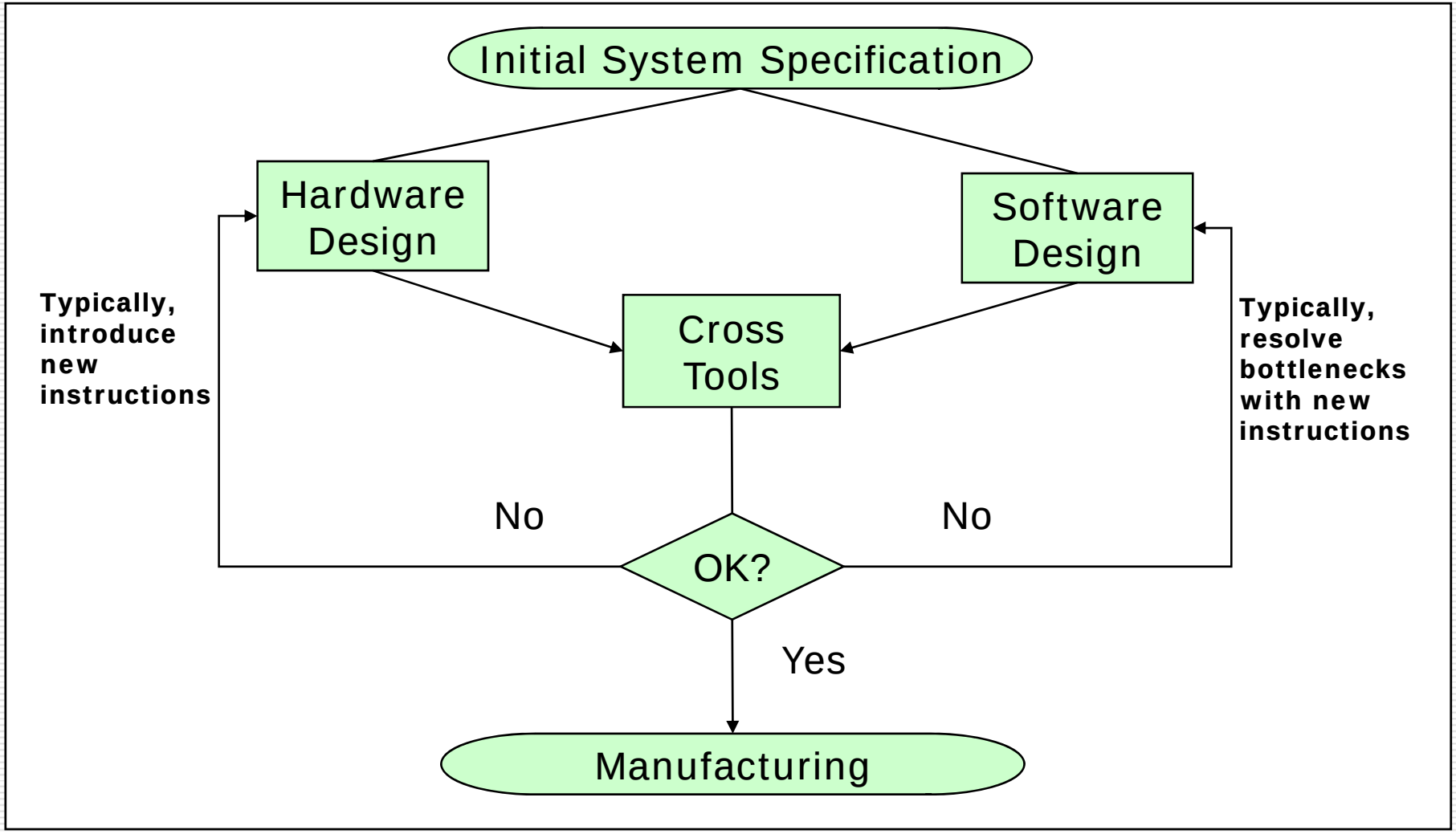
ISE Language: the ADL for Efficient Development of Cross Toolkits

Nikolay Pakulin **npak@ispras.ru**
Vladimir Rubanov vrub@ispras.ru
Institute for System Programming
Russian Academy of Sciences

Using Cross Tools in Embedded System Design



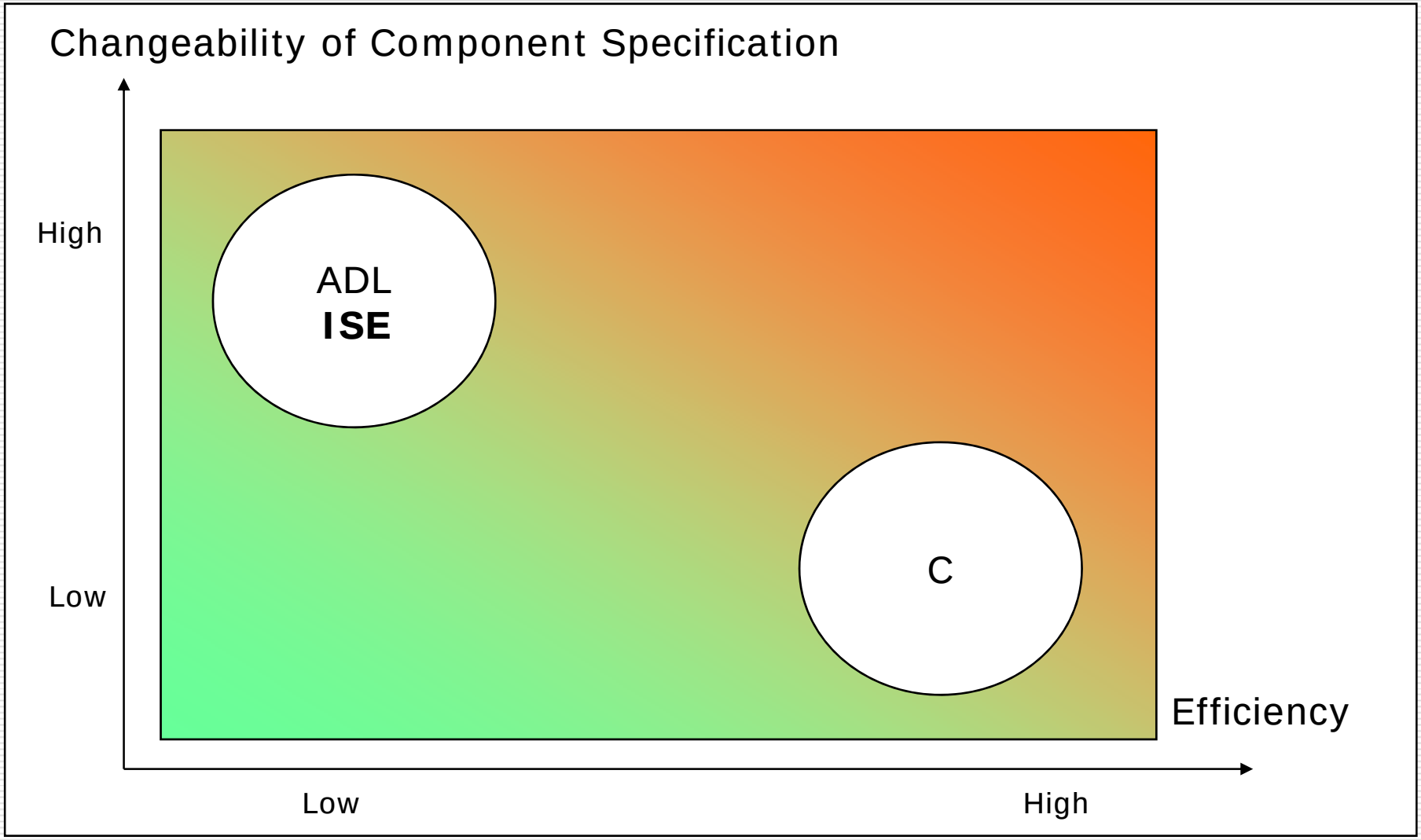
Requirements for Cross Tools Development

1. Complete cross toolkit:
   - Building tools: C compiler, assembler, disassembler, linker, firmware generator.
   - Execution tools: debugger, simulator, profiler.
2. Cycle-accurate simulation and profiling
3. High performance
   - e.g. 10 MCPS on 2 GHz PC
4. Availability at early development stages
5. Quick response to design changes
6. Method's concepts and notations should look familiar to industrial developers

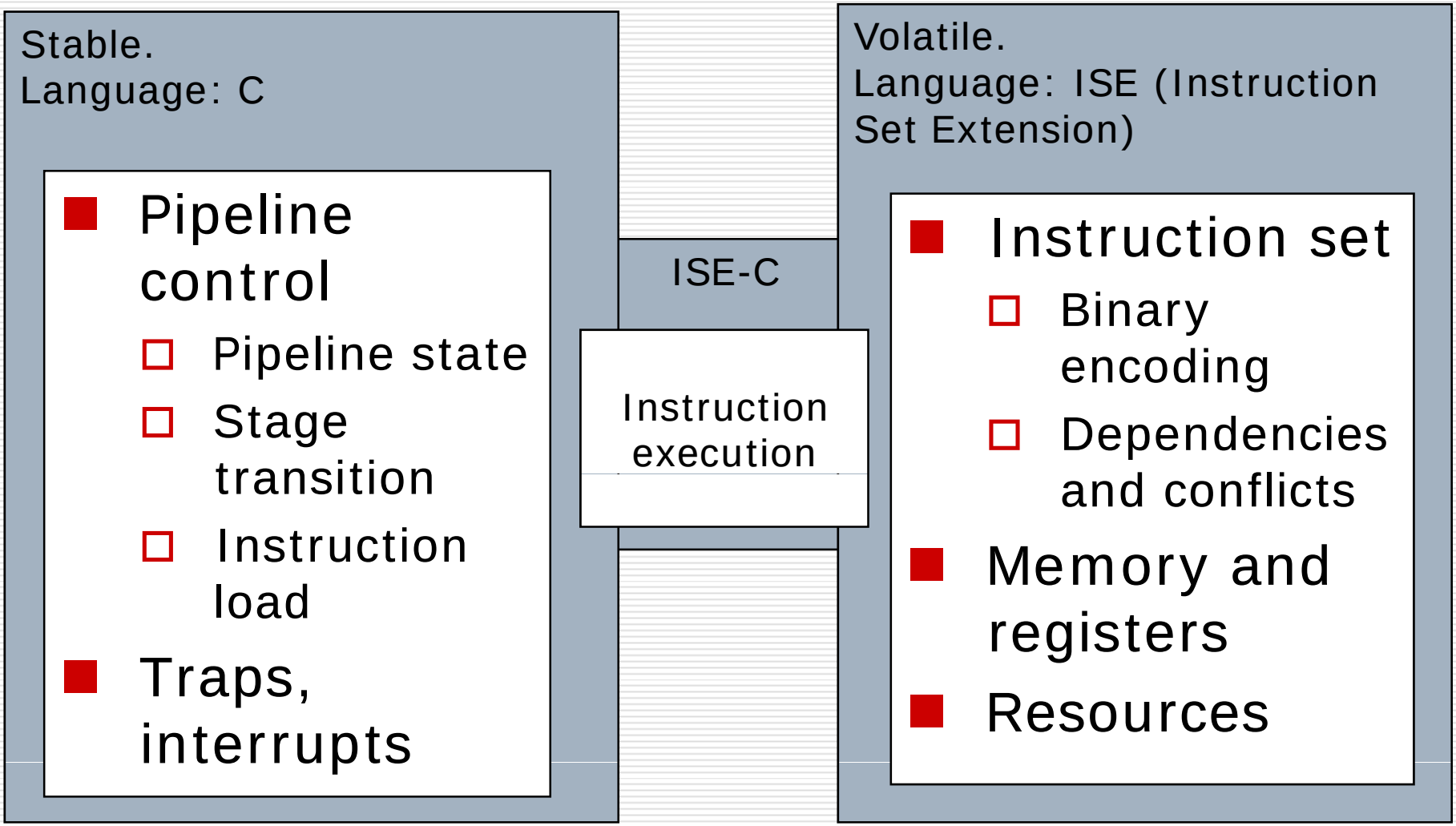
“Pure Specification” Approaches

- HDL (VHDL, SystemC)
  - Extremely low simulator performance
  - Late stage in the development process
  - No explicit instruction set specification – only simulator available
- ADL (nML, EXPRESSION, ISDL, ...)
  - No cycle-accurate simulation
  - Low simulator performance
  - Unfamiliar for engineers
- C/C++
  - Low responsiveness to instruction set changes

Hybrid Specification: Changeability vs. Efficiency



Primary Components



ISE Language Design Goals

- ISE specification structure should be aligned to “Instruction set reference” style
- Efficient support for irregular instruction encoding
  - Diverse encoding formats for different instructions
- C-like notation for functional specification

ISE Language Features

- Global architecture specification
  - Number of pipeline stages
  - CPU resources (buses, shift modules, accumulators, etc.)
- Memory and registers specification
  - Register pool
  - Memory banks, memory regions
- Operand definition
  - Register operand
  - Memory operand
  - Indirect addressing
- Instruction specification
  - Binary encoding
  - Cycle-precise; multi-stage operations
- Inter-instruction conflicts specification

Memory Specification

```
.storage
GR WORD16 [0..7] // 8 16-bit registers
AR WORD16 [0..3] // 4 address registers
ACR NBIT<36> [0..1]          // 2 36-bit accumulators
internal alu_temp WORD32     // 32-bit ALU accumulator
internal mul_res  WORD32     // 32-bit internal MAC
PREG PC unsigned int         // program counter
// program memory (256 Kwords)
PMEM PM WORD16 [0..2**18-1]
// 2 banks of data memory
DM {
   DM0 WORD16 [0..65535] // main memory(64 Kwords)
   TM0 WORD16 [USERDEF]  // DFFT coefficients
}
```

Operand Types

```
.otypes
// General purpose register
GRs         {GRn : SSS}
// Constant embedded in instruction code
C4          {const4 : CCCC}
// Memory bank identifier
DMx         {DMa : X}
// Indirect addressing
DM_GRs_C4{DM_GR_offset : M-SSS-CCCC}
```

Instruction Specification

```
.instructions
<MOV002> MOVE {DMa:M}({GRs}), {GRt}
         00M0-0000-1SSS-1TTT
         “MOVE DMa(GRs), GRt”
         properties [ rgrn:GRs, rgrn:GRt ]
         action {
                 DMa[GRs] = GRt;
         }
<MAC01>  MAC {ACRa}, {GRs}, {GRt}
         0111-00A0-1SSS-1TTT
         “MAC ACRa, GRs, GRt”
         properties [ racr:ACRa, wacr_2:ACRa,
                              rgrn:GRs, rgrn:GRt ]
         resources  [ EXE_MAC(mac) ]
         action {
                 mul_res = GRs * GRt;
         }
         action:EXE_MAC {
                 ACRa = ACRa + mul_res;
         }
```

Syntax Rule pattern

Encoding

Human readable mnemonics

Properties

Behavior

Conflicts specification

- Errors

  (I1:P:wacr_2 == I2:P:wacr)

  :: error “ACR conflict on write”

- Warnings

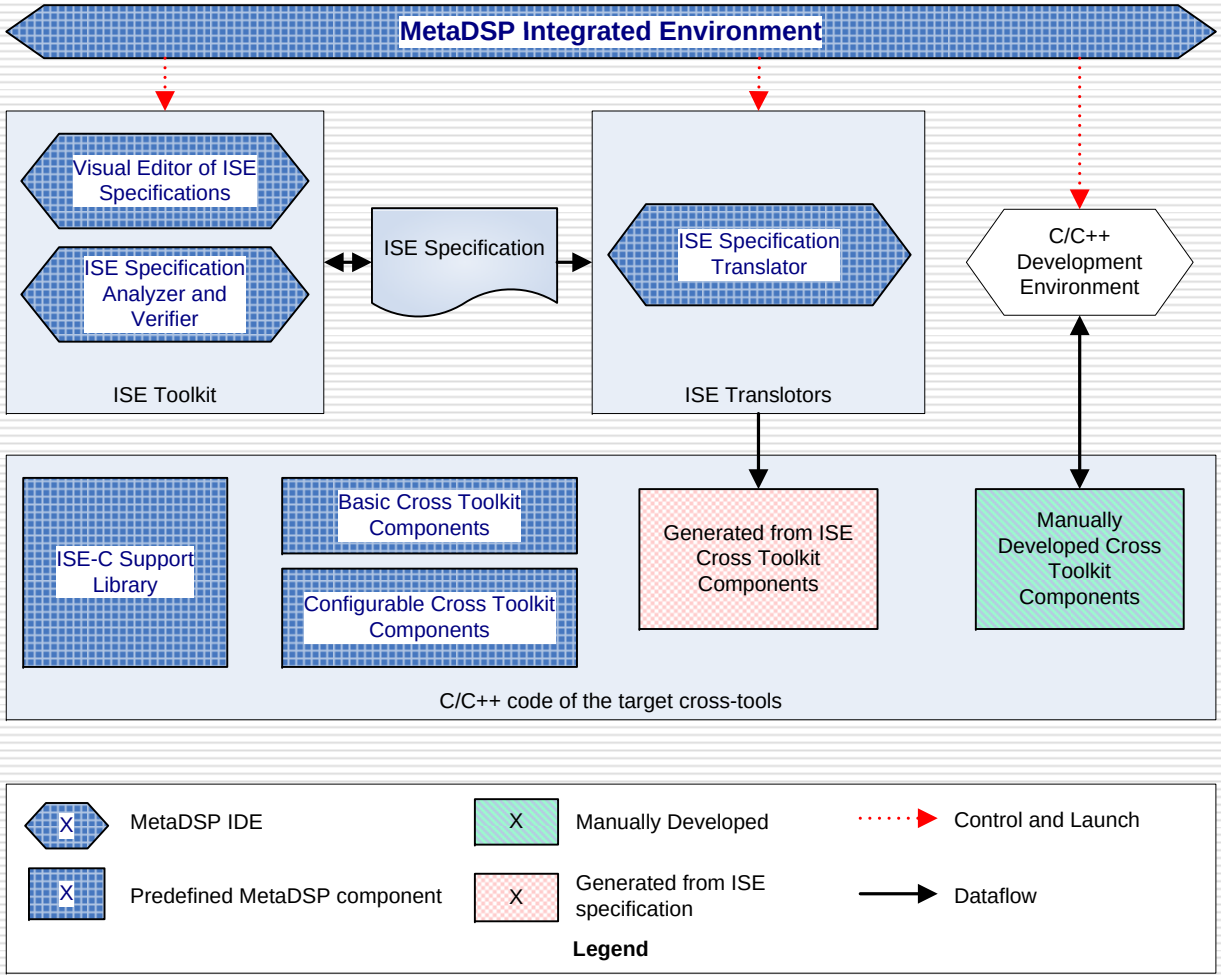
  (I1:R:mac && I2:P:clr)

  :: warning “Clear instruction after MAC instruction, result unspecified”

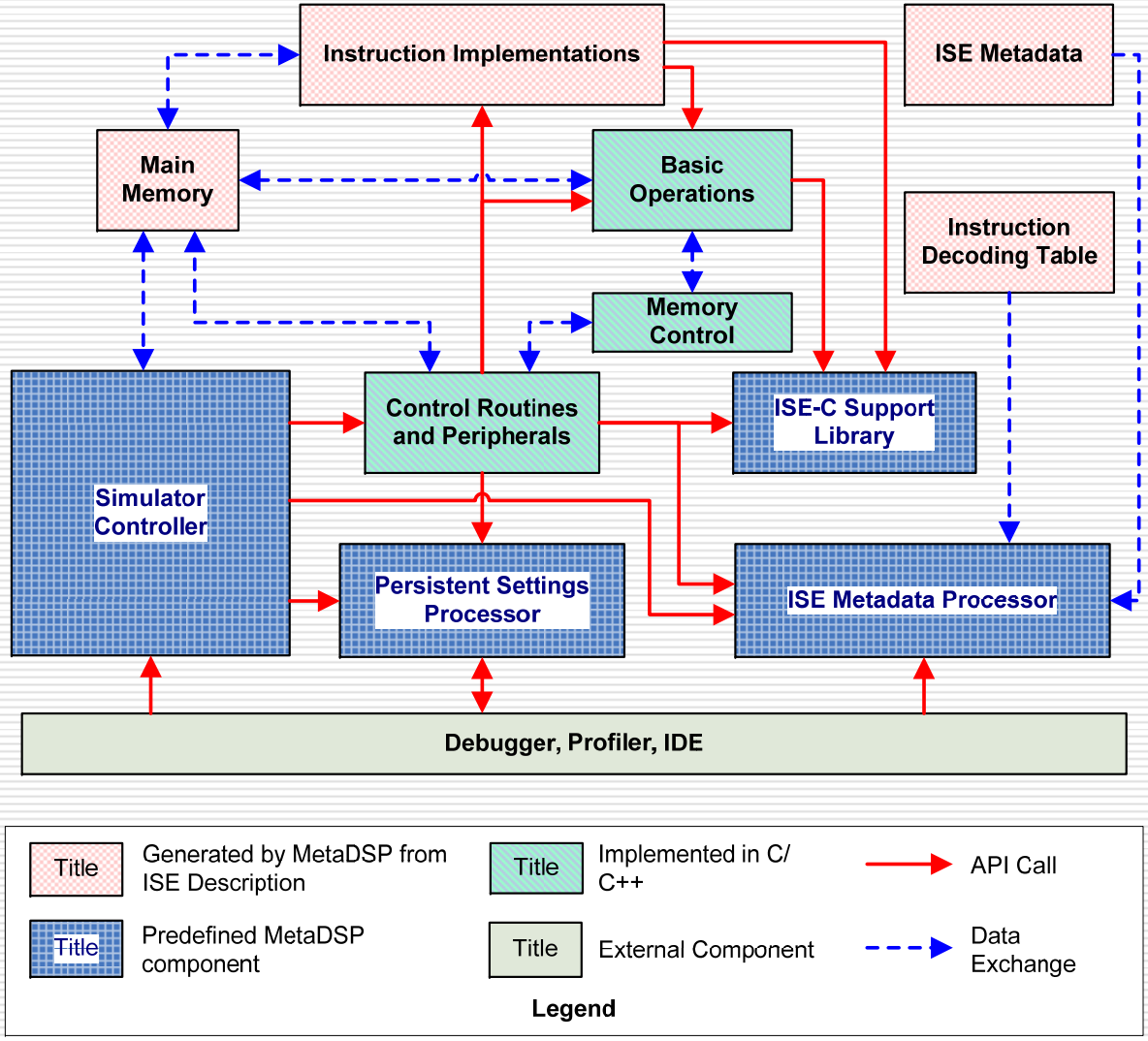
Toolkit Support

- MetaDSP Toolkit:
  - ISE specification analysis and verification tools
  - ISE translator
  - Reusable software components
    - Simulator core modules
    - Ready-to-use configurable modules for cross-tools construction
- OSCAR IDE:
  - Workbench for application developer
  - C/C++/Assembler app development
  - Debugging
  - Simulation / profiling

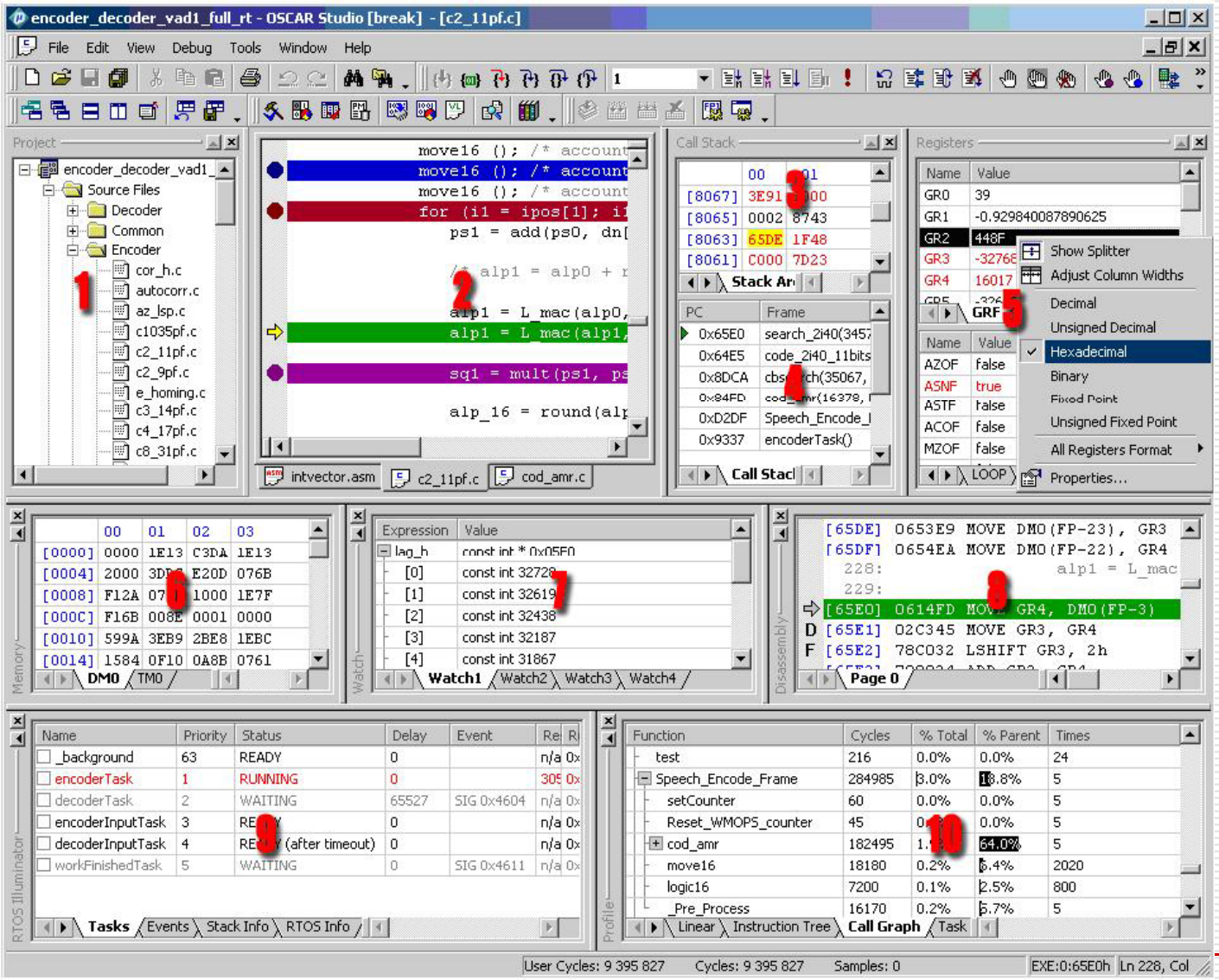
MetaDSP Framework



MetaDSP: Simulator Architecture



OSCAR IDE: Execution Environment



Industrial Applications

- **50** revisions of cross tools for Freehand Platf. 2; 2001, Freehand DSP AB (Sweden).
- **45** revisions of cross tools for MicroDSP 1.0; 2002-2003, VIA Technologies (Taiwan).
- **27** revisions of cross tools for MicroDSP 1.1 and extensions (DFFT, ALBI bus) ; 2003-2004, VIA Technologies (Taiwan).
- Cross tools for stable ZAC CPU; 2004, VIA Cores (USA).
- **33** revisions of cross tools for VIA DSP 2 variations and extensions; 2005, VIA Technologies (Taiwan).

ISE-Driven Cross Tools Development

- Application Domain:

  YES: modern 16/32 bit DSP, 32-bit ARM

  NO: contemporary general-purpose CPU

- Efficiency compared to manual C/C++ Cross Tools Development

Specification size	Reduced by a factor of **12**
Development Efforts	Reduced by **60%**
Responsiveness to Instruction Set changes	Improved by a factor of **10**