



FP7 – 216693 - MULTICUBE Project

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

Deliverable D4.2.4: Final report on the application of MULTICUBE design flow to the demonstrators

Revision [8]

Delivery due date: M30 (June 2010)


Actual submission date: October 1st, 2010

Lead beneficiary: POLIMI

Dissemination Level of Deliverable		
PU	Public	X
PP	Restricted to other program 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	
	Other	



Authors:	Gianluca Palermo (POLIMI), Marcos Martinez (DS2), Sara Bocchio (STM), Prahabat Avasare (IMEC), Geert Vanmeerbeeck (IMEC), Hector Posadas (UC)		
Reviewer(s):	Cristina Silvano (POLIMI)		
WP/Task No:	WP4/T4.2	Number of pages:	46
Identifier:	D4.2.4_V8	Dissemination level:	PU
Issue Date:	October 1st, 2010		

Keywords:	Multi-Objective Design Space Exploration, Multi-Processor SoC Architectures, Exploration Algorithms
Abstract:	<p>This PUBLIC deliverable (issued at month 30) describes the final assessment of the MULTICUBE design flow for the design space exploration of the project demonstrators. This deliverable complements the results already reported in the accompanying reports of the three project demonstrators released as D4.2.1, D4.2.2 and D4.2.3 (PUBLIC). The demonstrators have been used in WP4 to validate the capabilities of the MULTICUBE design flow and tools developed in WP2 and WP3 to speed up the design space exploration process. Other results have been previously shown in the PUBLIC deliverable D3.2 “Implementation and Evaluation of the Exploration Algorithms” mainly in terms of the accuracy and quality of the optimization algorithms validation process.</p> <p>This deliverable has two main objectives. First, the deliverable describes the benefits that can be obtained by using simulators at several levels of abstractions during the DSE process of the multimedia use case based on MPEG4 encoder. Second, the deliverable reports the results of the application of the automatic DSE assessment procedure as defined in D4.1.2 “Final Evaluation Plan” to the DS2 Powerline use case, STM SP2 low-power processor use case and to the IMEC design flow. Mainly, the reported results prove the benefits of the MULTICUBE approach in terms of design turnaround time.</p> <p>Overall, a strong validation effort has been done by all partners in WP4 under the leadership of DS2 to assess the project results in terms of the application of the assessment procedures to the industrial use cases developed in the project and to check the assessment criteria (defined in D4.1.2). In my opinion, this deliverable represents one of the most important achievements of the third reporting period (Year 3) of the project. Being a public report, this deliverable can be easily accessed from the web for a wide dissemination of the project results.</p> <p>The Project Coordinator</p>
Approved by the Project Coordinator:	Date:
	October 1 st , 2010

Index

1	Executive Summary.....	4
2	DSE based on simulators at several abstraction levels.....	5
2.1	Multi-Simulator based DSE Approach.....	5
2.1.1	Key Enablers for DSE approach.....	7
2.2	Multimedia Use Case Modeling and Experimental Results.....	10
2.2.1	Use case modeling and experimental results with HLSim and CoWare-based simulators.....	10
2.2.2	Use Case Modeling and experimental results with M3-SCoPE.....	14
3	Automatic Design Space Exploration Assessment Procedure.....	17
3.1	DS2 Automatic DSE Assessment Procedure.....	21
3.1.1	Description of DS2 Automatic DSE Assessment Procedure.....	21
3.1.2	Comparison Results.....	26
3.2	STM Automatic DSE Assessment Procedure.....	29
3.2.1	Description of STM Automatic DSE Assessment Procedure.....	32
3.2.2	Comparison Results.....	33
3.2.3	Other benefits of the automatic DSE.....	36
3.3	IMEC Automatic DSE Assessment Procedure.....	37
3.3.1	Scratch-Pad Memory Management.....	37
3.3.2	MPSoC Platform Optimisation.....	38
3.3.3	Dynamic MPSoC Runtime Management evaluation.....	39
3.3.4	Benefits of Automated vs Manual DSE at IMEC.....	40
4	Common Assessment Criteria.....	41
5	Final considerations about MULTICUBE DSE Flow.....	45

1 Executive Summary

This deliverable describes the final assessment of the MULTICUBE design flow for the design space exploration of the demonstrators developed in the project. This deliverable complements the results already reported in the accompanying reports of the demonstrators **D4.2.1**, **D4.2.2** and **D4.2.3**. The demonstrators have been used in **WP4** to validate the capabilities of the MULTICUBE design flow and tools developed in **WP2** and **WP3** to speed up the design space exploration process.

The deliverable is organized as follows.

First, the deliverable describes in **Section 2** the main results obtained by using simulators at several levels of abstractions during the DSE process of the multimedia use case.

Second, this deliverable reports in **Section 3** the results of the application of the automatic DSE assessment procedure as defined in **D4.1.2 (Final Evaluation Plan)** to the DS2 Powerline use case (**Section 3.1**), STM SP2 low-power processor use case (**Section 3.2**) and to the IMEC design flow (**Section 3.3**) including the scratch-pad memory management, MPSoC platform and run-time management.

Third, in **Section 4**, a final table has been reported to demonstrate that the common assessment criteria (as defined in **D4.1.2**) have been successfully met at the end of the project.

Finally, in **Section 5**, some final considerations have been drawn to summarize the benefits that can be obtained by using an automatic DSE process in several design contexts to encourage the future adoption of the MULTICUBE approach by designers and researchers outside the project.

2 DSE based on simulators at several abstraction levels

Performing a Design Space Exploration (DSE) for an application running on an embedded platform requires a large number of application runs. Usually in a design cycle, these application runs are performed on an embedded platform simulator. There is always a trade-off between accuracy of the simulation results and the time taken for each simulation e.g. a more accurate simulation (such as cycle-accurate Transaction-Level Model (TLM)) will take much more time to complete the simulation than a pure functional-level simulation. But the accuracy of the results for TLM-based simulations is usually much more reliable than the functional-level one. Due to a time-to-market pressure, designers get limited time to perform DSE. On the other hand, growing complexity of embedded systems increases difficulty in predicting optimal operating point(s) for the system. These restrictions usually results in sub-optimal DSE task performed only on a selected set of design parameters, thus potentially missing many other interesting design configurations.

In the context of MULTICUBE project, we have developed a methodology to avoid this pitfall of potentially sub-optimal DSE results. We propose using multiple platform simulators to run the application at different abstraction levels i.e. first an exhaustive DSE is done with a large set of application runs using faster higher-level simulators (e.g. HLSim and then the derived interesting operating points (usually clusters of operating points) are explored using more accurate simulators (e.g. TLM-based and/or cycle-accurate simulator). We have validated our methodology by doing DSE for an MPEG4 encoder application provided by IMEC by using three different simulators interfaced with a available DSE tools: *modeFRONTIER* and *M3Explorer*.

2.1 **Multi-Simulator based DSE Approach**

In MULTICUBE DSE approach, we use multiple simulators in combination with a commercially available sophisticated DSE tool to achieve the desired DSE in a significantly less amount of time.

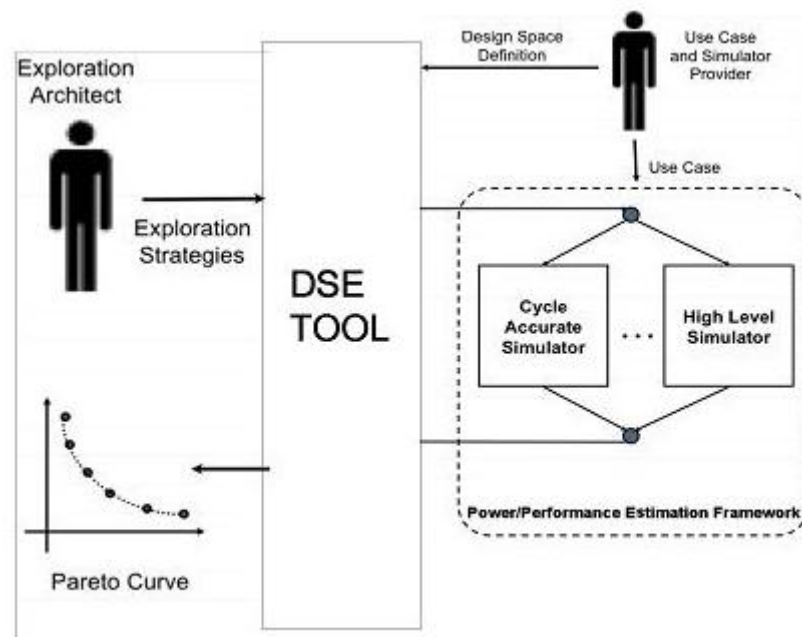


Figure 1 : DSE tool interfaced with multiple simulators using XML-based interfaces

Figure 1 displays how DSE tool interacts with different simulators during its exploration phase. The main idea is that the DSE tool takes, as input, constraints imposed on the design space parameters e.g. it can take range of instruction cache sizes that one would like to explore as an input parameter (assuming that there is a simulator available which takes instruction cache as an input parameter). There are various phases of interaction between user, DSE tool and simulators. For all these interactions, we have used XML interfaces as defined in MULTICUBE project.

First interaction is between user and DSE tool. User conveys the DSE tool a design space definition which includes ranges of input parameters that the tool can explore e.g. range of interconnect width on the platform. When user specifies a particular parameter to be explored, it inherently implies that the user has an access to a simulator or system in which it can take that parameter as an input. Hence in practice, user has to provide a (set of) combination of input parameter and corresponding simulator.

Next in the exploration cycle, DSE exploration architect (most of times this is the same person as the user) decides DSE exploration strategies e.g. which DSE algorithms to use depending on various factors e.g. available resources (hardware to run DSE, time ...). At the end of this decision, DSE tool generates its dataset for Design of Experiments (DoE). These datasets are simulated on provided simulator(s), in parallel if possible.

Second interaction is between DSE tool and the simulator. DSE tool specifies parameter(s) for each simulation to the simulator and simulator in return gives back simulation results to DSE tool, both in the form of pre-defined XML interfaces, essentially a list of parameter name-value pairs.

Running across different abstraction levels in different simulators is an iterative process. This is mainly to feed the performance (mainly timing) annotations back into high-level simulations from the ones measured from detailed (e.g. TLM-model) simulations. Also it is important to validate simulators at different abstraction levels by running them with exactly same input parameters. Currently mixing of different simulations via the DSE tool is handled manually as it is heavily dependent on the simulators available and the application.

At the end of DSE, the tool collects the outputs of all the simulations. This output can then be interpreted or viewed with different graphing tools to get the desired design space analysis. Usually, DSE is an iterative process: after doing initial analysis, DSE tool might derive interesting regions of design space which it might want to explore further in detail. Here is where application user can guide the DSE tool to switch between available simulation abstraction levels. Such a decision is complementary to the sophisticated data analysis support provided by commercially available DSE tools (e.g. modeFRONTIER).

2.1.1 Key Enablers for DSE approach

To make our DSE approach feasible, following three ingredients need to be developed:

- Common pre-defined XML-based interface between DSE tool and simulators
- API to define simulator-specific part of the application.
- Back annotations (timing) to validate different simulators used

These three elements are described in detail hereafter.

2.1.1.1 Common XML interfaces

First, key ingredient necessary for our approach is common predefined interfaces between user, DSE tool and different simulators (Figure 1). XML-based interfaces are used for this purpose are the ones developed in WP1 of MULTICUBE project.

First interface is between user and DSE tool to describe the design space. This interface describes possible range of different design parameters in the design space e.g. what are the clock frequencies of the processors that can be explored, or what is the maximum amount of parallelism (number of threads) available in the application implementations. This interface can provide some additional rules in design space definition to help DSE tool in restricting the design space. These rules can specify combination of sub-optimal (or sometimes unavailable) input parameters which user can easily predict e.g. if we have specified a range of processor clock frequencies as a parameter to explore using a platform simulator, but we guess from application QoS requirements that a certain range of frequencies is not worthwhile to explore, we can specify that with an additional rule for non-usable frequencies of the processor.

Second interface is between DSE tool and the simulator. This has two parts: one from DSE tool to the simulator and the other vice versa. This interface is essentially a list of parameter name-value pairs. This interface defines input parameters that are to be fed to the simulator e.g. it specifies to the simulator clock frequency of each processor running on the platform. This interface also demands each simulation to produce a predefined output format as

expected by DSE tool e.g. a predefined output from the simulation can contain estimated execution time and power consumption which DSE is trying to optimize.

We also envisage third interface between the DSE tool and run-time management handler on the platform. This interface will link DSE tool output (Pareto set of optimal operating points) to a run-time management tool. This interface is still in early phases.

2.1.1.2 Run-Time library (RTLib)

Another key feature of our approach is to keep consistency of application across multiple abstraction levels by using the same code across different simulators at different abstraction levels. Inside the application code, the simulator-dependent part is shielded from the rest by using a Run-Time Library (RTLib) API. We have identified certain set of primitives which can act as an API between simulator dependent part of the application with the rest. These primitives are described further.

This RTLib API is implemented separately on each simulator (i.e. at each abstraction level). This is shown in following two examples.

Take a case of thread spawning by a master processor on a slave processor on an MPSoC platform. In a functional-level simulator, this thread spawning (and joining) can be implemented by making the slave processor to poll at a certain memory location in a shared memory. But on a TLM-based cycle-accurate simulator, the same thread-spawning can be implemented accurately by using a processor-to-processor pre-defined message-passing protocol. Take another case of FIFO-based communication between threads running on different processors. This communication can be implemented accurately on a cycle-accurate TLM platform simulator by modelling it as a transaction over a Network-on-Chip (NoC) between the processors. Whereas for a functional-level simulator, FIFO communication can be implemented using C++ Standard Template Library (STL) based "Queue" primitives.

Every primitive defined in RTLib forms the basis of creating platform-dependent part of the application. Every simulator needs to have its own implementation of RTLib API so that the same application code can be run on any of the simulator by linking with corresponding RTLib API implemented for that simulator. Note that one can use a refinement process for implementation of these primitives for a particular simulator to make it more accurate.

Our RTLib layer consists of following main primitives:

Thread spawning and joining: these primitives implement how a master processor in an embedded system spawns a thread on a slave processor. These primitives include argument passing between master and slave threads.

Thread communication and synchronization: these primitives implement ways for different process communication and synchronization. They include blocking and non-blocking process communication e.g. FIFO based data and control communication between threads. They also include thread synchronization primitives similar to a semaphore.



2.1.1.3 Validations between different simulators

There are various approaches possible for a validation between different simulators and different abstraction levels. As previously described, typical approaches use a refinement procedure to validate across different abstraction levels. We are using a different method which is aimed towards validation of execution times across simulators. This method forms the third key ingredient in our DSE approach.

The main idea is to measure 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 into high-level timed simulations to achieve validation across simulators.

Our approach makes a practical sense for simulating data-dominated applications, especially multimedia applications. There, usually execution time is dominated by execution in various nested loops (also called *kernels*). Moreover these kernel executions are solely dependent on processor architectures and not on any platform-dependent component. By accurately timing all the kernels in the application on a cycle-accurate simulator, we can keep an accurate timing of the application execution. During a simulation on a cycle-accurate simulator, processor cycles needed to execute kernel iteration is measured and stored in a database. Each entry in the database identifies how many processor cycles an individual kernel took for an individual execution (kernels can take variable number of cycles across iterations due to existing control flow, such as if-statements, inside kernels). During the high-level simulations supporting timed annotations to the executions, the same database can be accessed to get the cycle count for a specific kernel at a specific iteration count. Size of such a kernel execution database can be easily managed using data compression techniques e.g. disk space needed for storing all executions of all kernels in our MPEG4 encoder example simulating 300 frames is of the order of few megabytes.

Following application code displays our use of back annotations between simulators:

```
arp_enter_kernel(threadid, loopid, iterid);
for (i=0, i++; i<100)
  for (j=0, j++; j<10)
    if(...)
      {...}
    else
      {...}
arp_leave_kernel(threadid, loopid, iterid);
```

The calls *arp_enter_kernel()* and *arp_leave_kernel()* are implemented differently on each simulator abstraction level. These functions display mainly two behaviours : on a cycle-accurate simulator, they record number of processor cycles it takes to execute the kernel iteration whereas on the high-level timed simulator implementation of the same *arp_** functions playback the stored cycle counts and use them in the timed simulations to advance time. This way there is always a close timing validation between simulators. Note that

recording of kernel timings can also be done on an Instruction Set Simulator (ISS) for the processor, if available.

2.2 Multimedia Use Case Modeling and Experimental Results

This section discusses modeling techniques and experimental results obtained by using multi-simulator approach on MPEG4 encoder using three different simulators: HLSim, M3-SCoPE and CoWare-based simulator.

2.2.1 Use case modeling and experimental results with HLSim and CoWare-based simulators

This section details the use in IMEC of our approach on MULTICUBE multimedia use case, namely MPEG4 encoder, using several simulators interfaced with available DSE tools.

We applied our multi-simulator based DSE approach to a MPEG4 encoder example (Figure 2). To explore parallelism during the design cycle, we have prepared number of multi-threaded versions of the same MPEG4 encoder, threads ranging from two to eight, using an in-house parallelization tool MPA. This tool takes parallelization specifications (which functions or loops to parallelize) as an input along with sequential MPEG4 encoder code to generate parallelized MPEG4 encoder code. This tool also inserts RTLib API automatically in the parallelized code.

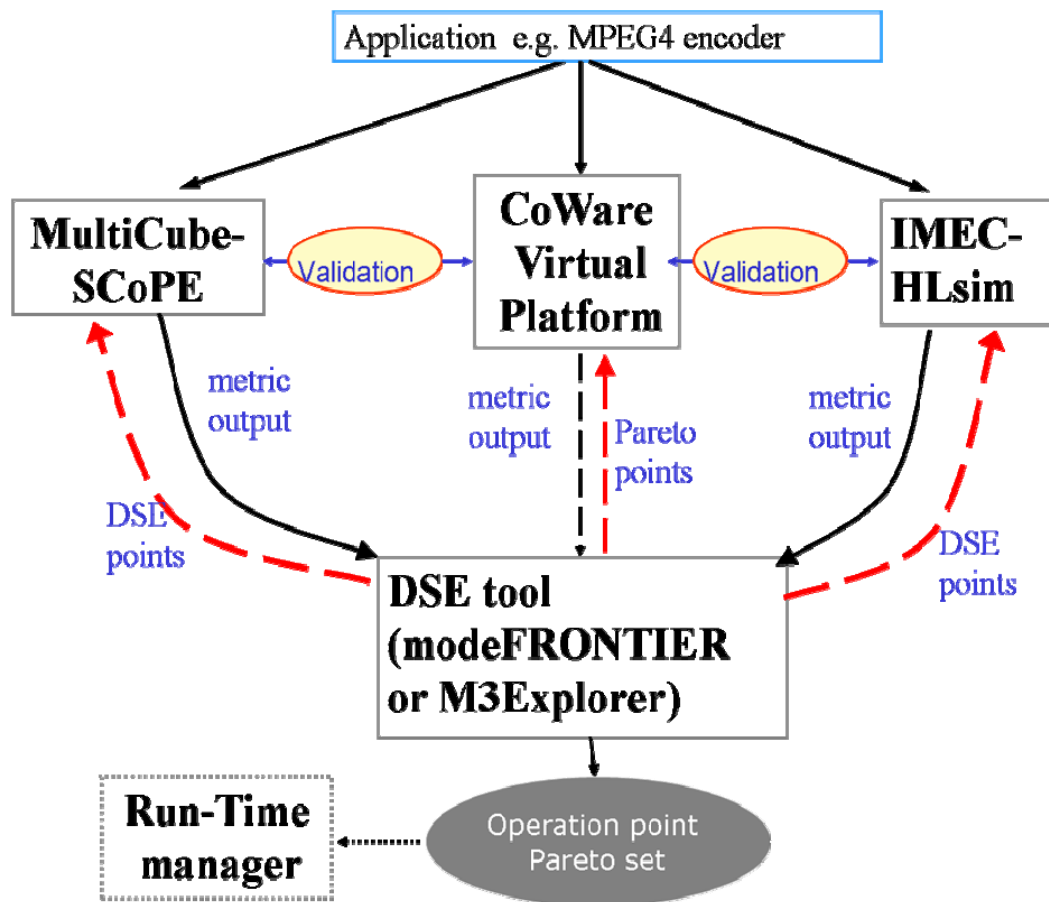


Figure 2 : Using DSE tool(s) with multiple simulators: HLSim, M3SCoPE and TLM-simulator

Our aim is to run DSE to evaluate execution time versus power consumption trade-off. The input parameters that we can explore are different clock frequencies of the processors and a particular parallelization used. We are using two different simulators for this purpose: one high-level timed simulator HLSim and another SystemC-based cycle-accurate Transaction-Level Model, TLM-Simulator built using CoWare virtual platform prototyping tools. These simulators are interfaced with a commercially available DSE tool *modeFrontier*. We have also used other DSE tool *m3explorer* in order to verify DSE results against each other. Note that we are using the same application code to map on two different platform simulators by using different compilers and linking with different RTLlib implementations.

Time-annotated High-level simulator HLSim that we are using is more than just a functional-level simulator:

- It can annotate timing information during the execution, i.e. it has an in-built notion of simulated time
- It has high-level models of many of the platform components e.g. a Network-on-Chip (NoC) is modelled at a high level as a full cross-bar switch (with a fixed delay between any two nodes and without modelling any collisions in the NoC)
- By having a timed simulation, it can measure execution time for MPEG4 encoder and also at the same time by looking at the resource consumption it can estimate power consumption of the simulation run by using power models for memories and processors.

On the other hand, TLM-simulator ,built using CoWare tools, has a SystemC-based cycle-accurate Transaction-Level Model of the platform containing multiple processor cores connected with a NoC as an interconnect (Figure 3). MPEG4 application code is mapped on this platform by using a cross-compiler for the processor and by linking it with RTLlib implementation of the platform. At the end of simulation, along with standard XML simulation output, TLM-simulator spits out many important statistics (e.g. memory access counts, processor stall cycles) which can be used to validate with HLSim.

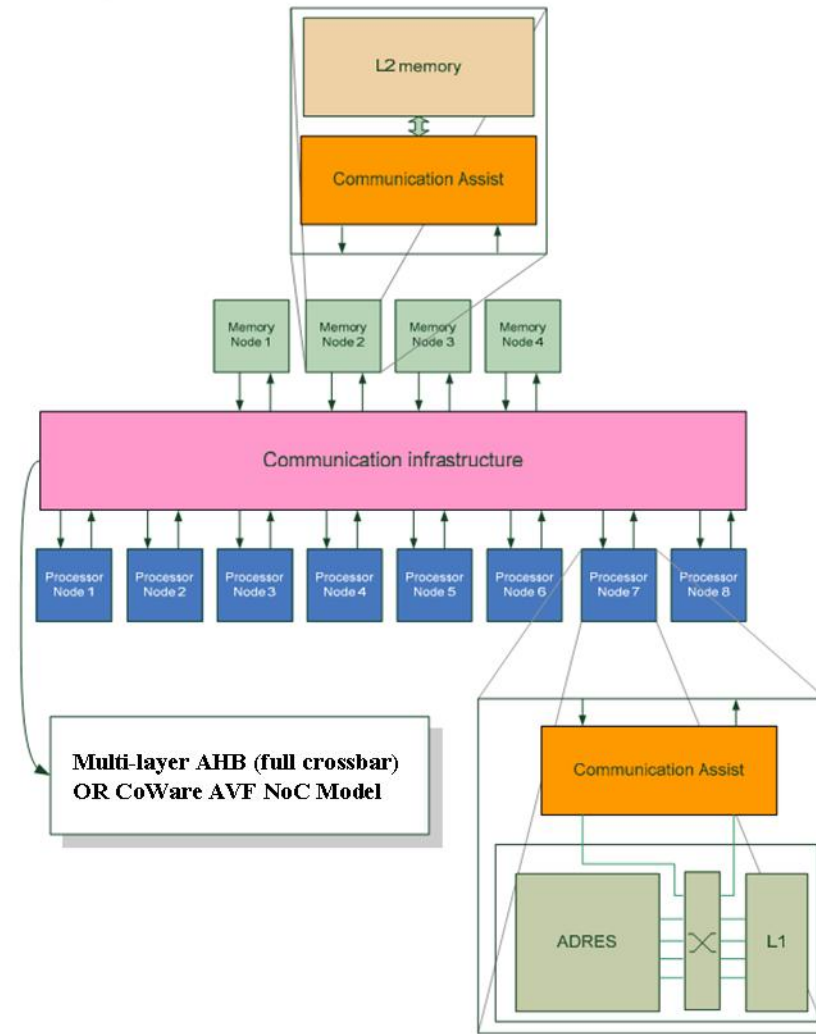


Figure 3 : Cycle-accurate TLM simulator of the platform built using CoWare tools

Note that it takes around four hours for TLM-simulator to simulate MPEG4 encoder encoding 10 frames whereas HLSim takes 90 seconds to achieve the same. In our design exploration phase, we first ran the application on TLM-simulator to extract cycle counts for computation-intensive inner loops. These cycle counts are then fed to HLSim simulations to reach accurate execution time estimations. Some of these estimations can be validated by running exactly same input parameters on both simulators. DSE tool modeFrontier fires extensive number of simulations using HLSim and generates a Pareto curve (Figure 4). Interesting regions pointed out by this Pareto curve are then further explored (and verified) using TLM-simulator with modeFRONTIER.



Figure 4 : Pareto graph generated by modeFRONTIER using HLSim with processor clock frequencies between 20 and 200MHz

After initial run on TLM-simulator to extract timing information of application execution, it takes modeFRONTIER only couple of hours to explore along hundreds of design points. The most interested regions of Pareto points are then further explored at a great detail by using TLM-simulator with exactly the same setup of DSE tool and application code. In our HLSim based explorations we had found that six-threaded parallelisation is the most desirable parallelisation in terms of execution and power consumption.

Hence we explored (and validated) six-threaded parallelisation using TLM-simulator by exploring for clock frequencies much beyond the range used in HLSim (and also some other platform configurable parameters). This TLM-simulator based DSE helps in two ways: it focuses on the most interesting region and also it validates HLSim-based simulation results. Figure 5 shows the result of that exploration in terms of Pareto set generated by M3Explorer tool.

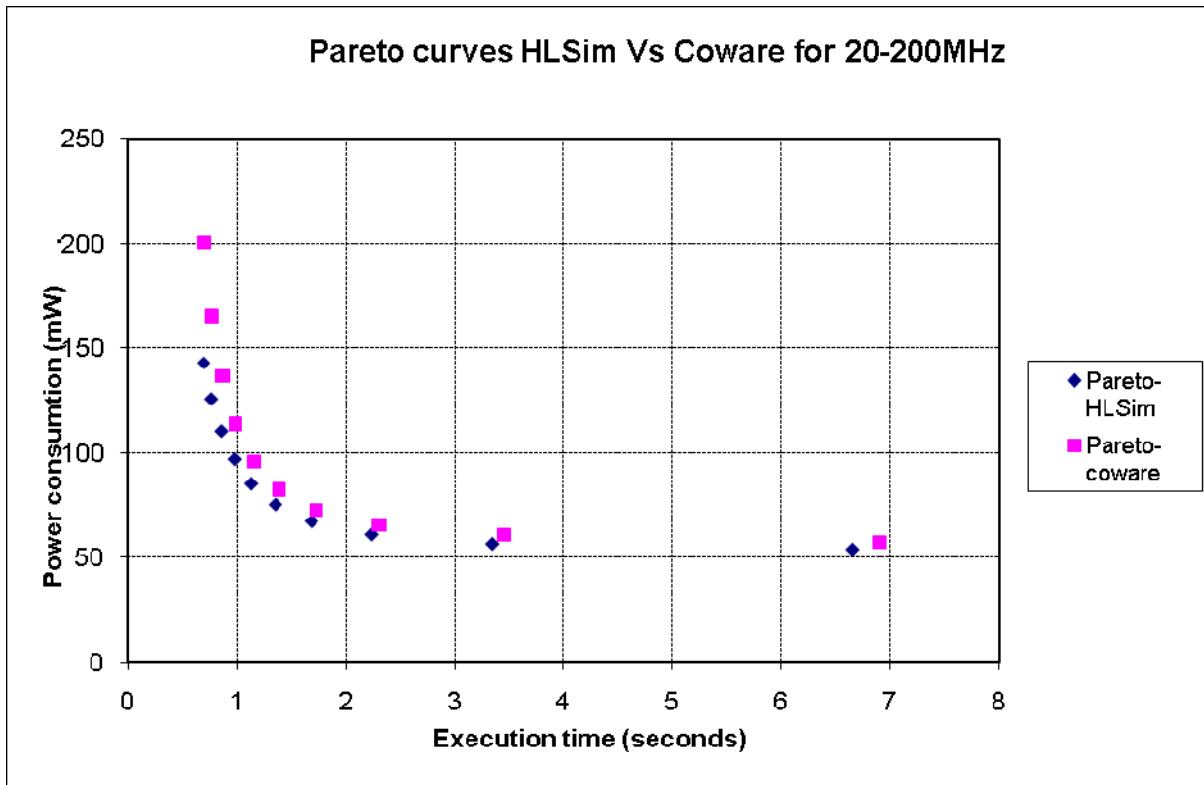


Figure 5 : DSE Output plot after m3explorer integration with TLM model (generated by m3explorer by using clock frequencies between 20 and 500MHz)

Initially when one does such an exercise to run simulations using CoWare simulator, one can find some difference between experimental results of the two simulators. This is where the validation part becomes very important one. This validation will involve back-annotating TLM-simulator based execution time figures into HLSim-based one. From the figures above (Figure 4 and Figure 5) one can see that after doing such an back-annotating exercise, both simulator results converge nicely.

In practice, switch between HLSim and TLM-simulator during DSE step can happen repeatedly, depending on amount of time available for DSE. But with our approach, user can reach DSE results more accurately in a considerably less amount of time.

2.2.2 Use Case Modeling and experimental results with M3-SCoPE

Among all three simulators used to model the Multimedia use case, M3-SCoPE is the tool developed by UC that provides an intermediate balance in terms of speed/accuracy. Additionally, it is important to note that it is the only one that has not been specifically developed for modeling this use case. M3-SCoPE is a generic simulator capable of modeling a wide range of system architectures and designs.

SCoPE infrastructure is based on a native co-simulation technique. The system is simulated combining native execution of annotated SW code with loosely timed TLM models of the HW components. As a consequence, the main difference with respect to other TLM modeling technologies is that there is no a processor model, as an Instruction Set Simulator (ISS), or just-in-time binary translation, like in Qemu.

SW modeling and its interaction with the HW platform model is based on source-code annotation and the use of a RTOS model. The annotation extends the original functional SW code with information about the time, power consumption and cache misses required to execute each basic block of SW in the target platform. During execution, the kernel of the RTOS model controls the overall simulation, interleaving the execution of the application SW and the SystemC HW platform. To do so, it manages all the information annotated in the SW in order to adequately model the time advance and the bus traffic. Additionally, the RTOS model provides all the functionality required for SW/SW and SW/HW communication, automatically performing the required access to the HW platform model.

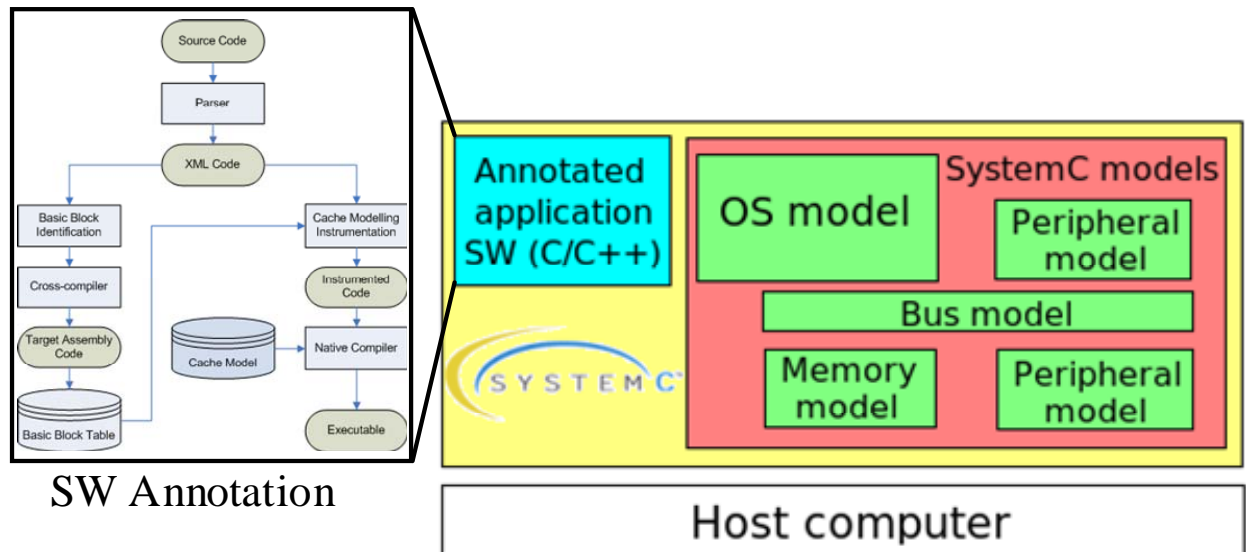


Figure 6. Basic structure of native co-simulation

In the Multimedia use case, there is not a real operating system, as there is only one thread per processor. The SW infrastructure is limited to some functions for data transfer among tasks and between memory and scratch-pad memories. Thus, to make compatible the original SW code and M3-SCoPE it is required a layer to transform the original interface (MPS) to POSIX. As a consequence, a MPS2POSIX library has been developed by IMEC.

Nevertheless, a RTOS model is indispensable for SW modeling. It has been supposed that the eight processor nodes make up a symmetric multi-processor system (SMP). Thus, a single OS is instantiated, controlling all the processors. This solution enables the use of inter-processor communication (IPC) mechanisms for SW/SW communication. Additionally, it enables automatic exploration of fixed task allocation and dynamic task re-allocation by the OS. Nevertheless, these options have not been considered in the use case definition, as they are not supported by the other two simulators, and subsequently results of these options are not included in the present document.

About performance modeling of the SW code, most of the annotations added by the parser before compilation have been disabled. The parser transforms the SW code in three ways:

- Transform the system calls in order to call the OS model functions instead of the host OS. To do so, the “uc_” prefix is added to all these functions.
- Add SW execution times.
- Add cache miss calls.

The ADRES processor included in the platform has no caches. A scratch-pad and a small memory are used to store data and instructions respectively. Thus, no cache miss annotation is required. Additionally, the automatic SW time estimation system of SCoPE has been disabled. Instead, the ARP library from IMEC provides the estimated time for each code segment. A pair of SW functions has been developed to interconnect M3-SCoPE and the ARP library. Thus, only the first transformation of the parser is required.

It is important to note that, although the real system has no caches and no miss annotation has been performed, in the platform model all processors have caches associated, to model the power consumption of the local memories (static power and processor accesses).

Finally, in order to enable exploring the 6 different codes for the different multimedia parallelizations, dynamic libraries have been used. The code for each parallelization is encapsulated in a different “.so” file, opening the adequate one depending on the configuration selected by the DSE tool (i. e. M3explorer) for the experiment.

The modeling of the Multimedia use case with M3-SCoPE is part of the open-source demonstrator of the MULTICUBE project. Thus, a complete description can be found in D4.2.3.

2.2.2.1 Results

Results shows a deviation with respect to the cycle-accurate model developed by IMEC of a 2.25% maximum error in time estimation and 5,77% in power consumption. Considering that the tool is a generic tool, not specifically developed for that use case, and that the knowledge about the platform internal details that the University of Cantabria has is much lower that the information managed by IMEC in order to create the other two models, the error rate demonstrate a very high accuracy and validate the use of M3-SCoPE for fast modeling of similar electronic systems.

Parameters		M3-SCoPE		CoWare		Deviation	
freq	threads	Time (seg)	Power (mW)	Time	Power	Time error %	Power error %
20	6	6,75	55,76	6,9	57,29	2,25	2,68
40	6	3,39	58,33	3,46	60,66	2,02	3,84
60	6	2,26	62,63	2,31	65,48	1,79	4,36
80	6	1,71	69,43	1,73	72,47	1,61	4,2
100	6	1,37	79,81	1,39	82,36	1,28	3,09
120	6	1,15	94,4	1,16	95,86	1,16	1,52
140	6	0,99	114,3	0,99	113,72	0,79	0,51
160	6	0,87	140,01	0,87	136,62	0,61	2,48
180	6	0,77	172,35	0,78	165,28	0,55	4,28
200	6	0,7	212,02	0,7	200,46	0,09	5,77

Regarding the simulation speed, the execution of the different configurations shows that the simulation of each use case with M3-SCoPE takes about one minute of host time.

3 Automatic Design Space Exploration Assessment Procedure

As described in **D4.1.2 “Final Evaluation Plan”**, the general procedure to assess the results 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 design target, but also the impact of such a technology on the entire design process within the corporate. 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 automatic DSE assessment procedure (already described in **D4.1.2**) has been reported in this section to make the present deliverable self-contained.

The specific design process activities can be analyzed and classified in order 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 phase:

- **Model Setup:** preparation of an initial model of the virtual platform.
- **Simulation:** the simulation of the executable model with a single configuration of parameters is carried out.
- **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 7 shows a scheme that represents 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 human operator is informed about it and handles the simulation tools to get the results. Note that this idle time can be very short if the designer is informed about the end of the simulation immediately or can be significant if the designer is not on duty.

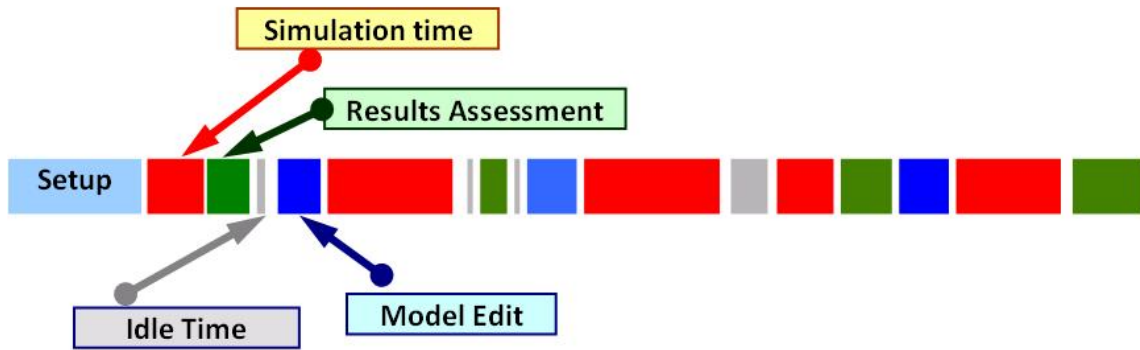


Figure 7: Manual exploration procedure

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.
- **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 designs database.

Figure 8 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. Note that 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.

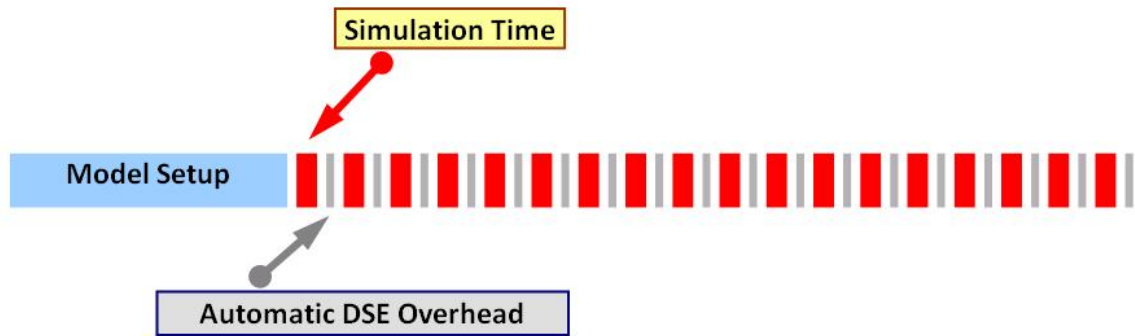


Figure 8: Automatic exploration procedure

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 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 man-day (10 hours), the designer can run at most 70 designs. In the same problem, experience shows that an automatic approach can handle something like 600 designs per hour, which means about 14400 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 structured database. The designer, not only will not be stuck on repetitive operations, but can focus his/her attention (and profit from his 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.

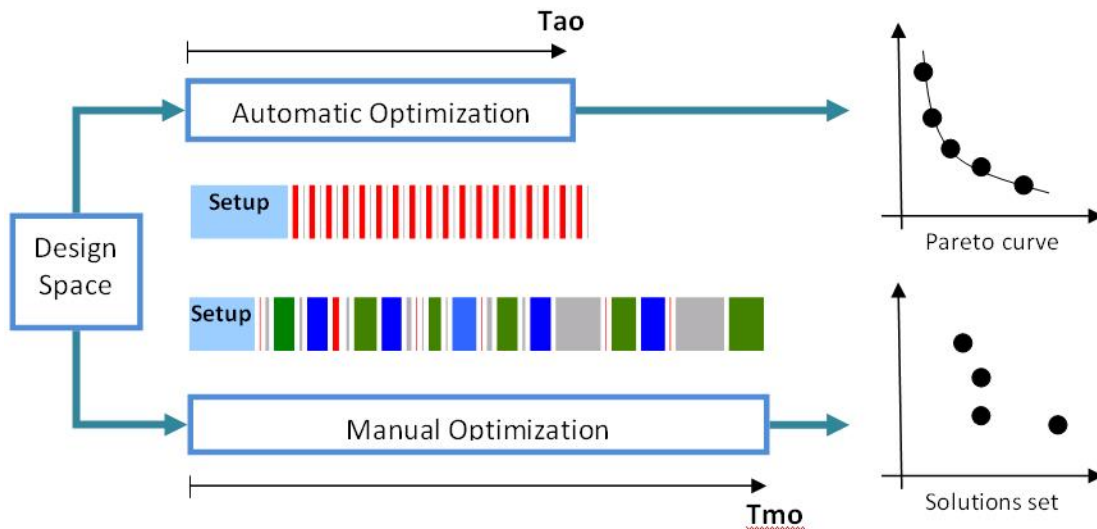


Figure 9: Comparison between manual and automatic optimization

Figure 9 presents a comparison between manual and automatic optimization. The global lead time (**T_{mo}**) 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 (**T_{ao}**) in the automatic procedure case is only given 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. Other difference concerns the type of results produced by both types of optimization: the automatic optimization produces a set of designs that are likely to belong to the Pareto front, while the manual optimization produces 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 procedure that has just been presented will be the basis for the validation of each of the use cases of the project. In order to apply this procedure, the responsible for each of the use cases will adapt however the procedure to the specificities of its particular application and context but without modifying the aim and the general steps described above.

3.1 **DS2 Automatic DSE Assessment Procedure**

3.1.1 **Description of DS2 Automatic DSE Assessment Procedure**

Design of Systems on Silicon (DS2) is a fabless silicon design house and one of the leading suppliers of silicon and software for Power line Communications (PLC). Founded in 1998, DS2 produced first time working silicon and is currently marketing chipsets with performance rates of up to 200Mbps. DS2 technology converts the existing electricity grid and domestic power lines into a high-speed communications network supporting voice, video and data services, at low cost. DS2 technology not only delivers high performance it is also flexible and efficient supporting co-existence between the PLC access and LAN markets.

PLC stands for “*Powerline Communication*”, an advanced telecommunications system allowing the fast and reliable transfer of audio, video and data information using the most ubiquitous transmission system: the power lines. This transmission line is used to exchange information between different equipment connected to the network using the most advanced coding techniques like OFDM (“*Orthogonal Frequency Division Multiplexing*”).

In order to stress the different procedures and methodologies proposed in MULTICUBE project, DS2 has proposed to optimize a powerline system based on the new ITU-T G.hn standard ratified early 2010. This way, the high level model of an ITU G.hn system has been used as test case. This test case models a network of several powerline nodes connected through a powerline channel model. The model of PLC nodes include the different layers that are described in the standard, focusing specially on the MAC layer, where the main design decisions have to be taken.

The model is adapted to the future hardware platform that will be used in the real platform-based ASIC implementation where an embedded microprocessor will run a PLC SW stack code that will make use of different hardware resources, accessible through a bus system. A simplified (with only two nodes) high level description of such a platform is shown in the following Figure 14.

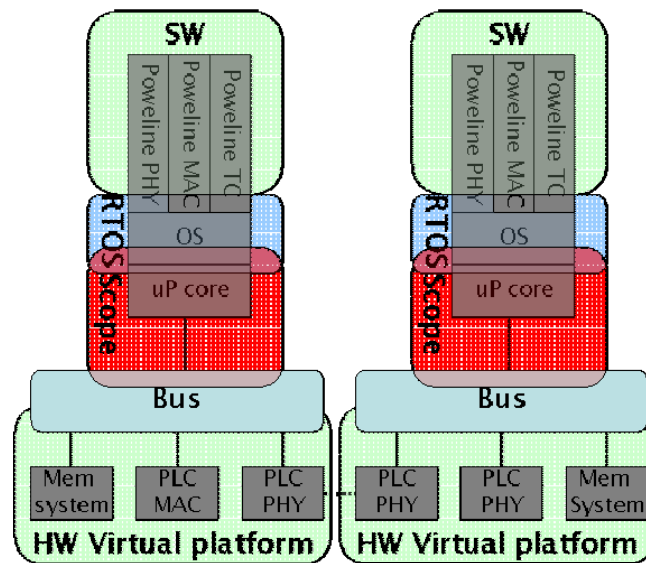


Figure 10: PLC network model high level description

This high level description of a Powerline network shows two PLC nodes connected through a physical electrical connection (dotted line).

Each of the nodes is the high level model of a platform-oriented System-on-Chip. The architecture of these nodes is quite classical: around a bus subsystem, we plug the necessary components (both hardware and software) to perform the communication and the management of the system. These components are usually event-driven modules that are used to monitor and control the powerline communication protocols, the interfaces towards the user and memories and the Quality-of-Service of the overall link.

At a hardware level, we find in the platform usual peripherals as memories, I/Os, bus controllers together with different PLC-specific IP blocks, which are included in hardware due to their high speed necessities or their complexity.

At a software level, we can find a microprocessor system (which, in the high level platform is still at an abstract phase) with a simple real time operating system. In top of this operating system, we run a PLC-Specific code together with more generic software stacks (IP stack, etc...). It is interesting to note that the C code included in the platform is almost the final application software that will run in the final ASIC.

While other use cases of the project are focused on applications that look for the highest computing power with the available resources, powerline use case is centered in optimizing a medium-end platform that run a control-oriented complex protocol. The overall goal is to optimize the cost of the overall solution while maintaining the necessary performances for the specific application but without trying to increase as much as possible the platform processing power.

The main differences with the other use cases presented in MULTICUBE project can be summarized as follows:

Particularity	Effect on the analysis
The platform is less computing-intensive than the other use cases defined within the project	The metrics to analyze are different from the other use case
The focus of the optimization phase is put on cost-effectiveness of the final solution and not really on its final performance	The metrics are more difficult to measure since they can rely on subjective analysis
The structure of the use case is data flow oriented. The optimization is done over the control of this data flow	The simulation results rely strongly on the data flow contents and behavior
Many parameters of the platform are fixed by the standard or by legacy hardware	Less freedom to choose some of the parameters
The optimization procedure was already semi-automatic in past designs	The evaluation procedure is different from other use cases.

Table 1: Powerline use case main particularities

These differences had an effect on the generic assessment procedure described in Section 3, therefore we have defined an adapted process that will be described in this chapter.

As a reminder, we can show the following diagram, described in detail in deliverable D4.2.1: “Demonstrator of use cases: Powerline application” where we presented the different flows between the main actors in the optimization process

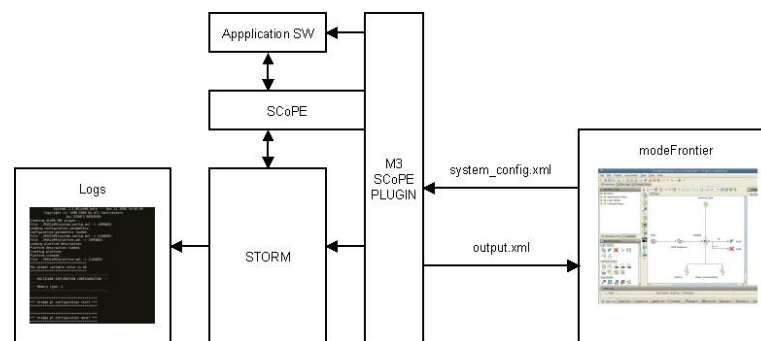


Figure 11: Powerline use case process flows

The use case goal is to show how we have applied the different methodologies and tools crested in the project to the specificities of the use case. For that, it is interesting to show the traditional semi-automatic procedure used within the company and the new, updated procedure incorporating MULTICUBE processes. Both processes are shown in the following Figure 12 and explained in detail in next paragraphs.

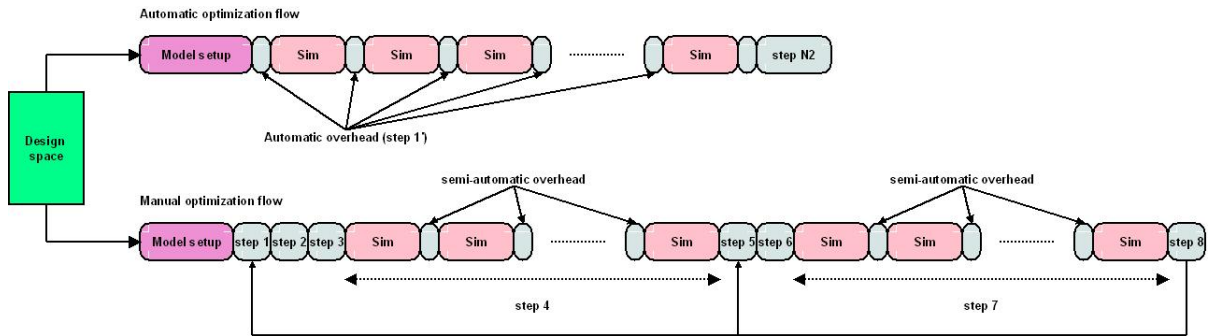


Figure 12: Assessment procedures updated for DS2 Powerline use case

3.1.1.1 DS2 Semi-automatic procedure (or “traditional procedure”)

In order to optimize the different chipsets DS2 has designed in the last years, a semi-automated procedure was implemented. The procedure is possible thanks to the high flexibility and configurability of STORM platform, which is based on the analysis of command files in ASCII format. In a traditional optimization flow, many simulations are run in a sequential way. Each of these simulations is parameterized through the abovementioned ASCII command files.

Next table shows each of the steps in the process, together with a brief description of the actions taken in each of them.

STEP	Description
Step 0	Build a configurable platform using ASCII files (DS2 proprietary language) to configure the platform
Step 1	Select a subset of 2-3 parameters and 2-3 metrics among all the possibilities offered by the platform
Step 2	For the non-selected metrics, fix the rules of what are the accepted boundaries. For the non-selected parameters, fix the configuration that will be used through the first steps of the process
Step 3	Write automation scripts (usually in perl) to generate all selected combinations and to filter the requested metrics from the output files and facilitate manual inspection
Step 4	Run simulations for all possible selected combinations with the automated script and automatically extract metrics from the simulation results. Build a database.
Step 5	Manual inspection of the metric values in the database. Depending on these values, select a set of the parameter combinations that seems to be the better adapted the problem. Selection is done mainly through designer's past experiences
Step 6	Generate new platforms views with the selected combination of parameters and using corner cases for the non-selected parameters (based on designer's past experiences)
Step 7	Run simulations with the new set of platforms. This process is also partially automated through the use of Perl scripts
Step 8	<p>Verify that the ALL the metrics in each of the simulations of step 8 meet the rules specified in step 2.</p> <p>If not, investigate causes. If the cause can be corrected by changing one of the selected parameters select a new combination in step 5.</p> <p>If the cause cannot be corrected by changing one of the selected parameters Change the selected combination of parameters (include new ones) and come back to step 1</p>

Table 2: DS2 Semi-automated procedure

3.1.1.2 DS2 Automatic procedure (MULTICUBE flow)

The automatic procedure proposed by MULTICUBE project is highly automated through the use of standard interfaces that are understood by the different developed tools. Then, it is normal that the number of steps is also highly reduced.

STEP	Description
Step 0	Build a configurable platform using ASCII files (DS2 proprietary language) to configure the platform. Prepare the platform to overwrite some of these configurations by MULTICUBE own tools. In this step we also consider the time for configuring modeFRONTIER and generating the design experiments
Step 1'	Automated parameter update procedure and metric extraction using MULTICUBE interfaces and tools
Step 2'	Final analysis of the solutions in order to select the optimized combination of parameters

Table 3: MULTICUBE automated procedure

3.1.2 Comparison Results

The two procedures proposed in the precedent chapter have been applied to the platform in order to optimize a set of parameters as described in **D4.2.1**. The main results of this comparison are presented in this chapter where we have divided the results in two main groups: *objective assessment and subjective assessment*.

3.1.2.1 DS2 Objective Assessment

From an objective point of view, after the assessment of the procedure proposed in MULTICUBE project for system optimization, the overall conclusion is that *the proposed flow can save up to a 80% of designer time while achieving better results in terms of performance since much more simulations can be run and analyzed following this flow obtaining ten times more possible combinations than with the semi-automatic design flow*. Also, with just one run we can already identify the optimum for the design parameters. A second or third iteration would fine tune the design in order to obtain the final parameters.

As an example, we have run a test case where we have tried to optimize the parameters described in **D4.2.1** with the following data:

- Number of parameters: 5 parameters
- Number of metrics: 8 metrics
- Simulation real time: 0,5 seconds (real time)
- Simulation time: 1,5 minutes

- Optimization-time using automated procedure: 14,88 hours
- Number of simulations in semi-automated procedure: 45 simulations

Under these conditions, we have estimated and reported in Figure 13 the time required by each step composing the semi-automated (left side) and automated (right side) procedures as well as the total time spent by both procedures.

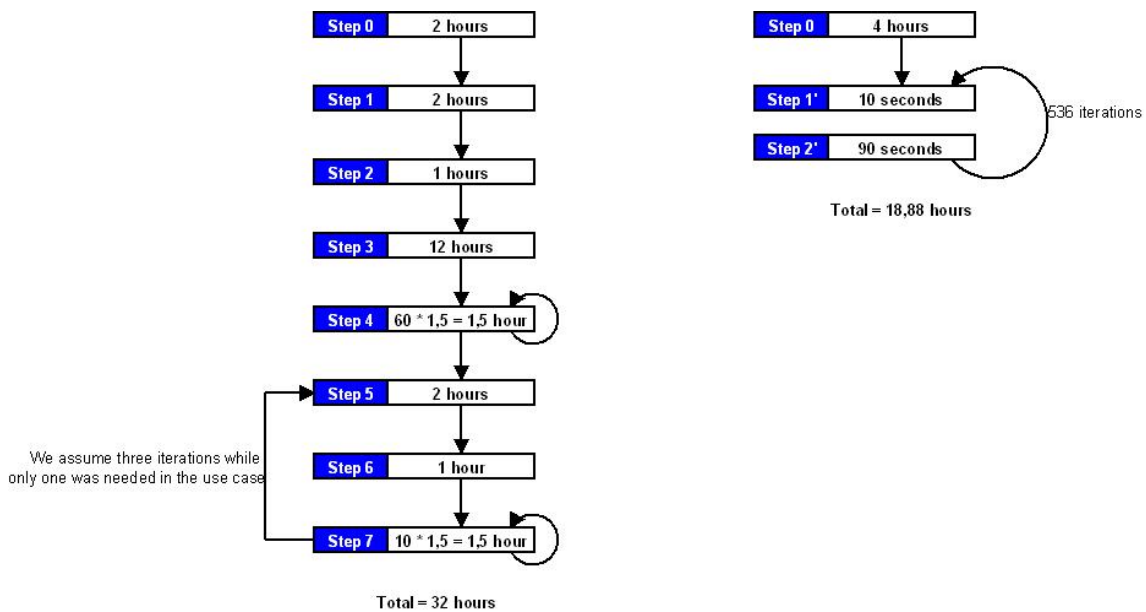


Figure 13: Assessment procedures comparison (semi-automated vs automated)

As we can clearly see from Figure 13, the automated procedure provides:

- **Higher number of design points:** The number of design points that can be analyzed by following MULTICUBE methodology is much higher than the ones obtained using the semi-automated procedure
- **Better quality of the results:** The quality of the results obtained through the use of the automated procedure is higher since modeFRONTIER guarantees the correct generation of experiments in order to maximize/minimize the metrics following users' guidelines.
- **Less complexity of the flow:** The complexity of the flow is much less in the case of the automated procedure. The number of steps to be accomplished is much lower. In this sense, it is interesting to mention that a fewer number of steps represents a lower possibility of errors.
- **Faster convergence times:** The convergence towards the optimum point is achieved faster with the automated procedure than with the semi-automated procedure: an overall speed up of 1.69 has been measured (32 hours required by the semi-automatic optimization procedure compared with 18.88 hours required by the automatic one).

3.1.2.1.1 DS2 Subjective Assessment

In parallel with the objective assessment presented in the precedent paragraph, we can mention some subjective facts that have been identified by the engineers that have run the design flow in the DS2 use case. The main conclusions of this subjective analysis can be:

MULTICUBE design flow is also a perfect tool to guarantee the coherence and correctness of the design as we have seen in our study. The analytical capacities of modeFRONTIER allow to stress the model and to analyze its behavior under different running conditions allowing investigating if the model behaves as expected since we can analyze a larger number of operating conditions. As a simulator tool alone, it adds value to the current simulation methodologies in place.

We can mention that the procedure proposed by MULTICUBE makes the work of the embedded designer much easier by proposing an excellent front-end tool and by automating all the processes necessary to update the platform in each simulation cycle. The intervention of the designer is reduced to a minimum, minimizing also the possibility of human errors.

MULTICUBE procedure benefits also of the use of standardized interfaces. It is possible then to envision the usage of other analysis tools quite easily. In addition to powerful analysis tools from commercial suites like modeFRONTIER, that have been proven in other technical domains, we can use other suites developed internally in the company or to adapt it to future tools.

The overall conclusion of the utilization of MULTICUBE design flow over a real industrial use case like the one just discussed is that the added value overcomes largely the cost of setting up the whole process, converging much faster to the optimum solution for a given technical problem.



3.2 STM Automatic DSE Assessment Procedure

The STM use case goal is to show how the different methodologies and tools developed in the MULTICUBE project can be applied in a common industrial case study and to evaluate possible benefits and/or limitations. STM has proposed as an advanced computing platform a low power processor for generic computing applications as specified in D1.3. The STM SP2 processor is designed as a low power and high performance microprocessor offering comparable performance to entry-level desktop microprocessors; in Figure 14 the block diagram of the processor is shown. It is designed for generic applications and mobile applications that need low power dissipation and high peak performance. With the presence of SIMD coprocessor, it will also well support multimedia applications. The deliverable sp2sim is the cycle accurate SP2 simulator with full privilege architecture resource support.

The applications come from SPEC CPU2000 benchmark suite, which is offered and owned by Standard Performance Evaluation Corporation®. [8] Before year 2006, SPEC CPU2000 was the standard benchmarking suite for commercial desktop and server level microprocessors. For microprocessor designers, it offered a quantitative result comparable to the mainstream microprocessors 3-10 years ago.

SP2 microprocessor is compatible with MIPS Technologies Inc's MIPSISA32® Release2 Instruction Set Architecture (ISA) and Privilege Resource Architecture (PRA) [9]. Sp2sim is SP2's register transfer level cycle accurate simulator, which precisely runs its applications according to SP2's pipelines and configurations.

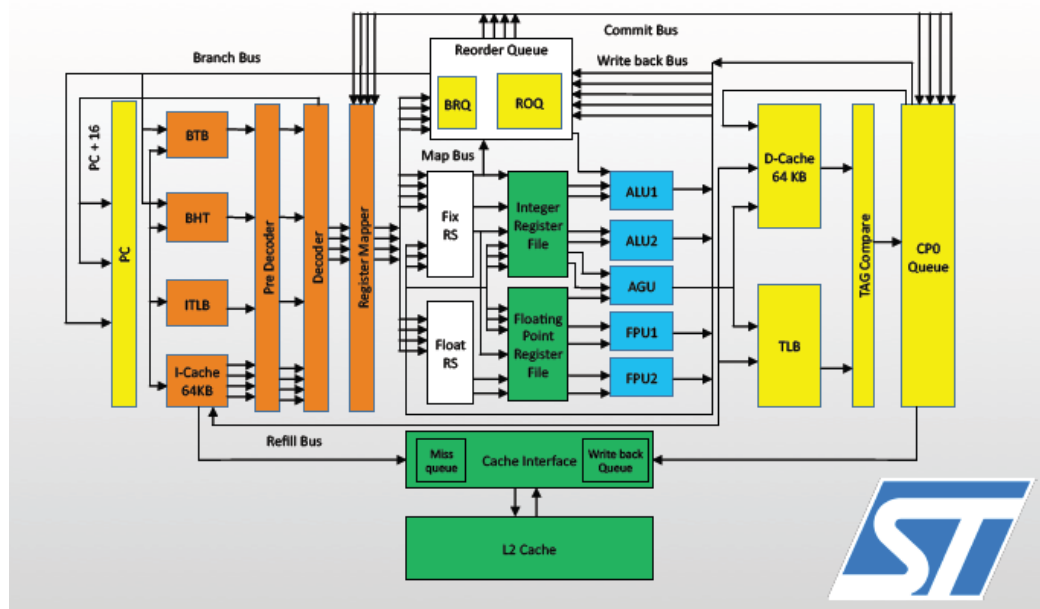


Figure 14: STM SP2 architectural block diagram

The SP2 main architectural features are listed below:

- MIPSISA32 release2 compatible instruction set architecture and privilege resource architecture
- 4-issue superscalar out-of-order execution

- Built-in 2 integer ALUs, 2 interleaved load-store units
- Split primary instruction and data cache
- 64-byte cache line, 4-way set associative primary caches
- 64-byte cache line, 8-way set associative unified exclusive secondary cache
- Configurable out-of-order parameters: renaming register number, instruction issue window width, reorder buffer depth etc
- Configurable cache parameters: cache size
- Runtime resizable secondary cache

The design space of this use case consists of 11 configuration parameters and 7 system metrics. In Table 4, the configuration parameters are summarized, while some design rules exist among them generating an overall design space composed of 1,161,216 points to be explored. The system metrics provided by Sp2sim are summarized in Table 5.

category	parameter	description	values
out of order execution	rob_depth	reorder buffer depth	32, 48, 64, 80, 96, 112, 128
	mreg_cnt	rename register number	16, 32, 48, 64
	iw_width	instruction window width	4, 8, 16, 24, 32
cache system	icache_size	instruction cache size	16, 32, 64
	dcache_size	data cache size	16, 32, 64
	s-cache_size	secondary cache size	0, 256, 512, 1024
	lq_size	load queue size	16, 24, 32
	sq_size	store queue size	16, 24, 32
	mshr_size	miss holding register size	4, 8
branch prediction	bht_size	branch history table size	512, 1024, 2048, 4096
	btb_size	branch target buffer size	16, 32, 64, 128

Table 4: STM SP2 configuration parameters

category	metric	description
performance	total_cycle	total cycle number
	total_instr	total instruction number
	IPC	instruction per cycle
power dissipation	total_energy	total energy consumed
	power_dissipation	average power dissipation
	peak_power_dissipation	peak power dissipation
area occupation	area	area occupied

Table 5: STM SP2 system metrics

Low power processor use case is used to explore one of the main trade-off in nowadays SoC design: performance-power trade-off. The use case is a well know architecture, therefore a Design Space Exploration (manual or automatic) should not lead to surprising results: this lead to STM to monitor the outcome from MULTICUBE design flow with confidence.

The main difference with the other use cases presented in MULTICUBE project is the huge dimension of the design space proposed; this aspect reflects actual SoC designs, where having so many possibilities is not so unusual. A good reference in STM product line could be the members of the STM32 family products. All STM32 devices are built with the same architecture (i.e. same model) and differ by memory sizes, CPU frequency, number of standard peripherals (GPIOs, timers, spi, usart) and the inclusion or not of application specific peripheral (like USB, CAN and Ethernet) in order to meet different constraints and performances. It's easy to imagine how many parameter a designer should configure to build a new member for this family.

Another difference is the fact that sp2sim model keeps the same actions with the HDL level RTL simulation; this lead to a simulation time in the order of hours, not minutes. Again this choice was driven by the purpose to reflect STM state of the art design process. In STM having high abstraction level model for a SoC is not so common and when it is present is often build after design choices are taken, with the purpose to provide a Virtual Platforms for embedded SW development and HW/SW Co-verification of the SoC. Often a design is build following a platform based approach, where integration of pre-existing IPs is a big part of the system, and the most complete IP models portfolio is in RTL. These considerations bring the fact that a methodology used in other use cases, i.e. having a TLM model, is not followed in this use case. A significant simulation time can also reflect situations where the system should be configured for different operating modes. A simple example can be a high level ISS that is tested with a specific benchmark, i.e. a set of different application; even if the simulations time for a single simple application is irrelevant, when you consider complex scenario the simulation times can grow up significantly.

3.2.1 Description of STM Automatic DSE Assessment Procedure

To better understand the benefits of using MULTICUBE design flow in an industrial practice, STM was willing to compare the traditional procedure used internally in STM with respect to the automatic DSE procedure proposed by MULTICUBE project.

3.2.1.1 Traditional Internal STM procedure

A generic architecture like a processor doesn't lead a single goal, even if it can be multi-objective. For a single IP it can be much useful to find different configurations that can meet different needs. As general rule, we try to figure out different scenarios, where the performance-power tradeoffs can have different priority.

The design space of the problems as defined in [1] consists of 1,161,216 design points. With a simulation time of approximately 2 hours for each design point, the time required to evaluate all these designs would be too large to be considered as a feasible option. Usually designers' experience can help to reduce the design space by choosing some main parameters that most influenced the system metrics for a first, fast design exploration. A number of possible good candidates are chosen; on this candidates a second phase of exploration is performed based on the remaining parameters that are considered affected the objective at a secondary level, for a refinement of the configuration.

The formalization of the traditional procedure described here just before and applied to the STM use cases defined in [1] with $N_p = 11$ parameters, is listed below:

1. Starting from the design space, identify the set of principal parameters (the number of the parameters chosen is referred as N_1) and 2/3 metrics among all the possibilities to perform a full search on those parameters. Fix also a proper configuration for the remaining parameters that will be used through the first phase of the exploration. The criteria are totally based on designers' past experience.

2. Write a script to launch the $M_1 = \prod_{i=1}^{N_1} p_i$ simulations and collect from every simulation the output in a proper file.

3. Run the M_1 simulation by executing the script.

4. Write a script that collects in a database the data (configuration parameters and system metrics) from the M_1 output simulation files and run it to collect the data.

5. Analyze the metric values in the database and identify B candidates that better satisfy the objectives of the exploration. Selection is done mainly through designer's experiences; therefore the number B can be unknown until this step is performed.

6. Write a script (or modify the script written at Step 2) to launch $B \times M_2$ simulations where

$$M_2 = \prod_{i=1}^{N_p - N_1} p_i$$

7. Run the script

8. Write a script to collect output data from the M_2 simulations and/or reuse the script from Step 3.

9. Analyze the metric values in the database and identify final best candidates

In particular, considering the customization of this *traditional procedure* on the Low-Power processor, the first set of parameters that are used for exploration are: `icache_size`,



`dcache_size`, `scache_size`, `bht_size`, `btb_size`, that lead a first set of the design space of 576 points. After this first exploration 3 points have been chosen with three different criteria: one for high performance, the other for lowest power and the first the best candidates for the trade off. The second exploration phase will perform a full factorial on this three configurations for the remaining input parameters (`rob_depth`, `rmreg_cnt`, `iw_width`, `lq_size`, `sq_size`, `mshr_size`) for a total of 7560 points.

3.2.1.2 MULTICUBE automatic procedure

Compared to the above described traditional internal STM procedure, the application of the automatic MULTICUBE procedure requires a fewer number of steps, thus reducing also the designer intervention. In particular these are the steps for the MULTICUBE design flow applied to STM use case:

1. Write a script to interface the sp2sim simulator to the automatic DSE tool according to the XML format specified in **D1.4.1** and configure the automatic DSE tool to generate the required optimization strategy; the objectives are the same as the traditional procedure.
2. Run the automatic DSE tool that automatically launches N simulations, where N is less than $(M_1 + B \times M_2)$.
3. Analyze the metric values in the database and identify final best candidates.

3.2.2 Comparison Results

To evaluate the effectiveness of the work done in the MULTICUBE project, the two different procedures have been applied to the use case, as highlighted in Figure 15.

The traditional STM internal procedure, where a subset of the design space is chosen based on the designers' experience.

The MULTICUBE DSE optimization algorithm procedure, where the huge dimension of the design space is faced by the adoption of an optimization algorithm, visiting a number of points reduced with respect to the traditional procedure.

In the automatic DSE procedure, the optimization step (*STEP 2*) is done by M3Explorer tool by using the NSGA-II algorithm. The choice to use the NSGA-II algorithm has been motivated by the fact that, as shown in **D3.2**, it offers the best accuracy for the target class of problems. The number of simulations to run automatically by M3Explorer at STEP 2 has been set to 1000 because of the Pareto set obtained is comparable to the Pareto set obtained by using the traditional procedure (see Figure 17). In both cases we consider as objective of the exploration the minimizations of two system metrics: `total_cycle` and `power_dissipation`.

A design space reduction of 1 to 8 has been reached for this example where 1000 design points compared to approximately 8000 design points have been simulated to get comparable Pareto sets.

The simulations have been run for both procedures in the STM computer farm machines, where up to 10 simulations can be performed in parallel. Both techniques are able to exploit this capability of using parallel machines. Obviously the utilization of the STM computer farm machine depends on the existing workload that can vary dynamically.



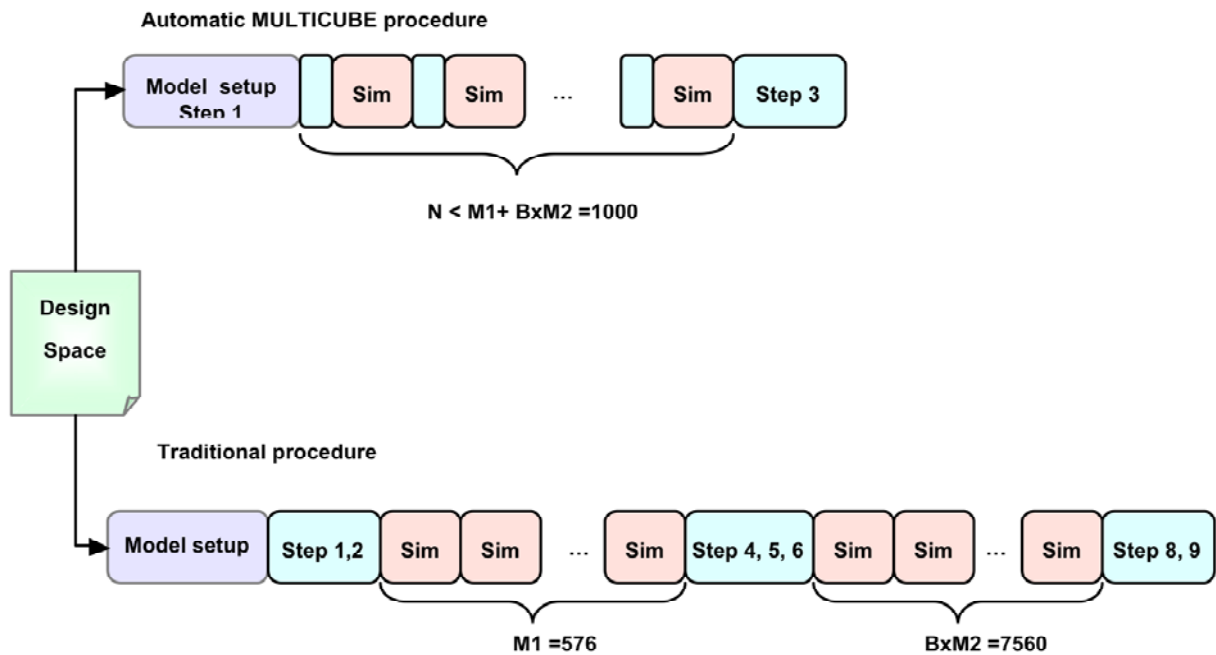


Figure 15: Comparison between manual and automatic optimization

The first comparison between the two procedures can be done in terms of the time spent to obtain tradeoffs results. As shown in Figure 16, the traditional and automatic DSE procedures require **34 days** with respect to **9 days** respectively corresponding to a **speed up of 3.77**.

The MULTICUBE automatic DSE can reduce the number of points to be simulated and so the simulation time spent during the exploration, but is also remove the intermediate intervention of the designer (which can be very time-expensive and whose duration can be unpredictable in some cases). We can also notice a huge reduction in the post processing data analysis part, since automatic DSE tools can automatically provide at the end of the exploration all the post-processed information that the designer needs for better understanding the design space. In the traditional procedure, designers start from a text file and have to elaborate data with the help of a general-purpose data processing tool (such as Excel or MATLAB for example). This help in the data analysis step can drastically reduce the effort for the designer spent in the pre and post processing steps.

The use of MULTICUBE optimization flow can save up to a 73% of the overall time while achieving comparable results in terms of power-performance trade-offs. In facts, as shown in Figure 17, the Pareto points obtained by the two procedures are comparable.

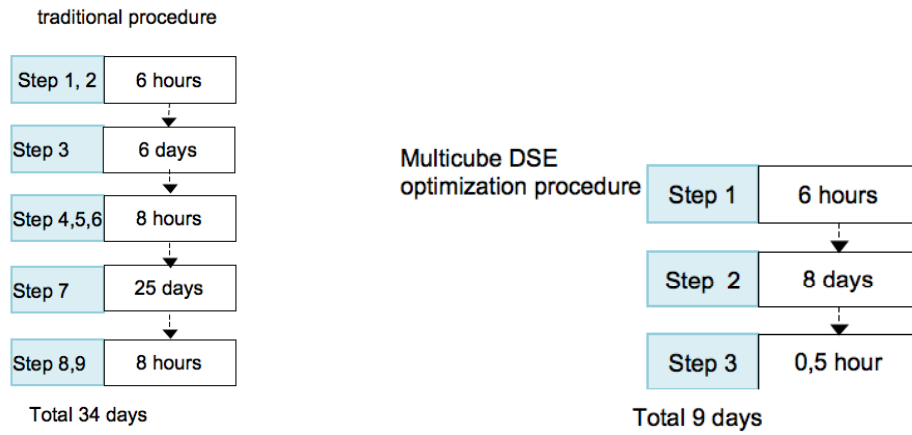


Figure 16: Assessment procedures comparison (traditional vs automated)

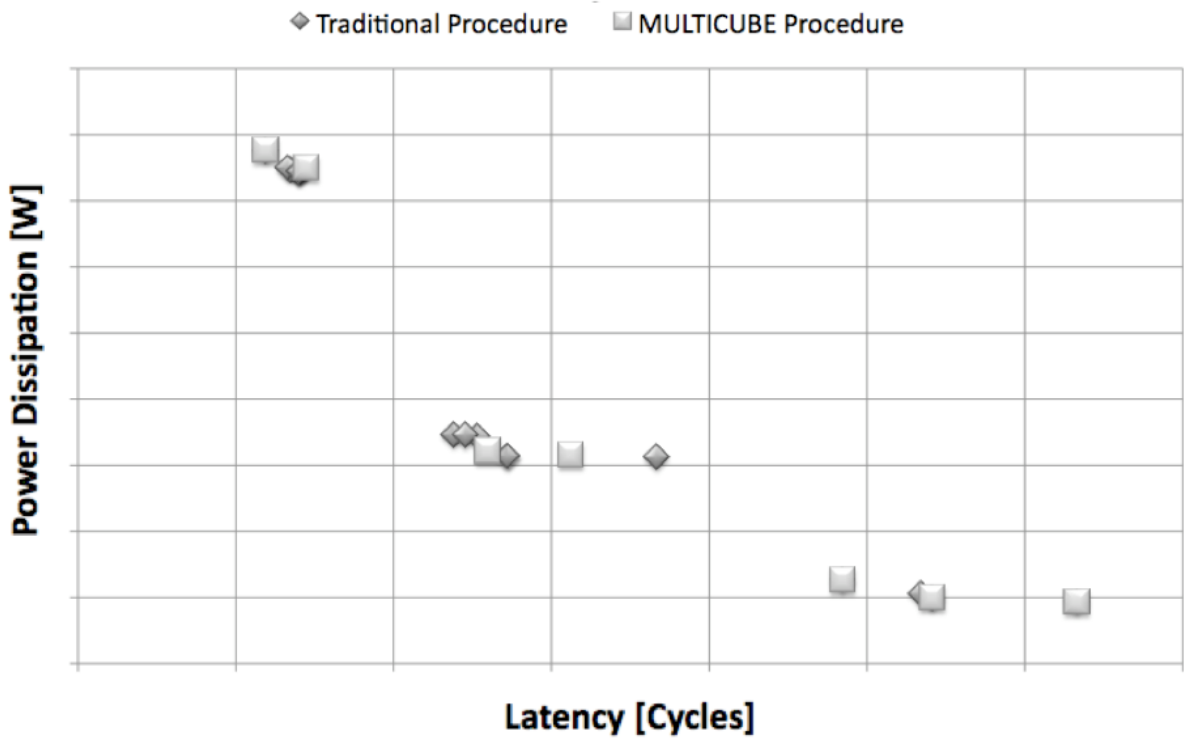


Figure 17: Comparison between traditional and automatic optimization: Power/Latency Pareto points for STM Low-Power Processor Use Case

Summarizing the presented results, the automated procedure provides:

- **Simpler flow:** The complexity of the flow is much less in the case of the automated procedure. The number of steps and the work involved to realize the flow is less. A simpler and automated procedure minimizes the possibility of introducing errors.
- **Faster times:** The convergence towards the optimum point is achieved faster with the optimization procedure with respect to the traditional procedure.
- **Fewer resources:** The possibility of achieving the same result, i.e. a comparable Pareto curve, with a significant reduction of the number of simulations, thus reducing the usage of the computational resources of the company.

3.2.3 Other benefits of the automatic DSE

Besides the quantitative assessment of the automatic DSE procedure presented in the precedent paragraph, we can mention some other more qualitative benefits that can be identified by the engineers that have run automatic DSE based on MULTICUBE design flow in STM for the low-power processor use case.

The MULTICUBE design flow makes the work of the designer simpler and in some cases limited to the data analysis, avoiding waste of time in developing scripts to run simulations and to collect results. This advantage can also be reported to future designs because of the definition of a standard XML interface for the interaction of the simulators with automatic DSE tool. Moreover the user friendly front-end and data analysis tool of the MULTICUBE DSE framework make the usage of the tools quite easy. Minimizing the human interventions, the usage of the automated DSE approach reduces also the possibility to introduce human errors.

The overall conclusion of the utilization of MULTICUBE design flow for an industrial use case is that the overhead in using the automated framework is very limited with respect to the large advantage that can be obtained in its usage, especially for complex designs requiring a long simulation run to evaluate a single point of the design space. As tested and demonstrated during the MULTICUBE project, the MULTICUBE framework is capable to reach much faster the same (or sometimes better) solutions obtained with a manual/traditional exploration procedure.

3.3 **IMEC Automatic DSE Assessment Procedure**

The assessment procedure for the final automated design space exploration as proposed within the MULTICUBE Project is different than for the other industrial partners within this Consortium. The main reason for this is that IMEC is a research institute, and hence the approach taken while doing engineering, the time frame and the end goals are very much different than for commercial companies.

In IMEC there is a lot of design space exploration performed within the design division. However, the reasons for this exploration are rarely to squeeze out the latest and greatest version of something. Since we are a research division we are searching for impact factors, alternate solutions, etc. Our main interest in doing DSE is to show a proof-of-concept of a new idea that IMEC is developing. Automatic DSE has helped us in guaranteed quality of design points as a larger design space is explored as compared to a manual DSE. But our focus has never been doing an exhaustive DSE to find the optimal design points. We perform a DSE till we find design points satisfying design goals to show proof-of-concept of our innovative research ideas.

To illustrate this, we will present *three* different design space exploration procedures and their goals. Last two of these DSE procedures were used in the context of MULTICUBE project.

3.3.1 **Scratch-Pad Memory Management**

For this research topic we want to out-perform a cache by using a (small) scratch-pad memory next to the processor and manage its data transfers manually (from within the application).

We will achieve this by doing code analysis techniques and source code transformations to introduce the required data-transfers using a pre-fetching technique to minimize the application waiting time.

The evaluation of this approach will be through comparison of both the overall performance and the overall system power.

The tool for doing the source code analysis and transformations proposes a (potentially large) number of possible solutions. This is because it does not only consider full array transfers, but also smaller sub-arrays. Depending on the application at hand, this can result in a large solution space.

A *first evaluation* of all these possible solutions is a *polynomial approach* : the number of memory accesses, data-transfers, the required memory size, etc. is used in a polynomial formula to evaluate an overall score for each solution.

A *second evaluation* is done by *simulating* the transformed application source code with a high-level simulator (HLSim). Since this is a time-annotated software simulator, it adds data-transfer arbitration and scheduling, and gives more accurate numbers for both power and performance.

A *final evaluation* is then performed more as a verification step by mapping the transformed application onto a complete *virtual prototype* of a platform. This virtual prototype will include instruction set simulators, bus models, DMA blocks and memory models. The result is a very detailed report of the behavior, performance and power.

The (manual) design space exploration procedure that is followed here includes a manual selection of the N most promising solution after every evaluation step. This is to reduce the



amount of lengthy simulations. In practice the analysis and transformation could produce as many as 100 possible solutions, HLSim will be used to evaluate about 10 different solutions and out of this only one or two solutions will be verified with a virtual prototype.

3.3.2 MPSoC Platform Optimisation

For this research idea we are investigating the impact of all the configuration parameters of a complex multi-processor platform. The platform will include several processors, busses and memories organized in a hierarchical way and interconnected by a central Network-on-Chip. This work is carried out in the context of MULTICUBE project (Deliverable D2.2.2 and D4.2.3).

These complex platforms can be tuned in many ways: clock-speeds of processors, busses, Noc, memory sizes, bus-widths, etc. And above all there are also many ways to configure the central NoC : topology, scheduling schemes, bit-widths, queuing schemes, quality of service, etc.

To evaluate the impact of these configuration parameters on a parallel application running on top of such an MPSoC platform requires running lengthy simulations onto virtual platforms. However to speed up the exploration of such a design space IMEC researchers have created and extended a simulator to use manual timing annotations of the parallel software to speed up these simulation runs, while trading of a minimal amount of accuracy (HLSim).

As a result, the huge design space (up to 100 different configuration parameters) which cannot be explored with the slow and detailed virtual platform simulator can now be evaluated with this faster simulation approach.

The (manual) optimization procedure will include first a strength and dependency analysis of the parameters using the HLSim. This is done using a very in-depth knowledge of both the parallel application as well as the MPSoC platform itself. Simulations are run via batch-scripts to be able to evaluate multiple results at once and extract both strength and dependency information among the configuration parameters. This evaluation step is also done manually or through specially devised scripts.

As a result an interesting sub-set (of configurations) is chosen to be mapped onto the virtual MPSoC platform simulator. These detailed runs will then validate the assumptions and/or conclusions drawn from other simulation runs.

In the context of MULTICUBE project, we had used high-level simulator HLSim with automatic exploration to deduce interesting sub-set of configurations. The design space had around 60 tunable parameters (mainly clocks and bus burst size). For our MPEG4 encoder parallelized application encoding 10 frames at 4CIF resolution, full space exploration would have required more than 3,000,000 simulations. Even with HLSim, which takes around 90 seconds to run a simulation, it was not practical to do a full-space exploration. Automatic exploration was done by M3 Explorer by using NSGA-II optimisation algorithm to derive interesting sub-sets of configurations to verify (and explore further) on MPSoC virtual platform simulator. This virtual platform simulator takes around 3.5 hours to finish one simulation. With HLSim based explorations using automatic DSE, we had obtained 20 configurations which formed the Pareto set for our application performance criteria of execution time and power consumption. We had verified these 20 configurations on MPSoC virtual platform simulator. To illustrate the amount of time saved by using automatic DSE:

- The exhaustive exploration (i.e., the simulation of all 3350886 configurations) with MPSoC virtual platform would take 103430544 hours; almost 153 years.



- Performing the NSGA-II optimizations using MPSoC virtual platform would require 16 months.
- The exhaustive exploration with HLSim would take about 6 months.
- Performing the NSGA-II optimizations using HLSim requires about 36 hours.

Clear advantage of automatic DSE was to reach desired operating points much quicker and in a more systematic, robust way than a normal manual DSE (which would have been typically based on intelligent guesses of input parameters). This is true even when in IMEC we need to achieve solution quality of only “proof-of-concept” and not the most optimal.

3.3.3 Dynamic MPSoC Runtime Management evaluation

Another research idea is to optimize the overall platform usage/behavior of MPSoC platforms when multiple (sequential and/or parallel) applications are running in parallel. Here every application will have a set of possible *operating points*. An operating point is a configuration of the application under which it can run and every point will require a certain amount of platform resources (# processor, amount of memory, communication bandwidth, etc.). These *operating points* are the result of a (manual) design space exploration done for every application separately.

The overall idea now is to evaluate and optimize the overall platform behavior according to external events (user interaction, battery status, etc.) by selecting the optimal operating point for every application and scheduling them accordingly. This work is carried out in the context of MULTICUBE project (Deliverable D3.3.2). Although there is some serious design space exploration that was done for this task, the main focus of this research idea is the runtime management aspect by itself. Hence the quality of the selected *operating points* is of lesser importance for this task. We are more interested in heuristics for doing the operating point selection and the actual move of all applications from one operating point to another at runtime. For this purpose we have developed an MPSoC simulator that also allows a top-level scheduling mechanism. Also a scheme is devised to represent every application with the required binary solutions (for every operating point) as well as a way to “describe” every operating point to the runtime-manager module.

In this research idea, for design space explorations, we were satisfied with the operating points which satisfied our output criteria (and not necessarily optimum). For proof-of-concept of our run-time management idea, we had used the same MPEG4 encoder parallelized application encoding 10 frames at 4CIF resolution. As described before, full space exploration for this application would have required more than 3,000,000 simulations. Even with a high-level simulator HLSim, which takes around 90 seconds to run a simulation, it was not practical to do a full-space exploration. Automatic exploration was done by using M3Explorer tool by choosing NSGA-II optimisation algorithm to derive interesting sub-sets of configurations using HLSim by doing only around 3000 simulations. To illustrate the amount of time saved by using automatic DSE:

- The exhaustive exploration with HLSim would take about 6 months.
- Performing the NSGA-II optimizations using HLSim requires about 36 hours.

Clear advantage of automatic DSE was to reach desired operating points much quicker and in a more systematic, robust way than a normal manual DSE (which would have been typically based on intelligent guesses of input parameters).

3.3.4 Benefits of Automated vs Manual DSE at IMEC

If we focus on the three exploration tasks explained within this section, we can evaluate what the added value would be of having an automated DSE approach versus a manual DSE as it is being done now:

In all three cases, we are relying on a very *in-depth knowledge* of application, platform, tools. However, having researchers skilled in many fields is not always easily possible. When we are doing manual explorations, we are limiting ourselves to a *limited number of solutions*. With this choice we are not guaranteeing to obtain the best or the most optimal solution, but it is a way to demonstrate the capabilities and results obtained from a new research idea.

In many cases, IMEC is only looking for a *proof-of-concept* solution. Again here, the solution obtained should prove a certain approach or technique but it is not necessarily required to do so in the most optimal way possible. We see that the advantage of having an automated DSE approach for these types of problem will reduce the in-depth knowledge requirements while improving on the overall results obtained. Also with problems like the MPSoC platform optimization problem, where the platform becomes too complex for human reasoning, an automated DSE approach will automatically extract strength and dependency results and could steer into new directions - directions which might get ignored in current manual explorations.

Looking at the two design space exploration case-studies (Section 3.3.2 and 3.3.3), it can be seen that using automatic DSE over manual full-space exploration has large benefits in terms of time-to-market and accuracy of exploration results. By using automatic DSE in dynamic runtime management evaluation case-study (Section 3.3.3), design space exploration time was reduced from 6 months (full-space) to 36 hours. In another case-study of MPSoC platform optimisation (Section 3.3.2), by using a simulator at higher abstraction level coupled with automatic DSE, we could reduce design space exploration time from worst case of 153 years (full-space) to 36 hours.

On the other hand, **the required effort of taking an automated DSE approach versus current manual DSE approach is relatively small**. In many cases researchers at IMEC are already partially moving into automatic DSE direction i.e. they are designing their applications tunable in terms of various parameters such that they can already run their manual DSE simulations in batches. The extra effort required for the automated DSE as proposed within MULTICUBE is a wrapper script around the XML interfaces and to learn usage of DSE tool(s). These efforts in the context of MULTICUBE project were of the order of few days of work (typically around two days) for the two case-studies mentioned above.

In conclusion, automatic DSE (performed in the context of MULTICUBE project) could produce **exploration results very close to the ones obtained by manual DSE, but at a fraction of the exploration time spent**. This guarantee on the obtained results has created a good confidence amongst the designers over capabilities of automatic DSE tools. At the same time, **the extra efforts required to build automatic DSE procedure were minimal**. Hence, these case-studies have shown potential benefits of using automatic DSE over manual DSE during MPSoC design space explorations.

4 Common Assessment Criteria

The aim of MULTICUBE project is to provide a main framework for performing design space exploration of complex architectures but taking always into account the necessary flexibility to handle the specific situations encountered in the different company structures and markets. In this sense, depending on the application, market and company constraints, these architectures can go from single core to many cores and can rely on open source simulators and models or proprietary simulators and models.

When designing MULTICUBE design flow, it was decided to provide the user the flexibility to follow its own path, chosen based on the particular constraints of its company, application and market. This way, the different MULTICUBE use cases represented a full set of possible expected situations when using the framework. In order to analyze the matching between user requirements and performances of the tool flow, we have defined a common assessment criteria table to assess the behavior of the tool flow versus a set of criteria that have been agreed as common to all possible use cases. The assessment criteria included in this common analysis have been reported in the following table (already shown in **D4.1.2**).

Category	Assessment criteria	Explanation / Motivation
I/O interface	Easiness in capturing use case design/ metrics	<i>Explanation:</i> The procedure to integrate new parameters/metrics in the use case has to be simple and with reduced manual interaction <i>Motivation:</i> The analysis process will be incremental
	Readability/analysis of results	<i>Explanation:</i> The obtained results have to be accessible easily by the designer in charge of the development of the architecture. <i>Motivation:</i> A faster analysis of the results will allow faster development cycles.
	Standard interface implementation	<i>Explanation:</i> The interfaces between the flow and the use case and between tools in the flow has to be as standard as possible <i>Motivation:</i> The use of a standardize approach will allow more flexibility when replacing, adding features (external or internal to the company) in the flow.
Modelling	Possibility to work with different abstraction levels	<i>Explanation:</i> MULTICUBE design flow does not have to rely on a particular abstraction level for the models of the use case. <i>Motivation:</i> The process of refinement of the design with different abstraction levels for a given module is an important step in the design of an embedded system. The industrial user has to be able to use MULTICUBE design exploration framework in all the steps of the refinement process.
Scalability	Integration in the industrial user's standard flow	<i>Explanation:</i> MULTICUBE design flow must be able to be easily integrated in the full design flow from the industrial company, <i>Motivation:</i> MULTICUBE design flow shall be one tool among others in the design flow of a company. In this sense, the needs from adaptation of the industry's design flow to MULTICUBE have to be reduced to the minimum.
	Scalability of metrics	<i>Explanation:</i> Capacity to evolve the metrics to be analyzed both in terms of quantity (number of metrics to be analyzed) and type (change the characteristics of a metric) <i>Motivation:</i> Increase the usability and flexibility of the system in all possible situations.
	Scalability of design	<i>Explanation:</i> Capacity to evolve the design under study in terms of complexity. <i>Motivation:</i> Allow to use MULTICUBE design flow in parallel with the development of the full system or with the refinement process
	Scalability of tool	<i>Explanation:</i> Capacity to evolve the features integrated in MULTICUBE design flow <i>Motivation:</i> Create an open, flexible tool that will be able to grow after the end of the project and be particularized to specific usages.
Operation	Easiness in operation of the flow	<i>Explanation:</i> MULTICUBE design flow has to be easily operated, with a short learning curve and with human interaction reduced to the minimum. <i>Motivation:</i> The interface with user has to deal with two different types of situations: a simple interface for the standard use and a more complex interface capability to particularize the toolset to specific situations (both interfaces are not incompatible)

Category	Assessment criteria	Explanation / Motivation
	Implementation of correct interface with user	<p><i>Explanation:</i> MULTICUBE design flow has to implement the interfaces to the user (GUI, scripts, etc...) which provides an easy operation of the flow but with the capacity to fine tune the toolset behavior to specific situations through the user interface</p> <p><i>Motivation:</i> The interface with user has to deal with two different types of situations: a simple interface for the standard use and a more complex interface capability to particularize the toolset to specific situations (both interfaces are not incompatible)</p>
Expected performance	Reliability/Accuracy of results	<p><i>Explanation:</i> The accuracy and reliability of the results have to be as expected in the requirements and have to allow the correct analysis of the architecture</p> <p><i>Motivation:</i> The accuracy of results is necessary to take the appropriate decisions in the architecture definition</p>
	Relevance of the results	<p><i>Explanation:</i> The results obtained through the MULTICUBE design flow have to be relevant and provide the necessary information to take the appropriate decisions</p> <p><i>Motivation:</i> Only the relevant results are important to take decisions in architecture definition.</p>
	Run-time performance	<p><i>Explanation:</i> The process of running MULTICUBE design flow has to take acceptable time.</p> <p><i>Motivation:</i> Design Space exploration tools have to help to take architecture definition decisions and, then, results have to be obtained in a reasonable time at early phases of the process</p>
	Completeness of the results	<p><i>Explanation:</i> Once the requirements and metrics are fixed, the tool has to perform the full analysis of the use case.</p> <p><i>Motivation:</i> MULTICUBE design flow has to deliver all the results necessary to fix the architecture after the analysis. Designer need to be sure that the all the possible (and reasonable) combinations have been investigated</p>

Table 6: Common assessment criteria table as from D4.1.2



At the end of the project, the following Table 10 summarizes the final assessment agreed by the industrial partners (namely DS2, STM and IMEC). All the assessment criteria have been successfully fulfilled by MULTICUBE design flow, while some detailed comments are summarized in the last column of Table 10.

Category	Assessment criteria	Final assessment	Comments
I/O interface	Easiness in capturing use case design / metrics	Good	Release of XML Interface Specification successfully used to release the Integrated Design Tool Prototype D1.4.2 including MULTICUBE design tools and use cases. The XML interface standardizes the formats to express the use case design parameters and metrics as well as the tool interfaces. <i>“Moreover the user friendly front-end and data analysis tool of the MULTICUBE DSE framework make the usage of the tools quite easy”, STM.</i>
	Readability/analysis of results	Good	
	Standard interface implementation	Good	
Modelling	Possibility to work with different abstraction levels	Yes	<i>“By using a simulator at higher abstraction level coupled with automatic DSE, we could reduce design space exploration time from worst case of 153 years (full-space) to 36 hours”, IMEC.</i>
Scalability	Integration in the industrial user’s standard flow	Good	Scalability and versatility of the M3 design flow has been proved by the successful application to several industrial scenarios (such as those provided by DS2, STM and IMEC)
	Scalability of metrics	Good	
	Scalability of design	Good	
	Scalability of tool	Good	
Operation	Easiness in operation of the flow	Good	<i>“Less complexity of the flow in the case of the automated procedure. The number of steps to be accomplished is much lower... a fewer number of steps represents a lower possibility of errors”, DS2</i> <i>“Simpler flow: the complexity of the flow is much less in the case of the automated procedure... A simpler and automated procedure minimizes the possibility of introducing errors”, STM</i>
	Implementation of correct interface with user	Good	
Expected performance	Reliability/Accuracy of results	Good	See next Section 5 “Final considerations about M3 DSE Flow”
	Relevance of the results	Good	
	Run-time performance	Good	
	Completeness of the results	Good	

Table 7: Final Automatic MULTICUBE DSE Flow assessment table



5 Final considerations about MULTICUBE DSE Flow

The main purpose of this deliverable is to demonstrate the effectiveness of the design techniques developed in the MULTICUBE project by means of final assessment and validation activities on real industrial design vehicles defined as the use cases in **D1.3**.

The assessment has mainly been based on the application of the automatic design space exploration assessment procedure as defined in **D4.1.2 “Final Evaluation Plan”**. The procedure has then been adapted by the responsible of each use case to comply with the specificities of its particular application and context, but without modifying the aim and the general steps of the original one.

DS2, the leading supplier of silicon and software for Powerline Communications (recently DS2 assets and technologies were bought by Marvell Technology Group) has applied the procedure to its Powerline use case suitable for advanced communication system to enable a fast a reliable transfer of audio, video and data information by using the most ubiquitous transmission system: the power lines. The general DSE assessment procedure has then be adapted to the peculiarities of the Powerline use case to test how the different methodologies and tools developed in the MULTICUBE project can be effectively applied to this industrial design flow. The main results of the comparison of the application of a manual or semi-automatic optimisation flow with respect to the MULTICUBE automatic optimisation flow can be summarized by the following statement:

“The proposed flow can save up to 80% of designer time while achieving better results in terms of performance since much more simulations can be run and analyzed following this flow obtaining ten times more possible combinations than with the semi-automatic design flow”.

STM, one of the world’s largest semiconductor companies, has applied the MULTICUBE design flow to design and optimize the SP2 low power advanced computing processor developed in STM Beijing for generic computing applications. The design space of the use case is very large including several architectural parameters concerning out-of-order execution parameters, memory hierarchy parameters and branch predictions parameters. To quantify the benefits of using the MULTICUBE design flow in an industrial practice, STM compared the traditional optimization procedure used internally with respect to the automatic DSE procedure proposed by MULTICUBE project. The benefits found from STM can be summarized by the following statement:

“The use of MULTICUBE optimization flow can save up to a 73% of the overall time and achieving comparable results in terms of power/performance tradeoffs.... The overall conclusion of the utilization of MULTICUBE design flow over a real industrial use case like the one just discussed is that the added value overcomes largely the cost of setting up the whole process, converging much faster to the optimum solution for a given technical problem.”

IMEC, the Europe’s leading independent research center for the development of microelectronics and ICT, was interested in evaluating the MULTICUBE DSE flow to show proof-of-concept of its innovative research ideas. IMEC was interested in evaluating the benefits of DSE flow not only considering the design-time tunable parameters, but also using the exploitation of the approach at run-time. For proof-of-concept of both design-time and run-time automatic DSE, IMEC used the same MPSoC virtual platform based on ADRES



core executing the MPEG4 encoder application. The benefits of automated DSE with respect to manual approach can be summarized by the following statements:

“Looking at the two design space exploration case-studies, it can be seen that using automatic DSE over manual full-space exploration has large benefits in terms of time-to-market and accuracy of exploration results. By using automatic DSE in dynamic runtime management evaluation case-study, design space exploration time was reduced from 6 months (full-space) to 36 hours. In another case-study of MPSoC platform optimization, by using a simulator at higher abstraction level coupled with automatic DSE, we could reduce design space exploration time from worst case of 153 years (full-space) to 36 hours... The extra efforts required to build automatic DSE procedure were minimal”.

Besides the benefits in terms of **design turnaround time** demonstrated in this deliverable by the application of DSE assessment procedure, previously we presented in **D3.2** “Implementation and Evaluation of the Exploration Algorithms” some PUBLIC results mainly in terms of the accuracy and quality of the optimization algorithms validation process. Other results of the application of the MULTICUBE design flow to the project demonstrators are described in the following deliverables: **D4.2.1** “Demonstrator of use cases: Powerline application”, **D4.2.2** “Demonstrator of use cases: Advanced computing platform”, and **D4.2.3** “Demonstrator of use cases: Multimedia Applications” (PUBLIC).