Design and Architectures for Embedded Systems

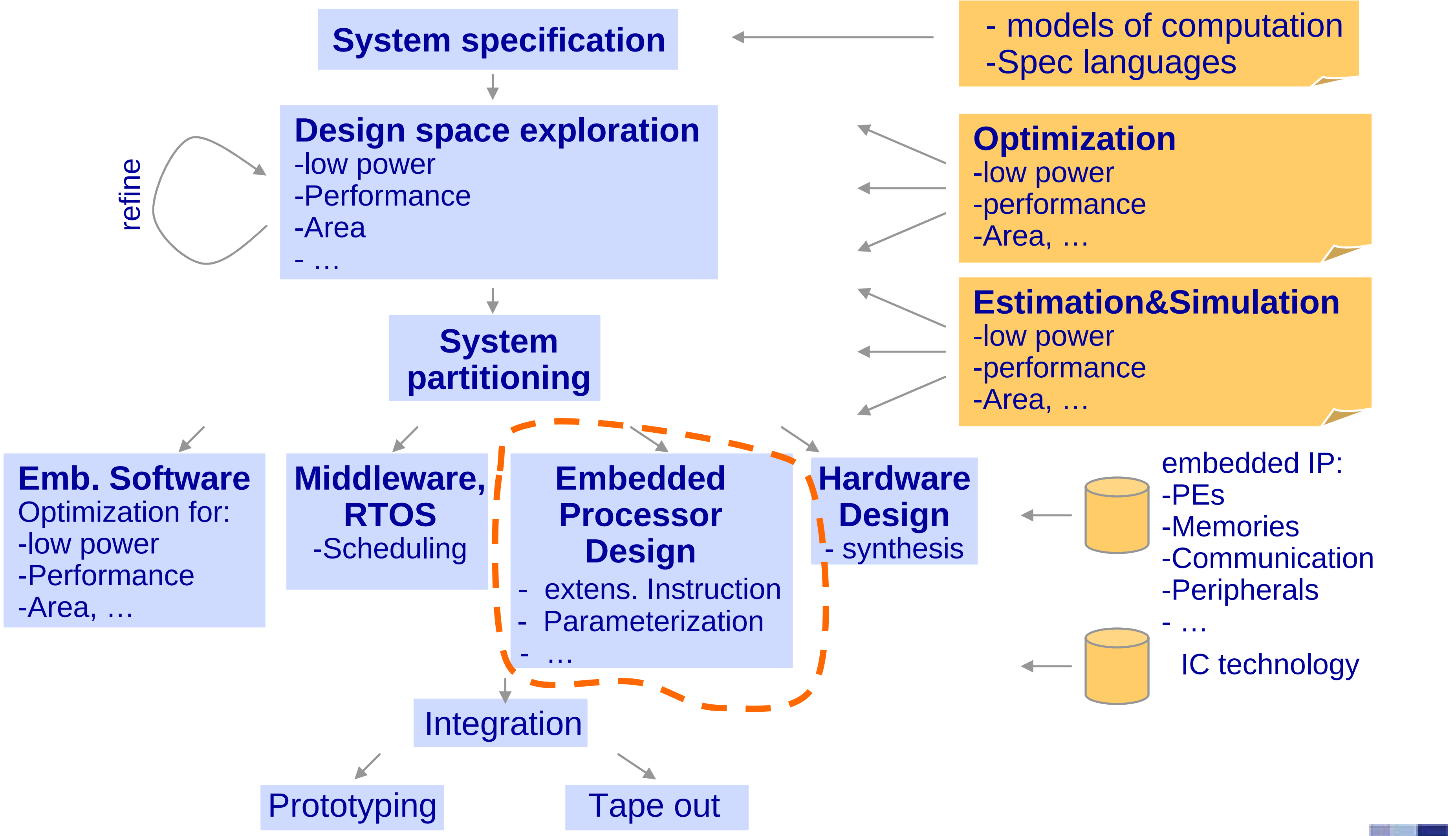
Lars Bauer (Lehrstuhl Prof. Dr. J. Henkel)

CES - Chair for Embedded Systems

University of Karlsruhe, Germany

Today: An exemplary Design-Environment
for extensible ASIPs

Where are we ?



System specification

- models of computation
-Spec languages

Design space exploration
-low power
-Performance
-Area
- …

refine

Optimization
-low power
-performance
-Area, …

System partitioning

Estimation&Simulation
-low power
-performance
-Area, …

Emb. Software
Optimization for:
-low power
-Performance
-Area, …

Middleware, RTOS
-Scheduling

Embedded Processor Design
- extens. Instruction
- Parameterization
- …

Hardware Design
- synthesis

embedded IP:
-PEs
-Memories
-Communication
-Peripherals
- …

IC technology

Integration

Prototyping

Tape out

Outline

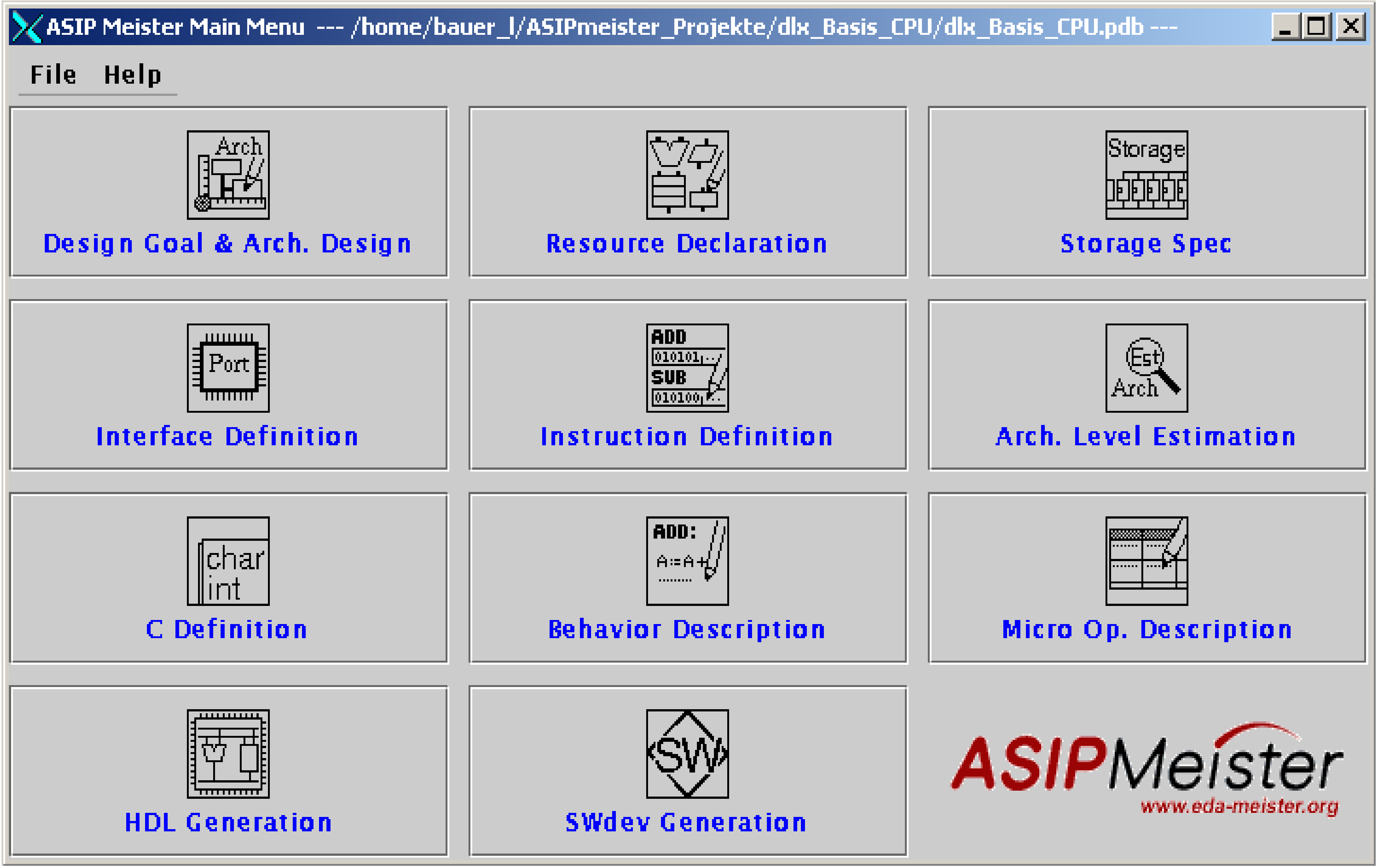
- ASIP Meister
  - A development environment for ASIPs
- CoSy Compiler
  - A retargetable compiler development system
- Platinum Edition
  - A state-of-the-art prototyping hardware

ASIP Meister

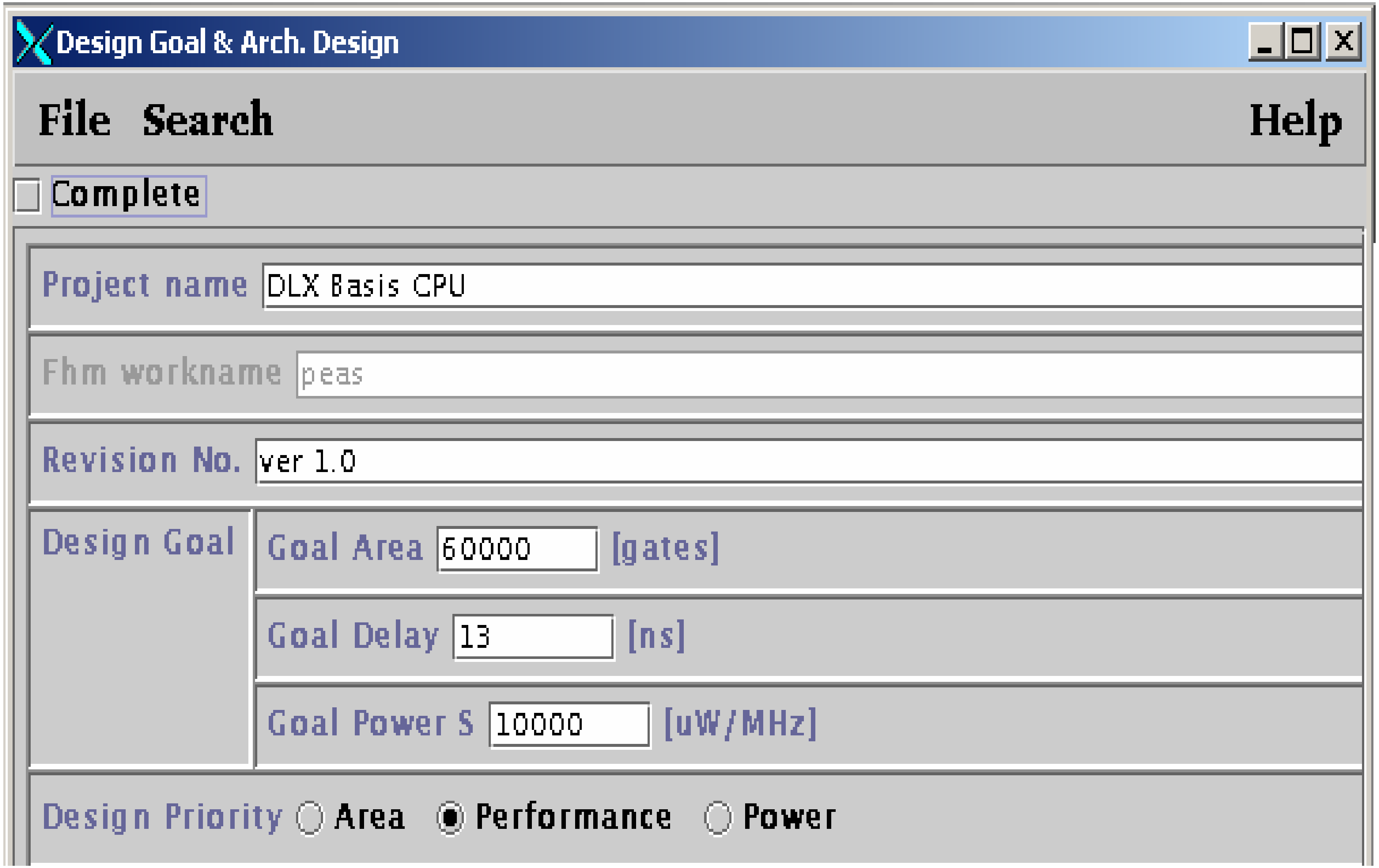
- GUI-based Design Environment for extensible ASIPs (Application Specific Instruction set Processors)

- Creates:
  - HDL-Description for Simulation
  - HDL-Description for Synthesis
  - Architecture-Description for Software-Tools (C-Compiler, Assembler)

ASIP Meister – Main Menu



ASIP Meister – Design Goal



□ Complete

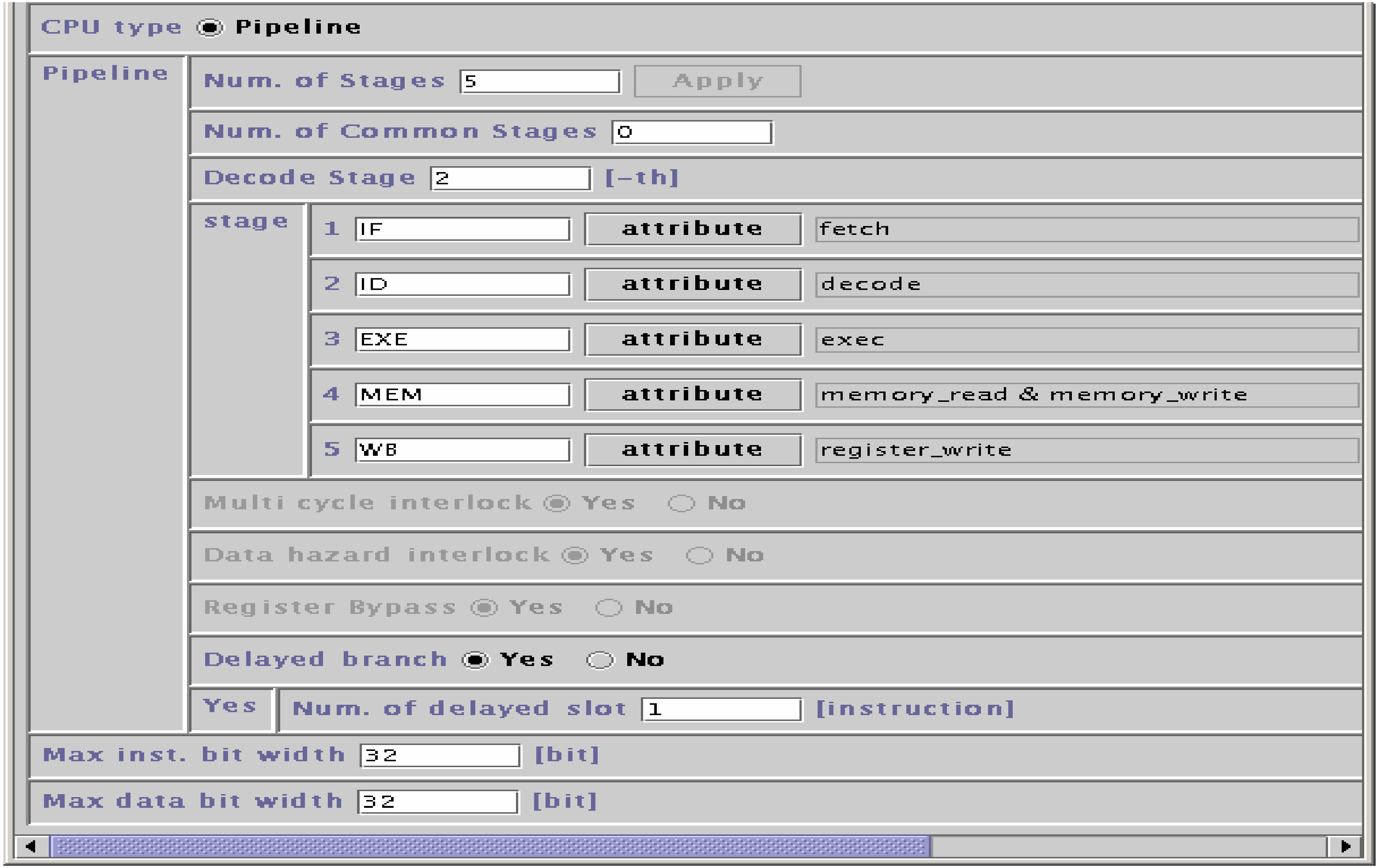
Project name	DLX Basis CPU
Fhm workname	peas
Revision No.	ver 1.0

Design Goal

Goal Area	60000	[gates]
Goal Delay	13	[ns]
Goal Power S	10000	[uW/MHz]

Design Priority ○ Area ● Performance ○ Power

ASIP Meister – Architecture Design



CPU type ● Pipeline

Pipeline

Num. of Stages 5 Apply

Num. of Common Stages 0

Decode Stage 2 [-th]

stage

1	IF	attribute	fetch
2	ID	attribute	decode
3	EXE	attribute	exec
4	MEM	attribute	memory_read & memory_write
5	WB	attribute	register_write

Multi cycle interlock ● Yes ○ No

Data hazard interlock ● Yes ○ No

Register Bypass ● Yes ○ No

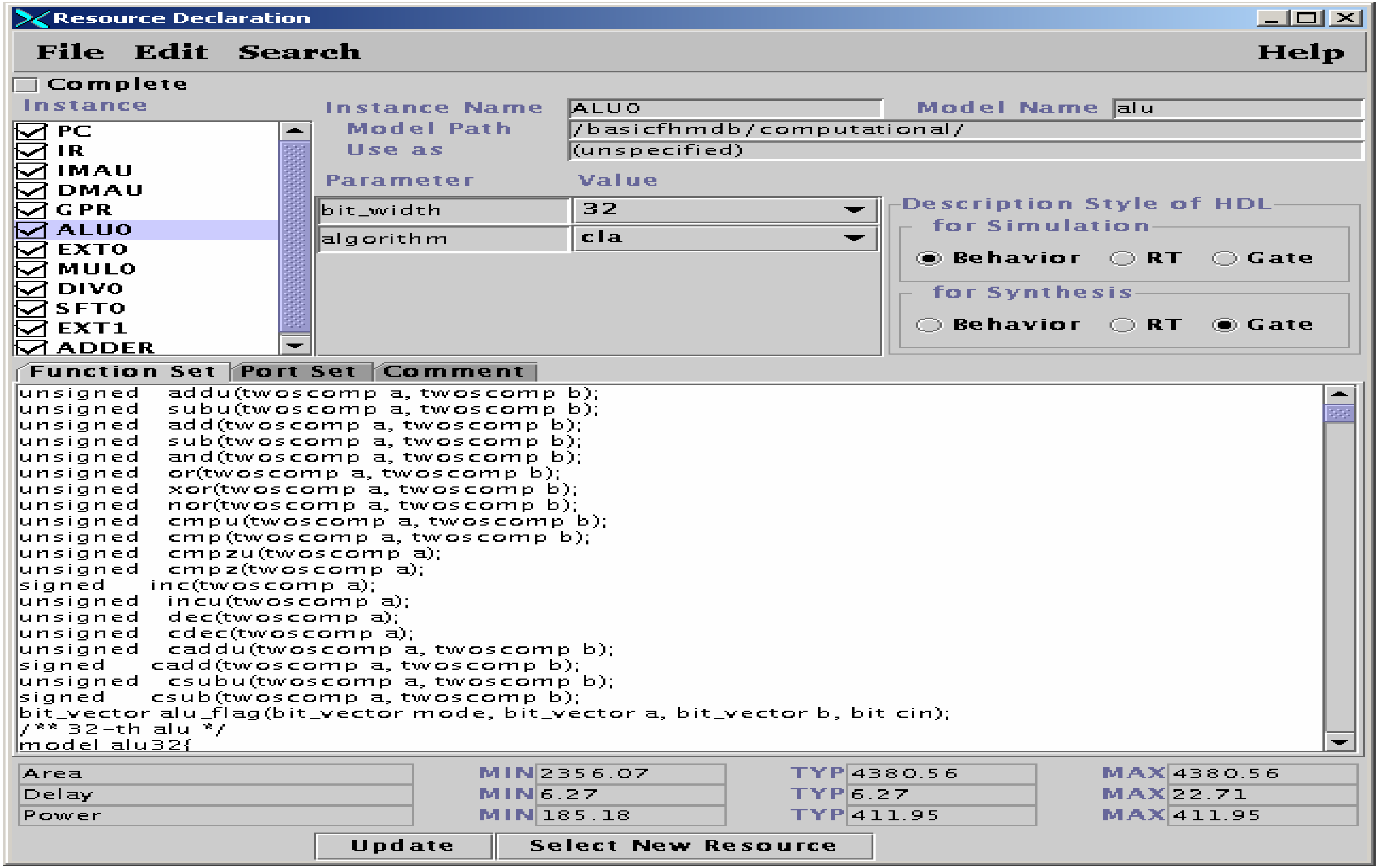
Delayed branch ● Yes ○ No

Yes Num. of delayed slot 1 [instruction]

Max inst. bit width 32 [bit]

Max data bit width 32 [bit]

ASIP Meister – Ressource Decl. ALU



□ Complete

Instance

☑ PC
☑ IR
☑ IMAU
☑ DMAU
☑ GPR
☑ ALU0
☑ EXT0
☑ MUL0
☑ DIV0
☑ SFT0
☑ EXT1
☑ ADDER

Instance Name: ALU0
Model Name: alu
Model Path: /basicfhmdb/computational/
Use as: (unspecified)

Parameter	Value
bit_width	32
algorithm	cla

Description Style of HDL

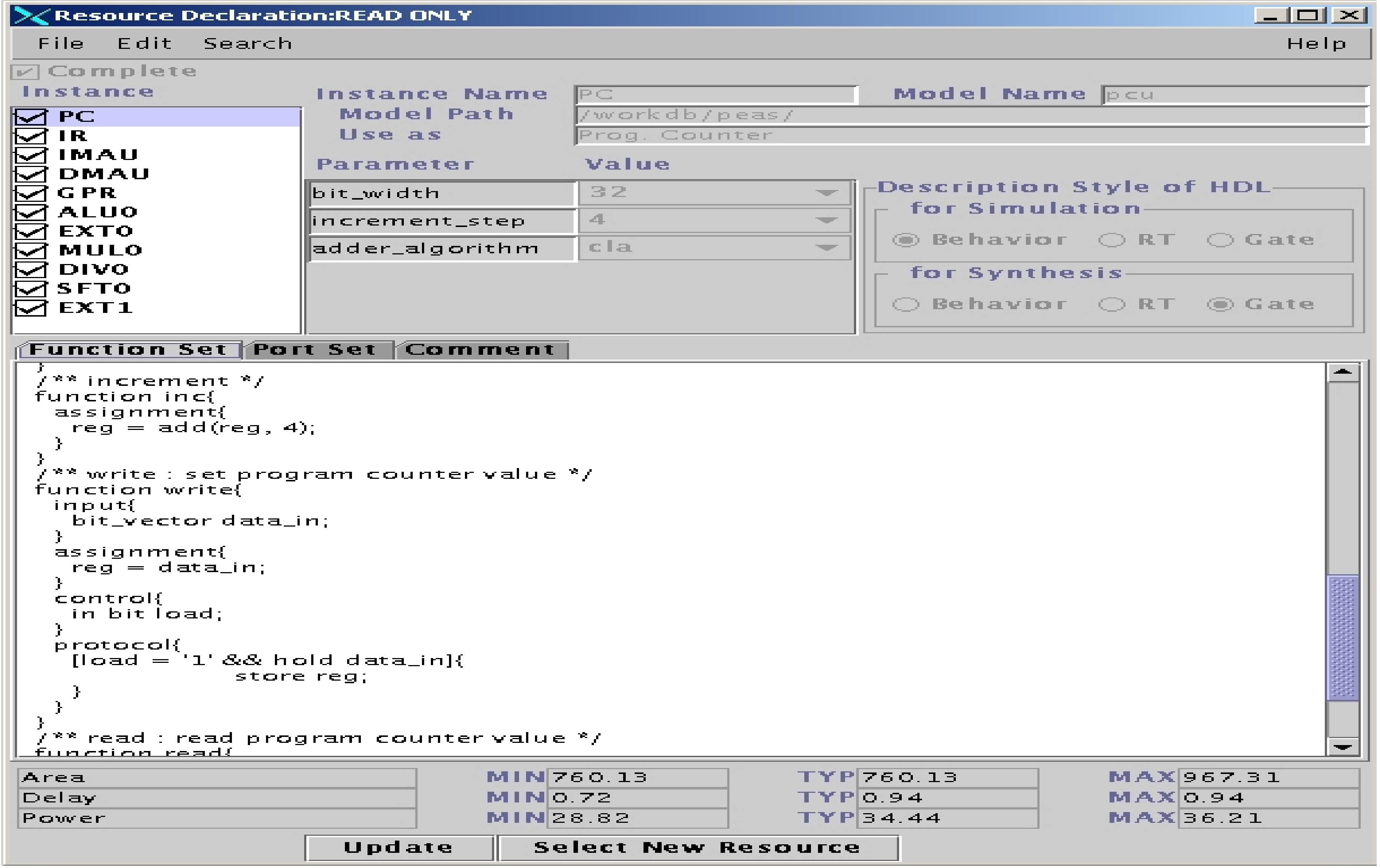
for Simulation: ● Behavior ○ RT ○ Gate

for Synthesis: ○ Behavior ○ RT ● Gate

```
unsigned   addu(twoscomp a, twoscomp b);
unsigned   subu(twoscomp a, twoscomp b);
unsigned   add(twoscomp a, twoscomp b);
unsigned   sub(twoscomp a, twoscomp b);
unsigned   and(twoscomp a, twoscomp b);
unsigned   or(twoscomp a, twoscomp b);
unsigned   xor(twoscomp a, twoscomp b);
unsigned   nor(twoscomp a, twoscomp b);
unsigned   cmpu(twoscomp a, twoscomp b);
unsigned   cmp(twoscomp a, twoscomp b);
unsigned   cmpzu(twoscomp a);
unsigned   cmpz(twoscomp a);
signed    inc(twoscomp a);
unsigned   incu(twoscomp a);
unsigned   dec(twoscomp a);
unsigned   cdec(twoscomp a);
unsigned   caddu(twoscomp a, twoscomp b);
signed    cadd(twoscomp a, twoscomp b);
unsigned   csubu(twoscomp a, twoscomp b);
signed    csub(twoscomp a, twoscomp b);
bit_vector alu_flag(bit_vector mode, bit_vector a, bit_vector b, bit cin);
/** 32-th alu */
model alu32{
```

Area	MIN	2356.07	TYP	4380.56	MAX	4380.56
Delay	MIN	6.27	TYP	6.27	MAX	22.71
Power	MIN	185.18	TYP	411.95	MAX	411.95

ASIP Meister – Ressource Decl. PC



☑ Complete

Instance

- ☑ PC
- ☑ IR
- ☑ IMAU
- ☑ DMAU
- ☑ GPR
- ☑ ALU0
- ☑ EXT0
- ☑ MUL0
- ☑ DIV0
- ☑ SFT0
- ☑ EXT1

Instance Name: PC

Model Name: pcu

Model Path: /workdb/peas/

Use as: Prog. Counter

Parameter	Value
bit_width	32
increment_step	4
adder_algorithm	cla

Description Style of HDL

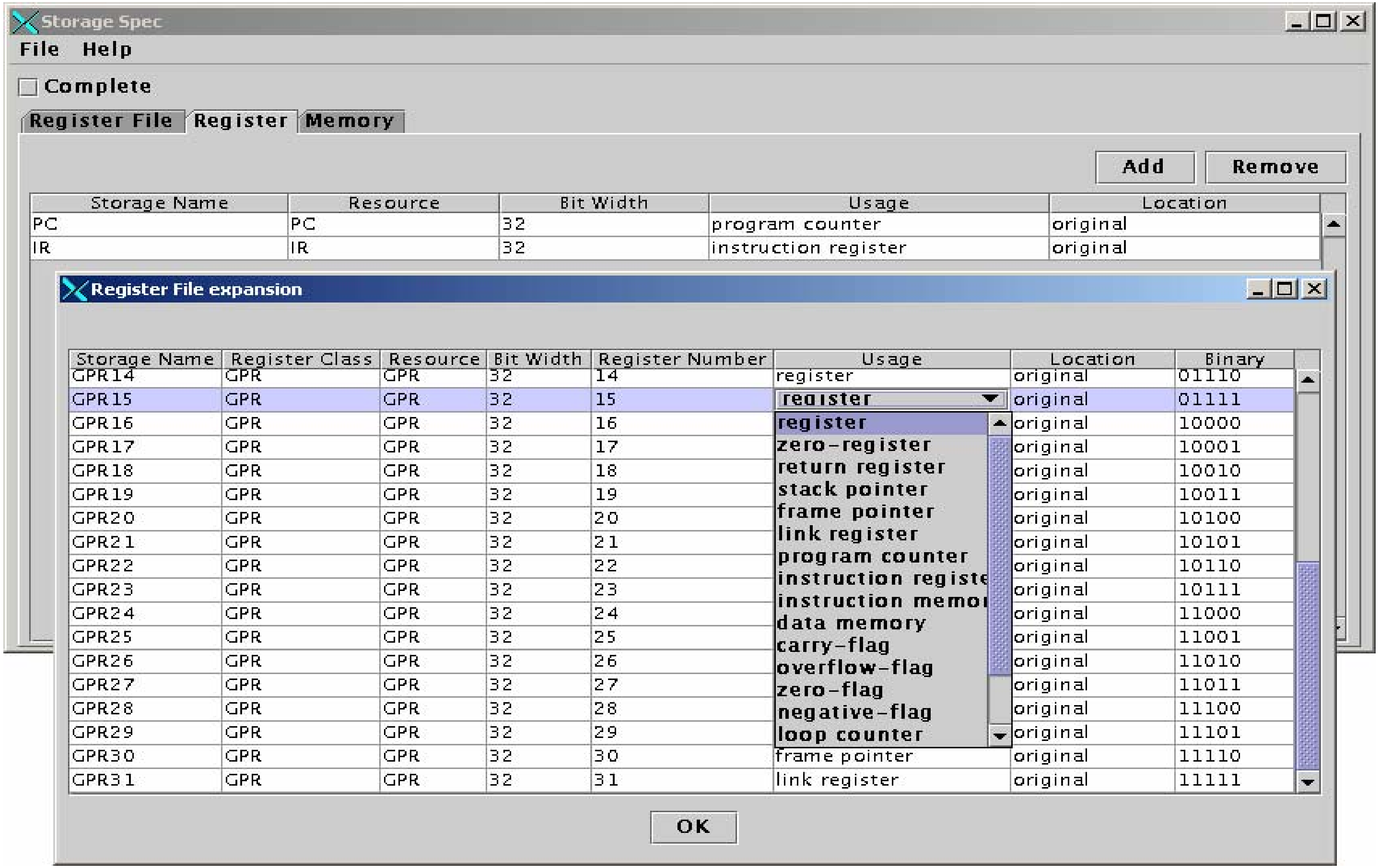
for Simulation: ● Behavior ○ RT ○ Gate

for Synthesis: ○ Behavior ○ RT ● Gate

```
}
/** increment */
function inc{
  assignment{
    reg = add(reg, 4);
  }
}
/** write : set program counter value */
function write{
  input{
    bit_vector data_in;
  }
  assignment{
    reg = data_in;
  }
  control{
    in bit load;
  }
  protocol{
    [load == '1' && hold data_in]{
                    store reg;
    }
  }
}
/** read : read program counter value */
function read{
```

Area	MIN	760.13	TYP	760.13	MAX	967.31
Delay	MIN	0.72	TYP	0.94	MAX	0.94
Power	MIN	28.82	TYP	34.44	MAX	36.21

ASIP Meister – Storage Spec

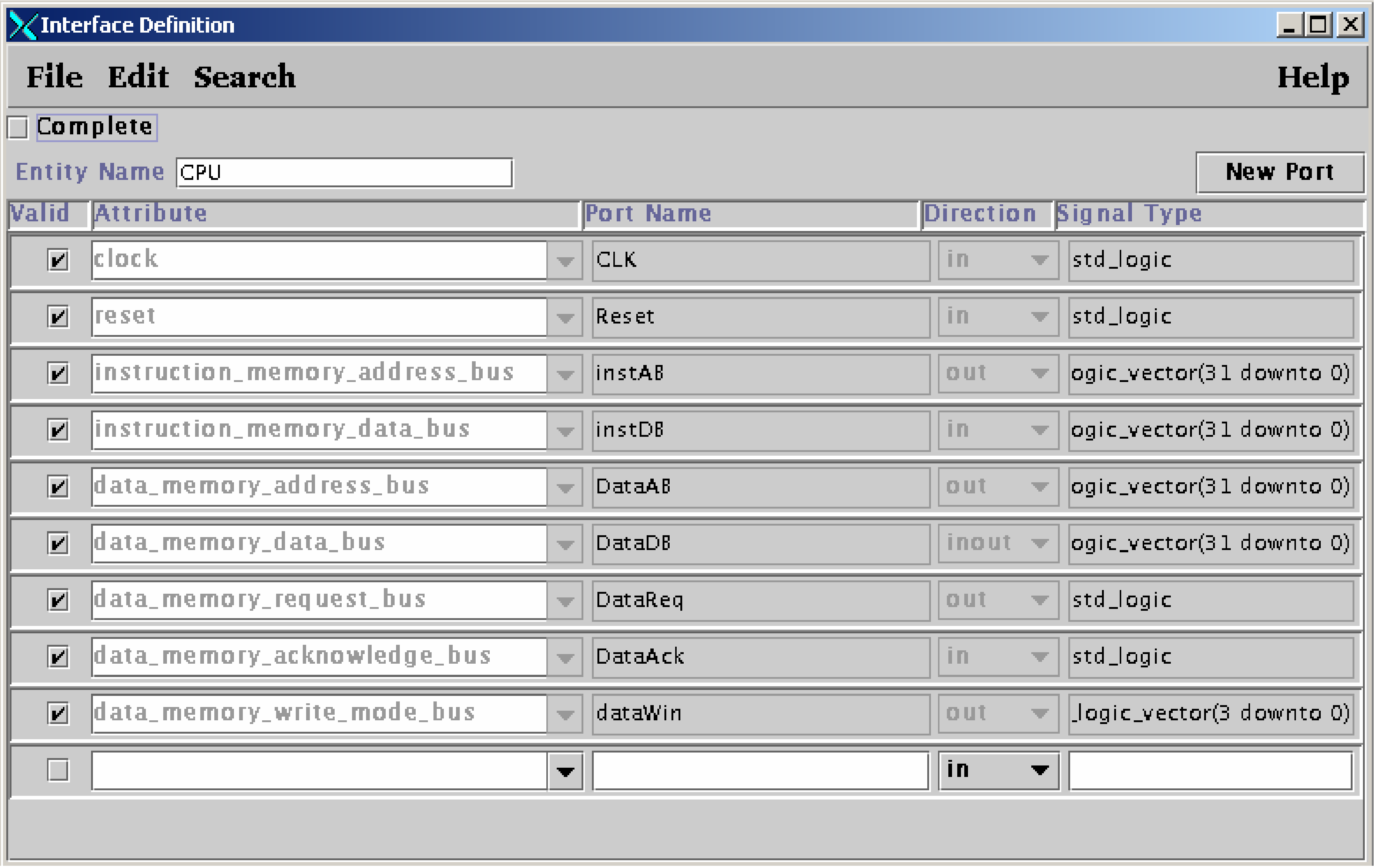


☐ Complete

Storage Name	Resource	Bit Width	Usage	Location
PC	PC	32	program counter	original
IR	IR	32	instruction register	original

Storage Name	Register Class	Resource	Bit Width	Register Number	Usage	Location	Binary
GPR14	GPR	GPR	32	14	register	original	01110
GPR15	GPR	GPR	32	15	register	original	01111
GPR16	GPR	GPR	32	16		original	10000
GPR17	GPR	GPR	32	17		original	10001
GPR18	GPR	GPR	32	18		original	10010
GPR19	GPR	GPR	32	19		original	10011
GPR20	GPR	GPR	32	20		original	10100
GPR21	GPR	GPR	32	21		original	10101
GPR22	GPR	GPR	32	22		original	10110
GPR23	GPR	GPR	32	23		original	10111
GPR24	GPR	GPR	32	24		original	11000
GPR25	GPR	GPR	32	25		original	11001
GPR26	GPR	GPR	32	26		original	11010
GPR27	GPR	GPR	32	27		original	11011
GPR28	GPR	GPR	32	28		original	11100
GPR29	GPR	GPR	32	29		original	11101
GPR30	GPR	GPR	32	30	frame pointer	original	11110
GPR31	GPR	GPR	32	31	link register	original	11111

ASIP Meister – Interface Definition

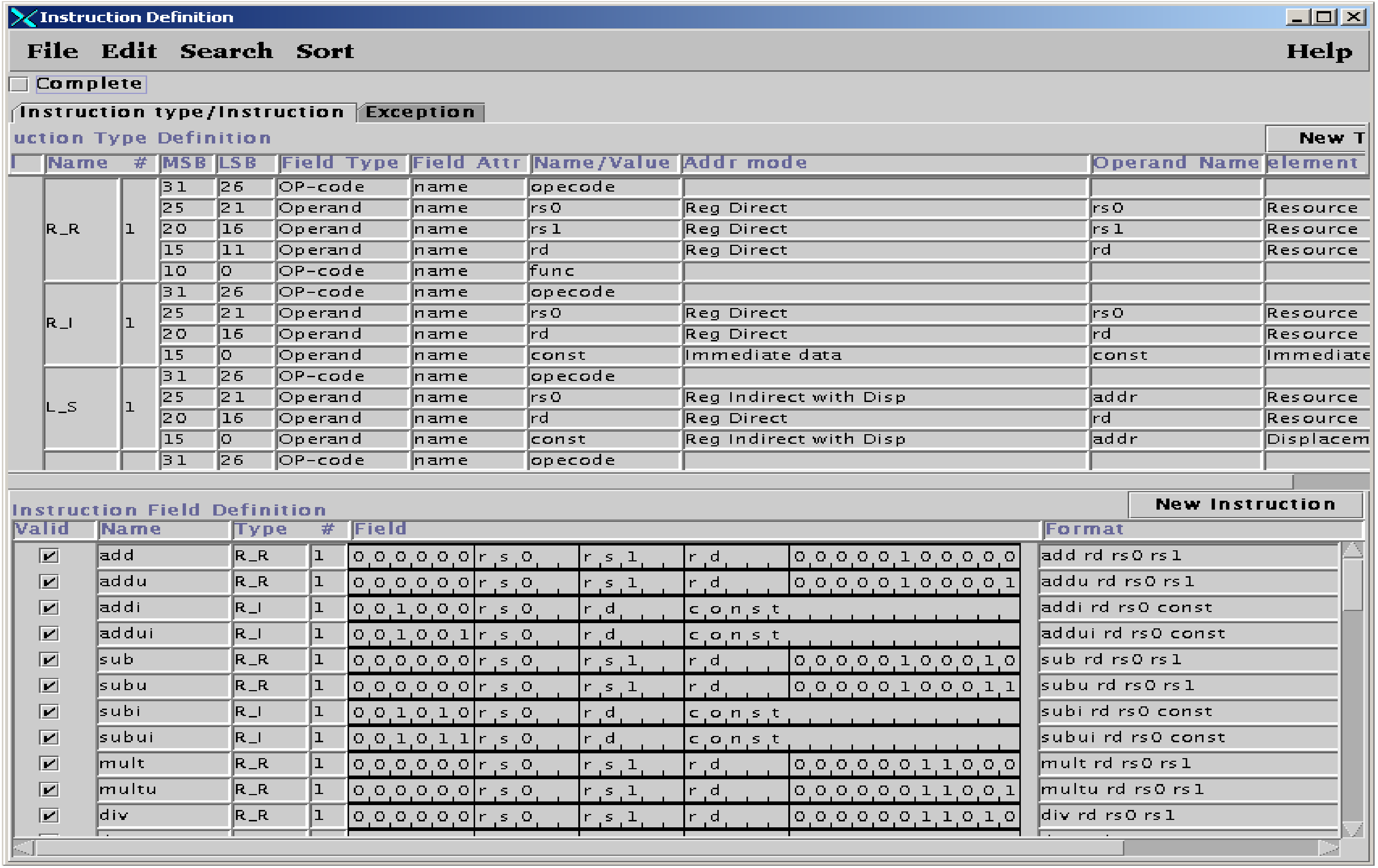


□ Complete

Entity Name: CPU

Valid	Attribute	Port Name	Direction	Signal Type
☑	clock	CLK	in	std_logic
☑	reset	Reset	in	std_logic
☑	instruction_memory_address_bus	instAB	out	ogic_vector(31 downto 0)
☑	instruction_memory_data_bus	instDB	in	ogic_vector(31 downto 0)
☑	data_memory_address_bus	DataAB	out	ogic_vector(31 downto 0)
☑	data_memory_data_bus	DataDB	inout	ogic_vector(31 downto 0)
☑	data_memory_request_bus	DataReq	out	std_logic
☑	data_memory_acknowledge_bus	DataAck	in	std_logic
☑	data_memory_write_mode_bus	dataWin	out	_logic_vector(3 downto 0)
□			in	

ASIP Meister – Instruction Definition



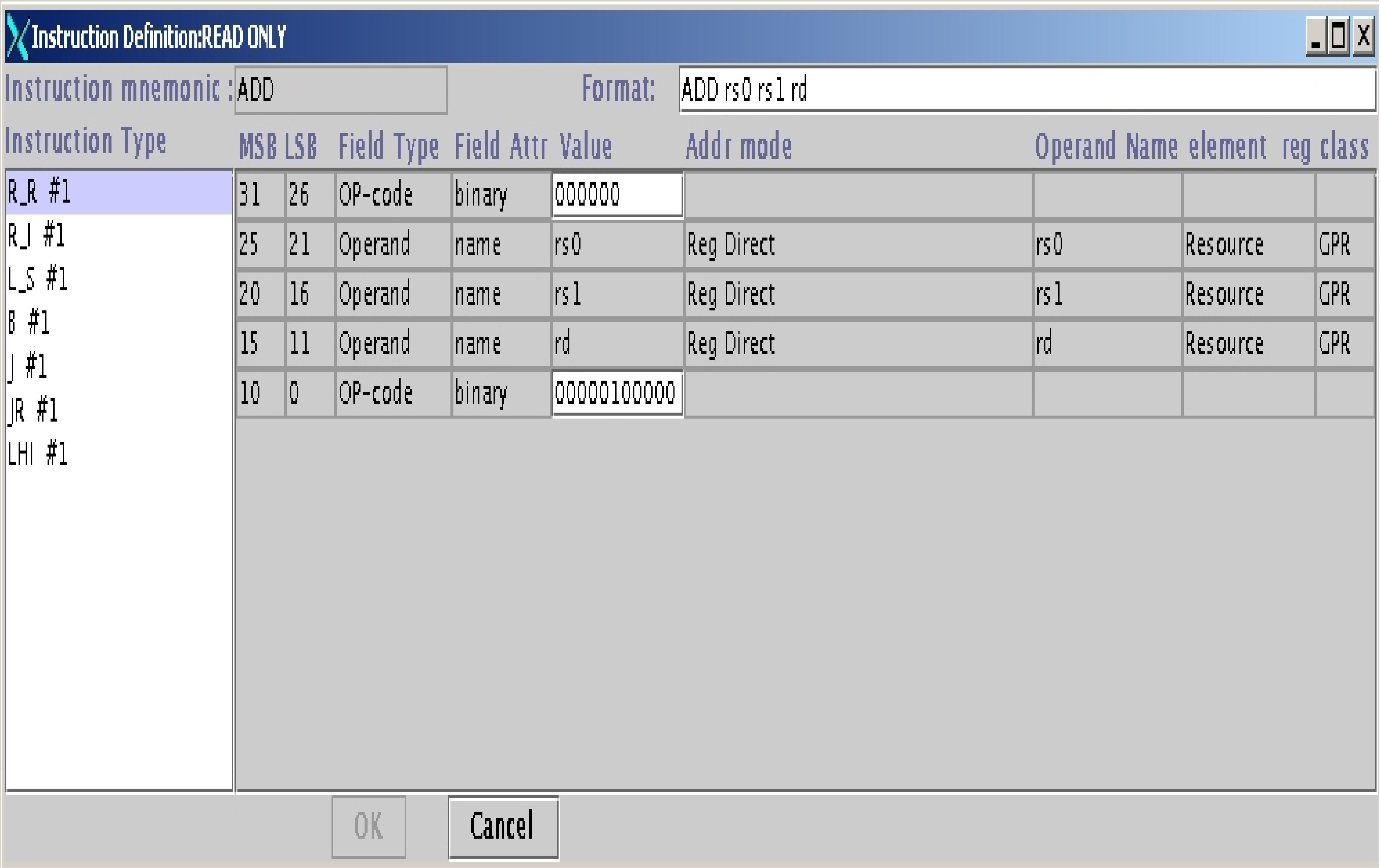
uction Type Definition

I	Name	#	MSB	LSB	Field Type	Field Attr	Name/Value	Addr mode	Operand Name	element
	R_R	1	31	26	OP-code	name	opecode			
			25	21	Operand	name	rs0	Reg Direct	rs0	Resource
			20	16	Operand	name	rs1	Reg Direct	rs1	Resource
			15	11	Operand	name	rd	Reg Direct	rd	Resource
			10	0	OP-code	name	func			
	R_I	1	31	26	OP-code	name	opecode			
			25	21	Operand	name	rs0	Reg Direct	rs0	Resource
			20	16	Operand	name	rd	Reg Direct	rd	Resource
			15	0	Operand	name	const	Immediate data	const	Immediate
	L_S	1	31	26	OP-code	name	opecode			
			25	21	Operand	name	rs0	Reg Indirect with Disp	addr	Resource
			20	16	Operand	name	rd	Reg Direct	rd	Resource
			15	0	Operand	name	const	Reg Indirect with Disp	addr	Displacem
			31	26	OP-code	name	opecode			

Instruction Field Definition

Valid	Name	Type	#	Field					Format
☑	add	R_R	1	000000	rs0	rs1	rd	00000100000	add rd rs0 rs1
☑	addu	R_R	1	000000	rs0	rs1	rd	00000100001	addu rd rs0 rs1
☑	addi	R_I	1	001000	rs0	rd	const		addi rd rs0 const
☑	addui	R_I	1	001001	rs0	rd	const		addui rd rs0 const
☑	sub	R_R	1	000000	rs0	rs1	rd	00000100010	sub rd rs0 rs1
☑	subu	R_R	1	000000	rs0	rs1	rd	00000100011	subu rd rs0 rs1
☑	subi	R_I	1	001010	rs0	rd	const		subi rd rs0 const
☑	subui	R_I	1	001011	rs0	rd	const		subui rd rs0 const
☑	mult	R_R	1	000000	rs0	rs1	rd	00000011000	mult rd rs0 rs1
☑	multu	R_R	1	000000	rs0	rs1	rd	00000011001	multu rd rs0 rs1
☑	div	R_R	1	000000	rs0	rs1	rd	00000011010	div rd rs0 rs1

ASIP Meister – ADD Instruction



Instruction mnemonic : ADD

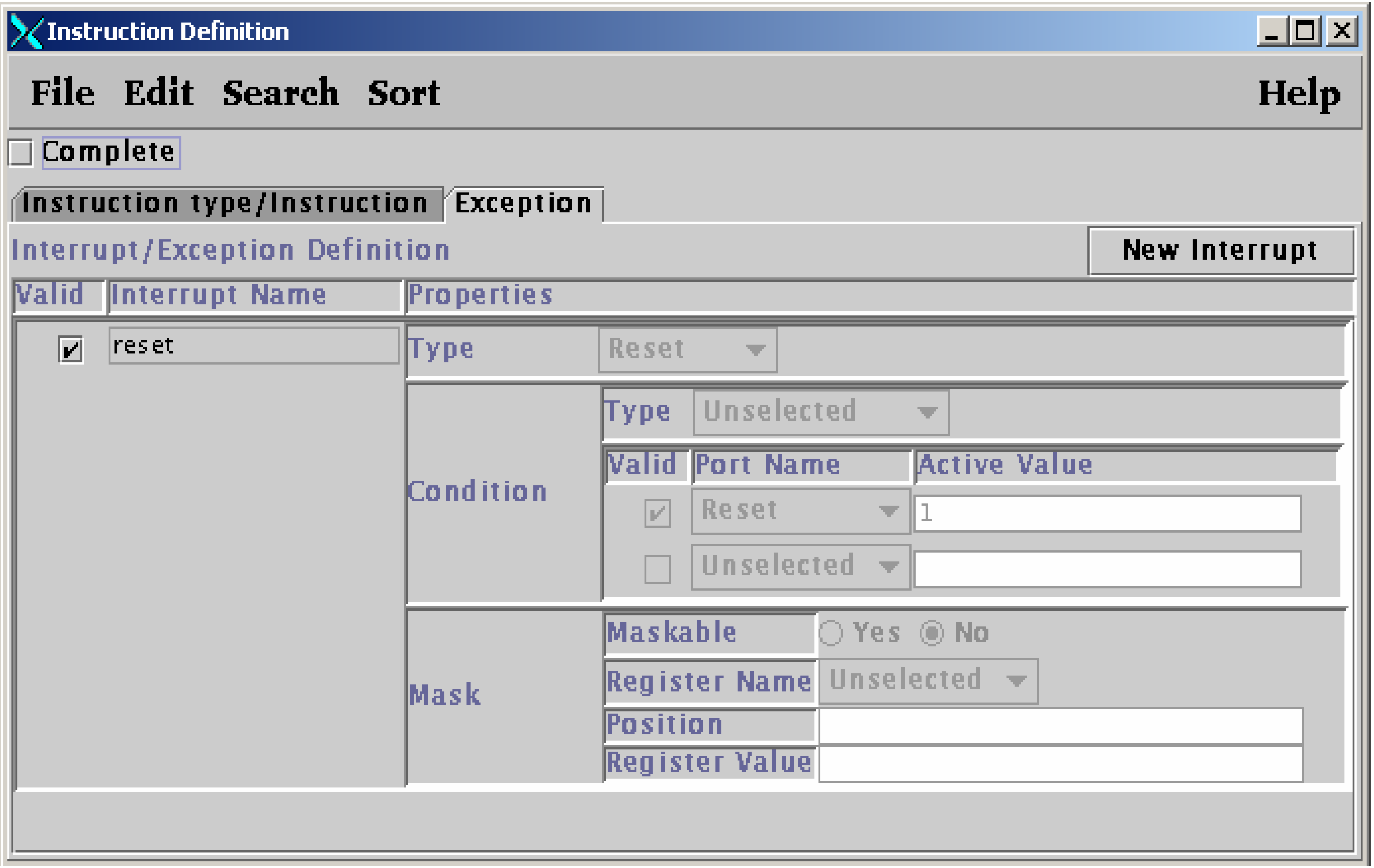
Format: ADD rs0 rs1 rd

Instruction Type

- R_R #1
- R_I #1
- L_S #1
- B #1
- J #1
- JR #1
- LHI #1

MSB	LSB	Field Type	Field Attr	Value	Addr mode	Operand Name	element	reg class
31	26	OP-code	binary	000000				
25	21	Operand	name	rs0	Reg Direct	rs0	Resource	GPR
20	16	Operand	name	rs1	Reg Direct	rs1	Resource	GPR
15	11	Operand	name	rd	Reg Direct	rd	Resource	GPR
10	0	OP-code	binary	00000100000				

ASIP Meister – Exception



□ Complete

Interrupt/Exception Definition

Valid	Interrupt Name	Properties
☑	reset	Type: Reset

Condition

Type: Unselected

Valid	Port Name	Active Value
☑	Reset	1
□	Unselected	

Mask

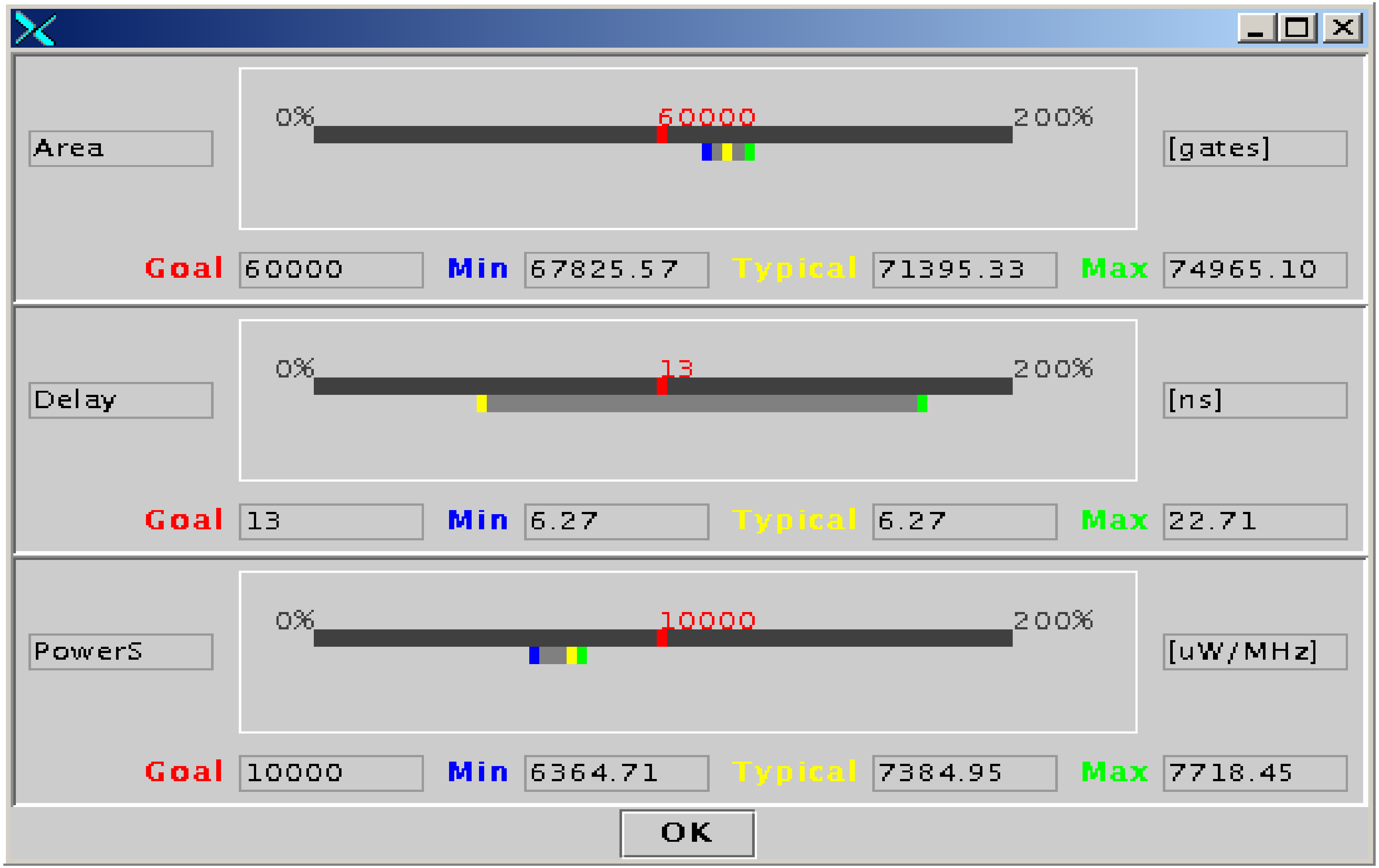
Maskable: ○ Yes ● No

Register Name: Unselected

Position:

Register Value:

ASIP Meister – Estimation



Area

[gates]

Goal 60000 Min 67825.57 Typical 71395.33 Max 74965.10

Delay

[ns]

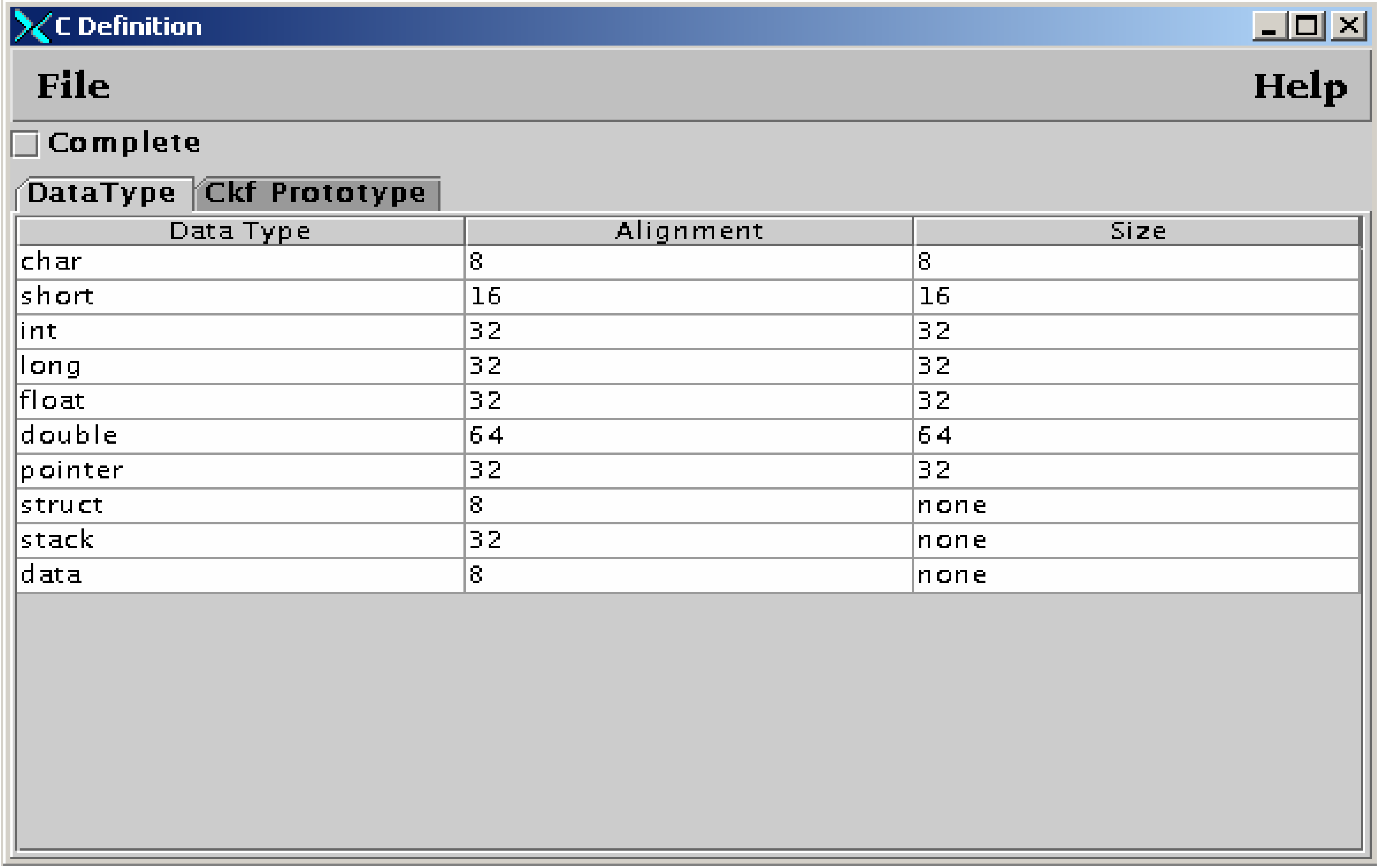
Goal 13 Min 6.27 Typical 6.27 Max 22.71

PowerS

[uW/MHz]

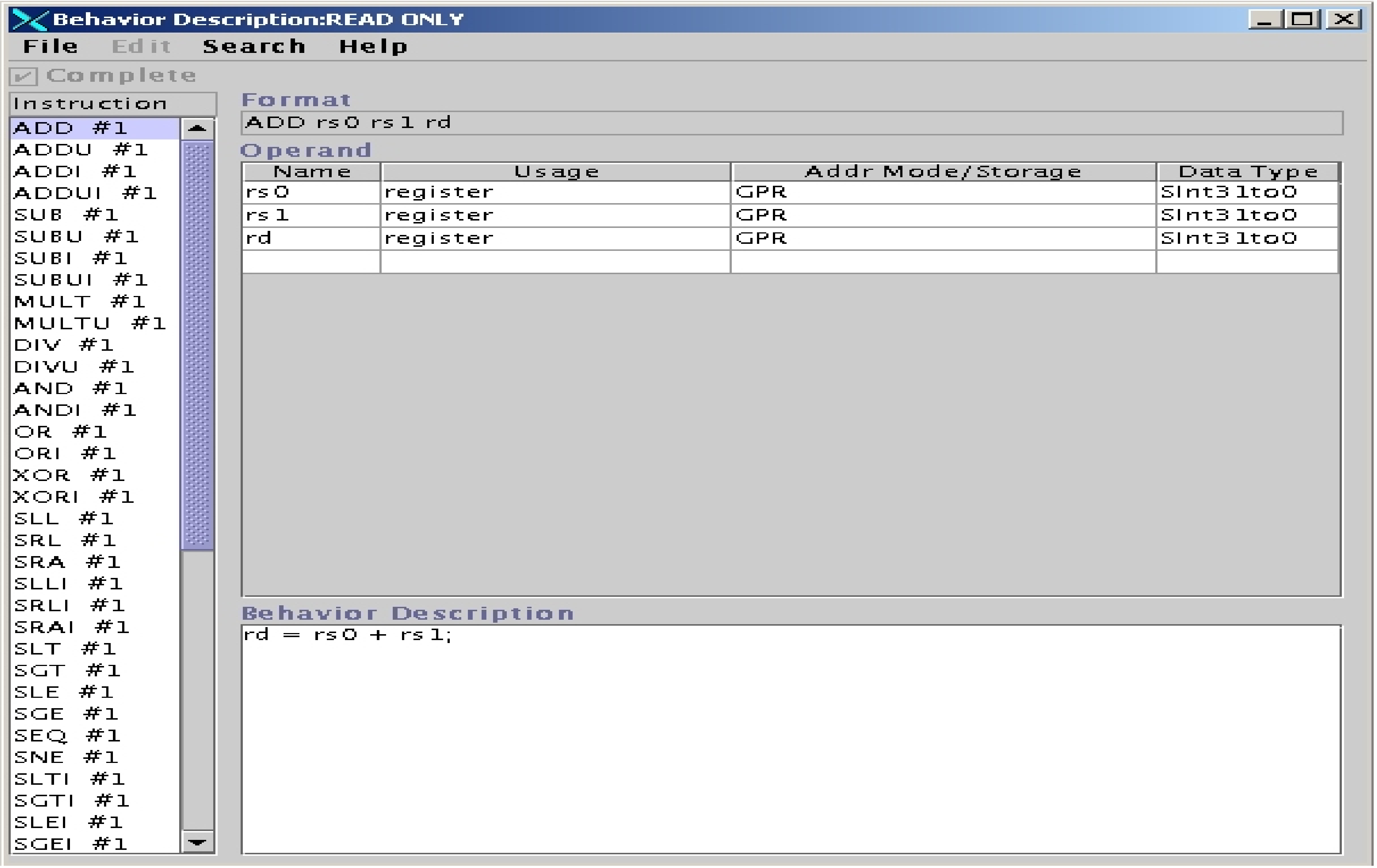
Goal 10000 Min 6364.71 Typical 7384.95 Max 7718.45

ASIP Meister – C Definition



Data Type	Alignment	Size
char	8	8
short	16	16
int	32	32
long	32	32
float	32	32
double	64	64
pointer	32	32
struct	8	none
stack	32	none
data	8	none

ASIP Meister – Behavior Description



Instruction

- ADD #1
- ADDU #1
- ADDI #1
- ADDUI #1
- SUB #1
- SUBU #1
- SUBI #1
- SUBUI #1
- MULT #1
- MULTU #1
- DIV #1
- DIVU #1
- AND #1
- ANDI #1
- OR #1
- ORI #1
- XOR #1
- XORI #1
- SLL #1
- SRL #1
- SRA #1
- SLLI #1
- SRLI #1
- SRAI #1
- SLT #1
- SGT #1
- SLE #1
- SGE #1
- SEQ #1
- SNE #1
- SLTI #1
- SGTI #1
- SLEI #1
- SGEI #1

Format

ADD rs0 rs1 rd

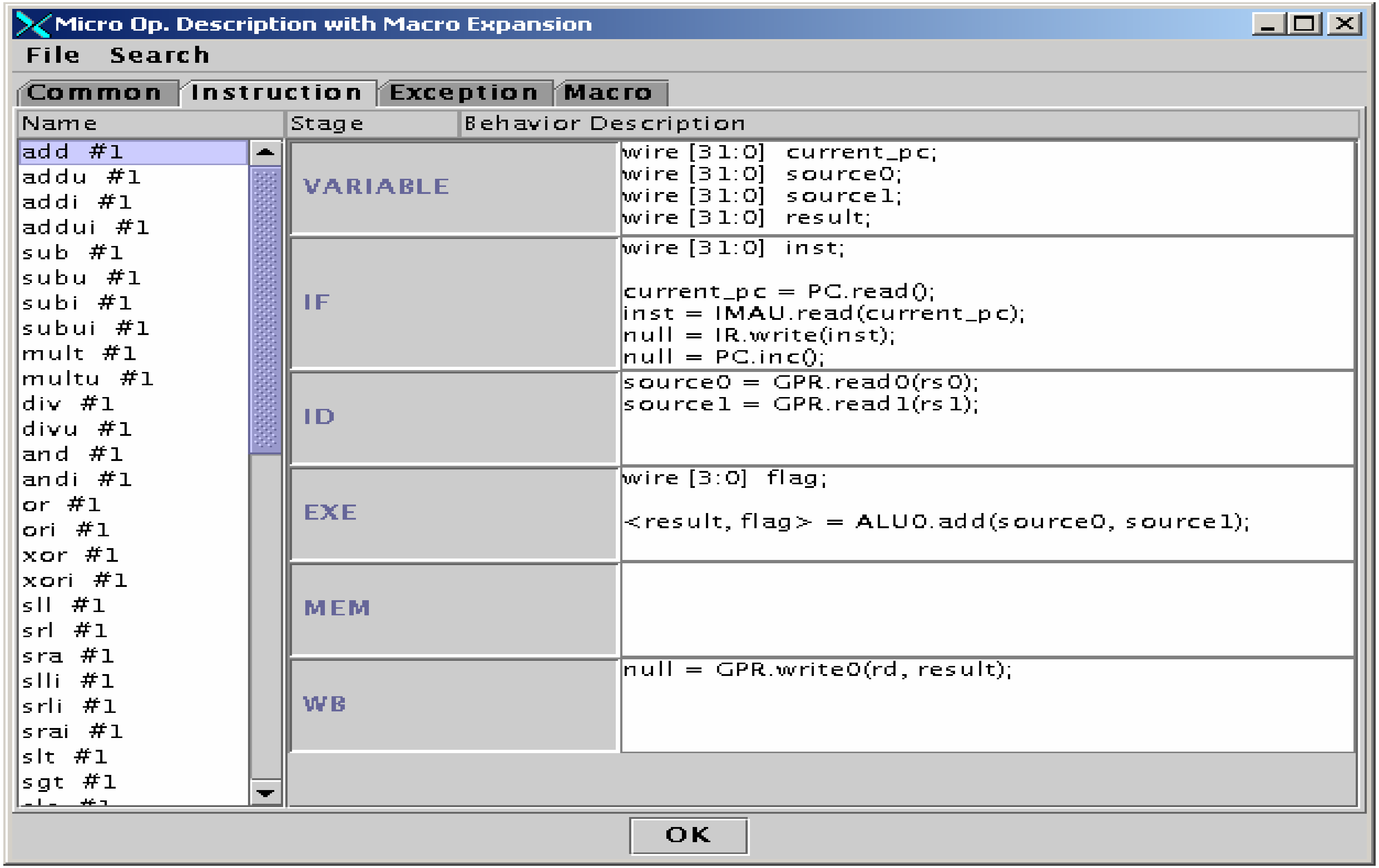
Operand

Name	Usage	Addr Mode/Storage	Data Type
rs0	register	GPR	SInt31to0
rs1	register	GPR	SInt31to0
rd	register	GPR	SInt31to0

Behavior Description

```
rd = rs0 + rs1;
```

ASIP Meister – MicroOp. Description



Name

- add #1
- addu #1
- addi #1
- addui #1
- sub #1
- subu #1
- subi #1
- subui #1
- mult #1
- multu #1
- div #1
- divu #1
- and #1
- andi #1
- or #1
- ori #1
- xor #1
- xori #1
- sll #1
- srl #1
- sra #1
- slli #1
- srli #1
- srai #1
- slt #1
- sgt #1

Stage	Behavior Description

VARIABLE	wire [31:0] current_pc;
	wire [31:0] source0;
	wire [31:0] source1;
	wire [31:0] result;

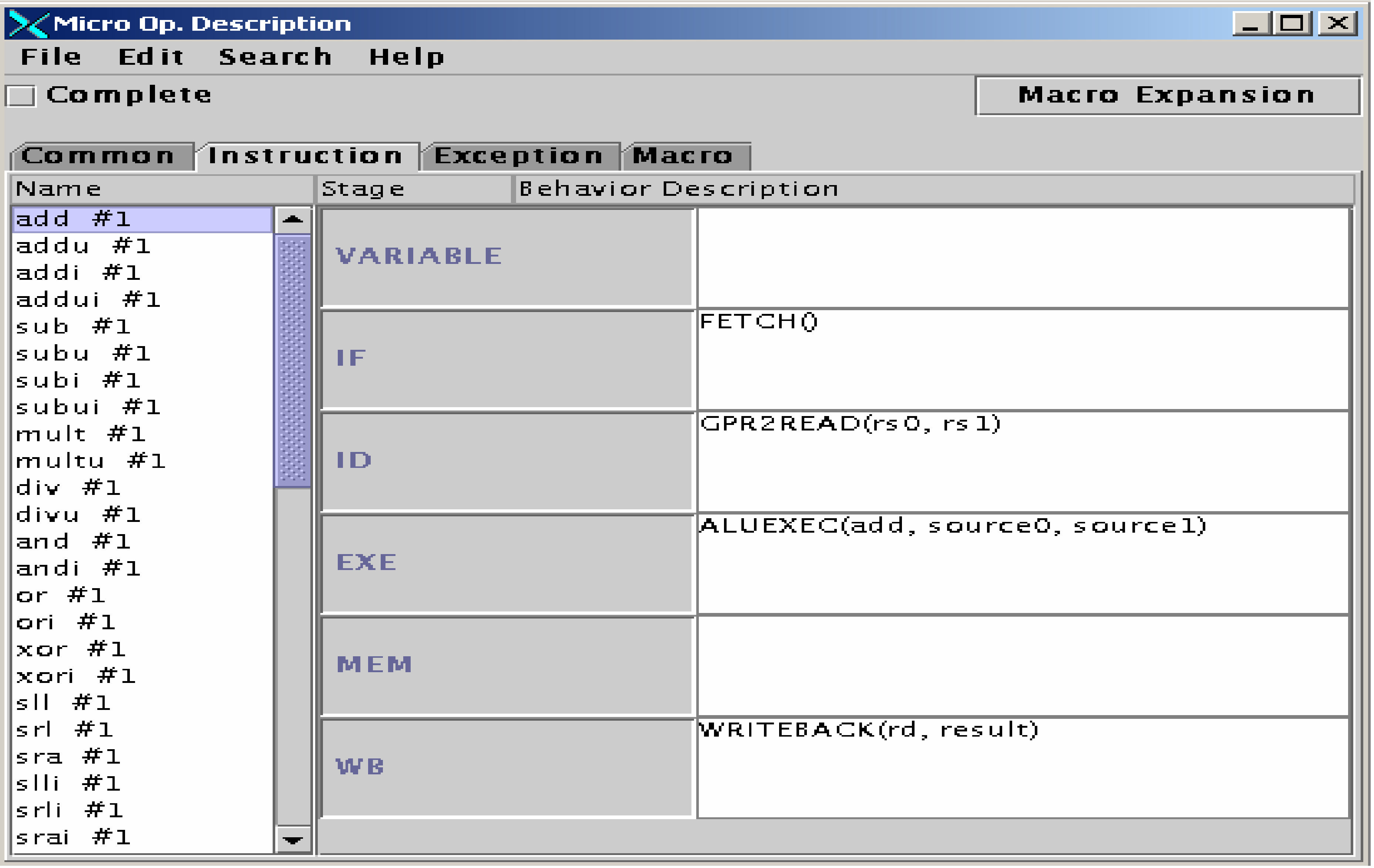
IF	wire [31:0] inst;
	
	current_pc = PC.read();
	inst = IMAU.read(current_pc);
	null = IR.write(inst);
	null = PC.inc();

ID	source0 = GPR.read0(rs0);
	source1 = GPR.read1(rs1);

EXE	wire [3:0] flag;
	
	<result, flag> = ALU0.add(source0, source1);

MEM	
WB	null = GPR.write0(rd, result);

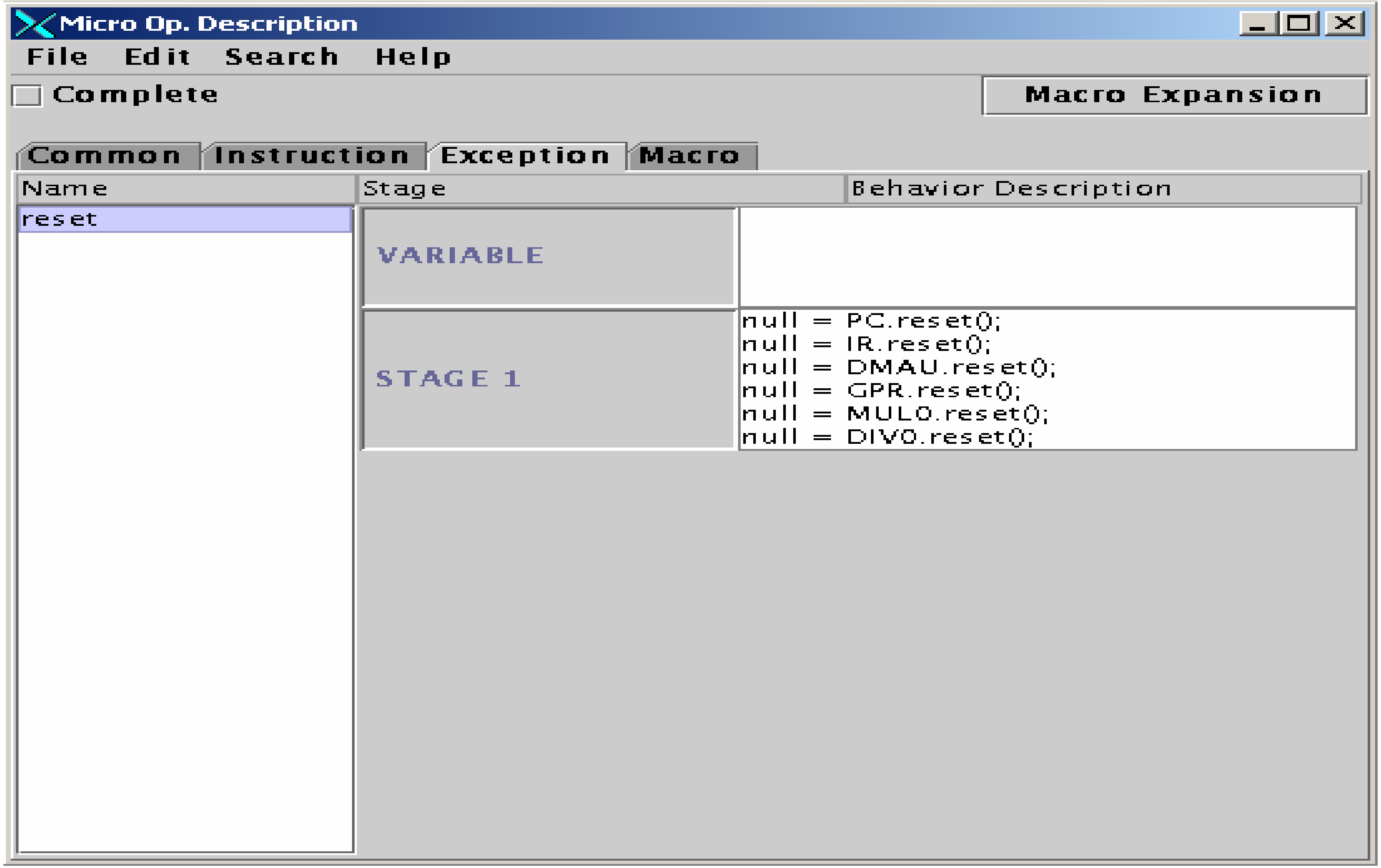
ASIP Meister – MicroOp. Macro



Name
add #1
addu #1
addi #1
addui #1
sub #1
subu #1
subi #1
subui #1
mult #1
multu #1
div #1
divu #1
and #1
andi #1
or #1
ori #1
xor #1
xori #1
sll #1
srl #1
sra #1
slli #1
srli #1
srai #1

Stage	Behavior Description
VARIABLE	
IF	FETCH()
ID	GPR2READ(rs0, rs1)
EXE	ALUEXEC(add, source0, source1)
MEM	
WB	WRITEBACK(rd, result)

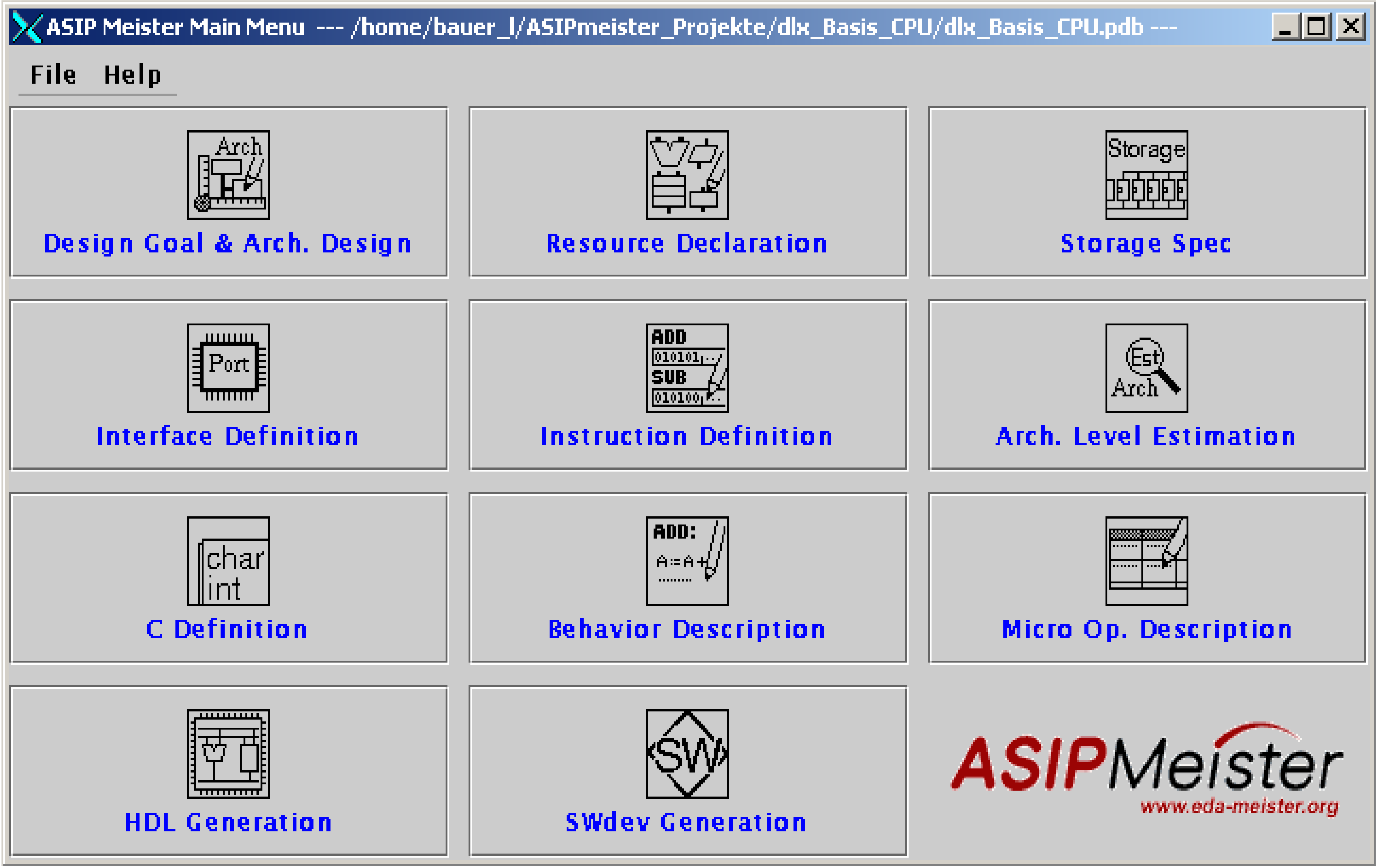
ASIP Meister – MicroOp. Exception



Name	Stage	Behavior Description
reset	VARIABLE	

	STAGE 1	null = PC.reset();
		null = IR.reset();
		null = DMAU.reset();
		null = GPR.reset();
		null = MULO.reset();
		null = DIVO.reset();

ASIP Meister – Main Menu



ASIP Meister advantages

- GUI, with partial consistence checking
- Early Estimation
  - Delay
  - Power
  - Area
- Configurable Pipeline properties
  - Number of stages
  - Behavior of a stage
  - Number of delay slots
- Configurable Hardware Blocks
  - Bit-size
  - Internal implementation (e.g. rca vs. cla)

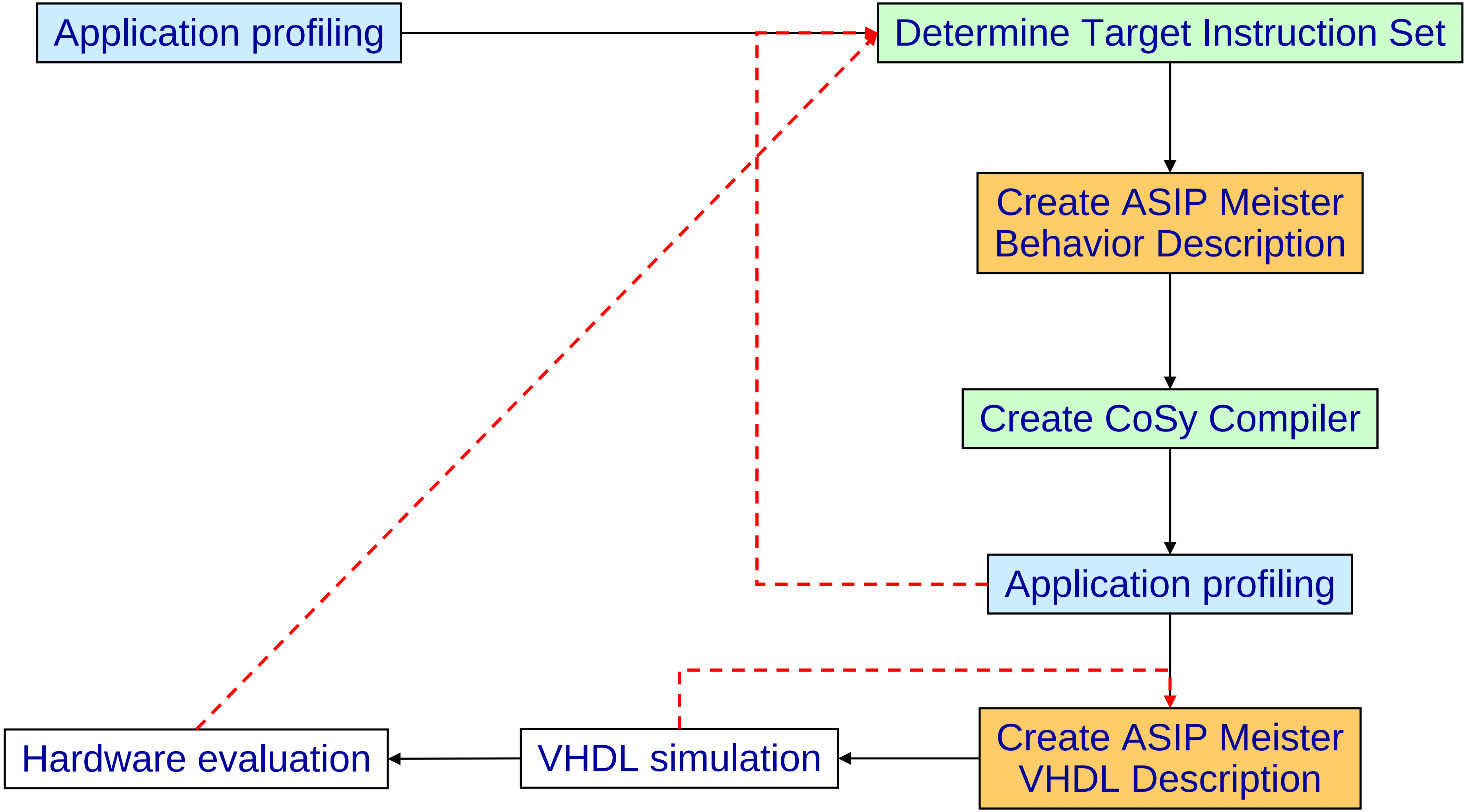
ASIP Meister advantages (cont’d)

- Automatically generated Control Unit
  - => Implement assembly-commands independent of each other
- VHDL Description at 3 different abstraction levels
  - Behavior
  - RT-Level
  - Gate-Level
- Software support
  - Assembler
  - C-Compiler

ASIP Meister disadvantages

- Installation
  - Not running with new JRE (> 1.3.x)
  - Binaries dynamically linked against old libstdc
- Only predefined Hardware Blocks
  - Documentation currently only available in Japanese
- Only very basic pipeline processors supported
  - No Data-forwarding
  - No multicycle-instruction without specific Hardware Block
- GUI consistence checking is not exhaustive
  - No assertion, that no internal reserved words are used (e.g. reg, fp, sp, mod)
  - No assertion, that all "must have's" are included (e.g. Reset-Exception, Instruction-Memory-Access-Unit)

Integrating ASIP Meister into the design flow



Application profiling

Determine Target Instruction Set

Create ASIP Meister Behavior Description

Create CoSy Compiler

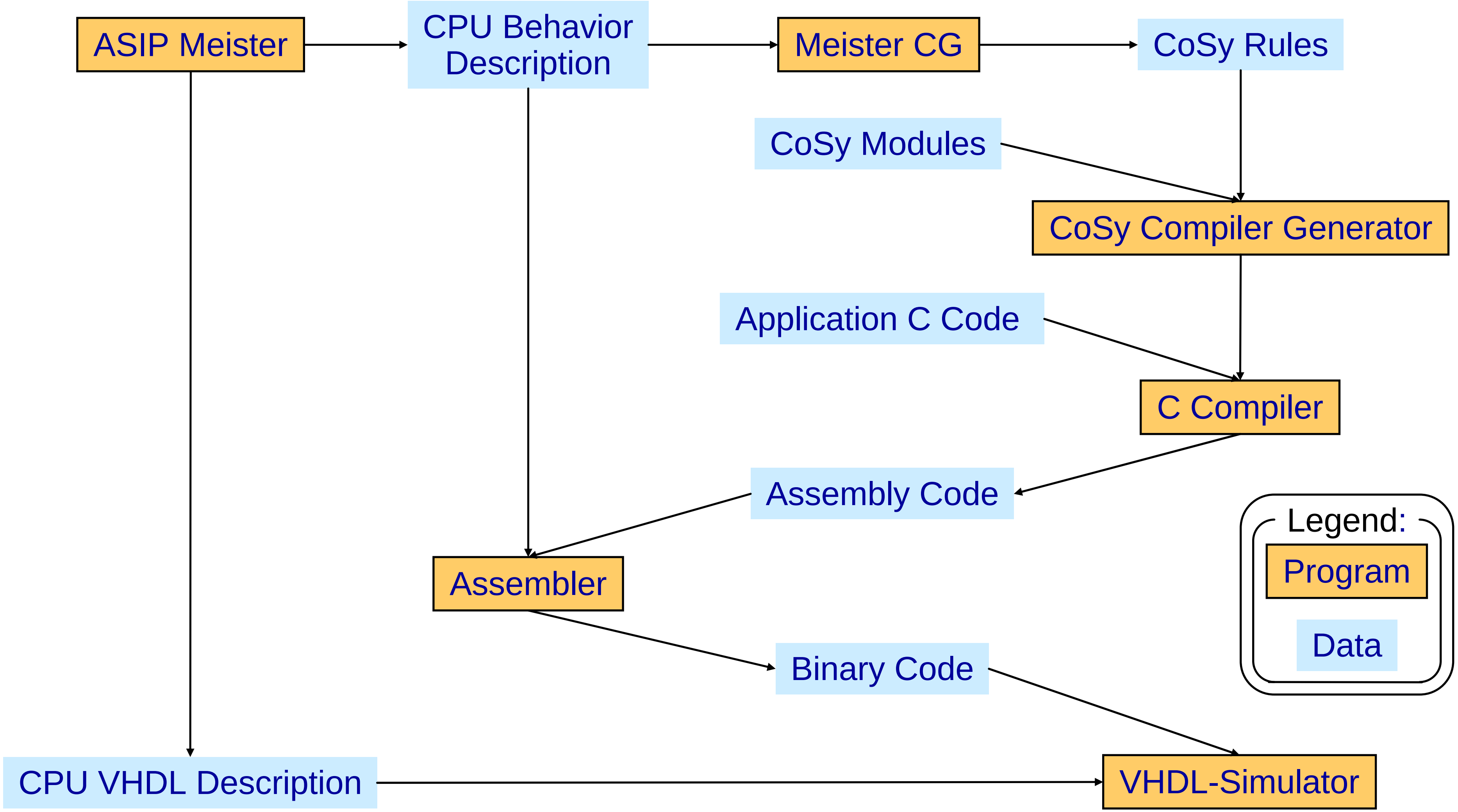
Application profiling

Create ASIP Meister VHDL Description

VHDL simulation

Hardware evaluation

Data flow from ASIP Meister to VHDL Simulation



ASIP Meister

CPU Behavior Description

Meister CG

CoSy Rules

CoSy Modules

CoSy Compiler Generator

Application C Code

C Compiler

Assembly Code

Assembler

Binary Code

CPU VHDL Description

VHDL-Simulator

Legend:

Program

Data

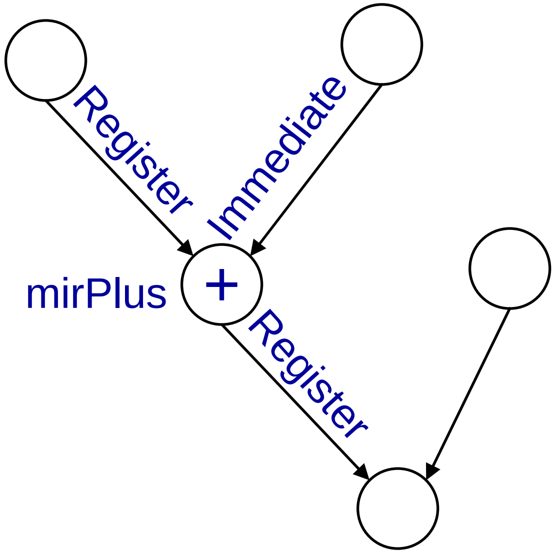
CoSy Compiler & Meister CG

- CoSy Compiler:
  - Retargetabel Compiler Generator
  - Creates Compiler for a specific architecture
  - Rules are used to describe the target architecure

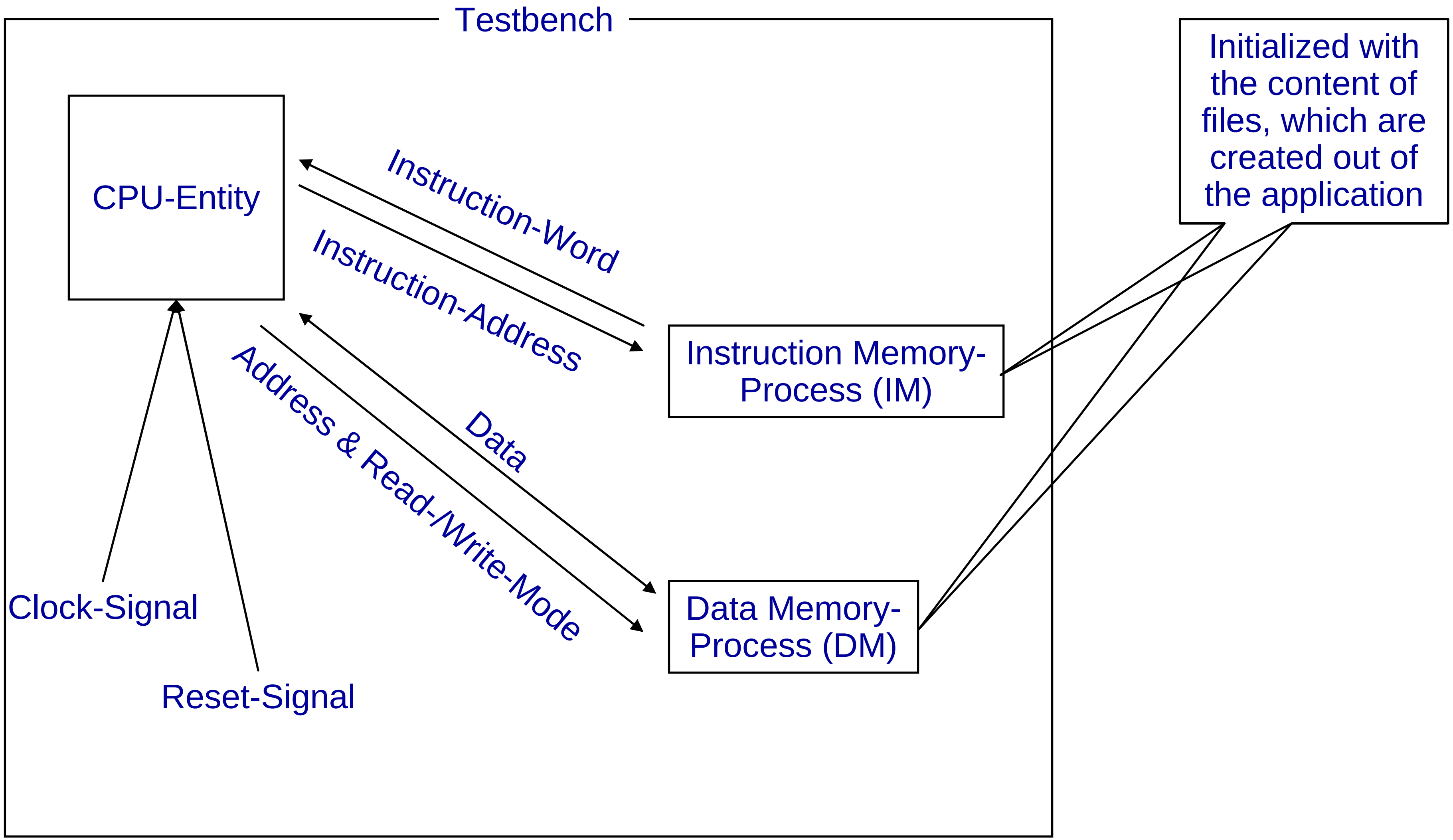
- Meister CG
  - New extension for ASIP Meister
  - Transforming CPU Behavior Description (i.e. Achitecture Description & Instruction Set Description) into CoSy rules

Example for CoSy Rule

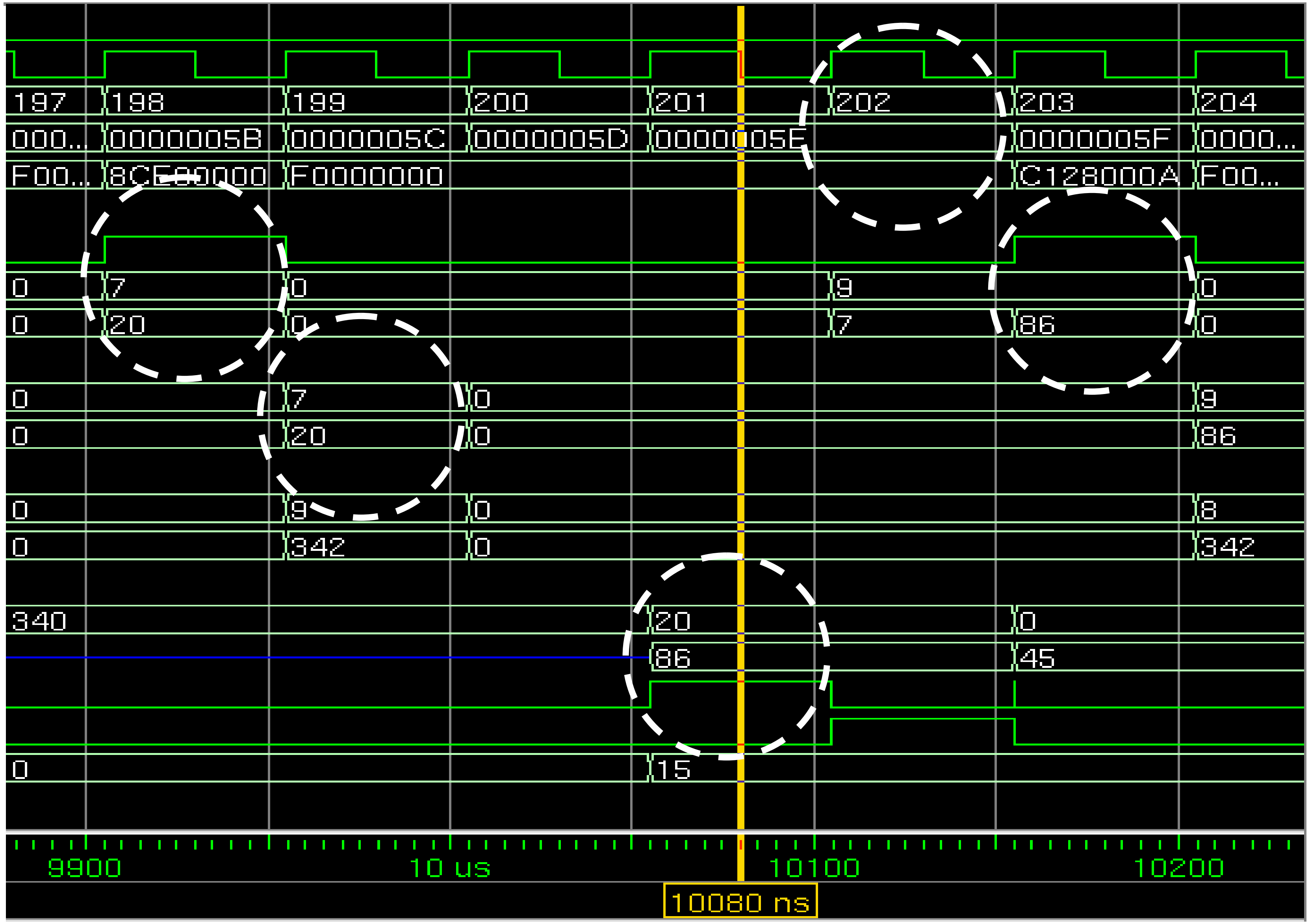
```
RULE [add_0] add:mirPlus (rs0:register, imm:immediate) -> rd:register;
CONDITION { IS_UNSIGNED(add.type) &&
                    (imm.value >=0) && (imm.value<256) }
COST 1;
EMIT {
    fprintf(outfile, “addui  %s, %s, $%i \n",
            REGNAME(rd), REGNAME(rs1),
            imm.value);
}
```



VHDL-simulation


Testbench
CPU-Entity
Instruction-Word
Instruction-Address
Instruction Memory-Process (IM)
Data
Address & Read-/Write-Mode
Data Memory-Process (DM)
Clock-Signal
Reset-Signal
Initialized with the content of files, which are created out of the application

Simulation-Output



Reset

Clock

Instr.Mem

Reg write

Reg read

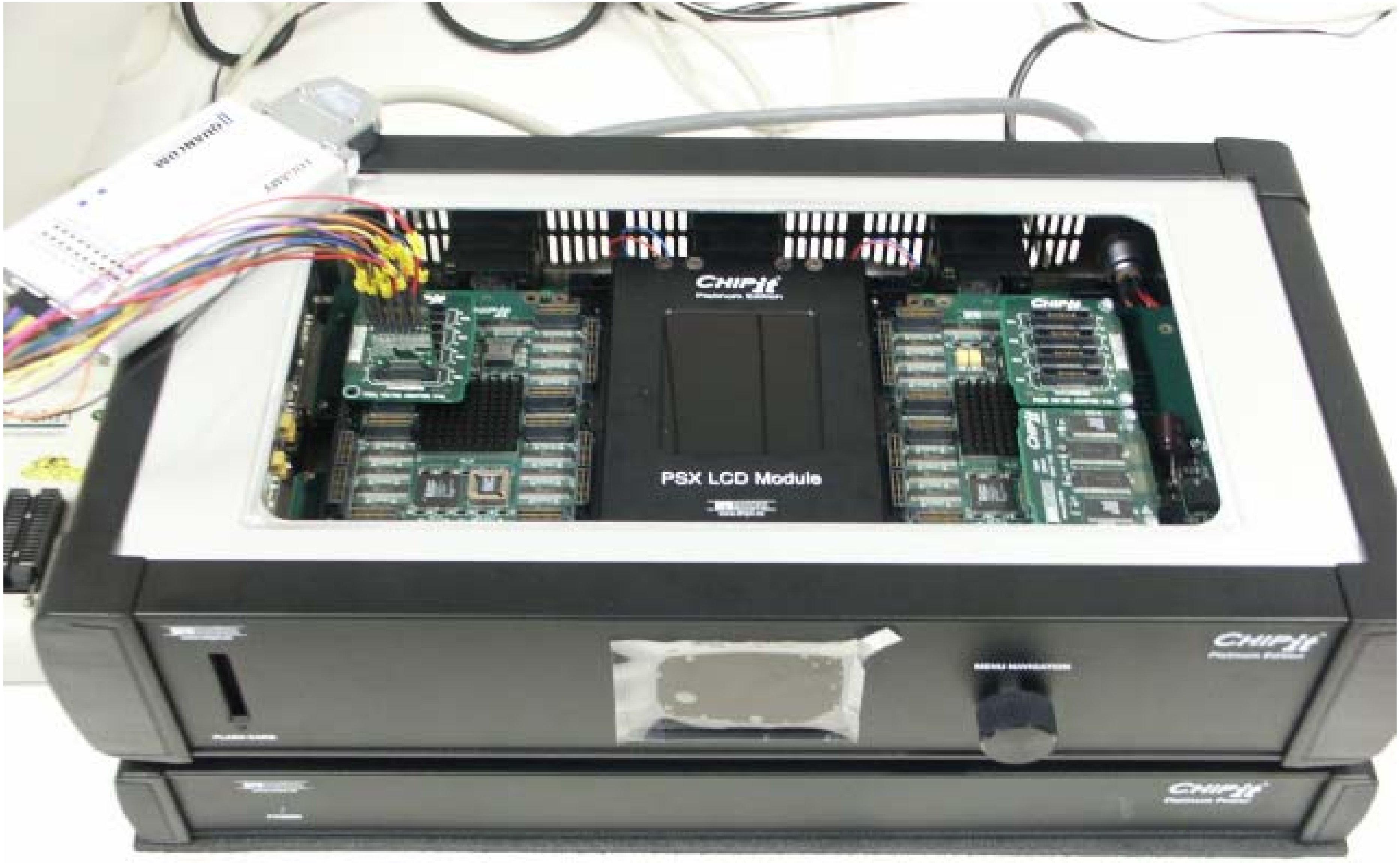
Reg read

DataMem

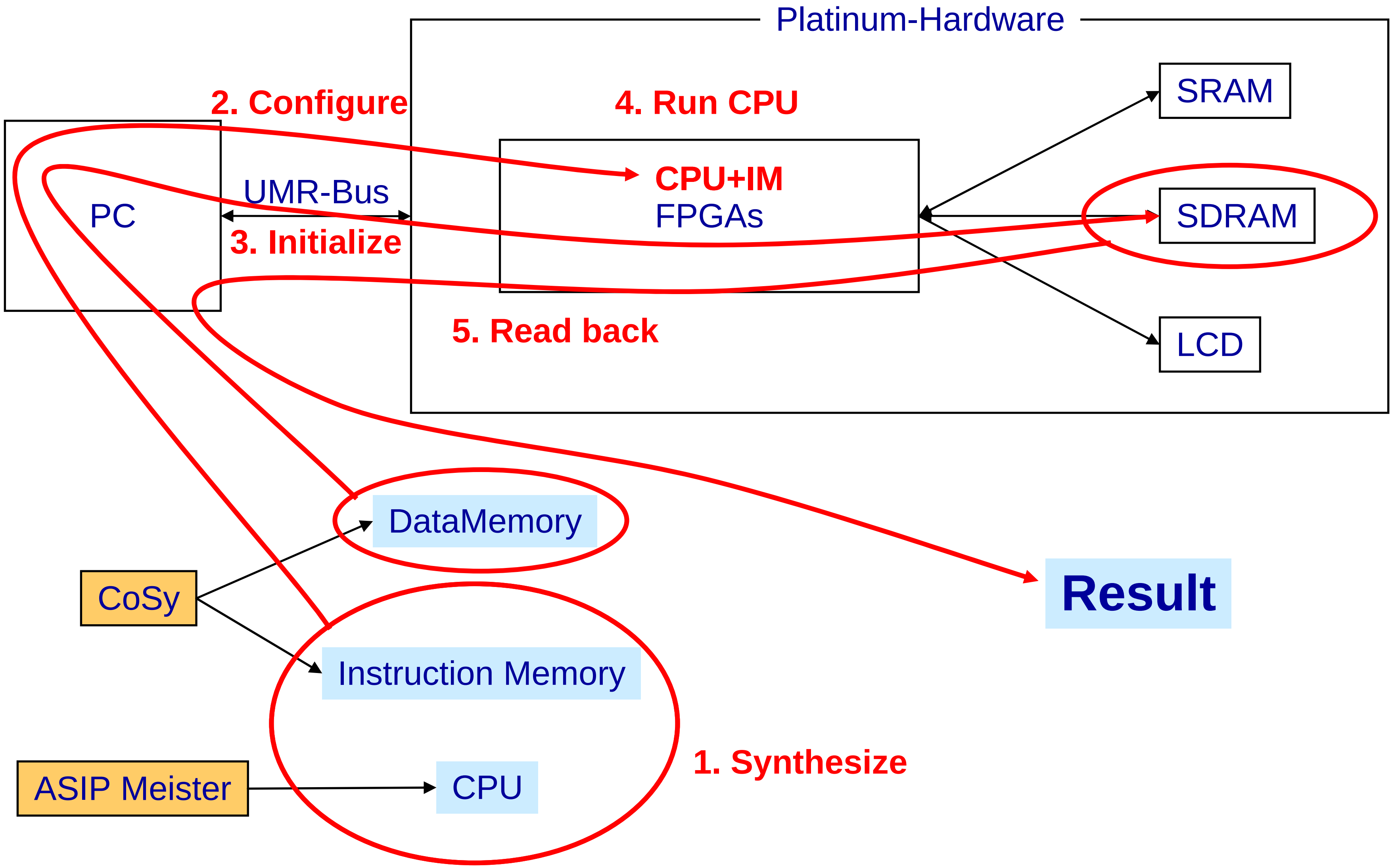
CHIPit Platinum Edition

- State-of-the-Art Prototyping-Board
- 3 Virtex-II FPGAs
- Extensible up to 9 FPGAs
- Extension boards:
  - 128 MB SDRAM / 2 MB SRAM
  - LC-Display
  - I/O for logic analyser

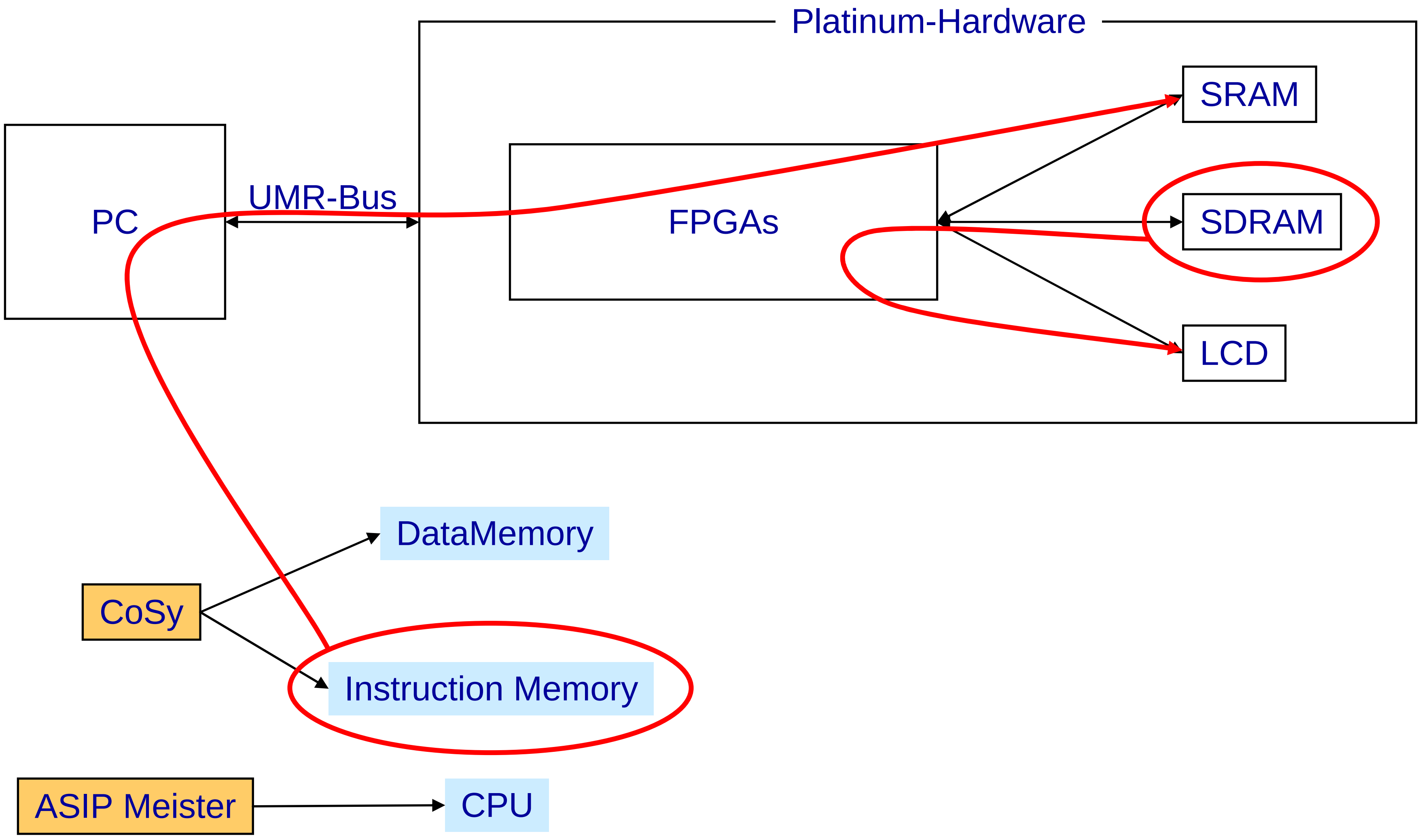
Picture of Platinum Edition



Process flow for Hardware evaluation



Future work


Platinum-Hardware
PC
UMR-Bus
FPGAs
SRAM
SDRAM
LCD
DataMemory
CoSy
Instruction Memory
ASIP Meister
CPU

Laboratory for students

- Used programs & tools:

ASIP Meister	(very detailed)
CoSy	(using)
ModelSim	(detailed)
Dlxsim	(detailed)
Xilinx ISE	(using)

- Learn, how to:
  - Design and create an ASIP
  - Find and implement new instructions
  - Simulate assembly-code with new instructions
  - Simulate a processor
  - Evaluate a design (processor & application)
  - Determine execution speed and power consumtion
  - Understand the side-effects of new instructions (area, power, speed)

References and Sources

- [ASIPMeister] ASIP Meister Homepage: http://www.eda-meister.org/asip-meister/
- [CoSy] Homepage von ACE, den Entwicklern des CoSy Übersetzers: www.ace.nl
- [ProDesign] ProDesign Homepage: http://www.prodesign-europe.com
- [Platinum] Broschüre zu der CHIPit Platinum Edition der Firma ProDesign: http://www.uchipit.com/ce/pdf/CHIPit_PlatinumEdition_Flyer.pdf