

# Information Societies Technology (IST) Program

## MNEMEE

**Memory management technology for adaptive and efficient design of embedded systems**

**Contract No IST-216224**



### Deliverable D6.2

### Quality Assurance Plan

Author/company: Arindam Mallik(IMEC)  
Contributors/company: Stylianos Mamagkakis(IMEC), Peter Lemmens(IMEC), Sander Stuijk (TUE), Robert Pyka (ICD), Christos Baloukas (ICCS)  
Status - Version: V0.9  
Date: 27/07/2009  
Confidentiality Level: Public  
ID number: Deliverable\_6.2\_V0.7doc

© Copyright by the MNEMEE Consortium

The MNEMEE Consortium consists of:

Interuniversity Microelectronics Centre (IMEC vzw)	Prime Contractor	Belgium
Institute of Communication and Computer Systems (ICCS)	Contractor	Greece
Technische Universiteit Eindhoven (TUE)	Contractor	Netherlands
Informatik Centrum Dortmund e.V. (ICD)	Contractor	Germany
INTRACOM S.A. Telecom Solutions (ICOM)	Contractor	Greece
THALES Communications S.A. (TCF)	Contractor	France

### Document revision history

Date	Version	Author/Contributor	Comments
07/06/2009	V0.0	Arindam Mallik	First draft
15/06/2009	V0.1	Stylios Mamagkakis	Edit
22/06/2009	V0.3	Sander Stuijk	Added TUE contribution
23/06/2009	V0.4	Robert Pyka	Added ICD contribution
24/06/2009	V0.5	Christos Baloukas	Added ICCS contribution
25/06/2009	V0.6	Arindam Mallik	Merged partner's contribution
29/06/2009	V0.7	Peter Lemmens	Added project mngt. contribution
01/07/2009	V0.8	Christos Baloukas	Modification from ICCS
27/07/2009	V0.9	Stylios Mamagkakis	Integrated toolflow and conclusions

## TABLE OF CONTENTS

<b>1. DISCLAIMER.....</b>	<b>5</b>
<b>2. ACKNOWLEDGEMENTS.....</b>	<b>5</b>
<b>3. PREFACE.....</b>	<b>5</b>
<b>4. ABSTRACT .....</b>	<b>6</b>
<b>5. INTRODUCTION.....</b>	<b>6</b>
<b>6. PROJECT MANAGEMENT AND COORDINATION METHODOLOGY .....</b>	<b>7</b>
<b>6.1. THE USE OF PROJECT MANAGEMENT BEST PRACTICES .....</b>	<b>7</b>
<b>6.2. APPLICATION OF PROJECT TEMPLATES .....</b>	<b>8</b>
<b>7. QUALITY ASSURANCE METHODOLOGY FOR INDIVIDUAL PARTNERS .....</b>	<b>9</b>
7.1. QUALITY ASSURANCE PLAN FOR IMEC TOOLS .....	9
7.1.1. <i>Documentation standard</i> .....	9
7.1.2. <i>Code Standard:</i> .....	9
7.1.3. <i>Benchmarking</i> .....	10
7.1.4. <i>Active Tool Support and 3<sup>rd</sup> party dependencies:</i> .....	10
7.1.5. <i>Stable Release:</i> .....	11
7.2. QUALITY ASSURANCE PLAN FOR ICD TOOLS.....	12
7.2.1. <i>Documentation standard</i> .....	12
7.2.2. <i>Code Standard:</i> .....	12
7.2.3. <i>Benchmarking</i> .....	14
7.2.4. <i>Active Tool Support and 3<sup>rd</sup> party dependencies:</i> .....	15
7.3. QUALITY ASSURANCE PLAN FOR TUE TOOLS.....	15
7.3.1. <i>Documentation</i> .....	15
7.3.2. <i>Benchmarking</i> .....	16
7.3.3. <i>Tool support and software development process</i> .....	16
7.3.4. <i>Software releases</i> .....	16
7.4. QUALITY ASSURANCE PLAN FOR ICCS TOOLS.....	16
7.4.1. <i>Documentation</i> .....	17
7.4.2. <i>Benchmarking</i> .....	17
7.4.3. <i>Active Tool Support</i> .....	17
7.4.4. <i>Stable Release</i> .....	17
<b>8. QUALITY ASSURANCE METHODOLOGY FOR INTEGRATED TOOLFLOW.....</b>	<b>17</b>
8.1. DOCUMENTATION.....	17
8.2. CODE STANDARD.....	18
8.3. BENCHMARKING.....	19
8.4. ACTIVE TOOL SUPPORT.....	20
<b>9. CONCLUSIONS .....</b>	<b>20</b>
<b>10. REFERENCES.....</b>	<b>21</b>
<b>11. GLOSSARY.....</b>	<b>21</b>

## List of Figures

Figure 1. Memory hierarchy in embedded systems.....	10
Figure 2. MACC eco-system.....	12
Figure 3. ICD-C compiler framework.....	13

## 1. Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## 2. Acknowledgements

The author acknowledges contributions by Stylianos Mamagkakis(IMEC), Peter Lemmens(IMEC), Sander Stuijk (TUE), Robert Pyka (ICD), Christos Blaoukas (ICCS).

## 3. Preface

The main objectives of the MNEMEE project are:

- Provide a novel solution for mapping applications cost-efficiently to the memory hierarchy of any MPSoC platform without significant optimization of the initial source code.
- Introducing an innovative supplementary source-to-source optimization design layer for data management between the state-of-the-art optimizations at the application functionality and the compiler design layer.
- The efficient data access and memory storage of both dynamically and statically allocated data and their assignment on the memory hierarchy.
- Use and dissemination of the results.

The expected results of the project are:

- 50% reduction in design time
- 30% reduction in memory footprint and bandwidth requirements
- 50% improved energy and power efficiency

The technological challenges of the project are:

- Provide a framework for source-to-source optimization methodologies, which targets both statically and dynamically allocated data.
- Introduce a novel source-to-source optimization design layer.
- Analyze the embedded software applications and highlight the source-to-source optimization opportunities.
- Develop a set of prototype tools.
- Provide data memory-hierarchy aware assignment and scheduling methodologies.

The MNEMEE project has started its activities in January 2008 and is planned to be completed by the end of December 2010. It is led by Mr. Stylianos Mamagkakis and Mr. Arindam Mallik of IMEC. The Project Coordinator is Mr Peter Lemmens. Five contractors (INTRACOM Telecom, Thales, ICD, ICCS and TUE) participate in the project. The total budget is 2.950 k€

#### **4. Abstract**

The goal of this deliverable is to provide details regarding the quality assurance as applied to the MNEMEE project. The document is divided in two main parts. First, we have described the methodology and criteria that assure the quality of the coordination, the quality of project management mechanisms and quality of project execution (focus on processes) in MNEMEE. In the second part, we have summarized the technical methodology and criteria to evaluate and assure the quality of the demonstrators, including benchmarks used by different partners.

#### **5. Introduction**

The MNEMEE project budget is funded by the European Commission and as thus by taxpayers money. Therefore it is important to make sure that what is being delivered as the outcome of this project can meet the quality standards and expectations of the consortium members, the EC and the external reviewers.

All of the consortium partners have a proven track record of successful project realisations either in the academic world or in the industry. It is the ultimate goal to disseminate the project results to the industry in particular and promote it to the wider public in general. At first instance it are the consortium members who aim to maintain high standards in project execution and high quality of the deliverables.

To avoid leaving things up to coincidence, a set of rules and activities was agreed upon by the consortium members at the beginning of this project. This set is based upon best practices in project management and should ensure that the invested means are used in a proper manner and achieved with appropriate quality.

The applied way of working is fit for the purpose of use in this funded project. The project team tries to avoid creating too much overhead, hence safeguarding the efficiency. Used principles are derived from practices as provided by the PMBOK (project management book of knowledge, by Project Management Institute). The agreed way of working is practical enough to ensure a consistent use. As a result, clear evidences of its application within the project can be provided.

The next section will describe in more detail which methods are used in practice. The third section will explain how the quality performance is measured and benchmarked against the objectives.

It is the responsibility of each of the individual partners to make sure conformance with quality expectations is achieved. At the end, the project coordinator (IMEC) is the end responsible for safeguarding and maintaining the methods used within this project and to mediate between partners when necessary.

The effectiveness of the used way of working within this project must be measured against the achieved quality and relevance of the project deliverables.

## 6. Project Management and Coordination Methodology

The set of rules as agreed upon by the project consortium consists of two key elements, namely “Best Practices” and “Uniform project Templates”.

The framework is used consistently throughout the entire duration of the project by all consortium members. Therefore it provides a guideline for the quality assurance.

### 6.1. The use of Project Management Best Practices

1. Definition of project objectives (WHAT) :
  - At the start of the project, the targeted outcome was defined and documented in an unambiguous way (e.g. percentage of reduction in memory footprint, tool suite availability, number of power savings, etc).
  - Each project meeting the actual status of the progress against the deliverables is measured and when necessary actions are taken to contain any deviation from the plan. In such a way effort is spend to avoid original targets are abandoned or diluted as time progresses.
2. Qualitative Evaluation of deliverables and reports (HOW) :
  - Review of each deliverable report and presentation material is carried out by each of the work package leaders prior to submission to the EC.
  - Sufficient time is foreseen to have a qualitative review and for the author to iterate based upon the received comments.
  - External (to the project team) reviewers are asked to provide their comments on important documents like year reports and dissemination events.
3. Monitoring & control of the actual progress versus the original plan (WHEN) :
  - Actual status of the work done is monitored by the individual work package leaders on a daily base. The project coordinator can decide to organize dedicated follow-up meetings
  - The high level planning for the next phases is reviewed at each project meeting. This assures that the future outlook remains in line with the original plan and that all enablers are put in place.
4. Risk Management :
  - A risk repository is prepared and updated on a regular base by the project coordinator. It includes the risk definitions, impact and mitigation actions.
  - Details have been provided in section 7.2 of the deliverable report D7.1
5. Keeping record of decisions made :
  - A central list of all the decisions made by the projectteam is kept and updated each meeting (not only for the project meetings and board meetings, but also for the intermediate technical meetings).
  - As such this list is a reference which can be consulted at any moment in time. It avoids any future unclarity as well as repeating discussion which were already concluded.
6. Documenting project meetings / workshops :
  - A uniform actiontracker is kept, updated and distributed among the consortium members, including due dates per action. The assigned work package leader assures the timely completion of the action, eventually assisted by the project coordinator.

## **6.2. Application of Project Templates**

The availability and use of uniform templates makes sure that throughout the project a proper flow of information can be enabled in an efficient way. It also gives an identity to the projectteam in case of external communication. The templates are stored on the MNEMEE website ([www.mneme.org](http://www.mneme.org)) and available to all consortium members.

The currently available templates are :

1. Deliverables register
2. Generic Tool flow overview
3. Risk management register
4. actiontracker & decision list template
5. deliverables and reports template
6. meeting agenda template
7. template for presentations
8. overall planning template

## 7. Quality Assurance Methodology for Individual Partners

This section summarises the methodology followed by individual partners in the MNEMEE project for maintaining the quality of the tools developed in the context of the MNEMEE project. Each partner will use their own benchmarking applications to demonstrate the gains obtained by each individual tool. Regarding the benchmarking of the tool flow, more details are provided in Section 8.

Quality Assurance (QA) [1] for a software tool is defined as a planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes, and procedures. QA includes the process of assuring that standards and procedures are established and are followed throughout the software acquisition life cycle. Compliance with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, and audits. Software development and control processes should include quality assurance approval points, where a QA evaluation of the product may be done in relation to the applicable standards.

To establish an assurance of standard and procedures for software development is critical, since these provide the framework from which the software evolves. Standards are the established criteria to which the software products are compared. Procedures are the established criteria to which the development and control processes are compared. Together, standards and procedures establish the prescribed methods for developing software; the QA role is to ensure their existence and adequacy.

### 7.1. Quality Assurance plan for IMEC tools

#### 7.1.1. Documentation standard

The first step of QA for the software tools is to provide detailed documentation of standards and procedures. These documentations provide a clear definition of process monitoring, product evaluation to measure project compliance.

From IMEC, every tool developed in the context of the MNEMEE project is supplied with a user manual. The user manual introduces the user to the syntax and semantics of the tool. It also includes a simple code example to illustrate the tool features and usage.

#### 7.1.2. Code Standard:

Code standards specify the language in which the code is written and define any restrictions on use of language features. They define legal language structures, style conventions, rules for data structures and interfaces, and internal code documentation.

The IMEC tool set supports applications written in C89 specifications. Due to the complexity of the C language, and the ways in which ATOMIUM needs to represent C programs internally, a number of restrictions have been imposed during application development stage. Details of these restrictions are available in the User's manual of the software package [2] [3].

IMEC tool code standards guide the designer to write standard codes for their tools in the form of the CleanC specification. The C language provides a lot of expressiveness to the designer, but as a drawback this expressiveness makes it hard to analyze the C program and to map it to a MP-SoC platform. Clean C gives guidelines to the designers to avoid usage of constructs that are not well analyzable.

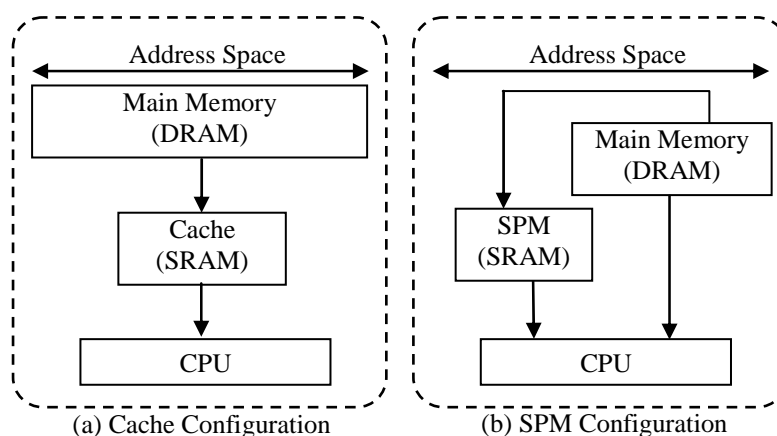
Clean C gives a set of rules, which can be divided into two categories as code restrictions and code guidelines. Code restrictions are constructs that are not allowed to be present in the input C code, while code guidelines describes how code should be written to achieve maximum accuracy from mapping tools.

A code cleaning tool suit provided by IMEC will help to identify the relevant parts for clean-up. While developing new application code, the Clean C rules can be followed immediately. Some of the proposed guidelines and restrictions details can be found in [4].

### 7.1.3. Benchmarking

Benchmarking is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. Benchmarking is usually associated with assessing performance characteristics of a tool. It provides a method of comparing the performance of various subsystems across different chip/system architectures.

Selection of a benchmark program is an important issue to measure the performance of a design tool. In the context of the MNEMEE project the key characteristic of the benchmark applications will be the intensive data transfer and storage and the need for efficient memory management. MPEG-4 encoding application is a representative of such a computationally intensive embedded systems application. For Benchmarking the MPMH tools from IMEC, MPEG-4 [5] is selected as the application of interest.



**Figure 1.** Memory hierarchy in embedded systems.

To prove the usability and the benefits of the MPMH tool, MPEG-4 application mapping on a scratch-pad based platform using MH will be compared to a mapping on a cache based platform. The platform is composed of a processor (a multi-media instance of ADRES [6] [7]) with its local memory (cache or scratch-pad), a bus and a main memory. The MPEG-4 encoder application has been optimized for the target processor. This ensures a realistic balance between computation and communication. The resulting encoder needs on average 40 processing cycles per pixel, assuming an L1 memory of sufficient size. A detailed description of the MPEG-4 application was discussed in an earlier deliverable [8]. The same optimizations will be applied for both the SPM and the cache mapping. The memory hierarchy tool will be applied after these source-code optimizations. The same application will be explored for parallelization benefits. Both functional and data level parallelism present within the application would be explored using the tool. The output of the MPMH tool would be a parallelized MPEG-4 code that is optimized for statically allocated data structures running on a SPM-based platform.

### 7.1.4. Active Tool Support and 3<sup>rd</sup> party dependencies:

For supporting the tool users and report of bugs in IMEC MPMH tools, IMEC uses the Trac system. Trac [9] is an enhanced wiki and issue tracking system for software development projects. Trac uses a systematic and centralized approach for web-based software project management. The Trac system is developed to help the application developers recognize and fix bugs in the software in an useful and organized way. It is very easy to integrate a Trac system within a software development framework as

it is independent of the actual development cycle. The software also provides integrated support for Subversion support and wiki management.

Each of the MNEMEE partners has been subscribed to the Trac system. The partners would share their common knowledge about the tool usage over the course of the project.

Atomium 3.3.4 is available for the following platforms:

- HP-UX 11.00 or later on PA-RISC 2.0, using version 03.27 or later of the aCC C++ compiler, or alternatively version 3.2.x of the g++ compiler. The HP-UX 11.00 version of Atomium does not support the older PA-RISC 1.x CPUs.
- Solaris 823 on SPARC, using the Sun Forte Developer 7 C++ compiler, or alternatively version 3.2.x of the g++ compiler.
- Red Hat Enterprise Linux 3 and 4 on Intel x86 or AMD, using version 3.2.3 of the g++ compiler.
- Red Hat Linux 7.2 on Intel x86, using version 2.96 of the g++ compiler, or alternatively version 3.2.x of the g++ compiler.
- Red Hat Linux 9.x on Intel x86 or AMD, using version 3.2.2 of the g++ compiler.

Users of Red Hat Enterprise Linux 2.x are advised to use the Atomium distribution for Red Hat Linux 7.2. Users of Red Hat Enterprise Linux 4.x are advised to use the Atomium distribution for Red Hat Enterprise Linux 3.x or Red Hat Linux 9.x.

In any case, Atomium 3.3.4 does not support the 64-bit modes of HP-UX 11 and Solaris 8. To allow to work around platform dependencies, Atomium predefines the same pre-processor macros as the respective C++ compilers do: `hpux` and `__hpux` on HP-UX, `sun` and `__sun` on Solaris, and `linux` and `__linux` on Linux.

#### 7.1.5. Stable Release:

The IMEC tools are updated on a regular basis through a stable release system. Based on reported bugs and requests for added features, the tools' developers from IMEC release a stable version of the tools.

The MNEMEE partners have received an initial release of the IMEC tools after signing a Software License Agreement (SLA) with IMEC. Through the software license IMEC grants to Licensee a non-exclusive and non-transferable license on the MPMH tools, without the right to grant sub-licenses, to use the executable code subject to the terms and conditions of this SLA for the sole purpose of performing the activities related to the work plan of the FP7 IST-216224 MNEMEE project.

The licensed Atomium software is a tool set consisting of 3 major components : Atomium/Analysis[1], Atomium/MH[10] (Memory Hierarchy), Atomium/MPA[11] (Multi-processor Parallelization Assistant).

## 7.2. Quality Assurance plan for ICD tools

### 7.2.1. Documentation standard

ICD provides a variety of documents concerning the tools used for the MNEMEE project. On the one side this documentation consists of hand crafted book style documents, describing general usage and features of ICD-C. On the other side, ICD enforces coding styles, which enable the automatic generation of Doxygen style documentation. This result in an easily accessible always up to date and precise documentation of APIs and further parts of the source code used and developed in this project.

### 7.2.2. Code Standard:

Our Tools for the MNEMEE Project are based on the MACC eco-system. Its underlying technology for code representation is based on the ICD-C compiler framework. ICD-C is a High-Level framework that handles the target application at Source Code Level. It is used by ICD as a compiler frontend, responsible for parsing the source code and performing High-Level optimizations.

In ICD-C, all high-level information is retained, excluding format information and comments. This feature makes ICD-C the optimal choice for the implementation of Source-to-Source optimizations as required for the MNEMEE project. Users can thus keep their environments and add performance-improving prepass optimizations as plug-in modules.

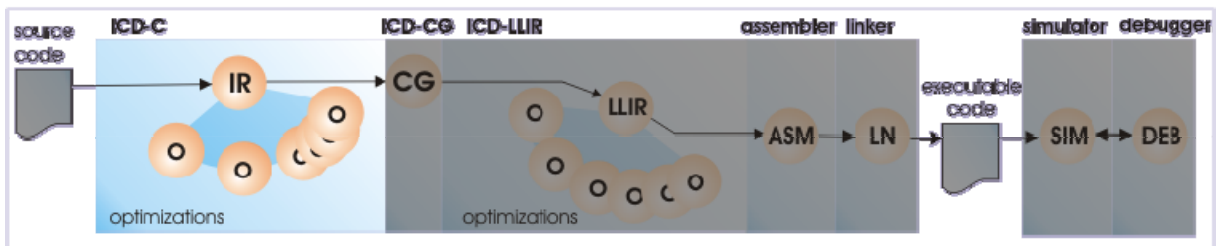
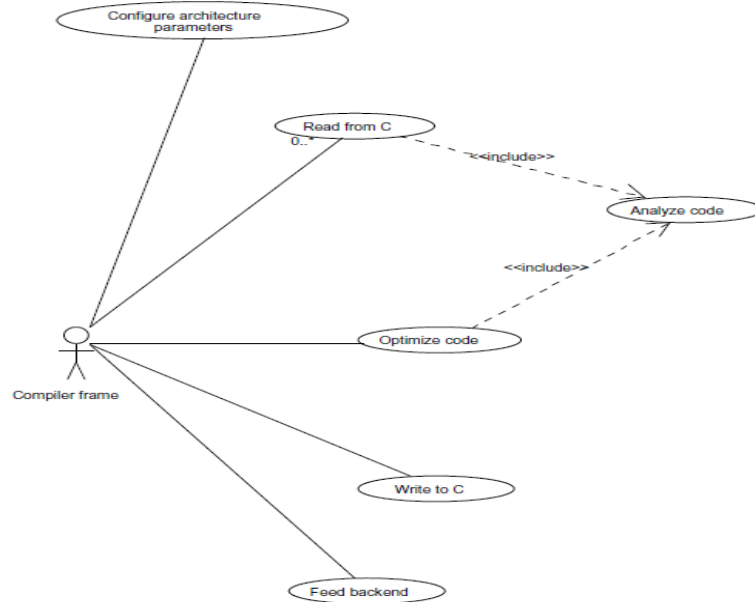


Figure 2. MACC eco-system

The ICD-C Compiler framework is designed as a compiler frontend framework to provide a platform independent intermediate representation (IR) for C code and based on it a set of platform independent code analysis and optimization functionality. Further, it is intended to feed either existing compiler toolchains by emitting compilable C code, as well as feeding compiler backends by supporting a dedicated code selection interface. The use case diagram shows the main applications of the ICD-C class.



**Figure 3.** ICD-C compiler framework

Basic feature of ICD-C are

- Input language support: Full C99 compliant, including C99 parser Optional deactivation of undesired C99 features
- High level intermediate representation: Preservation of entire C code structure, even including comments High-level to mid-level transformation module
- Support for stand-alone Source-to-Source optimizations: Facility for generating compilable C code Allows pre-pass optimizations in a given toolchain
- Analyses: Control flow, data flow, function callgraph analysis Automatic maintenance of consistency
- Multiple Compilation Units Allows really global optimizations Global analysis of functions and symbols
- C++ interface: Full access to ICD-C's data structures Allows code analysis, manipulation and optimization

### **Restrictions for the Task Extraction Tool**

There are almost no restrictions to the code, which is used as input for the Task Extraction Tool. Because a separate profiling tool is used to extract the required dependencies and execution counts, plain C code can be used. The only restriction, which is imposed by the current profiler based approach, is that the code may not contain any indirect called functions via pointers. These kinds of pointers make the behaviour of a program unpredictable to generate a valid parallelization for it.

If later on the profiling-based analysis tool gets replaced by static analysis techniques, which provide more precise results, additional restrictions will occur. Nevertheless, before taking such a transition we have to ensure, that these restrictions will be a subset of the CleanC coding rules. This means that further restrictions which may be added to the Task Graph Tool will have no influence on the restriction for the whole MNEMEE tool flow. All of these possible modifications will be well documented and made available to the partners.

### **Restrictions for the Scratchpad Memory Allocation Tool**

The Scratchpad Memory Allocation Tool can handle any source code which can be parsed by ICD-C and which complies with the following restriction: For an accurate analysis of the memory accesses the control flow of the program has to be well known. Therefore also for this tool no function pointers are allowed. Nevertheless, with some restrictions pointers to data objects may be used. Each pointer must not point to two different objects in the whole program. This applies to function parameters as well. Thus a function must always be called with the same data objects as parameters, if they are passed by reference.

### **Restrictions for the Mapping Optimization Tool**

The mapping optimization tool needs a task graph as input, as well as C-Code for each task. As already mentioned for both previous tools, restrictions to the code are already covered by the CleanC rules.

#### **7.2.3. Benchmarking**

The ICD-C compiler tools and the MACC framework are intensively tested. ICD-C utilizes a set of about hundred of test-cases which can be automatically applied to a release to validate stability and conformance to current language standards.

### **Benchmarking for the Task Extraction Tool**

The Task Extraction Tool is tested with all of afore mentioned test-cases of the ICD-C Compiler Framework in each step of the development process.

To check the behaviour of the tool on large programs, a couple of real world applications are also taken into account. Parts of these benchmarks are for example an implementation of an edge-detect benchmark (part of **Error! Reference source not found.**) and a JPEG 2000 encoder.

Right from the beginning of the tool development, the Task Extraction Tool is also tested with parts of the MPEG-4 encoder to ensure the fitness for optimizing the whole encoder at the end of the project.

### **Benchmarking for the Scratchpad Memory Allocation Tool**

A significant set of benchmarks is used to test the functionality of the Scratchpad Memory Allocation Tool. The benchmarks are used both to test the correctness of the applied code optimizations and to evaluate the gain of the optimization. For that purpose all tests from the already mentioned ICD-C test suite and data-dominated real world sample applications that respect the restrictions of the tool are used. An example for a real world application is a MPEG-4 encoder. Furthermore in the scope of the MNEMEE project programs provided by the industrial partners will be used for the evaluation of the computed allocation.

The tests will be made for at least two different systems that are modelled in the MACC eco-system. The Scratchpad Memory Allocation Tool will be tested whenever major changes have been made. The tool will be tested additionally before every release. The results of the tests will be recorded, so that either degradation can be detected or quality of improvements can be determined after each test run.

### **Benchmarking for the Mapping Optimization Tool**

The mapping optimization tool is tested with several benchmarks which were provided by ETH Zürich since these benchmarks are already given in the required input format. Next to several smaller benchmarks as for example the Fourier transformation or infinite impulse response (IIR) filter, also real-life benchmarks as MPEG-2 Video decoder and Wave field synthesis are given. Furthermore, other benchmarks as edge detect and MPEG-4 encoder are added to the existing benchmark suite. These benchmarks are used to test the functionality, correctness and performance of the mapping optimization tool.

#### **7.2.4. Active Tool Support and 3<sup>rd</sup> party dependencies:**

ICD offers a wide choice of Software Generation Tools for Embedded Systems. Based on a strong research background from the Embedded Systems group of the TU-Dortmund, we offer affordable custom tool solutions that allow customers to efficiently design their embedded software.

Our tool development service covers the entire tool chain from highly efficient C compilers down to fast cycle-true processor simulators, including Compiler Generation Tools and Consulting Services. ICD/ES offers products of industrial quality utilizing the most up-to-date techniques from research at TU-Dortmund.

In conjunction with the products developed at ICD full range, long term support and maintenance plans are offered. In general fast reaction (i.e. next working day) to bug reports and customer issues can be guaranteed.

### **7.3. Quality Assurance plan for TUE tools**

#### **7.3.1. Documentation**

Each tool developed by TUE within the MNEMEE project is distributed through a website at the TUE. This website contains reference manuals as well as a getting started guide. The website also provides references to additional documentation about the algorithms which are implemented in the tool. Finally, the website provides access to the documentation on the source code of the tool.

SDF3 [12] is one of the tools which is developed within the MNEMEE project. The interested reader can visit the website of SDF3 at [www.es.ele.tue.nl/sdf3](http://www.es.ele.tue.nl/sdf3) to see all documentation of this tool. Other tools which are developed by TUE within the MNEMEE project will get a similar website.

### 7.3.2. Benchmarking

Benchmarks are used to test the performance and correctness of the tools which are developed within the MNEMEE project. Whenever possible, existing benchmarks will be used for this purpose. If those benchmarks do not exist, we aim to provide such benchmarks through the website of the associated tool. A good example of this approach is SDF3. The website of this tool contains a large set of application models which are commonly used to benchmark dataflow analysis and mapping algorithms. Part of these application models have been developed by other researchers in the past and part of these models have been developed to benchmark new features of SDF3. The website collects all these models to provide researcher in this area with a complete set of benchmark applications. During the MNEMEE project, we plan to extend this set of benchmark applications with new real-life application models. The MPEG-4 encoder is one the first application which will be added to the benchmark since this application is used within MNEMEE to benchmark the overall tool flow.

Benchmarking not only requires representative inputs, but also the availability of state-of-the-art reference algorithms against which a comparison can be made. For this purpose, we use either existing implementations (if available) or we try to reconstruct these reference algorithms based on the details provided in publications that describe them. In the latter case, these algorithms are often integrated into our tools and will be provided on the tool website for use by the research community. SDF3 for example contains many state-of-the-art dataflow analyses and mapping algorithm. We will use those algorithms to benchmark the new algorithms that are developed by TUE within the MNEMEE project.

### 7.3.3. Tool support and software development process

Users of the tools developed by the TUE can get support through e-mail. The website of each tool contains the contact information of the maintainer of this tool. Communication through e-mail has proven to be a fast and easy method for users to interact with the tool maintainer.

The maintainer is also responsible for integrating new features into the stable version of the tool. Furthermore, he/she must enforce that the internal coding guidelines are used. To support the maintainer and developers of the tool, a version management system is used. The developers work on a separate branch of the source code. The maintainer is responsible for integrating the changes made by the developers on the main branch. This main branch is used to create the stable releases of the tool.

### 7.3.4. Software releases

The tools developed by TUE within the MNEMEE project are released under an open-source license (e.g. GPL, BSD). The source code of each tool can be downloaded from the website which is associated with the tool. A new version of a tool is released as soon as a paper is published which describes a new feature of the tool. Intermediate releases may also be provided to resolve bugs which have been discovered in the tool. The tools can be compiled and executed on all major Linux distributions using any g++ 3.x or 4.x compiler. Both 32bit and 64bit platforms are supported. Many tools can in fact also be compiled and executed on the Windows platform.

## **7.4. Quality Assurance plan for ICCS tools**

#### 7.4.1. Documentation

Each tool that is being developed by ICCS for the MNEMEE project is also being documented. The documentation explains the way to take advantage of the optimization tools for the designers by providing usage examples. Furthermore, the documentation sheds light on the internals of the developed tools for readers interested in extending them.

#### 7.4.2. Benchmarking

Choosing the right benchmarks was one of the top priorities for ICCS. Accurate benchmarking can facilitate the development of a toolset by providing bug and performance reference. Since ICCS's tools target the data structures optimization, benchmarks should contain a lot of data structures exhibiting a variety in access patterns and memory allocation.

In order to test the developed methodologies to the limit, a 3D game (Vdrift) [13] was chosen as the main benchmark application. Vdrift contains more than 25 data structures ranging from dynamic vectors to lists and trees. This set of optimization targets provides a rich range of access patterns and allocation behaviours.

Another reason for choosing Vdrift was the use of STL (Standard Template Library). All data structures except for the trees are coming from the STL. Since STL is a standard we will compare the performance of our optimized data structures to the original well documented data structures from the STL library. This will give us an accurate view of the achieved optimization.

#### 7.4.3. Active Tool Support

All tools developed by ICCS will be supported on an email-based scheme.

The tools will be available on Linux and Windows versions and both will be supported and further developed. New features of the tools will be added if requested by the end users. Close collaboration with the end users will be guaranteed by personal contact with one of our developers.

The source code base will be shared with potential users so that new the code base can be customized by the end users also.

#### 7.4.4. Stable Release

Once completed the ICCS' tools will be updated on a regular basis adding new features that maybe requested by the partners. The support and development will continue beyond the MNEMEE project. The source code base of all tools will be available for open source licensing.

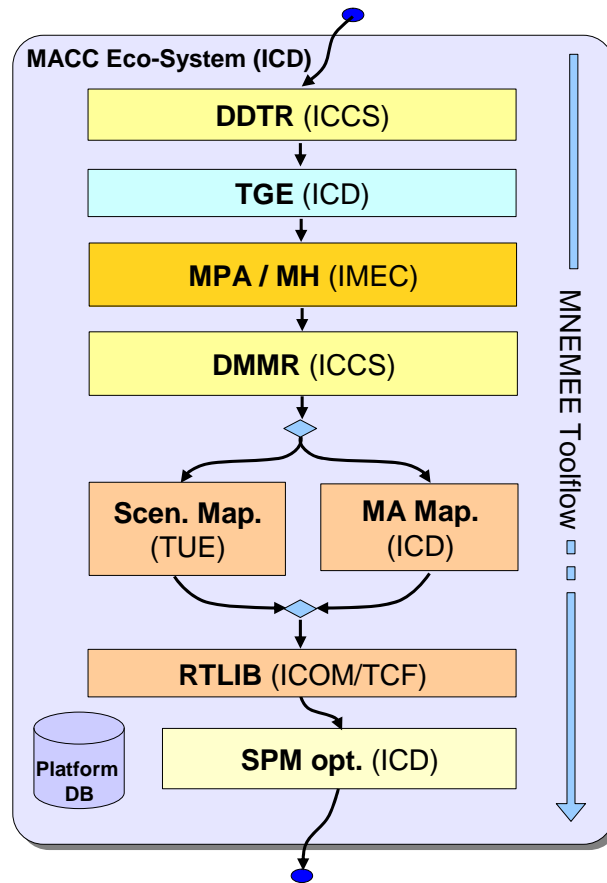
### **8. Quality Assurance Methodology for Integrated Toolflow**

#### **8.1. Documentation**

Every individual tool developed in the context of the MNEMEE project will be supplied with its individual user manual (which is described in detail in Section 7). In addition to the individual manuals, documentation will be provided for the integrated toolflow which will be delivered in Task 5.5. This documentation will describe the input and output format of the individual tools as well as the scripts and databases that will be used to glue the individual parts together. The documentation will include explanation on which parts of the toolflow are manual, semi-automated or fully automated. For the fully automated parts explanation for the GUI/command line controls will be provided. While regarding the manual steps or semi-automated steps, detailed instructions will be included in the documentation so that the user of the toolflow can replicate the steps of the related methodology.

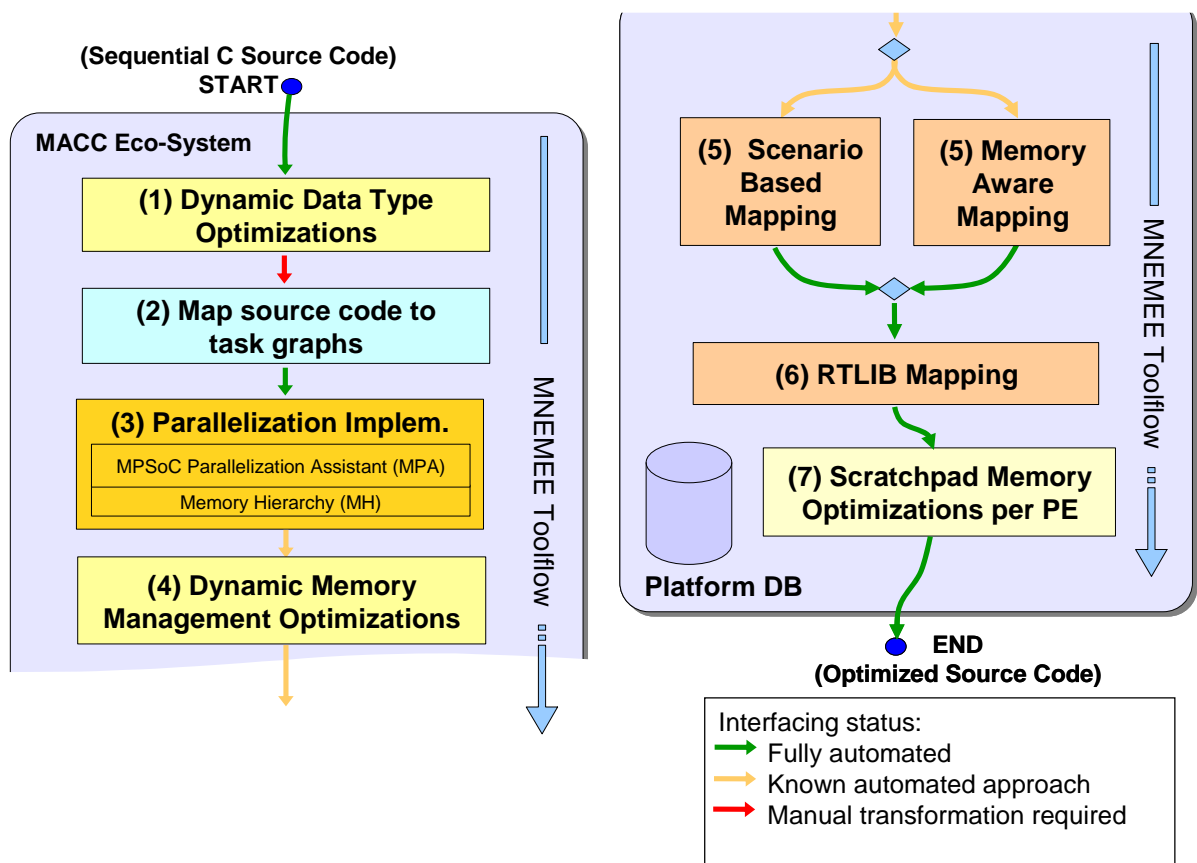
## 8.2. Code standard

The integrated MNEMEE toolflow will consist of the individual tools developed by IMEC, ICD, TU/e and ICCS, which will be integrated with the help of the MACC Eco-system which belongs to ICD, as can be seen in Figure 4.



**Figure 4 Toolflow overview**

The code standard for the integrated toolflow will consist of the code standards of the individual tools as they are explained in Section 7. In Figure 5, a more detailed view of the toolflow can be found where the individual tools are indicated in conjunction with the interfacing plans which are currently available. As the tool integration activity is just starting, we can not provide full details of the code standards that will be used for the interfacing automation scripts. These will become available in Deliverable 5.3 with the integrated toolflow.



**Figure 5 Toolflow detailed view**

### 8.3. Benchmarking

For the benchmarking we will use a software application that passes through the whole toolflow and makes use of every individual tool. This means that the software application should be using both dynamically and statically allocated data. Therefore, a video encoding application with wireless connectivity is chosen (e.g., as in a wireless camera).

More specifically, we will use the MPEG4 encoder which will feed data to a software implementation of the Transport or Network layer (as defined in the OSI reference model). A detailed description of the MPEG-4 application was discussed in an earlier deliverable [8]. A detailed description of the potential Transport or Network layer applications (e.g., implementation of IPv4 source code or the Deficit Round Robin scheduling source code will) can be found in the Netbench suite [14].

The main target of the integrated toolflow benchmarking will be to assess whether each individual tool can perform the individual source-to-source transformations and still produce a source code output that would be acceptable as source code input for the next tool in the toolflow.

Given the fact that the individual tools perform a variety of optimizations, which address a variety of potential problems in a software application, it is clear that it is not possible to test all the complimentary optimizations of all the individual tools in a single software application. Therefore, all the optimizations will be shown at the individual tool level with different applications that exhibit all the different problems/bottlenecks, which are necessary to showcase the respective optimizations (as discussed in Section 7).

Nevertheless, each of the industrial partners will use some of the tools in combination in Tasks 5.6 and 5.7 and will be able to report combined optimizations of the tools by comparing optimized and non-optimized versions of their source code. Additionally, the rest of the partners are exploring the option of running as many tools as possible in conjunction on a particular platform in the MPARM simulation framework [15] and reporting combined optimization results in Task 5.5.

#### **8.4. Active tool support**

The individual tools in the toolflow will be supported by their respective owners, as described in Section 7. Additionally, every tool and its respective owner is responsible to provide a source code output that is accepted as source code input by the tool further down the flow.

Finally, ICD has setup a webpage, as part of the dissemination activities, which offers to organize the respective service and maintenance as required, in cooperation with the developers of the project results.

The webpage can be accessed at: <http://www.icd.de/es/eu-results.html>

### **9. Conclusions**

The goal of this deliverable is to provide details regarding the quality assurance as applied to the MNEMEE project and regarding the management mechanisms and technical criteria. For the project management mechanisms, we presented a set of rules that was agreed upon by the project consortium and consists of “Best Practices” and “Uniform project Templates”, which guarantee the quality of the planning and execution within the project. Regarding the technical criteria, documentation and code standards, benchmarking and tool support mechanisms are explained for the individual partners and their respective tools as well as for the integrated MNEMEE toolflow.

## 10. References

- [1] NASA-STD-2201-93 "Software Assurance Standard", 10 November 1992
- [2] IMEC vzw, ATOMIUM Data Transfer and Storage Exploration optimization tools : <http://www.imec.be/design/atomium>
- [3] IMEC vzw, SPRINT source code parallelization tool : <http://www.imec.be/design/sprint>
- [4] Arnout Vandecappelle, E.D.G., Sven Wuytack, Clean C for MP-SoC. 2007
- [5] MPEG-4. Information technology coding of audio-visual objects. Technical report, ISO/IEC 14496-2
- [6] F. Bouwens, M. Berekovic, G. Gaydadjiev, B. De Sutter, "Architecture enhancements for the ADRES coarse-grained reconfigurable array", High Performance Embedded Architectures and Compilers. 3rd International Conference – HiPEAC, April 2008
- [7] J.-Y Mignolet, R. Wuyts, "Embedded multi-processor systems-on-chip programming," in Software, IEEE , vol.26, no.3, May-June 2009
- [8] MNEMEE deliverable D1.2, Scenario identification, analysis of static and dynamic data accesses and storage requirements in relevant benchmarks, December, 2008
- [9] Trac open source project, <http://trac.edgewall.org>
- [10] R Baert, E De Greef, E Brockmeyer, G Vanmeerbeeck, P Avasare, J Mignolet, M Cupak, "An automatic scratch pad memory management tool and MPEG-4 encoder case study", in the proceedings of The 45th Design Automation Conference (DAC), 2008
- [11] R Baert, E Brockmeyer, T Ashby, S Wuytack, "Exploring parallelizations of applications for MPSoC platforms using MPA", in the proceedings of Design Automation & Test in Europe, April, 2009
- [12] S. Stuijk, M.C.W. Geilen and T. Basten. "SDF3: SDF For Free". In 6th International Conference on Application of Concurrency to System Design, ACS D 2006, Proceedings. IEEE, June 2006, pp. 276-278. SDF3 is available via [www.es.ele.tue.nl/sdf3](http://www.es.ele.tue.nl/sdf3)
- [13] Vdrift 3D racing game, <http://vdrift.net/>
- [14] G. Memik and et al. Netbench: A benchmarking suite for network processors. In Proc. of ICCAD. IEEE Press, 2001.
- [15] MPARAM, a multi-processor cycle-accurate architectural simulator. <http://www-micrel.deis.unibo.it/sitonew/research/mparm.html>

## 11. Glossary

<b>MNEMEE</b>	Memory management technology for adaptive and efficient design of embedded systems
<b>CAD</b>	Computer Aided Design
<b>DLC</b>	
<b>DMT</b>	Discrete Multi-Tone
<b>DSP</b>	Digital Signal Processing
<b>FFT</b>	Fast Fourier Transform
<b>HW</b>	Hardware
<b>IC</b>	Integrated Circuit
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>MIMO</b>	Multiple Input Multiple Output
<b>MRC</b>	Maximum Ratio Combining
<b>OFDM</b>	Orthogonal Frequency Division Multiplexing
<b>OFDMA</b>	Orthogonal Frequency Division Multiple Access
<b>PHY</b>	Physical layer
<b>QoS</b>	Quality of Service
<b>SNR</b>	Sound to Noise Ratio
<b>SoC</b>	system on chip
<b>SOFDMA</b>	Scalable OFDMA
<b>STC</b>	Space-Time Coding
<b>SW</b>	Software
<b>WiMAX</b>	Worldwide Interoperability for Microwave Access

