

ISE Language: the ADL for Efficient Development of Cross Toolkits

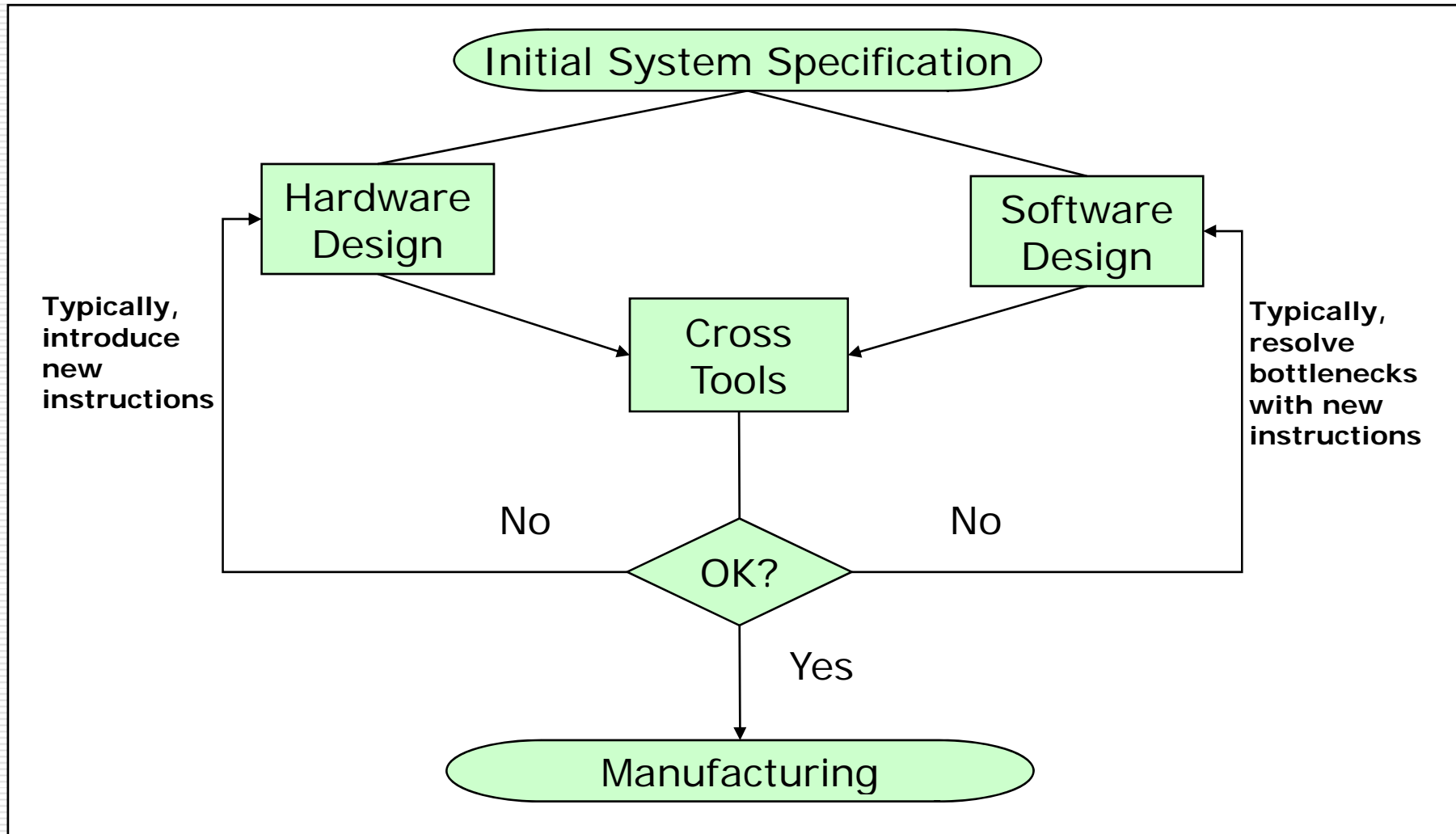
Nikolay Pakulin npak@ispras.ru

Vladimir Rubanov vrub@ispras.ru

Institute for System Programming

Russian Academy of Sciences

Using Cross Tools in Embedded System Design



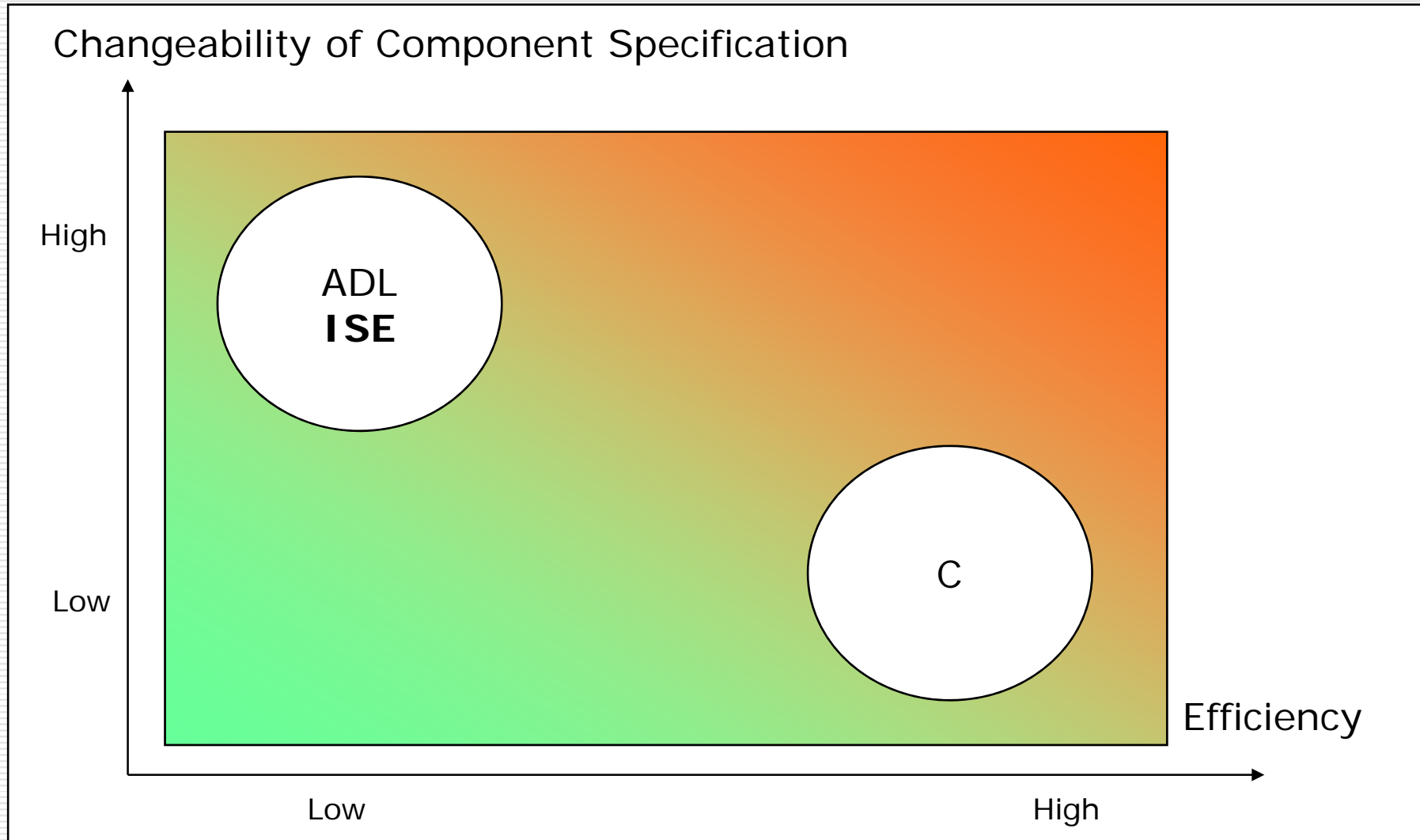
Requirements for Cross Tools Development

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

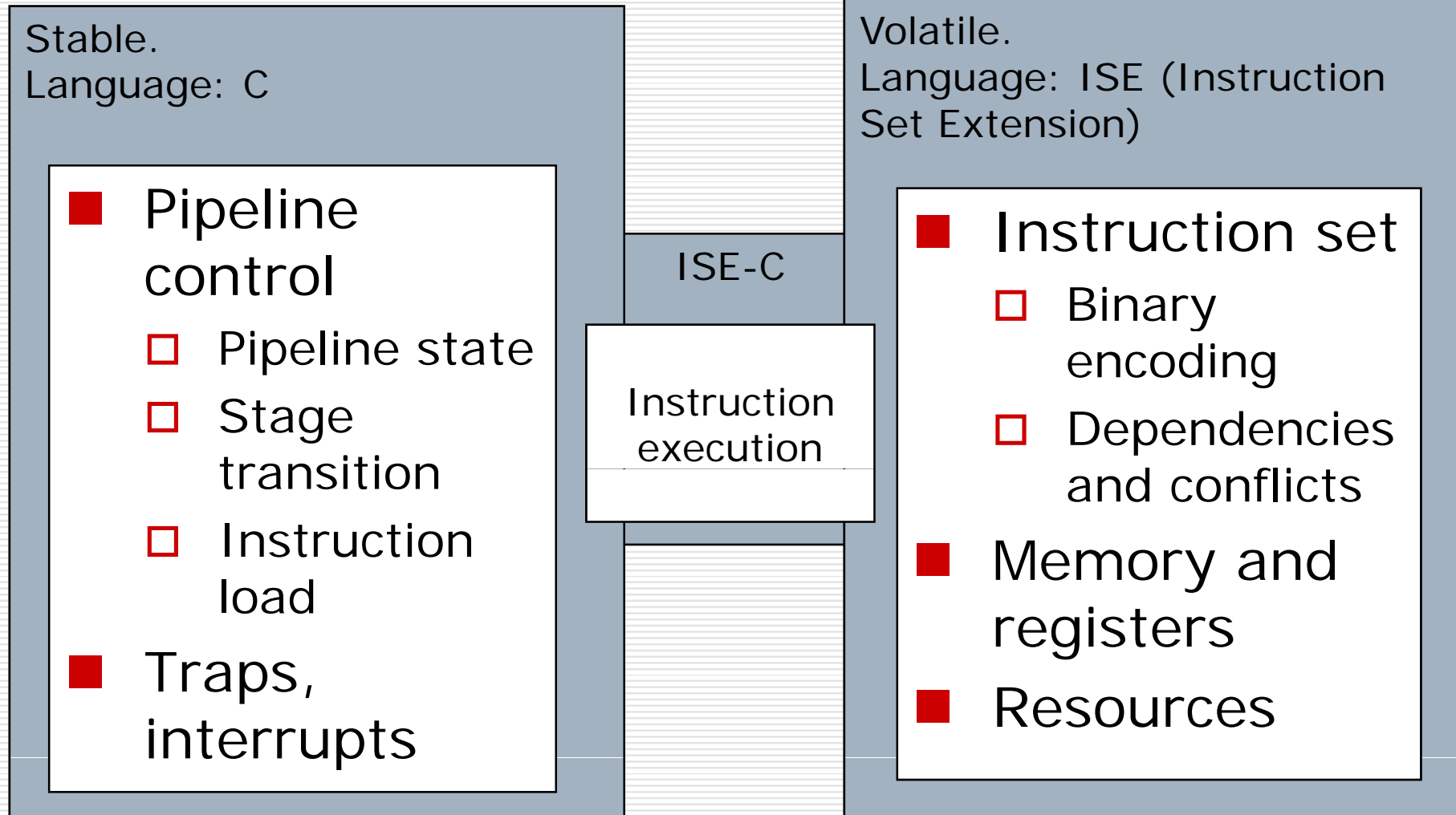
“Pure Specification” Approaches

- 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 {
```

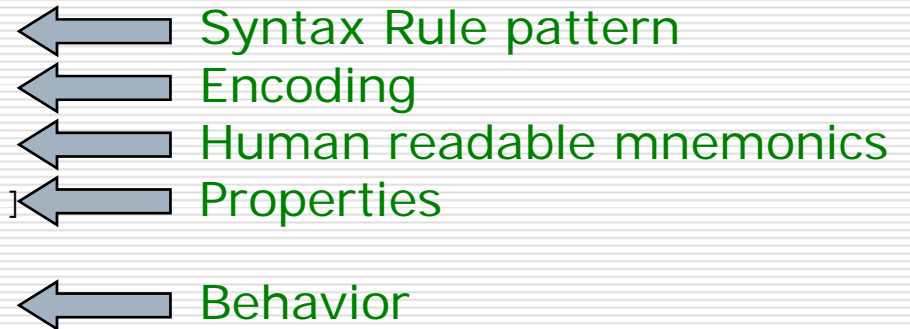
```
    mul_res = GRs * GRt;
```

```
}
```

```
action:EXE_MAC {
```

```
    ACRa = ACRa + mul_res;
```

```
}
```



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

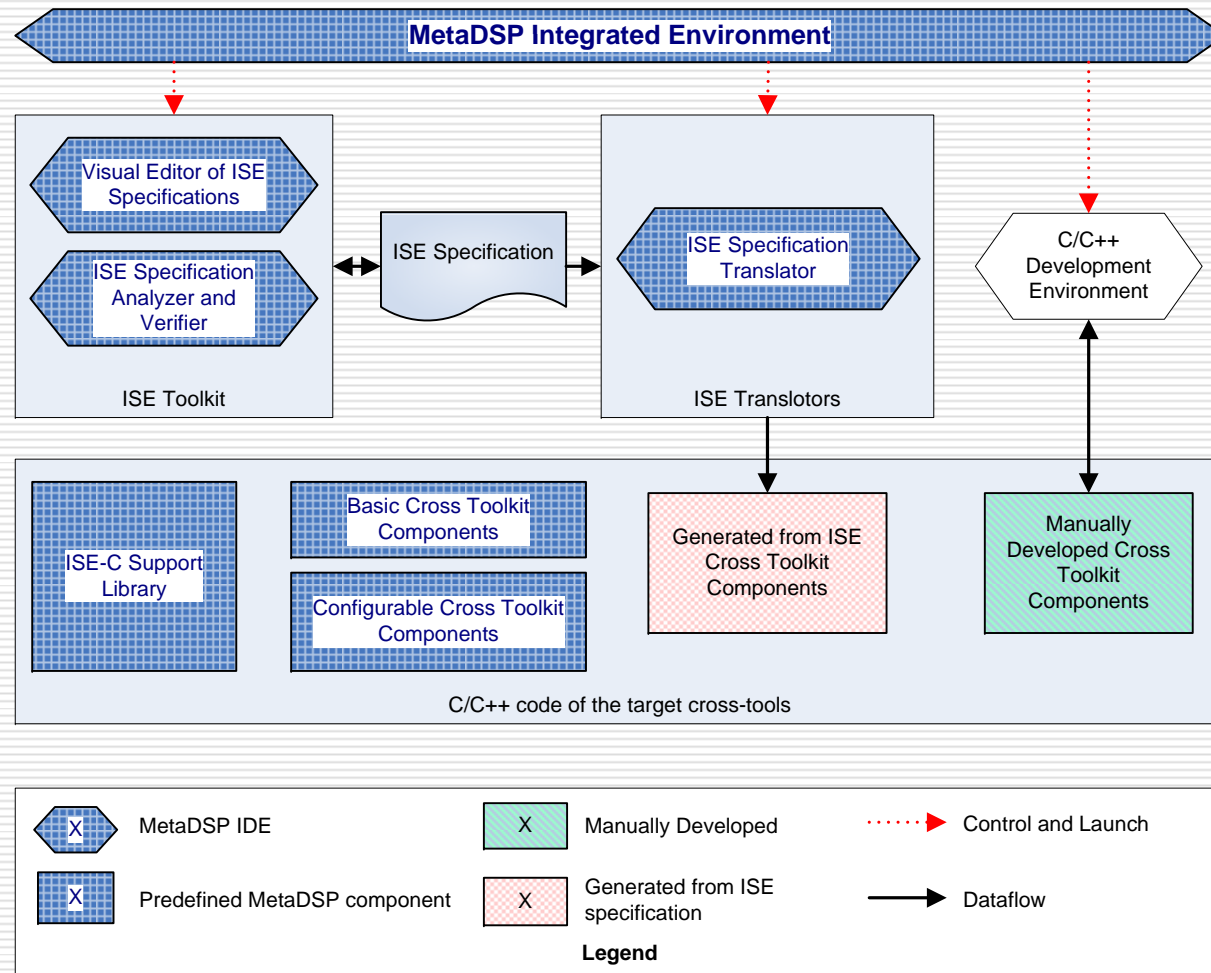
■ MetaDSP Toolkit:

- ISE specification analysis and verification tools
- ISE translator
- Reusable software components
 - Simulator core modules
 - Ready-to-use configurable modules for cross-tools construction

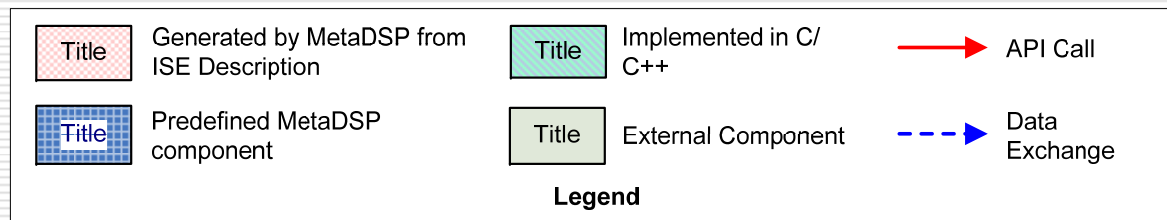
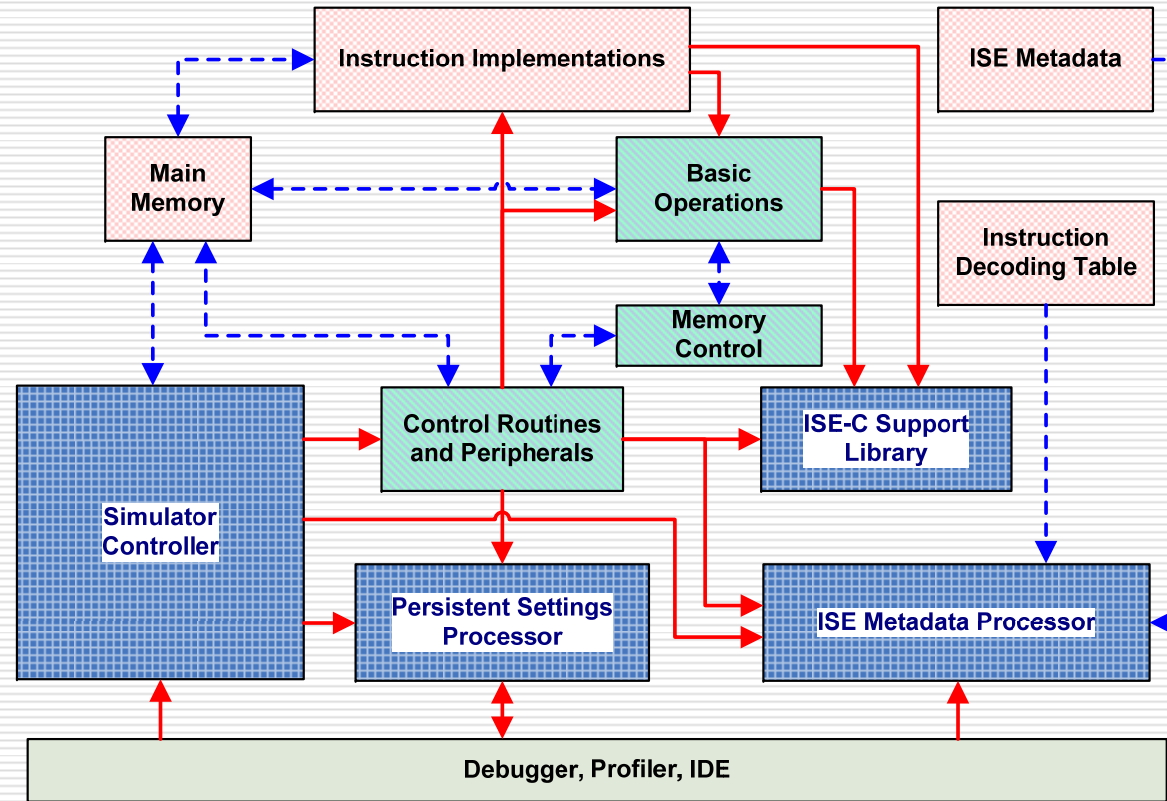
■ OSCAR IDE:

- Workbench for application developer
- C/C++/Assembler app development
- Debugging
- Simulation / profiling

MetaDSP Framework



MetaDSP: Simulator Architecture



OSCAR IDE: Execution Environment

The screenshot displays the OSCAR IDE interface during a debugging session. The main window shows the source code for `move16` and `for` loops. The `Registers` window shows the state of registers, with `GR2` highlighted. The `Memory` window shows the current memory location `DM0`. The `Disassembly` window shows the assembly instructions being executed. The `Tasks` window shows the current task `encoderTask` in a `RUNNING` state. The `Profile` window shows the execution time for various functions, with `cod_amr` being the most significant.

Registers:

Name	Value
GR0	39
GR1	-0.929840087890625
GR2	448F
GR3	-32768
GR4	16017
GR5	0
GR6	0
GR7	0
GR8	0
GR9	0
GR10	0
GR11	0
GR12	0
GR13	0
GR14	0
GR15	0

Memory:

Address	00	01	02	03
[0000]	0000	1E13	C3DA	1E13
[0004]	2000	3D1E	E20D	076B
[0008]	F12A	0710	1000	1E7F
[000C]	F16B	008E	0001	0000
[0010]	599A	3EB9	2BE8	1EBC
[0014]	1584	0F10	0A8B	0761

Disassembly:

Address	Instruction
[65DE]	0653E9 MOVE DMO (FP-23), GR3
[65DF]	0654EA MOVE DMO (FP-22), GR4
228:	alpl = L_mac
229:	
[65E0]	0614FD MOV GR4, DMO (FP-3)
D [65E1]	02C345 MOVE GR3, GR4
F [65E2]	78C032 LSHIFT GR3, 2h

Tasks:

Name	Priority	Status	Delay	Event	Re: R
_background	63	READY	0		n/a 0x
encoderTask	1	RUNNING	0		305 0x
decoderTask	2	WAITING	65527	SIG 0x4604	n/a 0x
encoderInputTask	3	READY	0		n/a 0x
decoderInputTask	4	READY (after timeout)	0		n/a 0x
workFinishedTask	5	WAITING	0	SIG 0x4611	n/a 0x

Profile:

Function	Cycles	% Total	% Parent	Times
test	216	0.0%	0.0%	24
Speech_Encode_Frame	284985	3.0%	10.8%	5
setCounter	60	0.0%	0.0%	5
Reset_WMOPS_counter	45	0.0%	0.0%	5
cod_amr	182495	1.9%	64.0%	5
move16	18180	0.2%	6.4%	2020
logic16	7200	0.1%	2.5%	800
_Pre_Process	16170	0.2%	5.7%	5

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