



# **Technological Educational Institute of Western Macedonia - Greece**

- **Dr Michael Dossis, Assist. Professor**
- **School of Kastoria**
- **Department of Informatics and  
Computer Technology**

# **A WEB SERVICE TO GENERATE PROGRAM COPROCESSORS**

- **Motivation and Background**
- **Hardware/software compilation flow and tools**
- **High-level synthesis inference transformations**
- **Web interface and compiler RDF relations**
- **Experimental results**

# MOTIVATION

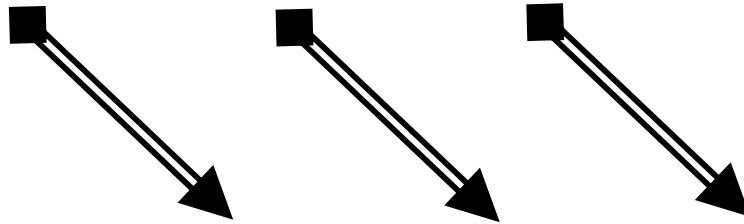
- **Traditional software development**
  - **Structured, functional, logic, object-oriented programming**
  - **Assemblers, compilers, interpreters**
  - **Debugging tools (trace)**
- **Traditional hardware development**
  - **Schematic entry**
  - **HDL entry (e.g. VHDL, Verilog)**
  - **Silicon compilers (e.g. using Model, Abel)**
  - **Behavioural and netlist simulators**
  - **Variety of vendor-specific back-end, layout level CAD tools**
- **Separate and dis-joint methods for system development**
  - **No design flow for mixed hardware-software embedded systems**
  - **Leads to long, tedious and repetitive spec-to-product cycles**
  - **Lack of unified automation leads to last-moment bug fixes**
  - **A lot of products miss the market window**

# **EVOLUTION of TOOLS and METHODS**

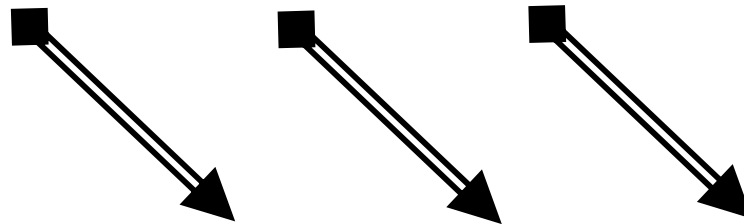
- **CASE tools**
  - **Diagrammatic and object-oriented programming**
  - **Data-flow, control-flow, object-class diagrams**
  - **UML, SysML**
  - **Vendor-specific, application-specific, non-portable and complex spec. formats**
- **High-level Synthesis**
  - **Portable, high-level exec. specification in behavioural program code**
  - **Algorithmic entry (C, Pascal, Ada, etc.)**
  - **High-level hardware compilers**
  - **Generates HDL RTL, commercially implementable model**
  - **Variety of vendor-specific back-end, layout level CAD tools**
- **Formal Verification and Model Checking**
- **Verification languages**
  - **System-C, System-Verilog, PSL (all IEEE standards)**

## Pressing need to raise the abstraction level in hardware specifications (VLSI & ULSI design)

VLSI technology advances allow for extremely high complexity of modern ICs



Design/development cycle often surpasses product lifetime in the market



Higher degree of abstraction and automation are required in the design of digital systems

# Why do we need higher Abstraction?

- **To achieve more dense and easily read spec. code**
  - **Building the system model in orders of magnitude shorter time is crucial**
- **To deliver technology-neutral exec. Specification code**
- **To intentionally hide implementation details from the system designer**
- **To shorten verification cycle time**
  - **To avoid or minimize time-consuming cycle-accurate or RTL simulations**
  - **Verification with compile and execute instead of simulations**
- **To shorten spec. to product time**
  - **So that system is delivered to market on time**
- **To eliminate functional bugs earlier in the dev. cycle**
- **To automatically deliver implementable models**
- **To achieve model portability across implementation tools**
- **To automate the design and verification cycles**

# MOTIVATION

- **Motivation for unified design of mixed SW/HW systems**
- **Use of number of MPU statistics in 1997**
  - **In personal computers : 30 million**
  - **In embedded systems : 3 billion (10.76 billion in 2009)**
- **Applications for micro-controllers and embedded systems**
  - **Consumer electronics**
    - **Microwave ovens**
    - **Cameras**
    - **Compact Disk units**
  - **Telecommunication and networks**
    - **Telephone switches**
    - **Cell (mobile) and wireless telephone devices**
    - **Portable and non-portable antenna systems**
    - **Portable and non-portable broadcast radio and receivers**
  - **Automobile industry**
    - **Engine controllers**
    - **Brake control systems (ABS)**
  - **Industrial control**
    - **CNC and robotic systems**
    - **Industrial installation control terminals**

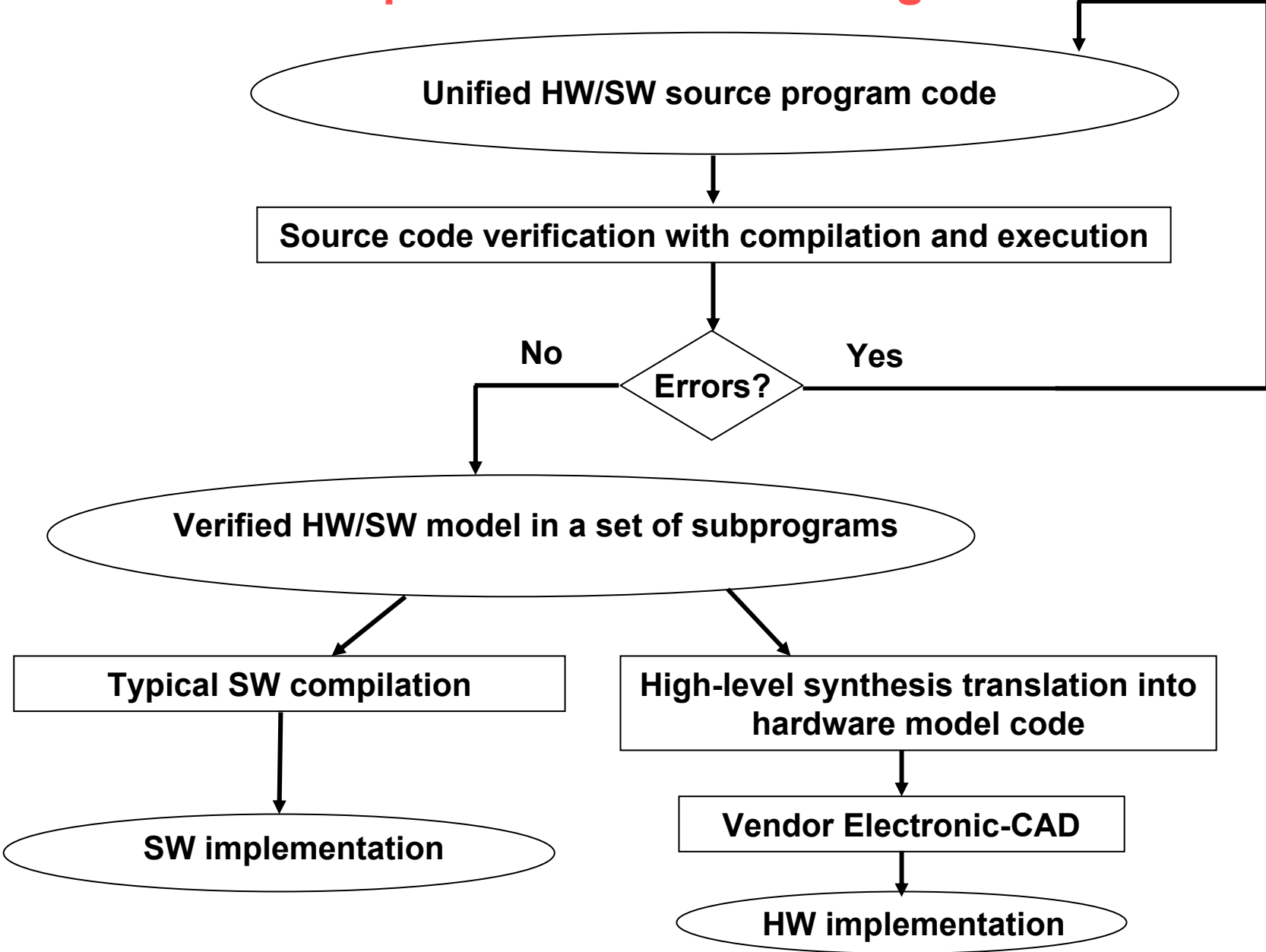
# Existing High-Level Synthesis technology

- **Some HLS tools use specialized, proprietary languages**
  - **New language skills are required**
- **Most accept only restricted, small syntactic subsets of computer languages**
  - **It is difficult to implement complete system models with structure and hierarchy**
- **Specifying systems in high level languages result in fewer lines of code**
  - **It's easy to maintain and with faster updates, than using traditional schematic entry or HDL RTL models**
  - **Faster to remove most of functional bugs earlier in the design flow**
  - **Executing high-level code is faster than simulation by orders of magnitude**
  - **Higher productivity levels are achieved**
- **HLS tools produce usually larger HDL code and with lower performance than the HDL code produced by experienced hardware designers**
  - **However, the HLS tool optimizations are continuously improving, offering an ever better quality of hardware implementations than in the past**

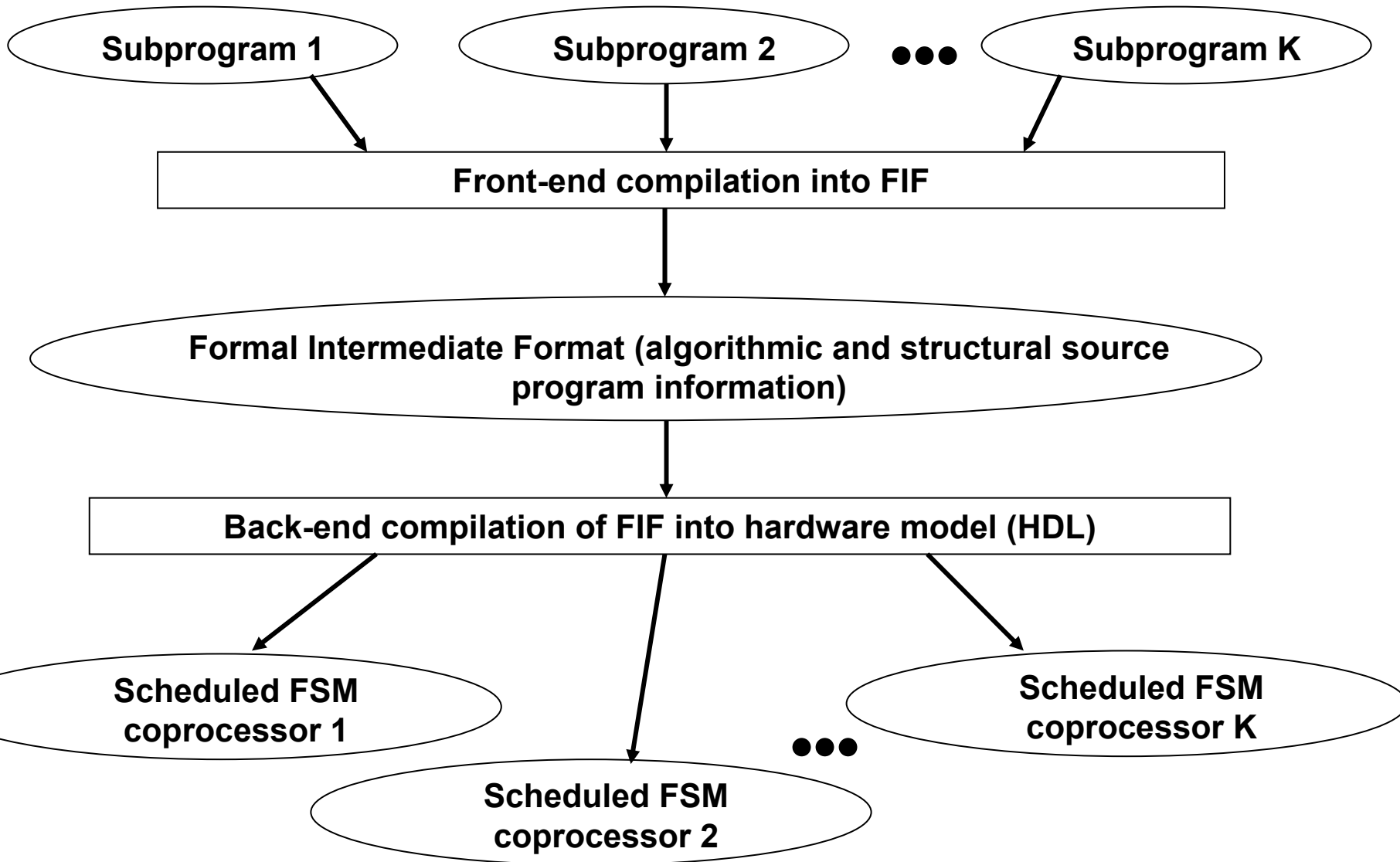
# Proposed HW/SW co-design methodology

- **Based on translation of customized co-processors from program code into hardware model (Custom Co-processor Compilation – C<sup>3</sup>)**
- **Our method starts with unified mixed HW/SW system executable specs in a popular system programming language**
  - Present version of prototype tools use ADA as source language
  - Extensions for other input languages such as ANSI C are under development
  - Allows to select a number of subprogram modules to translate into hardware
- **Achieves software implementation from verified executable specifications**
  - Using established structured programming
  - Directly and automatically using SW compilation
  - Automatically and without alterations of the source programs
- **Achieves hardware implementation from verified executable specifications**
  - Using established structured programming
  - Directly and automatically using the C<sup>3</sup> hardware compiler
  - Automatically and without alterations of the source programs

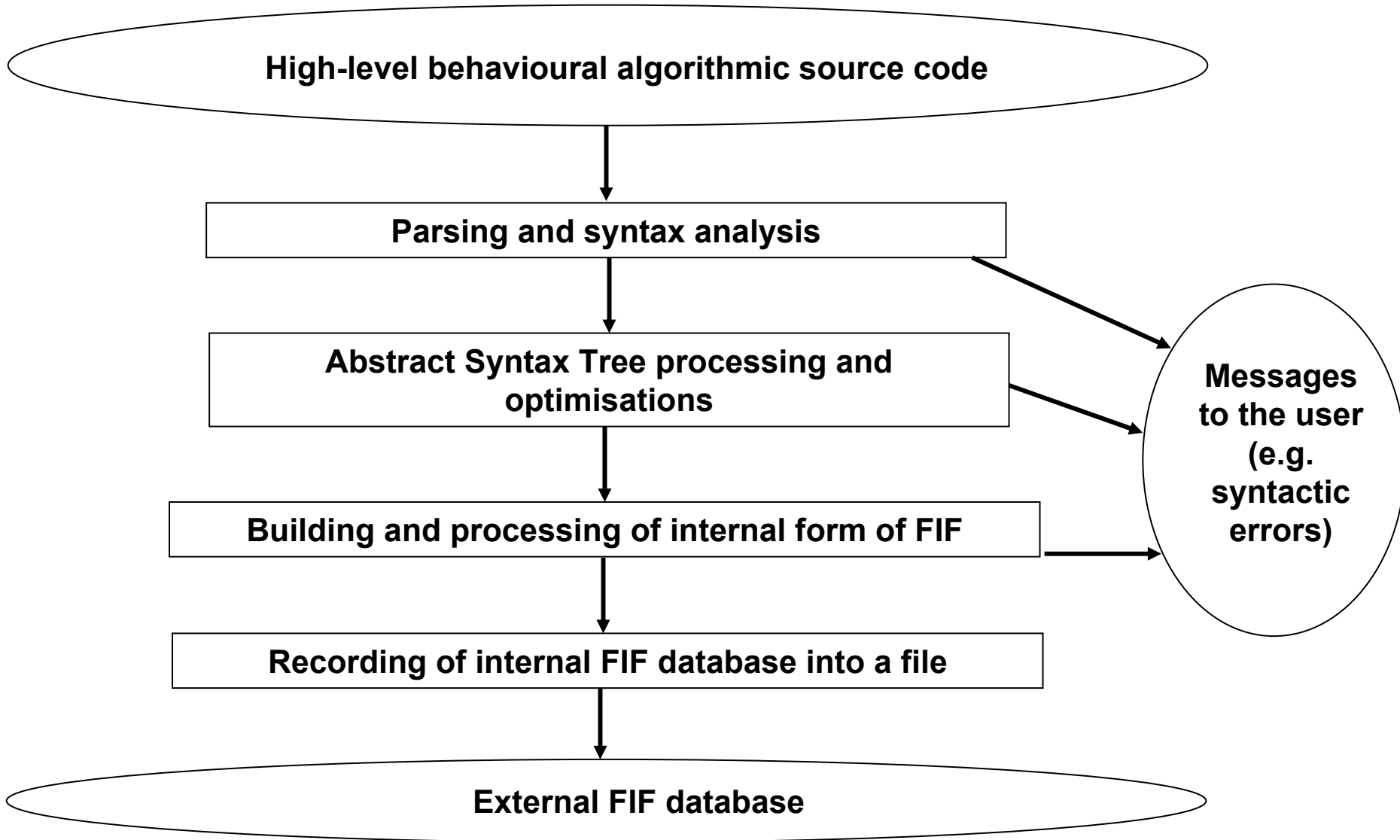
# Proposed SW/HW co-design flow



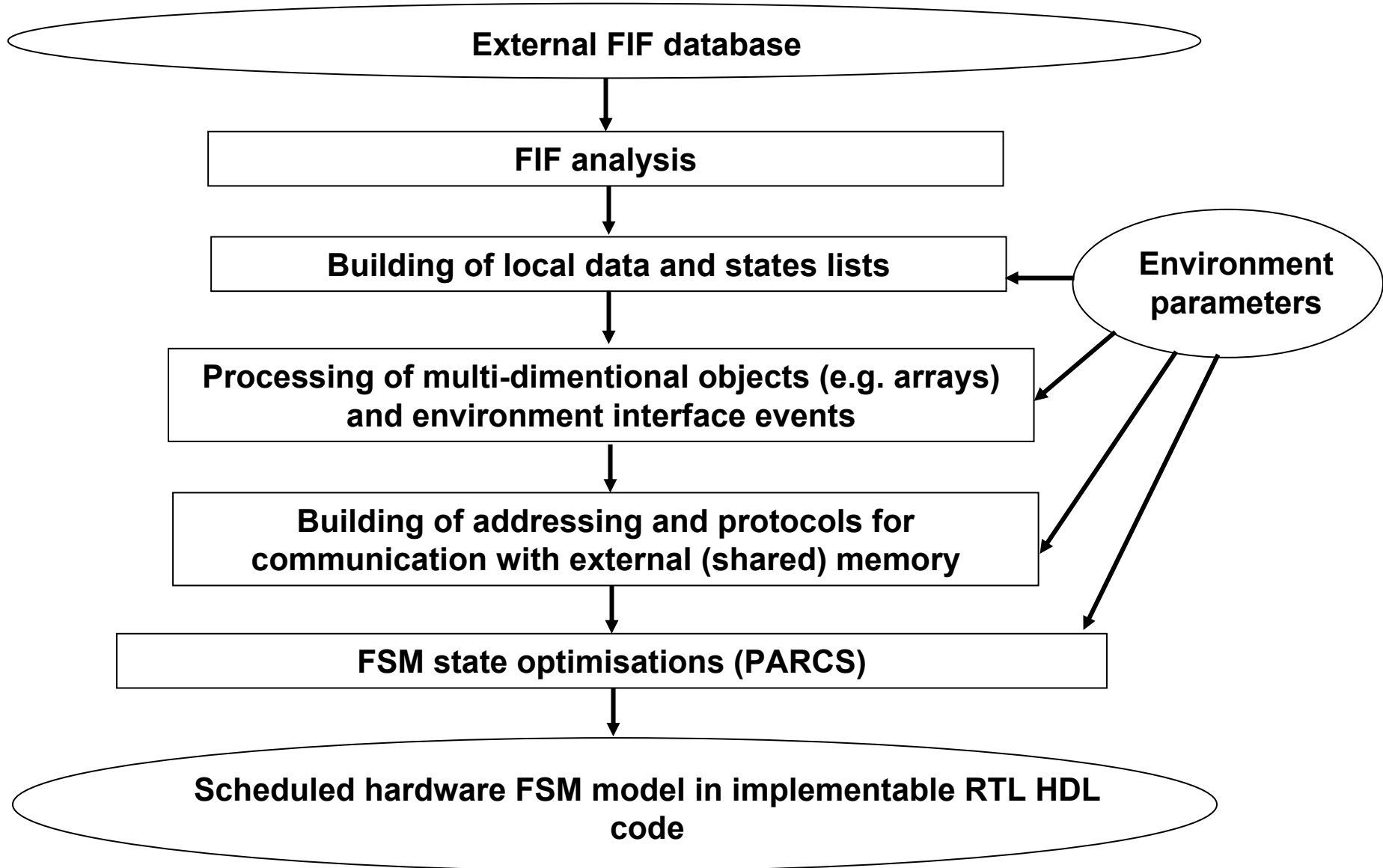
# C<sup>3</sup> hardware compiler



# Front-end compiler



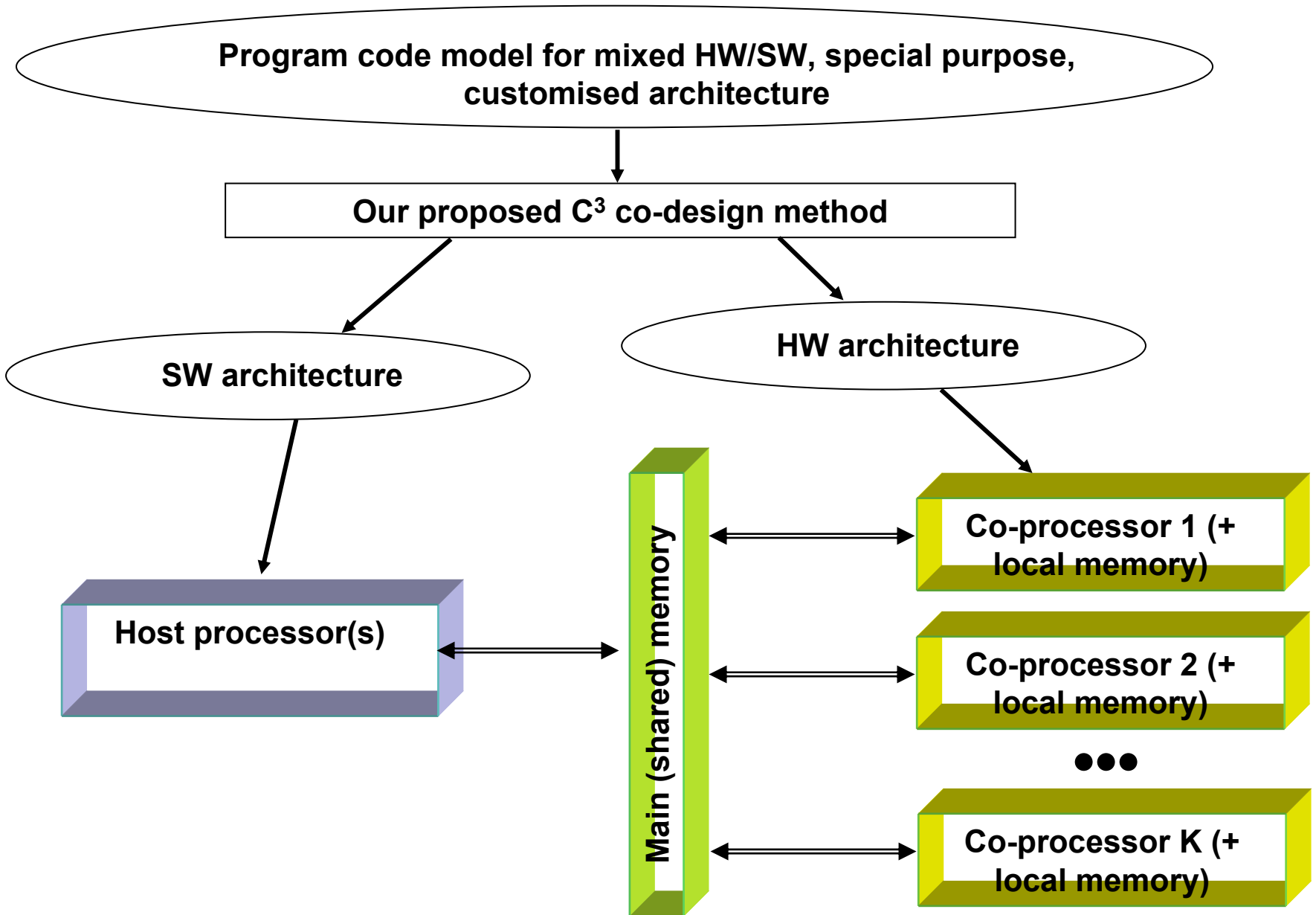
# Back-end compiler



# Scheduling optimisations

- **Optimisation of operations scheduling**
  - Parallel Abstract Resource – Constrained Scheduling (PARCS)
  - It is an aggressive scheduling algorithm which satisfies data dependencies
- **Processing of all program operations in a single step**
  - Including all source and special operations (e.g. communication and interface synchronization protocols)
- **Automatic generation of communication protocol and addressing**
- **Very fast compilation**
  - Most of tests were run in less than 5 seconds

# Co-execution of mixed HW/SW architecture



## Back-end compiler inference rules

- *Inference logic rules:*

$$A_0 \leftarrow A_1 \wedge A_2 \dots \wedge A_n \quad (n \geq 1)$$

- *FIF logic facts (Atomic formulas)  $A_n$  :*

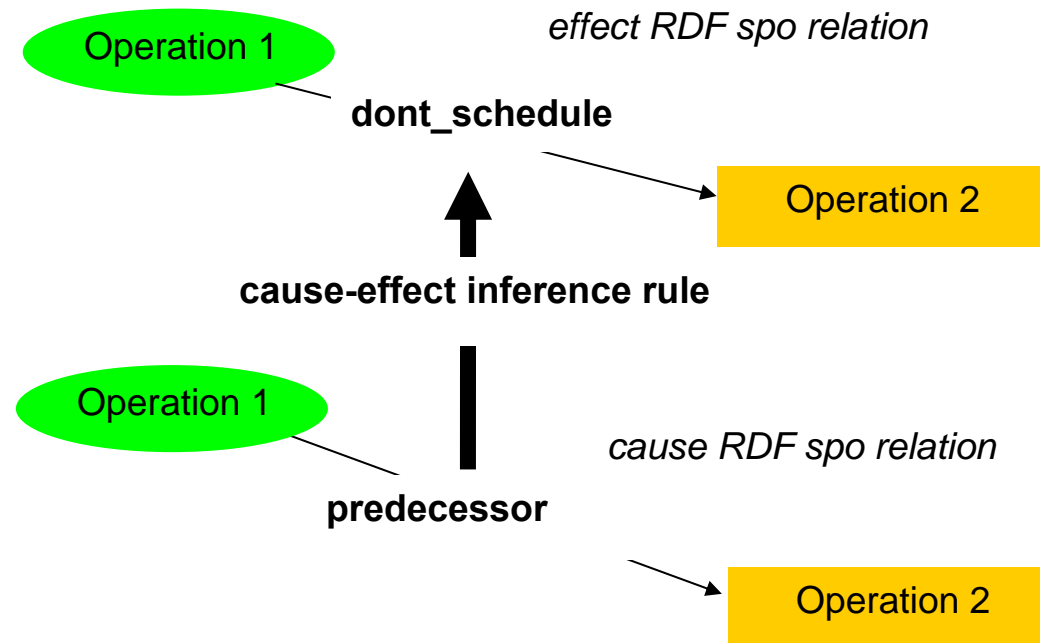
predicate\_symbol(Var\_1, Var\_2, ..., Var\_N)

- *Example inference rule*

dont\_schedule(Operation1, Operation2)  $\leftarrow$   
predecessor(Operation1, Operation2)

# RDF subject-predicate-object relations

- *Back-end compiler inference rule:*  
`dont_schedule(Operation1, Operation2) ← predecessor(Operation1, Operation2)`
- *Is a RDF rule relation:*



# The inference engine “concludes”

*RDF relations*

FIF  
database  
facts



*Inference  
rules with  
RDF  
relations*

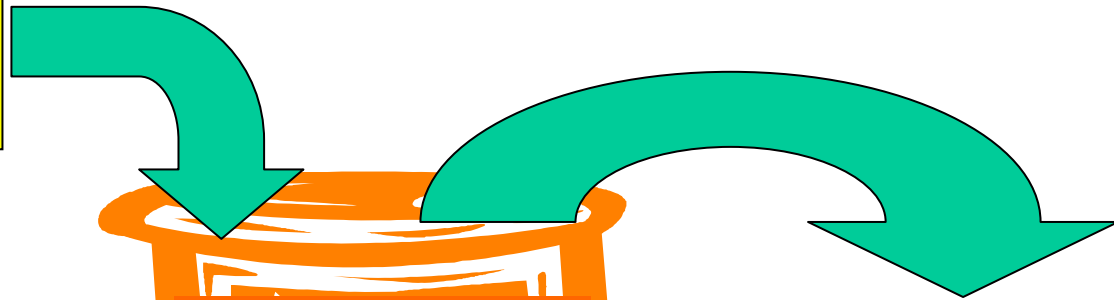
Back-end  
Inference  
engine



HW  
RTL



*Hardware model*



# The Web service client page:

The CCC project - Avant Browser

File Edit View Bookmarks AutoFill Tools Help

http://195.1 Go

Customize Links Free Hotmail My Yahoo! Windows Windows Marketplace Windows Media

The CCC project

Google Search Sign In

## Generating coprocessors on the web - The CCC project

### Frontend compiler

Choose an .ada file to upload

### Backend compiler

Choose a .cst file to upload (optional)

Choose a .mem file to upload (optional)

Command line argument (optional)

100% Internet

# BENCHMARKS and TESTS

- **Approximation algorithms**
  - Differential equation solving routine
- **Computer graphics**
  - Line drawing algorithm
- **Nested loops benchmarks**
  - Nested while and for loops within nested if then else constructs
- **Digital Signal Processing**
  - FIR DSP filters
- **Cryptography**
  - Public-key RSA applications
- **Flow Control design in network transmit-receive nodes**
  - Implementation of flow control and simulation of resulting hardware
- **Minsort algorithm**
- **Pseudo-random number generation algorithm**
- **Various benchmarks targeted to performance test of programming constructs**
  - Records, arrays, nested conditionals, etc.

## “Interesting” Links

- <http://www.verilog.com>
- <http://www.vhdl.org>
- <http://www.eda.org/>
- <http://www.eda.org/systemc/>
- <http://embedded.eecs.berkeley.edu/Respep/Research/hsc>
- <http://www.cs.washington.edu/research/projects/lis/www/chinook/>
- <http://mesl.ucsd.edu/spark/>