Chapter 1
The MULTICUBE Design Flow

Cristina Silvano,William Fornaciari, Gianluca Palermo, Vittorio Zaccaria, Fabrizio Castro, Marcos Martinez, Sara Bocchio, Roberto Zafalon, Prabhat Avasare, Geert Vanmeerbeeck, Chantal Ykman-Couvreur, Maryse Wouters, Carlos Kavka, Luka Onesti, Alessandro Turco, Umberto Bondi, Giovanni Mariani, Hector Posadas, Eugenio Villar, Chris Wu, Fan Dongrui, and Zhang Hao

Abstract This chapter introduces the design-flow of the MULTICUBE project whose main goal is the definition of an automatic multi-objective Design Space Exploration (DSE) framework to be used to tune the parameters of System-on-Chip architectures by taking into account the target set of metrics (e.g. energy, latency, throughput, etc.). One of the important goals of the automatic multi-objective DSE framework is to find design trade-offs that best meet system constraints and cost criteria which are indeed strongly dependent on the target application. A set of heuristic optimisation algorithms have been defined to reduce the overall optimization time by identifying an approximated Pareto set of parameter configurations with respect to a set of selected figures of merit. Once the approximated Pareto set is built, the designer can quickly apply decision criteria to select the best configuration satisfying the constraints. The DSE flow is based on the interaction of two frameworks to be used at design time: the Design Space Exploration Framework, a set of open-source and proprietary architectural exploration tools, and the Power/Performance Estimation Framework, a set of modeling and simulation tools (open-source and proprietary) operating at several levels of abstraction. The DSE flow also includes the specification of an XML integration interface to connect the exploration and estimation frameworks and a Run-time Resource Manager exploiting, at run-time, the best software configuration alternatives derived at design-time to optimize the usage of system resources.

1.1 Introduction

Many point tools exist to optimise particular aspects of embedded systems. However, an overall design space exploration framework is needed to combine all the decisions into a global search space, and a common interface to the optimisation and evaluation

C. Silvano (✉)
Dipartimento di Elettronica e Informazione Politecnico di Milano, Milano, Italy
e-mail: silvano@elet.polimi.it

tools. The MULTICUBE project focused on the definition of an automatic multi-objective Design Space Exploration (DSE) framework to be used to tune the System-on-Chip architecture for the target application evaluating a set of metrics (e.g. energy, latency, throughput, bandwidth, QoS, etc.) for the next generation of embedded multimedia platforms.

On one side, the MULTICUBE project defined an automatic multi-objective DSE framework to find design trade-offs that best meet system constraints and cost criteria, strongly dependent on the target application, but also to restrict the search space to crucial parameters to enable an efficient exploration. In the developed DSE framework, a set of heuristic optimisation algorithms have been defined to reduce the overall exploration time by computing an approximated Pareto set of configurations with respect to the selected figures of merit. Once the approximated Pareto set has been built, the designer can quickly select the best system configuration satisfying the target constraints.

On the other side, the MULTICUBE project defined a run-time DSE framework based on the results of the design-time exploration to optimise at run-time the allocation and scheduling of different application tasks. The design-time exploration flow results in a Pareto-optimal set of design trade-offs with different speed, energy, memory and communication bandwidth parameters. This information is used at run-time by a small OS kernel to make an informed decision about how the resources should be distributed over different tasks running on the multi-processor system on-chip. This resource distribution cannot be done during the design-time exploration itself, since it depends on which tasks are active at that time.

The goal of MULTICUBE design flow is to cover the gap between the system-level specification and the definition of the optimal application-specific architecture. The MULTICUBE activities have been driven by targeting the construction of a set of tools and accurate methodologies to support the design of application specific multi-core architectures.

In this context, a SystemC-based multi-level modeling methodology for multi-processors has been developed in the project. Once received the target architecture as input, the system model is provided to the simulator to evaluate different architectural trade-offs in terms of metrics. Then, the Design Space Exploration framework can be used to evaluate candidate configurations based on several heuristic optimisation algorithms. This step is implemented as an optimisation loop, where the selected architecture instance generated by the DSE framework is given back to the estimation framework for the metrics evaluation. The tool integration phase in MULTICUBE enabled to implement an automatic system optimisation engine to generate, for the target MPSoC architecture, either the best architectural alternative (if the exploration is done at design-time) or the best tasks scheduling and allocation (if the exploration is done at run-time).

To enable a widespread dissemination and use of the design flow in several application contexts, the following pre-requirements are introduced. First, the design flow aims at being independent from the particular language used for the description of the use case simulator. The design flow and the associated design tools should free the simulator provider from being tied to a specific programming language or model. Second, the interface between the design tools and the use case simulators should be specified and implemented by using a widely accepted and standardized interface. Standard interfaces are characterized by being supported by a large number of parsing and validation tools either in the public domain or commercially available while enabling a faster adoption of the design tool itself. Among the available interface specification languages, the most widely accepted and flexible is XML. XML enables to create efficient, customized data structures which are, at the same time, supported by industrial and academic tools for parsing, semantic evaluation and validation. These data structures can be used to facilitate the definition of tool interfaces.

The Chapter is organized as follows. Section 1.2 provides an overview of the MULTICUBE design flow, while Sect. 1.3 describes the design tools integration based on a common interface. Finally, Sect. 1.4 presents the advantages in using the automatic design space exploration approach.

1.2 Overview of the Design Flow

The MULTICUBE DSE flow (see Fig. 1.1) is based on the interaction of two frameworks to be used at design time: the Design Space Exploration Framework, an architecture exploration set of tools, and the Power/Performance Estimation Framework, a set of modeling and simulation tools operating at several levels of abstraction. The DSE flow also includes a Run-time Resource Manager to select at run-time the best design alternatives in terms of power/performance trade-offs generated during the design-time exploration phase.

According to the exploitation plan of the MULTICUBE project, both open-source and the proprietary exploitation models and tools co-exist into a single coherent view. This has been possible by making the design tools to adopt the same common MULTICUBE XML-based interface described in Sect. 1.3.

1.2.1 The Design Space Exploration Framework

The MULTICUBE Design Space Exploration Framework (see Fig. 1.1) consists of an architecture exploration set of tools providing the designers with the most appropriate optimisation of the multi-processor SoC considering several figures of merit such as performance and energy consumption.

The Design Space Exploration tools can be used at design time to automatically identify the Pareto optimal solutions of a multi-objective exploration given a set of design space parameters. During the MULTICUBE project, two design space exploration tools and some optimisation and analytical techniques have been developed and validated.

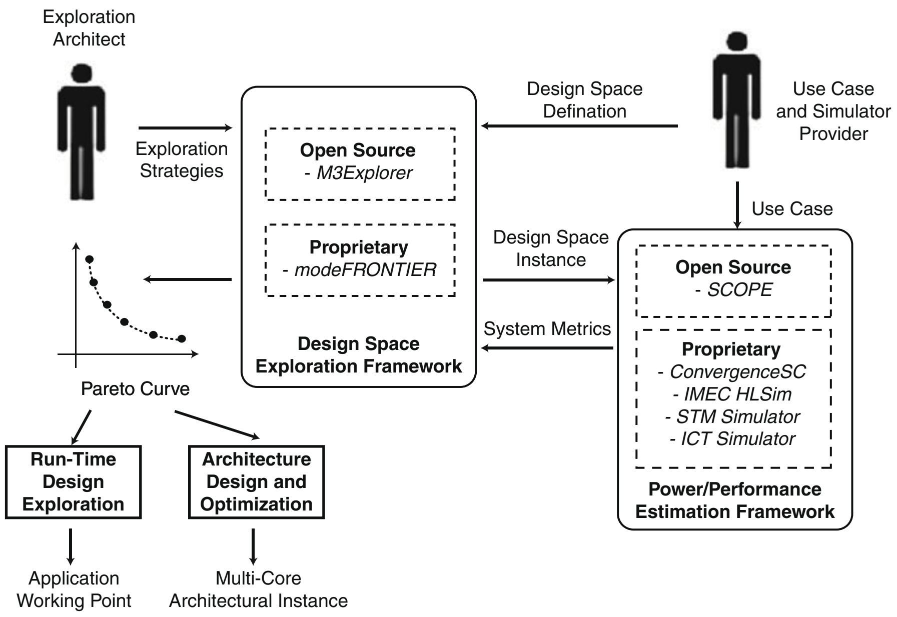

Fig. 1.1 Overview of the MULTICUBE design flow

- A new open source tool (Multicube Explorer available at http://www.multicube.eu) suitable for automatic design space exploration of MPSoC architectures. The tool enables a fast optimisation of parameterized system architecture towards a set of objective functions (e.g., energy, delay and area), by interacting with a system-level simulator through an open XML-based interface. Multicube Explorer provides a set of innovative sampling and optimisation techniques to support the designer in finding the best objective functions trade-offs. It also provides a set of Response Modeling Methods for speeding up the optimisation phase.
- An existing commercial tool (modeFRONTIER from ESTECO), widely adopted in other optimisation fields, has been re-targeted to support automatic DSE in the embedded systems field. The tool includes the most recent optimisation techniques available in literature, ranging from Design of Experiments to direct optimisers. modeFRONTIER (see also http://www.esteco.com) also provides meta-modeling support for the creation of interpolating surfaces from well statistically distributed designs to be used to perform the optimisation without computing any further analysis. The tool also supports multivariate statistical analysis and data mining tools directly integrated in the exploration process to enable the user to easily analyse complex data. The graphical user interface of modeFRONTIER provides access to all features of design experiment definition, exploration and analysis in a simple and intuitive way.

Both tools leverage a set of multi-objective optimisation algorithms that have been validated on several industrial use cases. In multi-objective optimisation problems there are more than one objective to be optimised (maximized or minimized), meaning that the outcome of the optimisation process is not a single solution but a set of solutions. This set of solutions, which is called the Pareto front, represents the best trade-off between the different (and possibly contradictory) objectives. The set of algorithms implemented includes state-of-the-art algorithms widely used in the field of multi-objective optimisation (ranging from evolutionary and genetic algorithms up to simulated annealing and particle swarm algorithms), enhanced versions of algorithms that were specifically targeted in the project for the multi-core SoC design, and new developed algorithms. The multi-objective optimisation algorithms developed in the MULTICUBE project are described in more detail in Chap. 3 of this book.

1.2.2 The Power/Performance Estimation Framework

The Power/Performance Estimation Framework (see Fig. 1.1) consists of a methodology and related tools that have been set up to provide accurate estimates for complexity, timing and power consumption at different abstraction levels and for different use cases. A set of tools has been used for the system modeling and estimation of several metrics such as energy consumption and execution time of the target MPSoC platforms among which:

- Multicube SCoPE: an extension of the open-source high-level SCoPE performance and power evaluation framework [7] developed by University of Cantabria for performing HW/SW co-simulation. Multicube SCoPE enables the definition of SystemC platform template models to evaluate performance and power consumption. Multicube SCoPE efficiency comes from the fact that performance and power estimations of the software side are performed at the application source code level by using back-annotation. The back-annotated software components are then linked to the hardware components by using standard SystemC interfaces. This modeling style is called *Timing Approximate*. Software back-annotation avoids instruction set simulation therefore decreasing of several orders of magnitude the simulation time and maintaining a fairly good accuracy with respect to cycle-accurate simulation. Multicube SCoPE also provides some hooks for enabling C/C++ software code to invoke operating system primitives compliant with POSIX and MicroC/OS. Multicube SCoPE is described in more detail in Chap. 2 of this book.
- A proprietary set of simulation tools developed by IMEC as SystemC-based transaction-level multi-core simulator built on top of the CoWare virtual prototyping environment to support platform-based design approach. The TLM-based prototype models an ADRES multi-core [6] and has been integrated with both modeFRONTIER and Multicube Explorer tools. The platform is composed of a variable number of processor nodes and memory nodes. All processor nodes

contain the IMEC proprietary ADRES VLIW processor and its scratch-pad local data (L1) memory. The processing nodes are connected to the memory nodes by a configurable communication infrastructure. It can be either a multi-layer AHB bus, which provides a full point-to-point connectivity, or a NoC model built with the CoWare AVF cross-connect IP.

- A High-Level time-annotated Simulator (HLSim, developed by IMEC) that provides a fast simulator of the ADRES platform at higher abstraction level to estimate metrics like performance and power consumption for a given platform architecture executing a parallelized version of the application. During the MULTICUBE project, HLSim has been extended with metrics on energy consumption derived from a multimedia use case for a relative comparison between different architectures and parallelizations. The introduction of HLSim in the design flow has provided several benefits such as speeding up the simulation and starting up the design exploration earlier than planned. HLSim-based explorations are much faster than TLM-based ones so as more extensive DSE was done by using HLSim to extract Pareto set information to be used at run-time.
- An instruction set simulator has been used for SP2 superscalar processor provided by STMicroelectronics and one simulator for the many-core architecture provided by ICT Chinese Academy of Science. Both simulators expose program execution time and power consumption as system-level metrics. More in detail, the ICT many-core architecture is a tiled MIMD machine composed of a bi-dimensional grid of homogeneous, general-purpose compute elements, called *tiles*. A 2D-mesh network architecture is used for connecting the cores to a non-shared memory sub-system.
- The DS2's STORM platform, a control-oriented architecture for powerline communication. The platform is used to model a PLC (Programmable Logic Controller) technology with several implementation choices. For this platform, both Ethernet QoS and internal communication are considered as metrics.

Given these target simulators, the MULTICUBE project developed a methodology, the multi-simulator based DSE approach shown in Fig. 1.2, to avoid potentially sub-optimal DSE results and to speed up the DSE process by exploiting multiple platform simulators to run the application at different abstraction levels.

The main idea is to get timing information (in terms of processor cycles) for an application execution on an accurate simulator (e.g. TLM-based cycle-accurate simulator) and feed this timing information back to a high-level timed simulator (e.g. HLSim) to achieve validation across simulators. Then, the DSE is done with a large number of application runs by using faster higher-level simulators (e.g. HLSim) and then the derived interesting operating points (usually clusters of operating points) are refined by using more accurate simulators (e.g. TLM-based and/or cycle-accurate simulators). The proposed methodology exploiting the synergy of multiple simulators at several abstraction level can be used to further speed up the DSE process while guaranteeing good accuracy of the simulation results. The methodology has been validated for the MPEG4 encoder application provided by IMEC by using three different

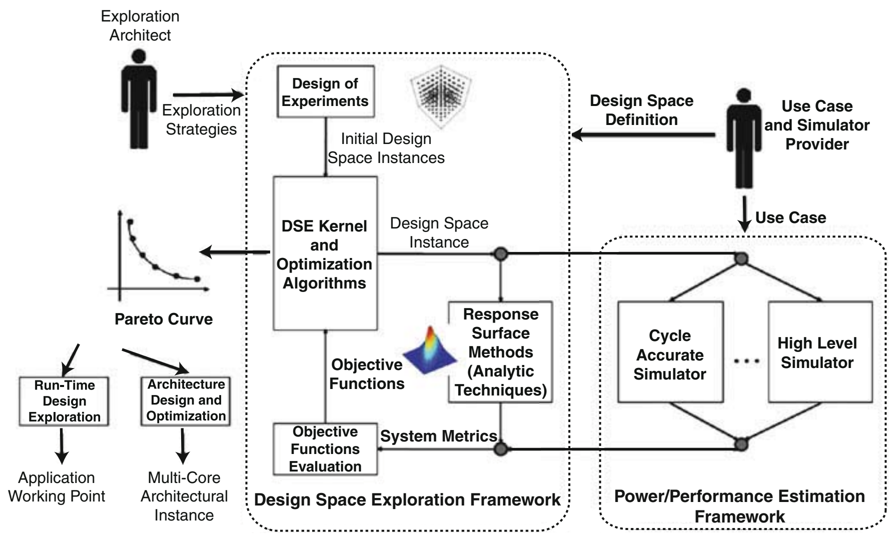
Exploration Architect
Exploration Strategies
Design of Experiments
Initial Design Space Instances
DSE Kernel and Optimization Algorithms
Design Space Instance
Response Surface Methods (Analytic Techniques)
Objective Functions
Objective Functions Evaluation
System Metrics
Design Space Exploration Framework
Design Space Definition
Use Case and Simulator Provider
Use Case
Cycle Accurate Simulator
...
High Level Simulator
Power/Performance Estimation Framework
Pareto Curve
Run-Time Design Exploration
Architecture Design and Optimization
Application Working Point
Multi-Core Architectural Instance

Fig. 1.2 Overview of the multi-simulator based DSE design flow

simulators (Multicube SCoPE, HLSim and TLM-based simulator) interfaced with the two available DSE tools (modeFRONTIER and Multicube Explorer). Overall, an acceptable relative accuracy has been found [1] with a significant speed-up in simulation time.

1.2.3 Response Surface Modeling Techniques

A set of analytical techniques (Response Surface Models, or RSMs) have been introduced to further speed up the design space exploration process (see Fig. 1.2). These techniques are key factors for developing a model of the system behavior without requiring the simulation of all the possible design configurations. RSMs have been proved to be an effective solution for analytically predicting the behavior of the system in terms of the target metrics without resorting to the system simulation.

Response Surface Modeling (RSM) techniques leverage the analytical dependence between several design parameters and one or more response variables by adopting both interpolation and regression techniques. The basic principle is to use a set of simulations either generated ad hoc by a Design of Experiment (DoE) phase or obtained by an exploration strategy previously applied to the design space, in order to obtain a response model of the system behavior. In the project, several RSM techniques have been implemented, among them Radial Basis Functions [3], Linear Regression [4, 5], Artificial Neural Networks [2] and Shepard's Interpolation. Every

RSM presented dramatic speed-up in terms of evaluation. Besides, it has been found that a peculiar mathematical transformation of input training set known as Box-Cox λ transform [4] has a great impact on the prediction accuracy. A sub-set of the above analytical techniques has been implemented and integrated in the MULTICUBE open-source tool while another sub-set was already available in the modeFRONTIER tool. RSM techniques for DSE are described in more detail in Chap. 4 of this book.

1.2.4 Run-Time Resource Management

The MULTICUBE design flow has been built to provide not only design-time support but also run-time support. In this scenario, at design time, the multi-objective design space exploration framework generates a set of Pareto-optimal operating points (for each application) annotated with system metrics like energy consumption, execution time, and memory and communication bandwidth values. The Pareto set can then be exploited at run time (while the application(s) are running) to optimise the overall system behavior (see Fig. 1.3). Specifically a separate Run-time Resource Manager (RRM) has been developed to exploit the set of operating points derived at design-time for all applications to steer the overall system behavior according to the imposed user requirements (quality, power, performance, etc.). The goal of the RRM is to use the Pareto information given by the design-time exploration on the operating points (of all applications) to make at run-time a decision to allocate the system resources to active applications based on the user requirements in terms of Quality of Service. Run-time Resource Management techniques are described in Chap. 5, while some more general concepts about resources management at the Operating System layer are presented in Chap. 6.

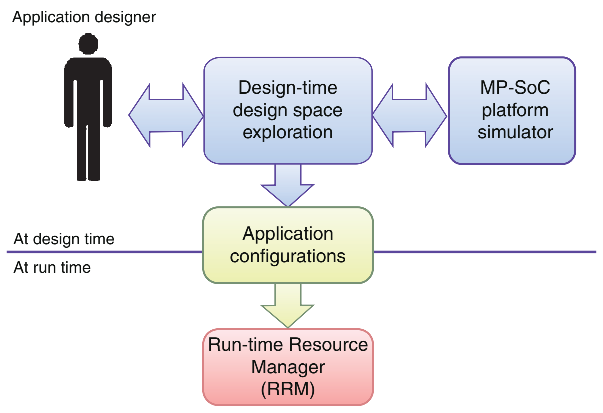
Application designer
Design-time design space exploration
MP-SoC platform simulator
At design time
At run time
Application configurations
Run-time Resource Manager (RRM)

Fig. 1.3 Overview of the run-time support

1.3 Design Tool Integration Based on the MULTICUBE XML Interface

Strategic importance from the point of view of the MULTICUBE exploitation is associated to the common XML Tool Interface Specification for the integration of the different tools and use cases. The common interface enabled the independent development of software modules and a seamless integration of design tools and data structures into a common design environment. The specification is defined in terms of XML, a widely used standard notation. To introduce the notation, let us highlight that, in the MULTICUBE design flow, there are two types of user agents to interact with the framework: the *use case and simulator provider* and the *exploration architect*. The simulator is the executable model of the use case and it is a single executable file (binary or script) which interacts with the DSE tool to provide the value of the estimated metrics, given an input configuration. The MULTICUBE project addressed the formalization of the interaction between the simulator and the DSE tools, that is essentially an automatic program-to-program interaction (see Fig. 1.4):

1. The DSE tool generates one feasible system configuration whose system metrics should be estimated by the simulator.
2. The simulator generates a set of system metrics to be fed back to the DSE tool.

To automatically link the use case simulator to the DSE tool, a design space definition file should be released by the use case and simulator provider together with the executable model of the use case (simulator). This file describes the set of configurable parameters of the simulator, their values range and the set of evaluation metrics that can be estimated by the simulator. This file describes also how to invoke the simulator as well as an optional set of rules with which the generated parameter values should be compliant. The rules are only used by the exploration tool to avoid the generation of invalid or unfeasible solutions during the automated exploration

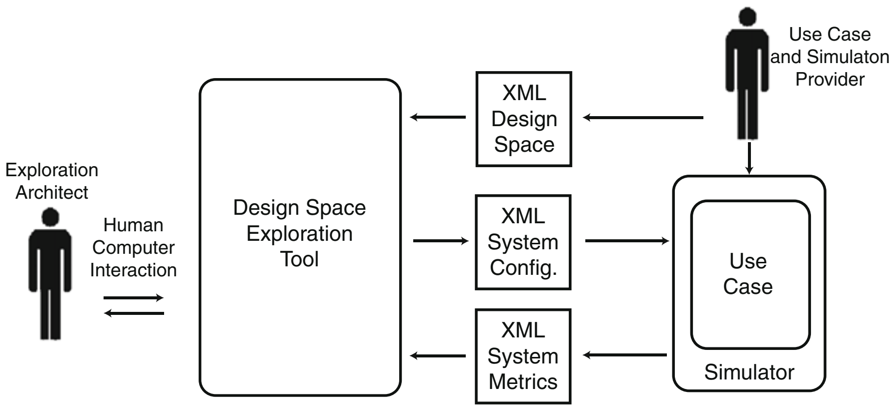

Fig. 1.4 Overview of the tool interfaces via XML

process. The above interaction has been addressed by creating a specification based on an XML based grammar for writing both the design space definition file and the simulator interface files. The grammar is defined by using the XSD schema language.

1.3.1 Design Space Definition

The definition of the design space is done by using an XML file that is composed of a preamble, which defines the name-space and supported version. The current release of the MULTICUBE XML interface specification is R1.4 and it is available on MULTICUBE web page (www.multicube.eu).

```
<?xml version="1.0" encoding="UTF-8"?>
<design_space xmlns="http://www.multicube.eu/" version="1.4">
<simulator> ... </simulator>
<parameters> ... </parameters>
<system_metrics> ... </system_metrics>
<rules> ... </rules>
</design_space>
```

The remaining part of the file describes the simulator invocation method (`<simulator>` tag), the set of parameters of the simulator which can be configured (`<parameters>` tag), the system metrics which can be estimated by the simulator (`<system_metrics>` tag) and the rules which have to be taken into account by the exploration engine to generate the feasible configurations.

1.3.1.1 Simulator Invocation

The `<simulator_executable>` marker is used for specifying the complete path name of the executable:

```
<simulator>
<simulator_executable path="/path/my_simulator_executable" />
</simulator>
```

1.3.1.2 Parameters Definition

The `<parameters>` tag is used by the use case and simulator provider to specify the names, the types and the ranges of the parameters that can be explored by the DSE tool. The section contains a list of `<parameter>` markers:

```
<parameter>
<parameter name="il1_cache_block_size_bytes"
   description="..." type="exp2" min="8" max="64"/>
<parameter name="bpred" description="b.p. type" type="string">
   <item value="nottaken"/>
   <item value="taken"/>
   <item value="perfect"/>
   <item value="bimod"/>
```

```
    <item value="2lev"/>
    <item value="comb"/>
</parameter>
...
</parameters>
```

For each parameter an unique name must be provided. The parameters types can be divided into two categories: scalar types, variable vector types. Scalar types can be **integer**, **boolean** (a subset of integers), **exp2** (progression of power of 2) and **string** (a type for defining categorical variables). Vector types can be used to describe combination of boolean values (*on-off-masks* or *permutations*). In particular, on-off-masks can be useful for describing the space of active processors while permutations can be used to describe the mapping of tasks on the available processors.

1.3.1.3 System Metrics Definition

The `<system_metrics>` section is used by the use case and simulator provider to specify the names, the types and the units of the system metrics that can be estimated by the simulator:

```
<system_metrics>
<system_metric name="cycles" type="integer" unit="cycles" />
<system_metric name="instructions" type="integer" unit="insts"/>
<system_metric name="powerconsumption" type="float" unit="W" />
<system_metric name="area" type="float" unit="mm2" />
</system_metrics>
```

A complex expression of the system metrics can be defined as one of the objective of the exploration algorithm.

1.3.2 Simulator Input/Output XML Interface

The simulator input file contains a preamble and a sequence of `<parameter>` sections where, for each parameter, the name and the value is specified. The number of `<parameter>` sections and the name of the parameters should be the same as defined in the XML Design Space description file. Similarly the simulator output file contains a preamble and a sequence of `<system_metric>` sections where, for each metric, the name and the value is specified. Besides, an appropriate error reporting syntax has been described in the specification.

1.4 Advantages of Automatic DSE

The procedure to assess the benefits of the introduction of an Automatic Design Space Exploration (or Optimization Methodology) has to address, not only the final objective quality and the improvement of the target design but also the impact of such a technology on the entire design process of embedded computing platforms. The benefits on the process can be measurable and tangible like the reduction of the overall design process lead time, and qualitative or intangible like the streamlining and the reduction of human error prone repetitive operations. These benefits are particularly valuable for design problems where the number of configuration parameters to be explored is quite large, like in MP-SoC designs.

The specific design process activities can be analyzed and classified to measure the various performance indicators. In a general way, we can consider the following steps as the basis for any manual design space exploration or optimization process:

- Model Setup: preparation of an initial model of the virtual platform;
- Simulation: execution of the simulation of the executable model with a single configuration of parameters;
- Results Assessment: meaningful measures are extracted and compared with historical and expected ones;
- Model Edit: the model is manually modified and resubmitted for a further analysis.

Figure 1.5 represents the steps of a typical manual exploration procedure. In a manual approach, the exploration of the design space is done by subjective assumptions of the human designer, who will modify at most one or two parameters per evaluation. The model simulation corresponds to a minimal portion of the time of the whole exploration procedure. A large amount of time is spent by the designer editing the configuration parameters and analyzing the results. There is also an Idle Time (from the point of view of the use of computational resources) that lasts from the end of the simulation till the moment in which the human operator is informed about it and handles the simulation tools to get the results. This idle time can be very short if the designer is immediately informed about the end of the simulation or can be significant if the designer is not on duty.

The automatic design space exploration process can be defined by identifying the following basic steps:

- Model setup: the model must be correctly parameterized in order to be easily managed by the automatic exploration tools;

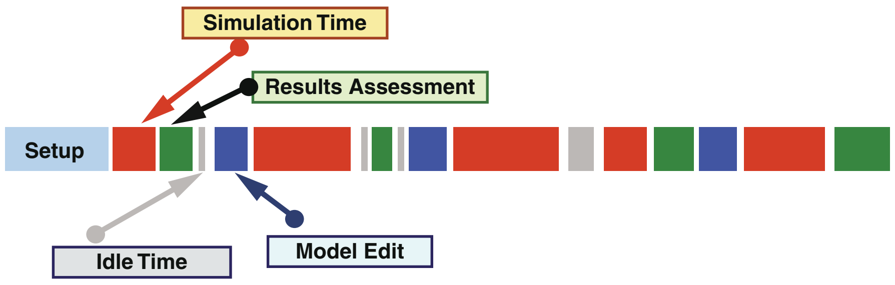

Fig. 1.5 Manual exploration procedure

- Simulation time: the simulation of the executable model with a single configuration of parameters is carried out (this phase is similar to the one that is performed with the manual approach);
- Automatic DSE overhead: this step includes the automatic assessment of the results, the automatic selection of the next configuration to be simulated (model selection) and the data transfer operations between the simulator and the design space exploration tool and its corresponding storage into the design database.

Figure 1.6 describes the automatic exploration procedure. The exploration of the design space is done by numerical/objective criteria. The Design Space Exploration tool (or "exploration engine") will change systematically all the parameters for each analysis and will evaluate the best result by adopting numerical formulas. The setup phase can be considerably longer than the set up of a manual exploration, since it usually requires an extended definition of the model to interact with the exploration engine, the definition of the proper optimization strategy and the definition of the multi-objective goals to be achieved. However, after that, the model evaluations (Simulation Time) is similar to the simulation time required in the manual approach, except for a further step (Automatic DSE overhead) that involves the automatic assessment of the results, the automatic selection of the next configuration to be simulated (model selection) and the time spent for data communication and storage.

Based on the past experience of ESTECO to deal with industrial customers, even if a single evaluation takes just a few seconds, it is difficult that a designer using a manual optimization procedure can evaluate more than seven designs of medium complexity within one hour. The designer has to go through all steps described before, editing the configuration parameters, running the analysis, reading and analyzing the results, etc. In this case, it is expected that in one person-day (10 h), the designer can run at most 70 designs. For the same problem, experience shows that an automatic approach can handle something like 600 designs per hour, which means about 14,400 designs per day. Since the automatic procedure can work 24 four hours a day, including weekends and holidays, the advantages of the automatic procedure are very clear.

However, there are other advantages of the automatic exploration procedure. In the automatic exploration, all data concerning previous evaluations are always stored in a

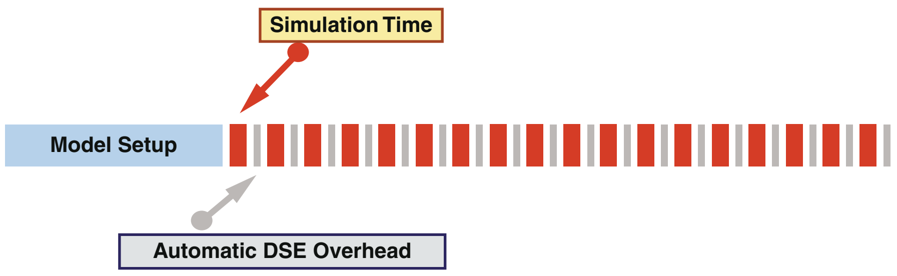

Fig. 1.6 Automatic exploration procedure

structured database. The designer, not only will not be stuck on repetitive operations, but can focus his/her attention (and profit from his/her experience) in analyzing the designs database in a statistical manner.

Moreover, the automatic exploration is driven by an optimization engine based on several optimization algorithms, whereas the manual exploration is based on designer ability and experience to assess the results and to move towards the next instance of the model to be simulated.

Figure 1.7 presents a direct comparison between manual and automatic optimization. The global lead time (Tmo) of the manual design exploration is determined by the number of manual iterations that are needed to reach the expected design improvement. The global lead time (Tao) in the automatic procedure case is dominated by the effective simulation time needed by the optimization strategy, given the overhead due to the automatic extraction and processing of measures, and communication latencies between the design exploration tool and the simulator. Another difference concerns the type of results produced by both types of optimization: the automatic optimization generates a set of designs that are likely to belong to the Pareto front, while the manual optimization generates just a set of designs that the designer found interesting. On the average, the manual approach suffers from the fact that it is affected by personal views, experience and background. This decreases the likelihood of finding points that belong to the Pareto front.

The reduction of the lead time of the design exploration phase is frequently reflected in the reduction of the overall time-to-market for the final product. The lead time must be always considered as one of the objectives for the introduction of the automatic exploration methodologies in industrial design processes, together with the more specific product related ones.

The general automatic DSE procedure described in this Chapter represented the common basis for the validation of each of the use cases of the MULTICUBE project. In order to apply this procedure, the responsible for each of the use cases adapted

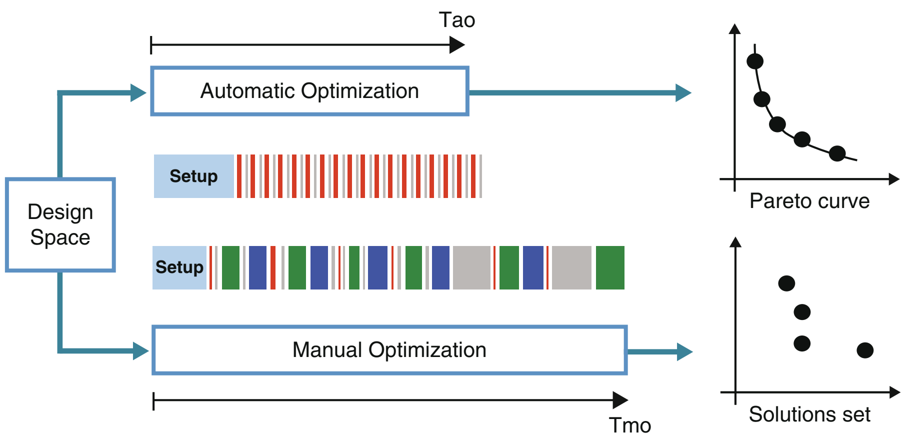

Fig. 1.7 Comparison between manual and automatic exploration procedures

however the general procedure to the specificities of its particular application and context, but without modifying the aim and the general steps described above.

1.5 Conclusions

In this chapter, the main structure of the MULTICUBE design flow has been introduced to be detailed in the next chapters. The founding principles of the proposed structure are meant to cover the gap between the system-level specification and the definition of the optimal application-specific architecture. The flow is based on the interaction of two frameworks to be used at design time: the Design Space Exploration Framework, an architecture exploration set of tools, and the Power/Performance Estimation Framework, a set of modeling and simulation tools operating at several levels of abstraction. The DSE flow also includes a Run-time Resource Manager able to select at run-time the best design alternatives in terms of power/performance trade-offs generated during the design-time exploration phase.

References

1. Avasare, P., Vanmeerbeeck, G., Kavka, C., Mariani, G.: Practical approach to design space explorations using simulators at multiple abstraction levels. In: Design Automation Conference (DAC) User Track Sessions. Anaheim, USA (2010)
2. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (2002)
3. Joseph, P., Vaswani, K., Thazhuthaveetil, M.: Construction and use of linear regression models for processor performance analysis. High-Performance Computer Architecture, 2006. The Twelfth International Symposium on pp. 99–108 (2006)
4. Joseph, P.J., Vaswani, K., Thazhuthaveetil, M.J.: A predictive performance model for superscalar processors. In: MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 161–170. IEEE Computer Society, Washington, DC, USA (2006). DOI http://dx.doi.org/10.1109/MICRO.2006.6
5. Lee, B.C., Brooks, D.M.: Accurate and efficient regression modeling for microarchitectural performance and power prediction. Proceedings of the 12th international conference on Architectural support for programming languages and operating systems **40**(5), 185–194 (2006). DOI http://doi.acm.org/10.1145/1168917.1168881
6. Mei, B., Sutter, B., Aa, T., Wouters, M., Kanstein, A., Dupont, S.: Implementation of a coarse-grained reconfigurable media processor for avc decoder. J. Signal Process. Syst. **51**(3), 225–243 (2008). DOI http://dx.doi.org/10.1007/s11265-007-0152-8
7. Posadas, H., Castillo, J., Quijano, D., Fernandez, V., Villar, E., Martinez, M.: SystemC platform modeling for behavioral simulation and performance estimation of embedded systems. Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation pp. 219–243 (2010)