DSPACE: LISA Methodology

Maximilian Odendahl

ESA DSP Day, Noordwjik, 28.8.2012

Institute for Communication Technologies and Embedded Systems

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Outline

- **Introduction**
  - DSPACE Requirements
  - LISA Methodology
- **LISA 2.0 Language**
  - Overview
  - Resources
  - Operations
- **DSPACE Results**
- **Demo**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

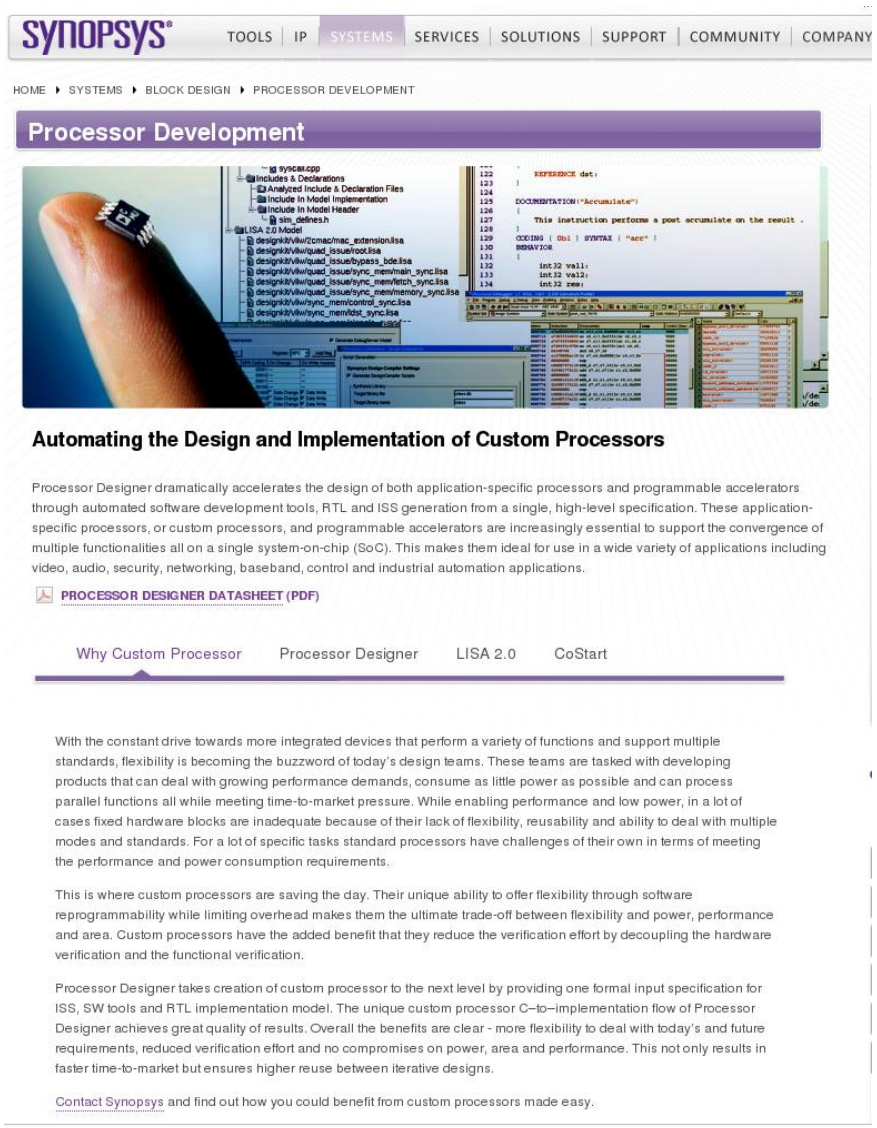
- **Issues and Requirements:**
  - Final architecture not known at the beginning
  - Should be easily adaptable and extendable to cope with future standards and algorithms
  - Needs a mature and stable software development environment
  - Relatively short project time frame

➢ **Raise the abstraction level of the processor model**

➢ **Use of a Processor Development Environment**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

- **History:**
  - LISA project at ICE
  - LISATek Inc.
  - CoWare Inc.
- **Integrated, embedded processor development environment**
- **Unified processor model in LISA 2.0 language**
- **Automatic generation of:**
  - SW tools
  - HW models



SYNOPSYS® TOOLS | IP | SYSTEMS | SERVICES | SOLUTIONS | SUPPORT | COMMUNITY | COMPANY

HOME ▸ SYSTEMS ▸ BLOCK DESIGN ▸ PROCESSOR DEVELOPMENT

Processor Development

Automating the Design and Implementation of Custom Processors

Processor Designer dramatically accelerates the design of both application-specific processors and programmable accelerators through automated software development tools, RTL and ISS generation from a single, high-level specification. These application-specific processors, or custom processors, and programmable accelerators are increasingly essential to support the convergence of multiple functionalities all on a single system-on-chip (SoC). This makes them ideal for use in a wide variety of applications including video, audio, security, networking, baseband, control and industrial automation applications.

PROCESSOR DESIGNER DATASHEET (PDF)

Why Custom Processor | Processor Designer | LISA 2.0 | CoStart

With the constant drive towards more integrated devices that perform a variety of functions and support multiple standards, flexibility is becoming the buzzword of today's design teams. These teams are tasked with developing products that can deal with growing performance demands, consume as little power as possible and can process parallel functions all while meeting time-to-market pressure. While enabling performance and low power, in a lot of cases fixed hardware blocks are inadequate because of their lack of flexibility, reusability and ability to deal with multiple modes and standards. For a lot of specific tasks standard processors have challenges of their own in terms of meeting the performance and power consumption requirements.

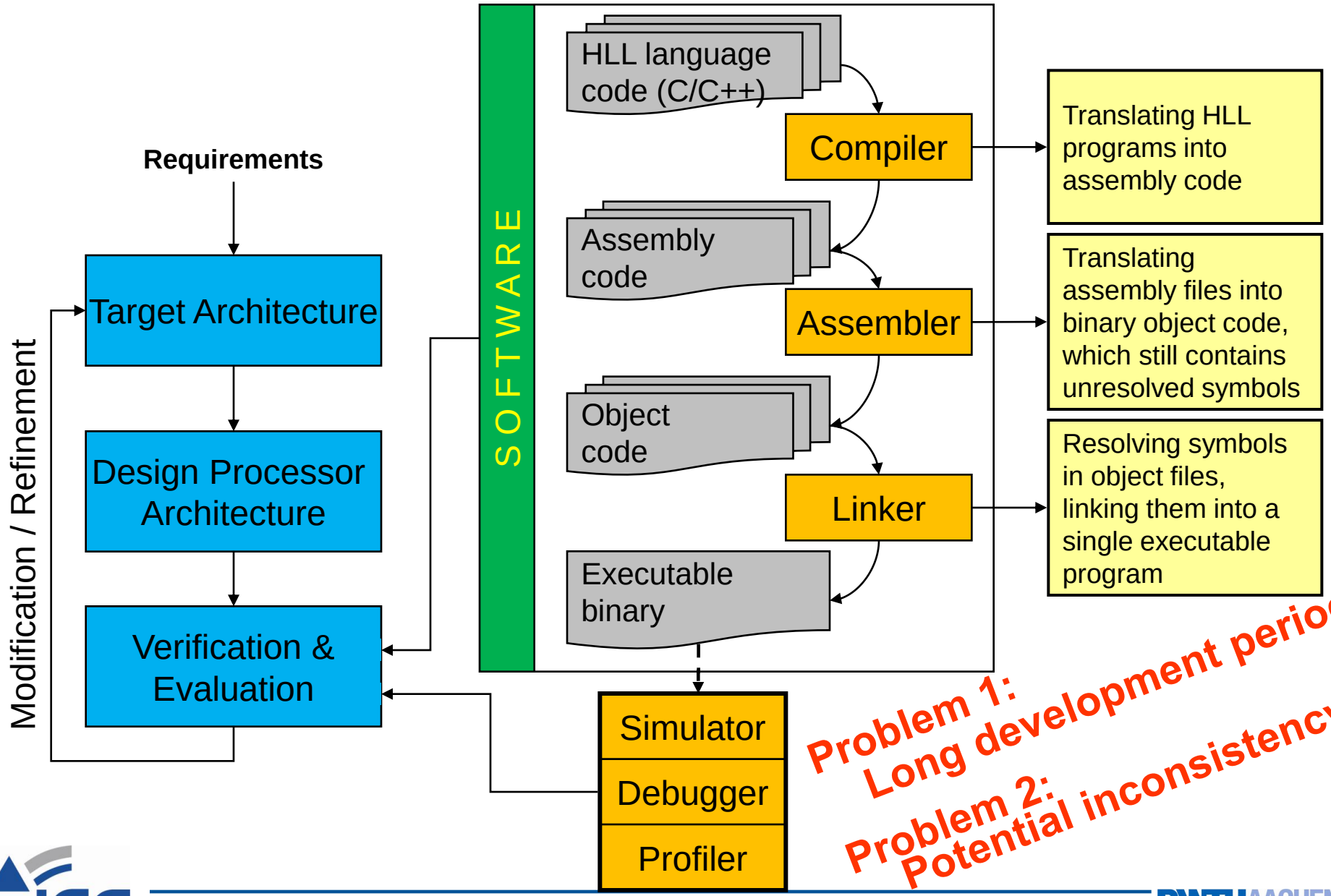
This is where custom processors are saving the day. Their unique ability to offer flexibility through software reprogrammability while limiting overhead makes them the ultimate trade-off between flexibility and power, performance and area. Custom processors have the added benefit that they reduce the verification effort by decoupling the hardware verification and the functional verification.

Processor Designer takes creation of custom processor to the next level by providing one formal input specification for ISS, SW tools and RTL implementation model. The unique custom processor C–to–implementation flow of Processor Designer achieves great quality of results. Overall the benefits are clear - more flexibility to deal with today's and future requirements, reduced verification effort and no compromises on power, area and performance. This not only results in faster time-to-market but ensures higher reuse between iterative designs.

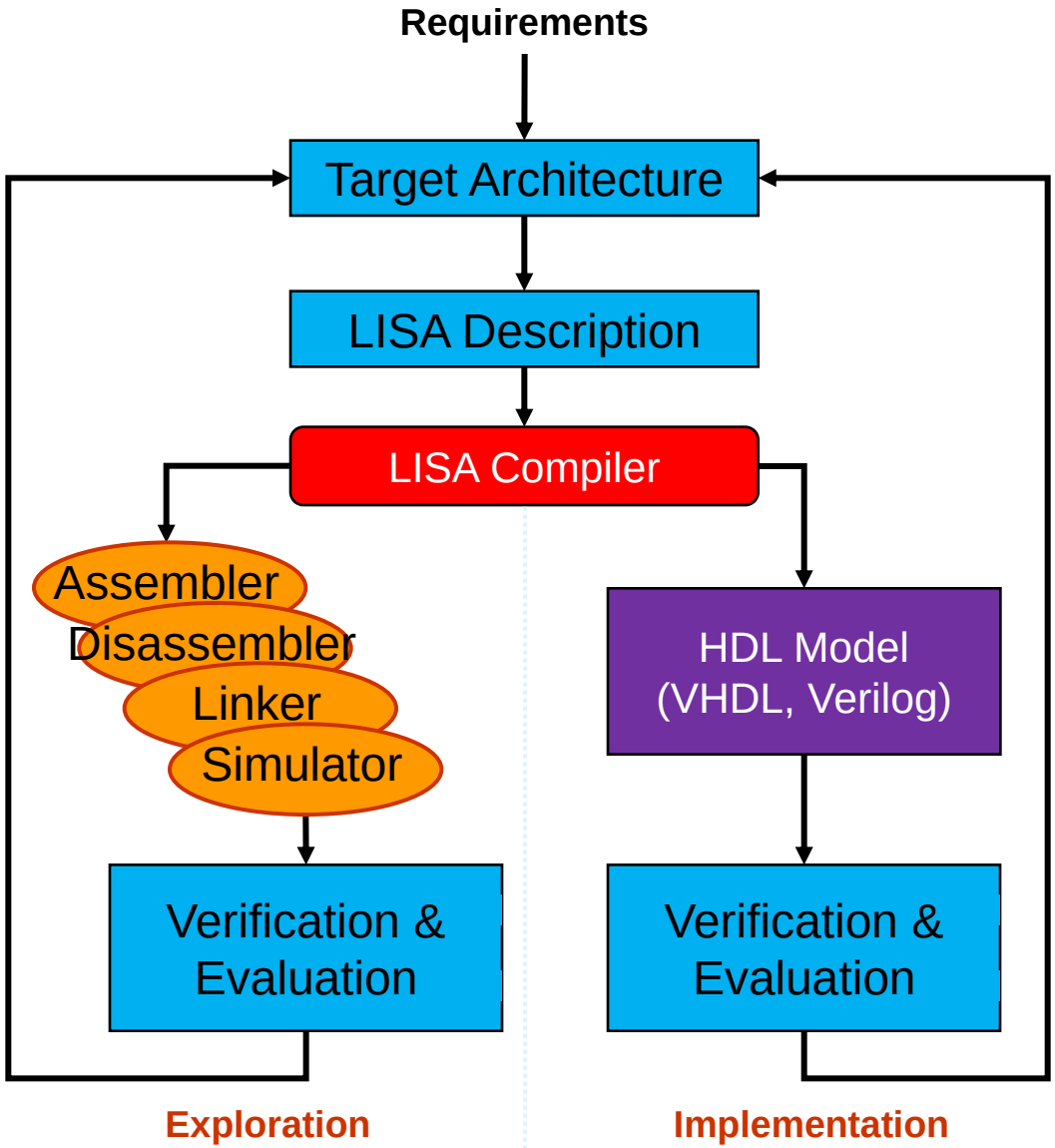
Contact Synopsys and find out how you could benefit from custom processors made easy.

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Traditional Processor Architecture Exploration



The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

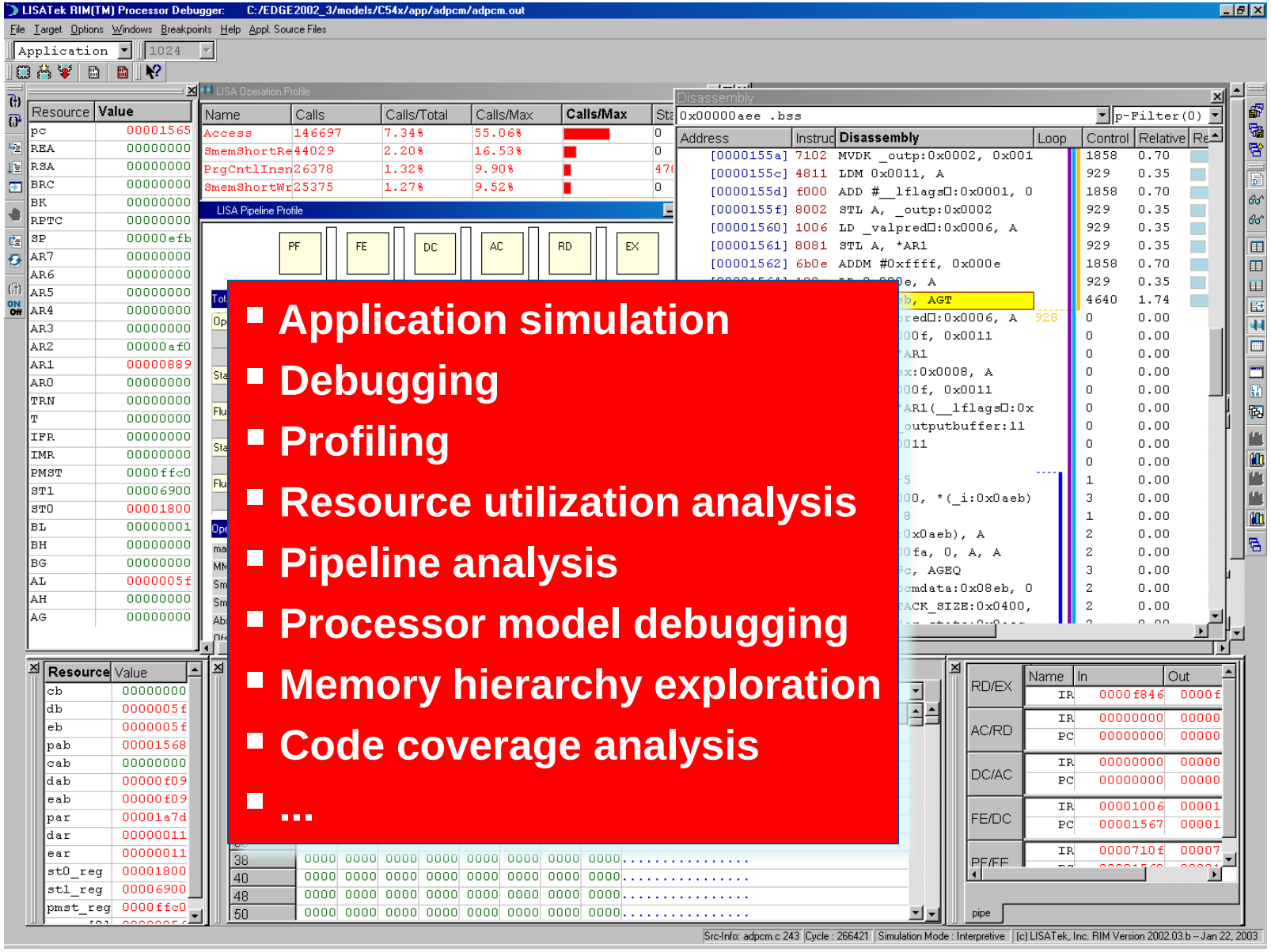


Main advantages of LISA:

- **Modeling and modification of processors at a high level of abstraction.**
- **Automatic generation of software development tools**
  - Shortening the development time
  - Consistency
- **Generation of HDL Description**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Exploration/debugger GUI



The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Outline

- **Introduction**
  - DSPACE Requirements
  - LISA Methodology
- **LISA 2.0 Language**
  - Overview
  - Resources
  - Operations
- **DSPACE Results**
- **Demo**

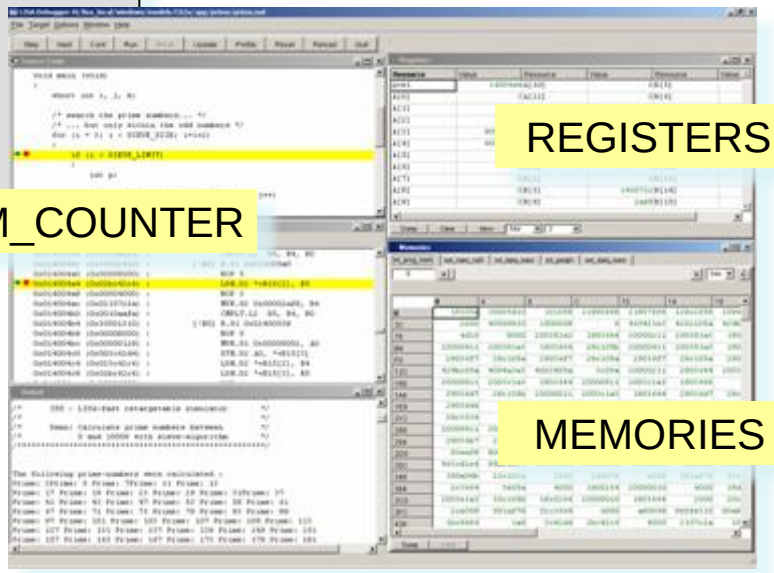
The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

- **Processor Description Language**
- **Formalized Description of**
  - Processor Resources
  - Instruction Set
  - Abstracted Hardware Behavior
  - Timing
- **Organized in Operations**
  - Modularity
  - Reusability
- **C/C++ based**
  - easy to learn
  - integrate existing IP
- **Multiple abstraction levels**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

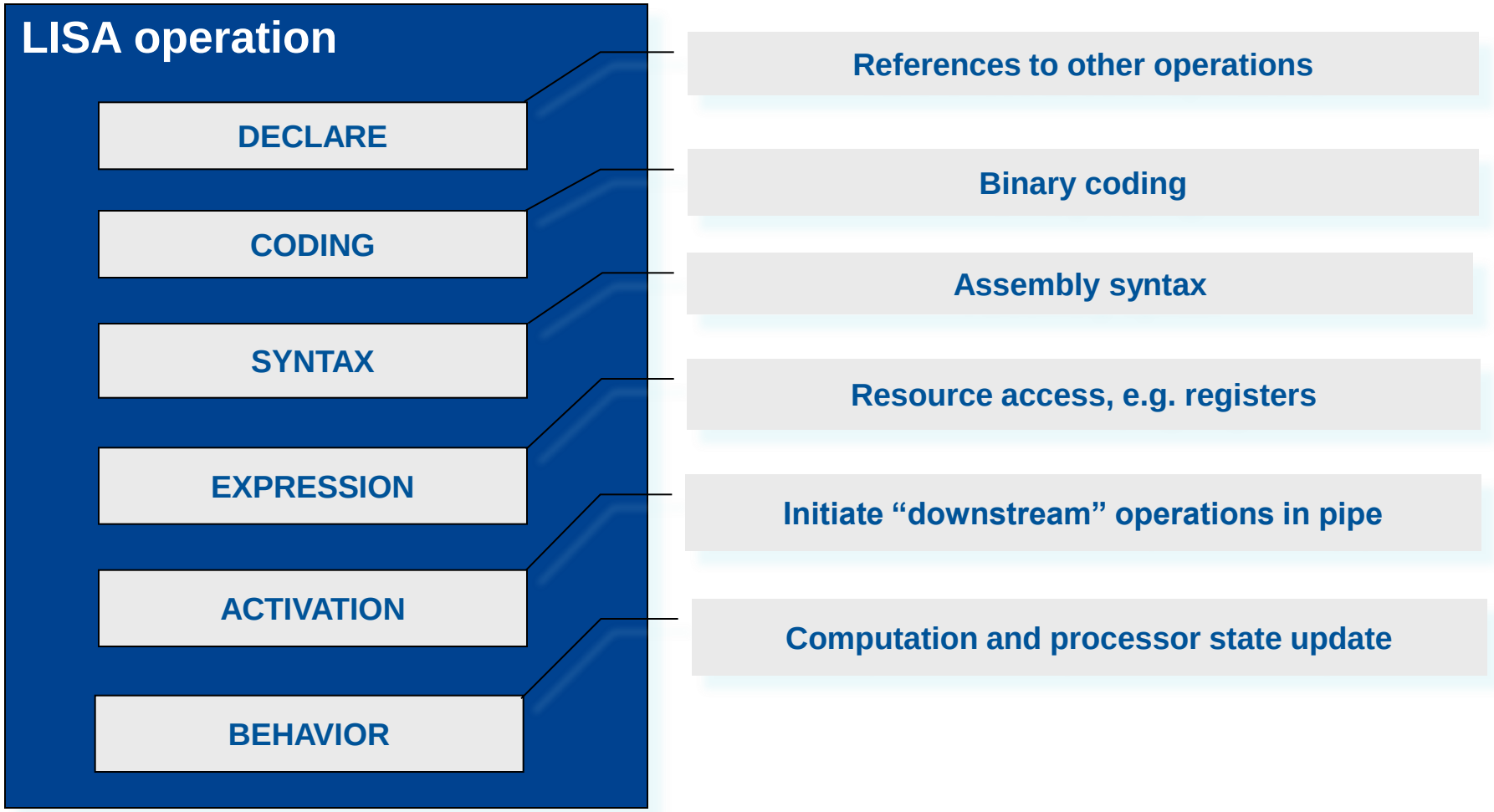
- **RESOURCE section provides a unique, centralized definition of all processor resources e.g. registers, memories, etc.**

```
RESOURCE
{
   PROGRAM_COUNTER long PC;
   REGISTER bit[12] R[0..31];
   CONTROL_REGISTER short ST;
   PIN IN bit[5] IRQ, NMI;
   RAM char pmem  { ... };
   RAM char dmem1 { ... };
   RAM int  dmem2 { ... };
}
```



The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

LISA Operation Sections



The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

```
OPERATION add
{




 BEHAVIOR {
   R[dest] = R[src1] + R[src2];
 }
}
```

C Code

- **OPERATIONs are used to describe the behavior of functional units in the processor.**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

```
OPERATION add
{
 DECLARE {
   GROUP src1 = { reg };
   GROUP src2 = { reg };
   GROUP dest = { reg };
 }


 BEHAVIOR {
   R[dest] = R[src1] + R[src2];
 }
}
```

- **OPERATIONs are used to describe the behavior of functional units in the processor.**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

```
OPERATION add
{
 DECLARE {
   GROUP src1 = { reg };
   GROUP src2 = { reg };
   GROUP dest = { reg };
 }
 CODING {0b10001 dest src1 src2}
 SYNTAX {"add" src1 "," src2 "," dest}
 BEHAVIOR {
   R[dest] = R[src1] + R[src2];
 }
}
```

- **Additionally, OPERATIONs contain information about coding and syntax.**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

OPERATION (4)

```
OPERATION add
{
 DECLARE {
    GROUP src1 = { reg_idx };
    GROUP src2 = { reg_idx };
    GROUP dest = { reg_idx };
 }
 SYNTAX {"add" src1 "," src2 "," dest}
 CODING {0b10001 dest src1 src2}
 BEHAVIOR {
    dest = src1 + src2;
 }
}
```

```
add r3,r4,r5
```

Assembler+Linker

```
10001001100000001
```

Loader+Decoder

```
r5 = r3 + r4;
```

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Outline

- **Introduction**
  - DSPACE Requirements
  - LISA Methodology
- **LISA 2.0 Language**
  - Overview
  - Resources
  - Operations

Results

- **Demo**

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

- **DSPACE core**
  - running at 125 MHz
    - 1.0 GOPS
    - 750 MFLOPS
    - Increase of 17 and 13 times, respectively, compared to Atmel‘s TSC21020F
  - area of 370 kgates
    (180nm standard-cell library, typical)
  - 10.000 lines of LISA source code
  - 300.000 lines of generated VHDL

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798

Thank you for your attention!

Questions?

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement n°262798