The successful usage of Hardware Description Languages like VHDL and Verilog in the design flow is mainly due to the availability of efficient synthesis methods and tools that enable the translation of RTL designs into optimized gate-level implementations. Many expect that the same approach could be effectively adapted at higher levels of abstraction despite the failure of previous attempts to behavioural synthesis from higher-level HDL descriptions.

High-level languages like SystemC or SystemVerilog that offer advanced modelling techniques and verification methods and tools based on them are in progress to enable more efficient verification process. But in order to provide the designers with an efficient automated path to implementation some kind of system synthesis is required.

Some automated approaches are nowadays proposed by industry and research to replace the manual recoding of high-level models into synthesize-able / implementable models:

- automated behavioural synthesis from higher abstraction levels
- translation of high-level models (e.g. in SystemC) to HDL RTL models and synthesis

The main expectations from the systems’ industry concern methods and tools supporting better management of the design complexity and reduction of the design cycle all together, breaking the trend to compromise evaluation of various design implementation options. Designing at higher levels of abstraction is an obvious direction as it allows to better manage the system design complexity, to verify earlier in the design process with higher performance and to increase IP reuse.

Thus, users formulate several crucial questions with regard to system synthesis:

- Is the so called “behavioural synthesis” an appropriate and efficient solution to get to the implementation?
- Is it optimal (the RTL optimized, targeted to downstream synthesis tools)?
- What benefit can users gain and at what price: how does it influence the designers productivity, the design flow and the way of exploring multiple implementation alternatives, trade-offs management, …?
- What about the verification flow (reuse of test-bench, back-annotation from implementation to specification)?
- How can IP reuse be strengthened by high-level synthesis: automatic retargeting to different technologies, exploration of implementation alternatives and trade-offs?

The ECSI Workshop on System Synthesis will provide an overview of existing and emerging solutions provided by both industrial partners (EDA companies) and best research institutions in this domain. It will give an outline (as complete as possible) of methods and tools available currently on the market for system synthesis, bring the details on their applicability, performances and strengths. Finally, the event will create a discussion platform for experience exchange between ECSI members.

Some workshop presentations will be composed of theoretical and practical (demonstration) parts.

### ECSI Institute System Synthesis Workshop Agenda

<table>
<thead>
<tr>
<th>Sessions</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 Obtaining More Easily and Rapidly Models Ready for System Synthesis with CoFluent Studio</td>
<td>CoFluent</td>
<td>09:00</td>
<td>09:40</td>
</tr>
<tr>
<td>S2 Synthesizing SystemC to Silicon</td>
<td>Celoxica</td>
<td>09:40</td>
<td>10:20</td>
</tr>
<tr>
<td>S3 C-Based Virtual Prototyping</td>
<td>Mentor</td>
<td>10:20</td>
<td>11:00</td>
</tr>
<tr>
<td>Break</td>
<td>11:00</td>
<td>11:15</td>
<td>0:15</td>
</tr>
<tr>
<td>S4 Bridging TLM and Implementation</td>
<td>Summit</td>
<td>11:15</td>
<td>11:55</td>
</tr>
<tr>
<td>S5 From SystemC Transaction-Level Model to Implementation</td>
<td>Forte Design</td>
<td>11:55</td>
<td>12:35</td>
</tr>
<tr>
<td>Lunch</td>
<td>12:35</td>
<td>13:35</td>
<td>1:00</td>
</tr>
<tr>
<td>S6 Synthesize RTL SystemC Thanks to the Prosiol’s SC2HDL Compiler</td>
<td>Prosiol</td>
<td>13:35</td>
<td>14:15</td>
</tr>
<tr>
<td>S7 SystemVerilog from a Synthesis Perspective</td>
<td>Synopsys</td>
<td>14:15</td>
<td>14:55</td>
</tr>
<tr>
<td>S8 System Studio Enables SystemVerilog Code Generation from Data-Flow Descriptions</td>
<td>Synopsys</td>
<td>14:55</td>
<td>15:35</td>
</tr>
<tr>
<td>Break</td>
<td>16:25</td>
<td>16:40</td>
<td>0:15</td>
</tr>
<tr>
<td>S9 Embedded Processor Design For Programmable ASIPs</td>
<td>CoWare</td>
<td>16:40</td>
<td>17:20</td>
</tr>
<tr>
<td>S10 A Straight Forward Extension of SystemC’s Synthesis Subset</td>
<td>OFFIS</td>
<td>15:35</td>
<td>16:00</td>
</tr>
<tr>
<td>S11 High Level Synthesis of IP Cores Under I/O and Timing Constraints</td>
<td>LESTER</td>
<td>16:00</td>
<td>16:25</td>
</tr>
<tr>
<td>Demonstration Sessions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstration Session 1: CoFluent</td>
<td>CoFluent</td>
<td>17:20</td>
<td>17:45</td>
</tr>
<tr>
<td>Demonstration Session 2: Celoxica</td>
<td>Celoxica</td>
<td>17:45</td>
<td>18:10</td>
</tr>
<tr>
<td>Demonstration Session 3: Synopsys</td>
<td>Synopsys</td>
<td>18:10</td>
<td>18:35</td>
</tr>
</tbody>
</table>
Session 1: “Obtaining More Easily and Rapidly Models Ready for System Synthesis with CoFluent Studio”

Abstract: The adoption of system synthesis techniques implies a shift in designers’ focus and efforts from implementation models to system-level models like SystemC that serve as input to synthesis. CoFluent Design provides CoFluent Studio, a system modelling and simulation environment complementary to system synthesis tools as it helps chip designers create and verify transactional-level SystemC models (application + platform) more effectively and rapidly. After HW/SW partitioning based on prospective performance analysis, software functions are automatically translated into C tasks for the targeted RTOS, while the SystemC model of the system’s HW platform can be used for further verification and, eventually, synthesis.

Presenter: Vincent Perrier, CoFluent - Duration: 40 min

Session 2: “Synthesizing SystemC to Silicon”

Abstract: SystemC is gaining popularity and momentum for modelling and verification. Architectural analysis and HW/SW trade-offs can be more easily understood and designers benefit from simulation performance that is orders of magnitude faster than conventional RT level simulation.

To extract the maximum value from their SystemC modelling and verification efforts designers are increasingly looking for implementation paths from SystemC model to silicon. In this workshop we will present deterministic hardware synthesis for SystemC. Using 3rd generation C-synthesis technology SystemC can now be synthesized directly to high-density FPGA, or to RTL VHDL and Verilog for ASIC & SoC design.

Presenter: Chris Sullivan, Celoxica - Duration: 40 min

Session 3: “C-Based Virtual Prototyping”

Abstract: Today single-purpose systems are becoming more varied and advanced in the features they include. This trend is forcing engineers to face and address serious problems in system level design, architectural specification, refinement and optimization; IP selection, evaluation and integration; software development and system verification; system-wide analysis of performance and power; and full system validation and implementation.

Next-generation C-based design tools will offer high performance, easy assembly of virtual prototypes, leverage existing architectural models and untimed system blocks, and provide key IP functions. The tool flow and models chosen must be sufficiently detailed for rapid system analysis, performance and power verification with maximum throughput. Further, such an environment will need to incorporate legacy RTL IP into the virtual model as well as provide a path to accelerated downstream implementation, reusing system level investment.

This session will examine automation opportunities as they apply to typical IP-dominant, bus-centric systems. Constraints introduced by these types of designs finally enable creation of effective system level design and implementation technologies. Tool categories will be defined and a range of solutions will be discussed, including C-based design, synthesis, analysis and verification linked to downstream software and hardware implementation.

Presenter: Staffan Berg – Mentor Graphics Product Specialist: HW/SW Co-design - Duration: 40 min

Session 4: “Bridging TLM and Implementation”

Abstract: System Level Design is generally accepted as the next step of the EDA to face the increasing complexity of system design and verification. To this extend SystemC is often the language of choice.

Adopting a new language, as well as a new methodology, always takes time. If significant progresses have been made in the methodology area, there is still a big stone missing to complete the foundations of this new approach; this is SystemC synthesis, an automatic way to change level of abstraction.

Since the year 2000, Summit-Design is providing tools to help designers addressing the System Level Design challenges. Not only Summit's Visual Elite solution allows creating and verifying complex systems, using any combination of C/C++/SystemC/VHDL and Verilog modules, but it also includes SystemC to HDLs translation. For each C/C++/SystemC elements of the system, user can decide to either use 3rd party synthesis technology to produce HDLs descriptions, or use SystemC RTL to HDL automatic translation. In a unique and scalable platform, engineers have access to state of the art System Level to RTL technologies and get ready for the future System synthesis.

Presenter: Jean-Marie Saint-Paul, Summit Design - Duration: 40 min

Session 5: “From SystemC Transaction-Level Model to Implementation”

Abstract: The presentation will elaborate on the design process from the starting point of a TLM behavioural model or a BCA behavioral model to the system description at the abstraction level appropriate for behavioural synthesis.

Presenter: Michael Meredith – Forte Design - Duration: 40 min

Session 6: “Synthesize RTL SystemC Thanks to the Prosilog’s SC2HDL Compiler”

Abstract: SystemC is a new standard language for systems description at several levels of abstraction (behavioural, transactional and also RTL) and a lot of companies have decided to integrate it in their design flow. However, SystemC is not dedicated to replace existing HDL languages, but rather cohabit with them. In this context, Prosilog has developed a set of efficient compilers, which ensure a fast translation between these languages.
For those who decided to use SystemC as an implementation language, from transactional down to RTL, the biggest issue that appears is the lack of a real offer of synthesizers taking SystemC as entry. The SystemC to VHDL or Verilog compiler (SC2HDL) carries out in a few seconds the extremely tedious translation task, which could take weeks or at least days in best cases. Thus, SystemC design flow can be seamlessly linked to the back-end flow in order to take advantage of existing tools such as HDL synthesizers.

In this workshop, after showing the interest of methodologies using SystemC as HDL language, we will focus on the synthesizable SystemC subset accepted in entry by the SC2HDL compiler; we will see that VHDL/Verilog designers can keep their coding style rules when moving to SystemC and several application examples will be presented to well understand how all RTL designs can be described in SystemC for synthesis purpose.

**Presenter:** Xavier SOLAU, Prosilog - **Duration:** 40 min

**Session 7: “SystemVerilog from a Synthesis Perspective”**

**Abstract:** SystemVerilog is the industry’s first standard Hardware Description and Verification Language (HDVL), unifying design and verification capabilities in a single language, delivered as an easy-to-learn extension of Verilog. With the addition of concepts such as communication interfaces, class system extensions, concise mechanisms for describing functionality, designers using SystemVerilog require 2-5X less code to capture the same functionality with no impact on synthesis QoR. This presentation will examine a number of design features in SystemVerilog for their synthesizability and their usefulness to designers.

**Presenter:** Peter Flake - Synopsys Scientist - **Duration:** 40 min

**Session 8: “System Studio Enables SystemVerilog Code Generation from Data-Flow Descriptions”**

**Abstract:** System Studio is the world-leading system-level design environment, tightly integrated into Synopsys’ Discovery verification platform. It is unique in its ability to serve two mission-critical system-level design areas for innovative system-on-chip (SoC) applications: algorithms and architectures. This presentation will address one aspect of the algorithm design flow.

In System Studio, algorithms are captured, verified and optimized using intuitive dataflow graphs. System Studio is the world’s fastest simulator of algorithmic control and dataflow systems. Traditionally the algorithm description was provided to an RTL designer, who would manually recode at RTL. With the latest release of System Studio this gap no longer exists: System Studio now allows to generate synthesizable SystemVerilog from the same C-based system description as used for algorithm simulation. Finally a single description is sufficient to serve two needs: enjoy algorithm simulation and analysis at highest speed, while being able to automatically take your algorithm to RTL. This enables consistency between algorithms and RTL at any time in the design flow.

**Presenter:** Philipp Krueckel - Synopsys R&D Manager - **Duration:** 40 min

**Session 9: “Embedded Processor Design For Programmable ASIPs”**

**Abstract:** ASIPs are a way to increase designers productivity and introduce design flexibility to address changing standards and specs. In this presentation we present the LISATEK toolset which allows to architect and design ASIPs and that also provides with the SW tool-chain.

**Presenter:** Bart Vanthournout - Product Architect - CoWare- **Duration:** 40 min

**Session 10: “A Straight Forward Extension of SystemC’s Synthesis Subset”**

**Abstract:** Though SystemC is completely implemented in C++, existing synthesis tools able to process SystemC do not support user-defined classes, at least not on a level which would go beyond simple records. This is a great pity, because a support of the C++ class concept would augment hardware design by object-oriented elements, and would therefore bear the prospect to raise the level of abstraction and increase design productivity.

Within the European FP5 project ODETTE, a, yet prototypic, tool was developed which is able to pre-process SystemC specifications including such C++ features, generating VHDL or SystemC code which can then be further processed with existing synthesis tools. Moreover, the ODETTE synthesis technology does not only extend the SystemC/C++ synthesis subset, it also extends SystemC itself by a pointer-free polymorphism concept, and monitor-like shared objects. The presentation will include an overview of the main features included in the synthesizable subset, called OSSS, and the underlying synthesis technology.

**Presenter:** Eike Grimpe, OFFIS - **Duration:** 25 min

**Session 11: “High Level Synthesis of IP Cores Under I/O and Timing Constraints”**

**Abstract:** Re-use of complex Digital Signal Processing coprocessors can be improved using IP cores described at a high abstraction level. System integration, that is a major step in SoC design, requires taking into account communication and timing constraints to design IP and wrappers. We present an IP design approach under I/O and timing constraints that relies on three main phases: constraints modelling, IP constraints analysis steps for feasibility checking and synthesis. We propose a set of techniques that guarantee an optimized IP integration by reducing I/O buffer size and protocol wrapper design. Based on a generic architecture of the communication unit, the presented method provides automatic generation of IP cores designed under integration constraints.

**Presenter:** Adel Baganne - LESTER Lab - **Duration:** 25 min