basis Functions

Figure 7 shows the validation results for the Shepard Interpolations. In the figure, the different lines represent the different radial function and different Box-Cox transformations.

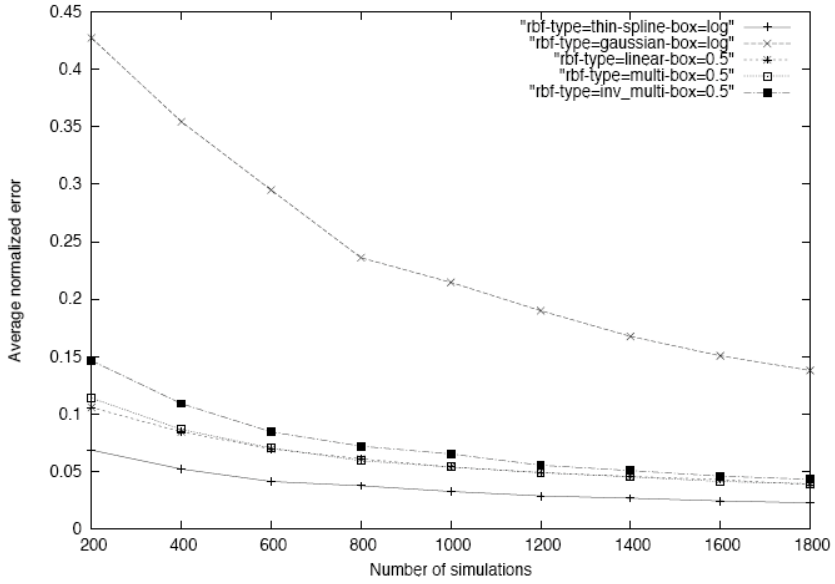

Figure 7 - Average normalized error for the Radial Basis Functions

Figure 7 shows only the RBF functions and preprocessing transformations with the best performance. As for the previous case, the accuracy of the selected models increases with the increment of the number of training point. The RBF using a thin-plate spline radial function where the data have been pre-processed by using a logarithmic Box-Cox transformation present the best results reaching an average normalized error down to 3%.

VII.4. Linear Regressions

To validate the results of the Linear Regression method, we added also some results obtained for the "adjusted" R^2 (see Figure 8) instead of showing only the average error graph (see Figure 9). In Figure 8 and 9 the different lines represent different linear models and different Box-Cox transformations.

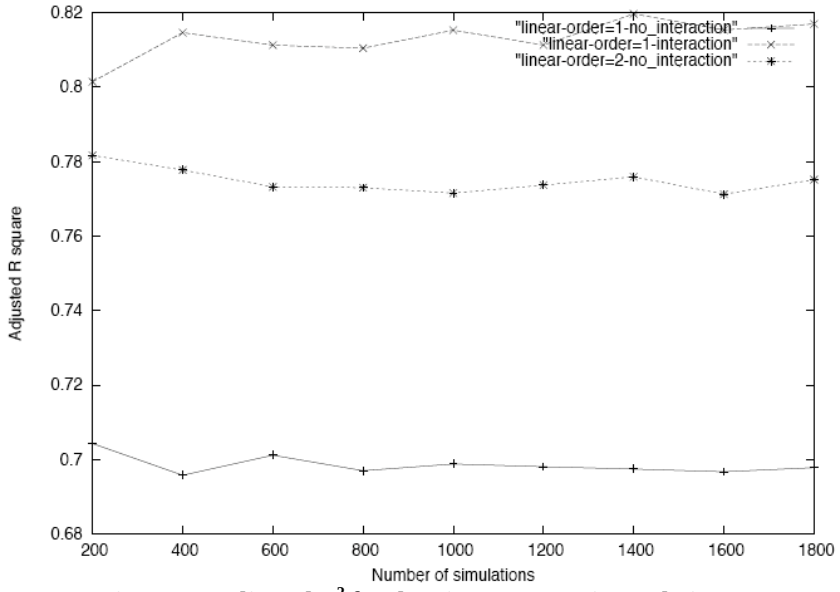

Figure 8 – Adjusted R^2 for the Linear Regression techniques

Figure 8 shows the average adjusted R^2 for the considered models by varying the number of simulations. The first order model with interaction between parameters presents the best results among all the possible model configurations in terms of ability to explain the linear dependence among all the parameters. The second order model without interaction between parameters is the second best, while the first order model without interaction is the worst one. The previous experimental considerations suggested us to choose the first order model with interaction as the candidate model to be further analyzed introducing the Box-Cox power transformations.

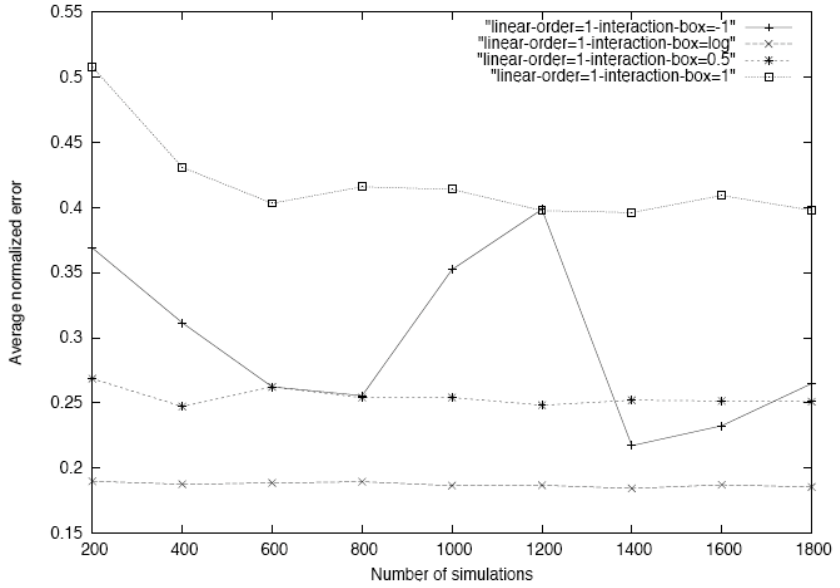

Figure 9 - Average normalized error for the Linear Regression techniques

Figure 9 shows that, among all the possible Box-Cox transformations, the logarithmic transformation is the most stable and provides the best approximation error for the considered model configuration. On the other hand, the $\lambda = -1$ transformation decreases the stability of

the prediction model despite the trends is on decreasing the average error.

VII.5. Artificial Neural Networks

Figure 10 shows the validation results for the Artificial Neural Networks. In the figure, the different lines represent different Box-Cox transformations applied to the Artificial Neural Network techniques.

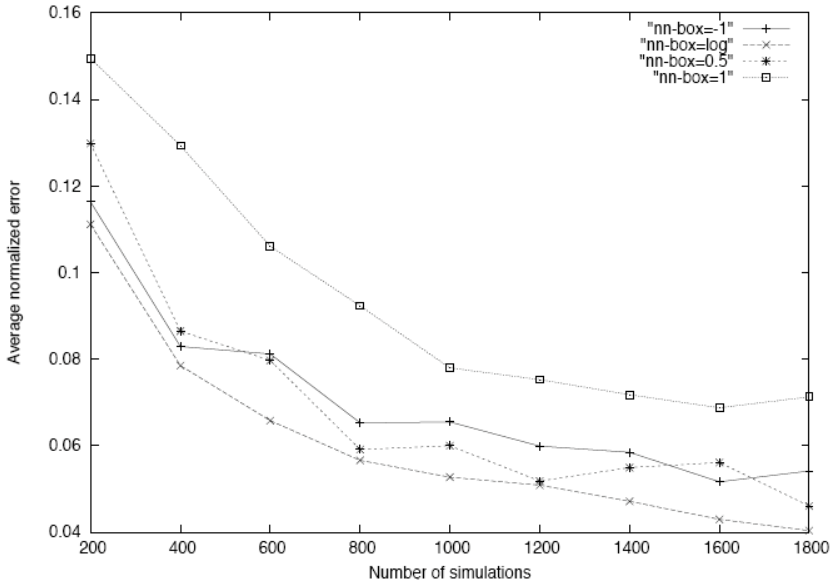

Figure 10 - Average normalized error for the Artificial Neural Networks

Although for every Box-Cox transform, the selected ANNs perform very well, with an error under 20%, the logarithmic Box-Cox transformation is the one that presents the most evident advantages. The ANN performances are comparable to the performance presented before by RBFs. For sure ANN needs more points to be trained, in fact the left part of the graph presents higher values with respect to RBF, but on average are less sensible to the Box-Cox transformation adopted during the data preprocessing step.

VIII. Development of the software modules

The previous described techniques have been implemented in software modules to be included in the final prototype of M3Explorer tool, one executable module for each analytical technique.

As interface for the analytical models, we adopted a standard format for the data. The common interface used to exchange information among modules is what is called XDR standard format (eXternal Data Representation, derived from IETF standard [15]) and it simply consists of a text format where each row represents a design point. There is a first field that represents how many data, separated by a space, will follow it and then, on the same row, can exist a new number to identify how many data will follow it an arbitrary number of times. When a row is finished a new line character is put so that is possible to identify a new row of the database. An example of a single line in the XDR format is given in Figure 11.

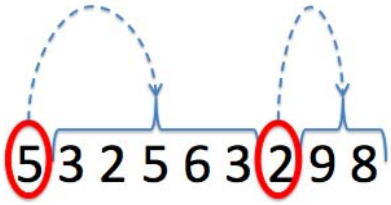

Figure 11 - Simple example of the XDR representation

This format is generic but it is also easy to be read. In fact, the meaning of the numbers in our specific case can be summarized in the following way: the size of the design vector (“5” in the example), the design vector (“3 2 5 6 3” in the example), the size of the response vector (“2” in the example) and the response vector (“9 8” in the example).

Considering not only a single line/point but also more points to be represented (as in the Figure 12) the design and response matrix can be easily identified.

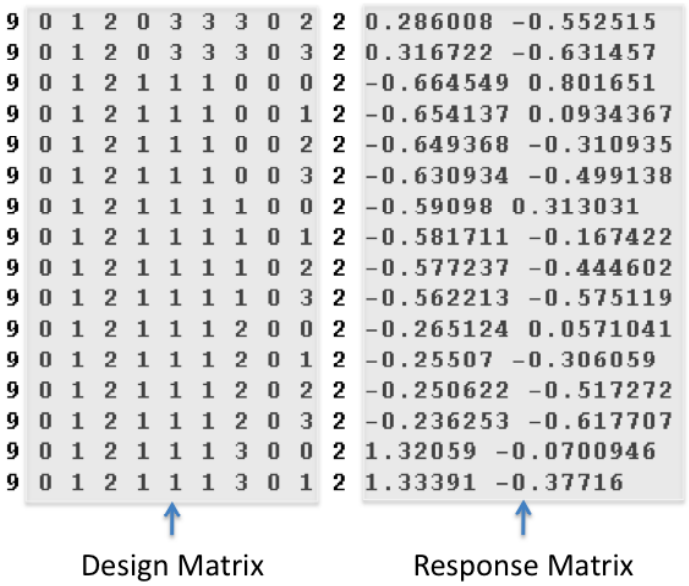

Figure 12 - Example XDR file with several design instances

The software modules implementing the analytical models will use the XDR format to take as input the points to be used for the training and to know what are the points to be predicted by the system response. The modules implementing the analytical models will generate as output the set of prediction values expressed by the XDR format.

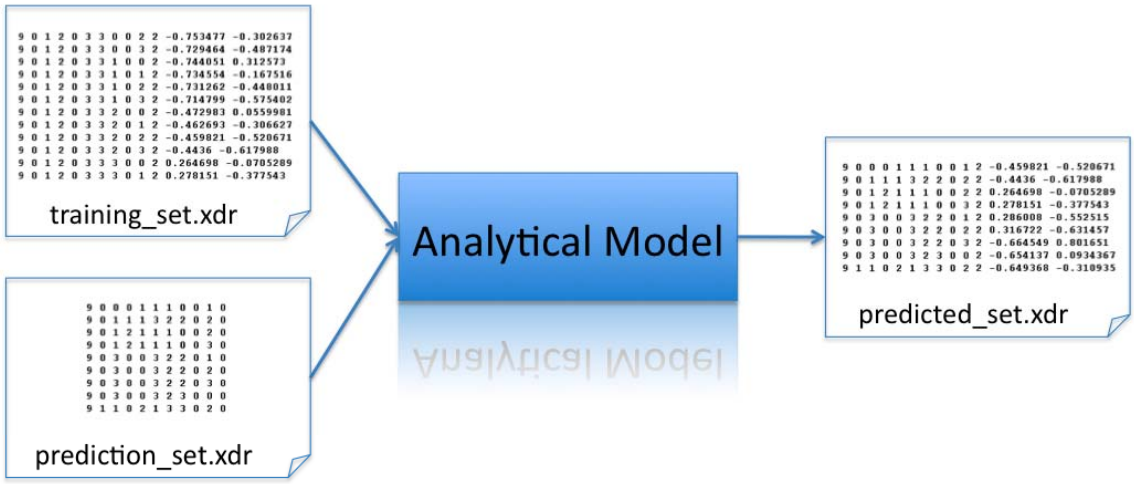

Figure 13 - Input and output files for the analytical models

Figure 13 shows how the analytical model interacts with the three different types of data represented by using the XDR format.

IX. Conclusions

This deliverable presented the set of analytical techniques (also called Response Surface Methods) developed in Task 2.3 up to M18. The developed techniques will be used in order to speed up the exploration time in the evaluation of the system metrics defined in D.1.1 (Definition of design tools requirements, and design evaluation metrics to be evaluated). Those analytical techniques will be trained through the results obtained by an initial simulation campaign and can be substituted by simulation-based system evaluation during the exploration phase.

In particular, the deliverable presents the description of both interpolative techniques (Shepard Interpolation and Radial Basis Function) and regressive techniques (Linear Regression and Artificial Neural Networks) showing also some preliminary evaluation results. The interfaces used to integrate the developed RSM modules into the exploration framework are also outlined.

A detailed evaluation of the presented techniques on the MULTICUBE use-case as well as the description of the complete integration of the techniques in the Multicube Explorer tool will be provided in the Deliverable D2.3.2 (Refined and extended multi-objective evaluation metrics) to be issued at M24.

X. References

[1]. A. Jerraya and W. Wolf, Multiprocessor Systems-on-Chips, Morgan Kaufman, San Francisco, Calif, USA, 2004.

[2]. Erez Perelman, Greg Hamerly, Michael Van Biesbrouck, Timothy Sherwood, and Brad Calder. Using simpoint for accurate and efficient simulation. In ACM SIGMETRICS Performance Evaluation Review, pages 318–319, 2003.

[3]. T. J. Santner, Williams B., and Notz W. The Design and Analysis of Computer Experiments. Springer-Verlag, 2003.

[4]. Douglas C. Montgomery. Design and Analysis of Experiments. John Wiley and Sons, 2005.

[5]. P.J Joseph, Kapil Vaswani, and M.J Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. High-Performance Computer Architecture, 2006. The Twelfth International Symposium on, pages 99– 108, 2006.

[6]. Karen Basso , Paulo Ricardo , De Ávila Zingano , Carla Maria , Dal Sasso Freitas. Modified Shepard Method , Investigating Alternatives for the Interpolation of Scattered Data. IEEE Symposium on Computer Graphics and Image Processing, 1999.

[7]. Buhmann, Martin D. (2003), Radial Basis Functions: Theory and Implementations, Cambridge University Press.

[8]. C. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 2002.

[9]. S. E. Fahlman and C. Lebiere, The Cascade-Correlation Learning Architecture, 1991.

[10]. D. Culler, J. P. Singh, and A. Gupta, Parallel Computer Architecture: A Hardware/ Software Approach (The Morgan Kaufmann Series in Computer Architecture and Design). Morgan Kaufmann, August 1998

[11]. S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. Splash-2 programs: characterization and methodological considerations. Proceedings of the 22th International Symposium on Computer Architecture, page 2436, 1995.

[12]. Jose Renau, Basilio Fraguela, James Tuck, Wei Liu, Milos Prvulovic, Luis Ceze, Smruti Sarangi, Paul Sack, Karin Strauss, and Pablo Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[13]. S. Wilton and N. Jouppi. CACTI:An Enhanced Cache Access and Cycle Time Model. IEEE Journal of Solid-State Circuits, volume 31, pages 677–688, 1996.

[14]. David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In Proceedings ISCA 2000: International Symposium on Computer Architecture, pages 83–94, 2000.

[15]. R. Srinivasan, RFC: 1832, “XDR: External Data Representation Standard”, Sun Microsystems, August 1995 http://www.ietf.org/rfc/rfc1832.txt