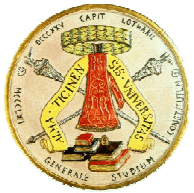


Università degli Studi di Pavia
Facoltà di Ingegneria
Dipartimento di Elettronica

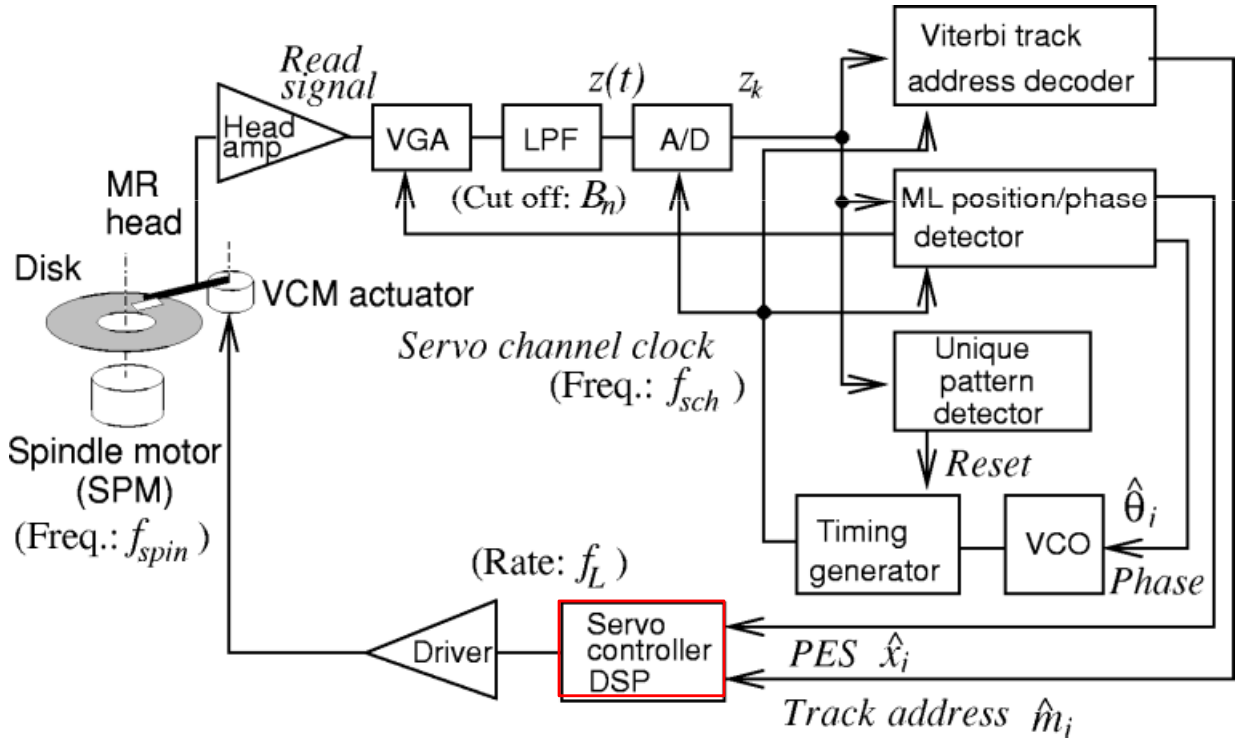
A.A. 2008/09
Dottorato in Microelettronica
VIII Ciclo Nuova Serie (XXII Ciclo)

Application Specific Instruction-set Processor for Hard Disk Drive Servo operation: the Microprogrammed Servo Sequencer

Dottorando: Paola BALDRIGHI
Tutore: Prof. Carla VACCHI

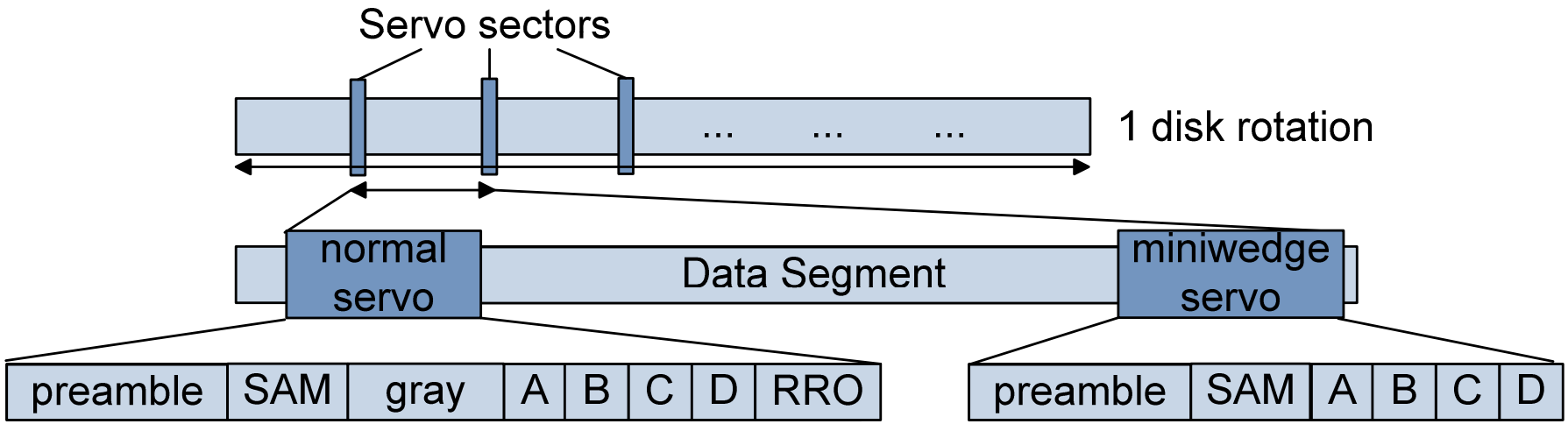
Servo System

- implementation of a part of the Servo System: the Regular Servo Sequencer (RSS) control logic (Finite State Machine)



Servo sectors

- Normal Servo for track seek
- Miniwedge Servo for track following



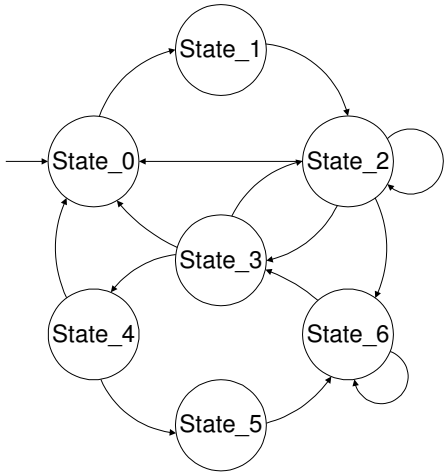
Servo System redesign

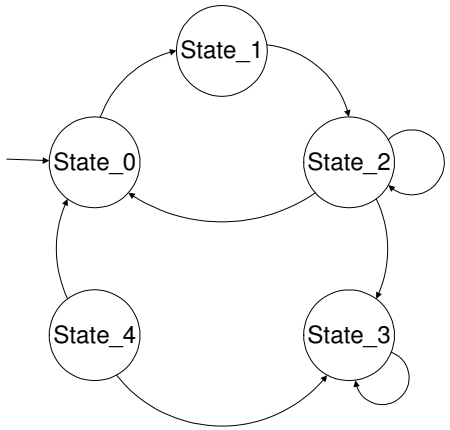
All microelectronics companies uses a FSM approach for Servo System implementation

Distinct market segments require different Servo data

Redesign of Regular Servo Sequencer

Servo 1



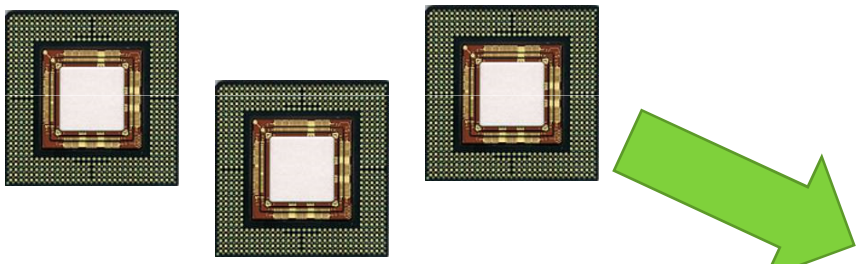


Servo 2

Reconfiguration vs Redesign

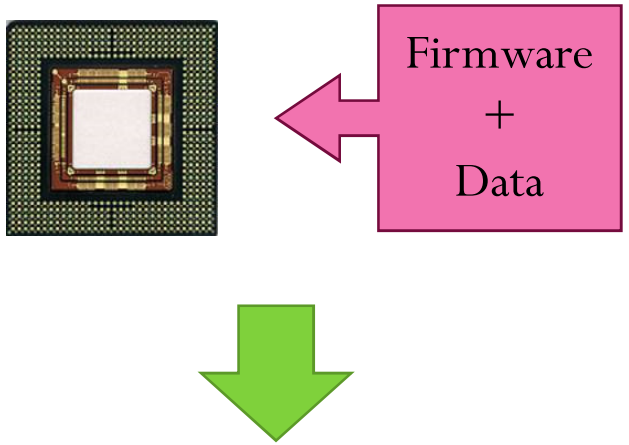
State of the art:

different Servo subsystem (Regular Servo Sequencer)



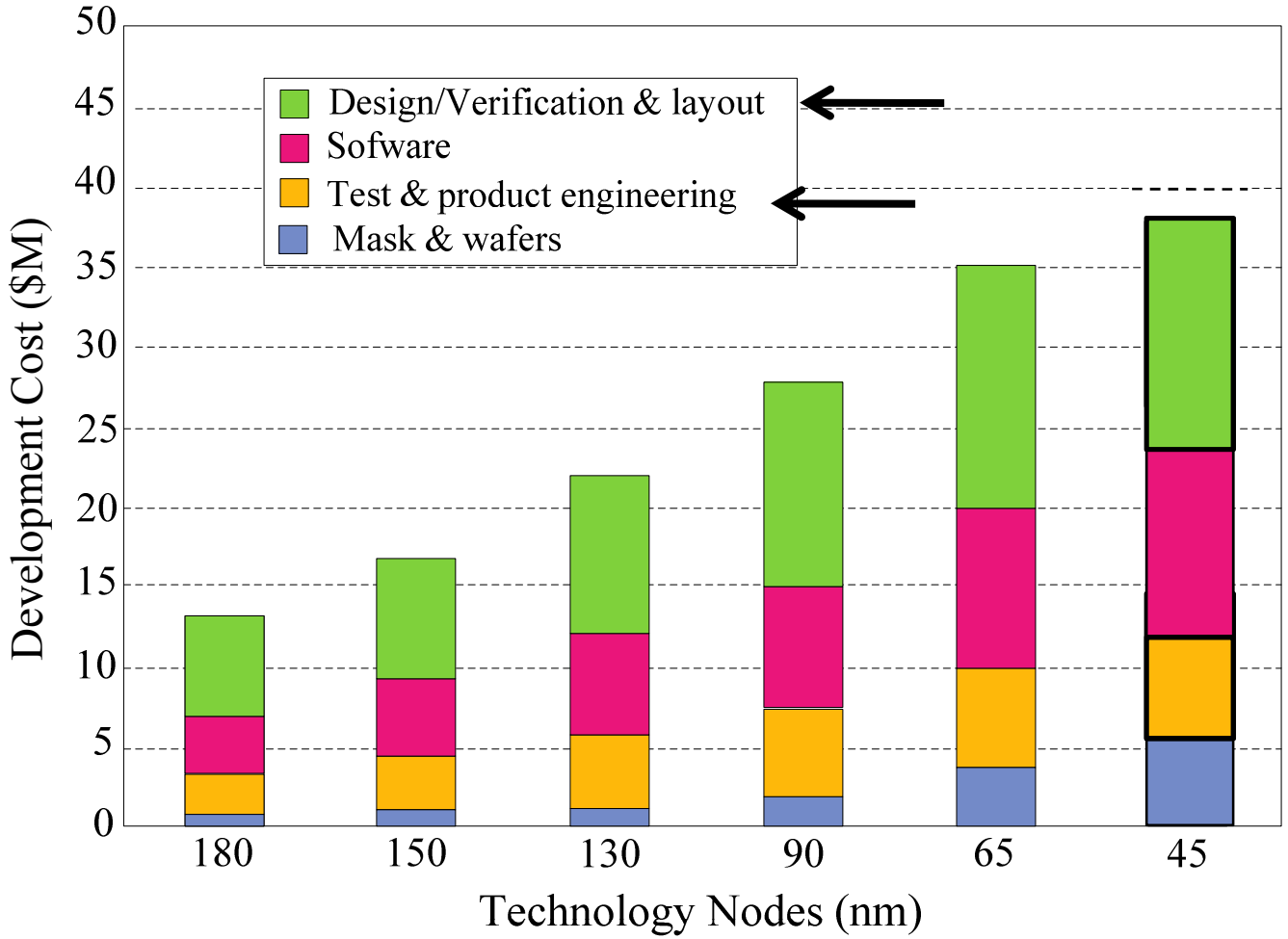
	RSS	MSS
area	☺	☹
power	☺	☹
design	☹	☺

single ASIP with different behaviors, being described through custom firmwares. (Microprogrammed Servo Sequencer)



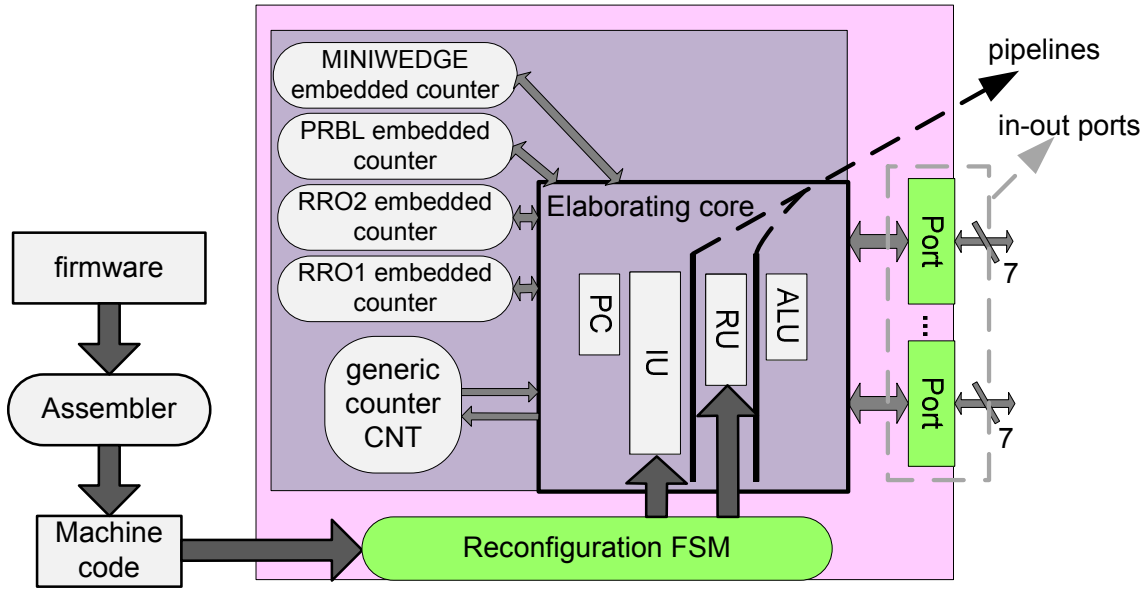
Redesign cost reduction

CMOS integrated circuit non recurrent cost



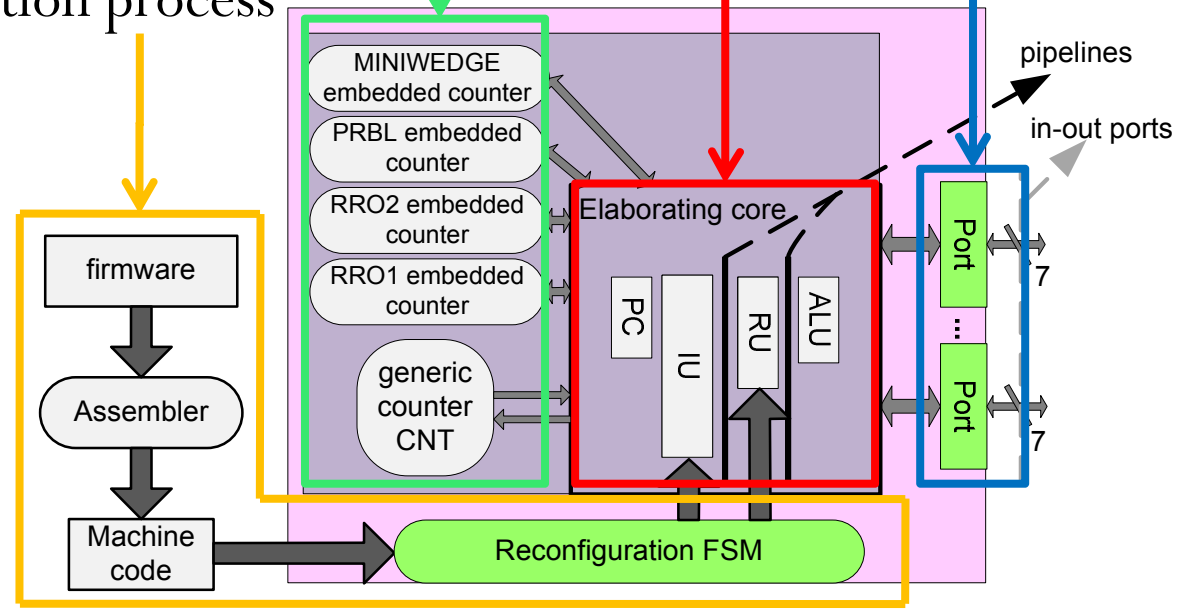
Microprogrammed Servo Sequencer

- **NO** commercial µP (ex. ARM)
- **single** ASIP with customized ISA
  - **different behaviors** described through firmwares
  - startup reconfiguration Finite State Machine

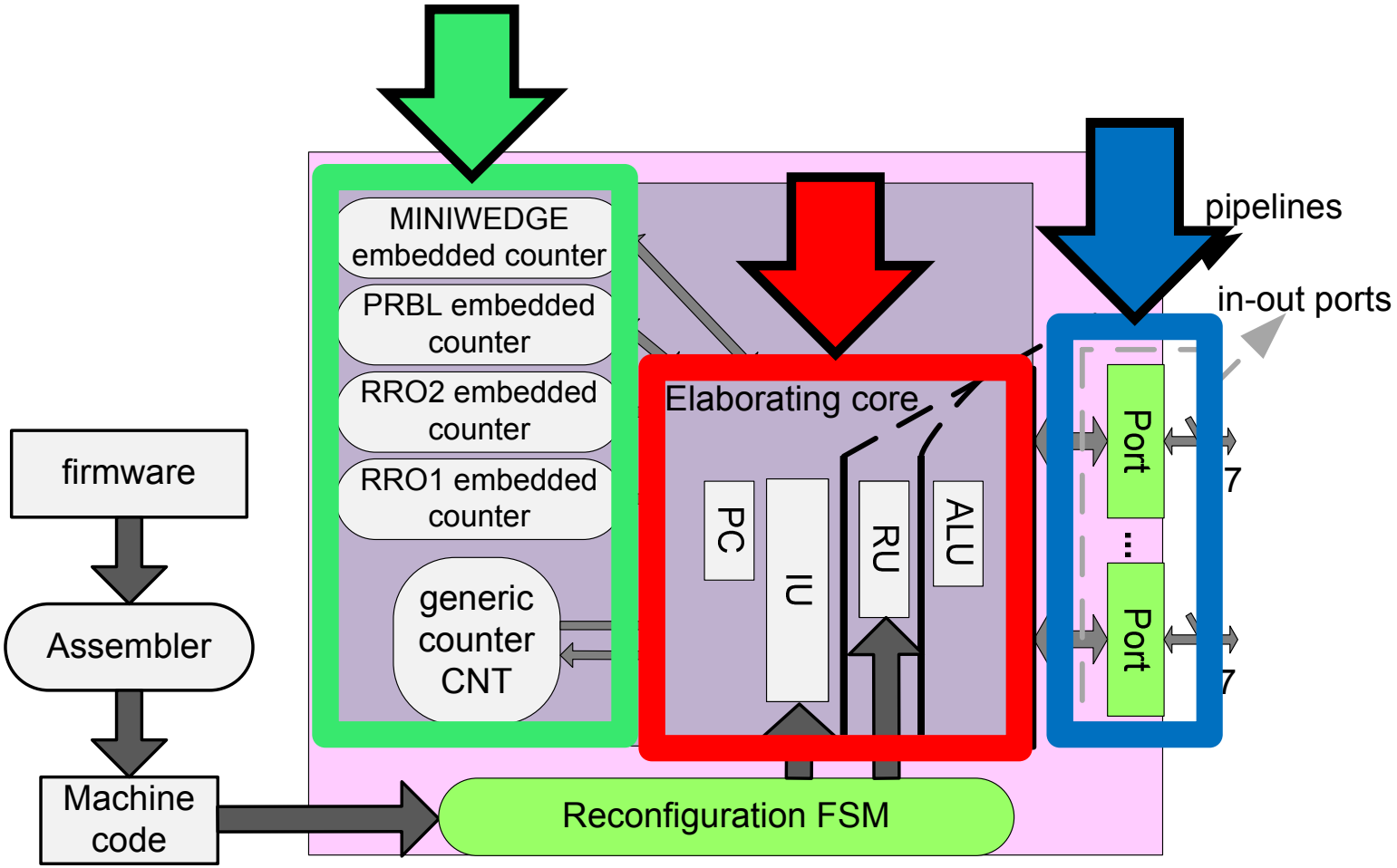


Microprogrammed Servo Sequencer

- Microprogrammed Servo Sequencer characteristics:
  - elaborating core
  - reconfigurable I/O ports
  - embedded counters
  - Reconfiguration process

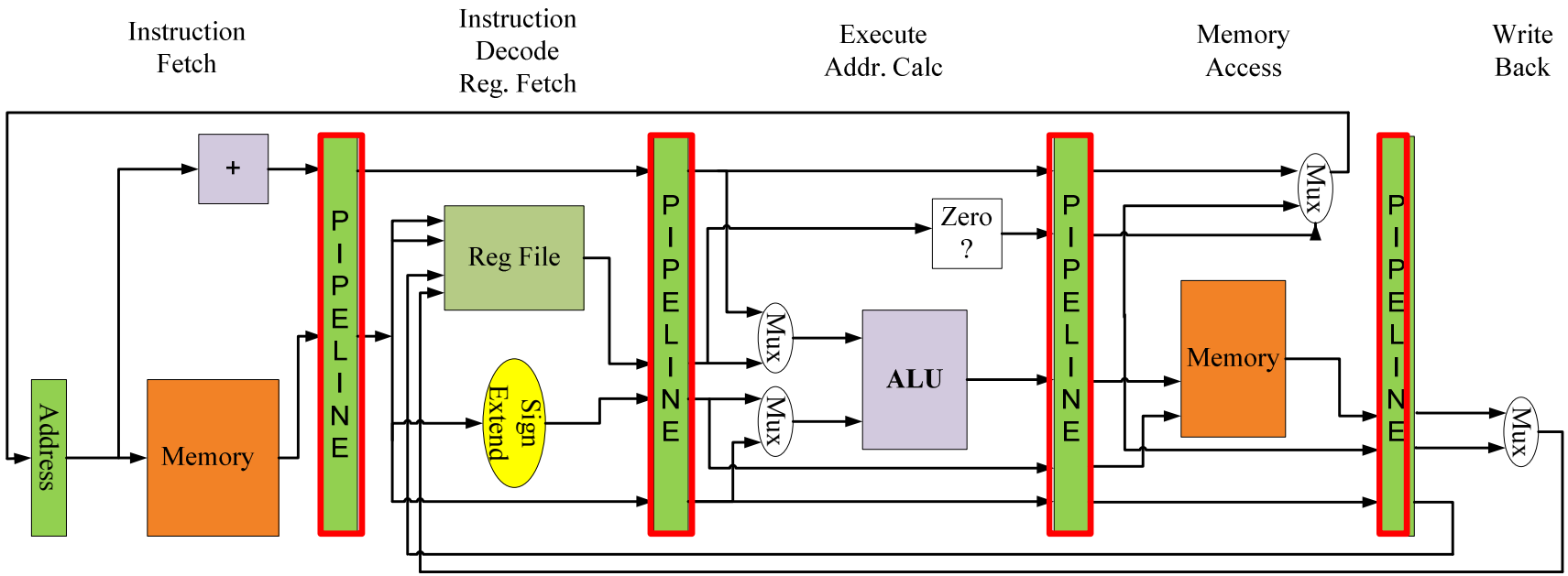


Elaborating core



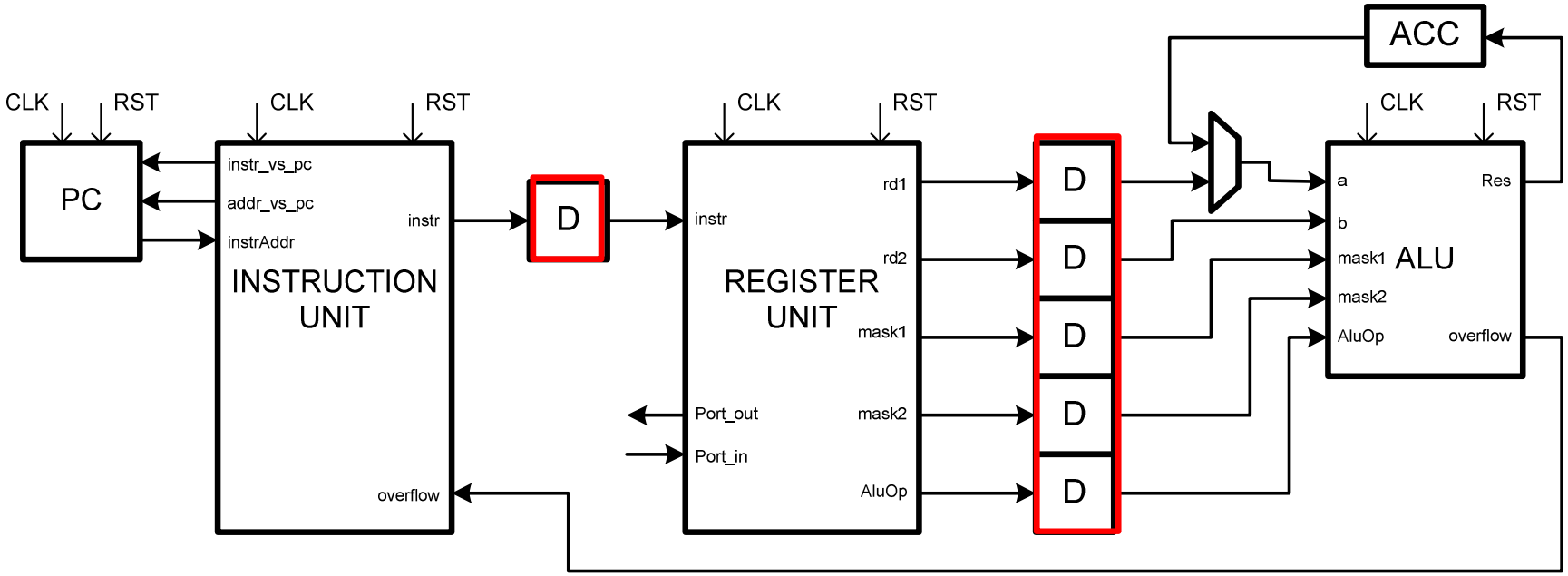
MIPS Elaborating core

- MIPS processor
  - RISC
  - Harvard approach

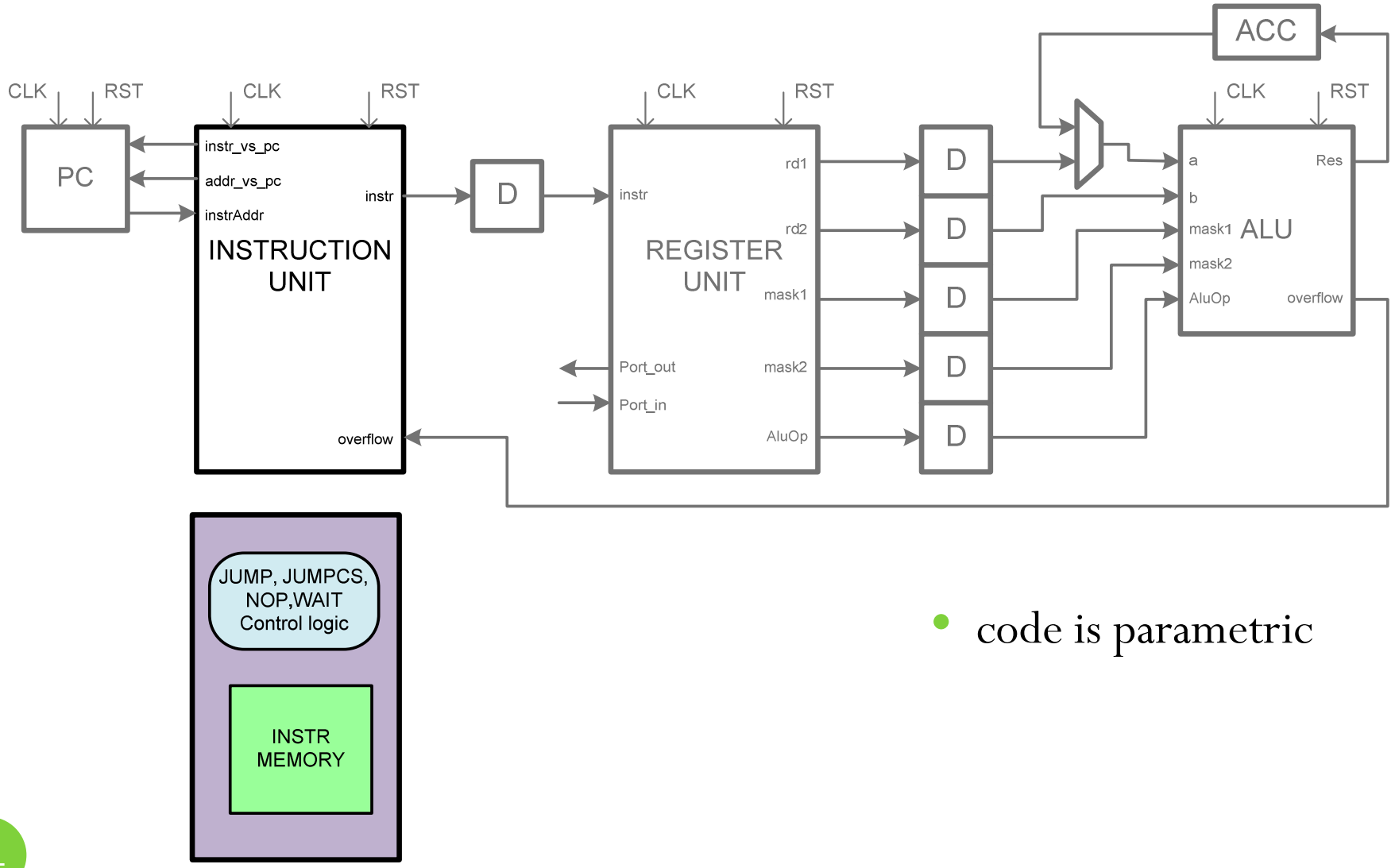


MSS Elaborating core

- code is parametric
- only 2 pipelines

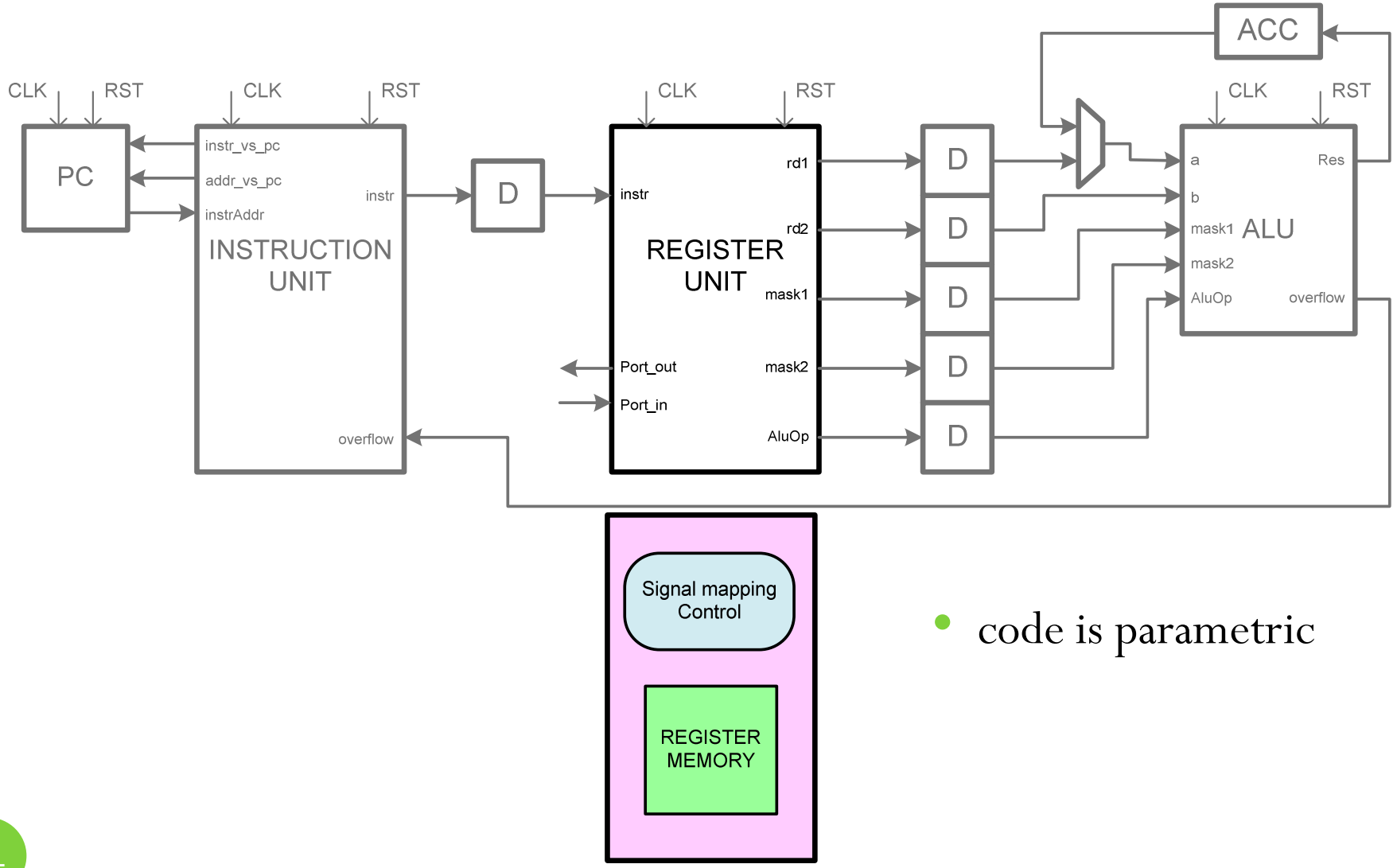


Instruction Unit



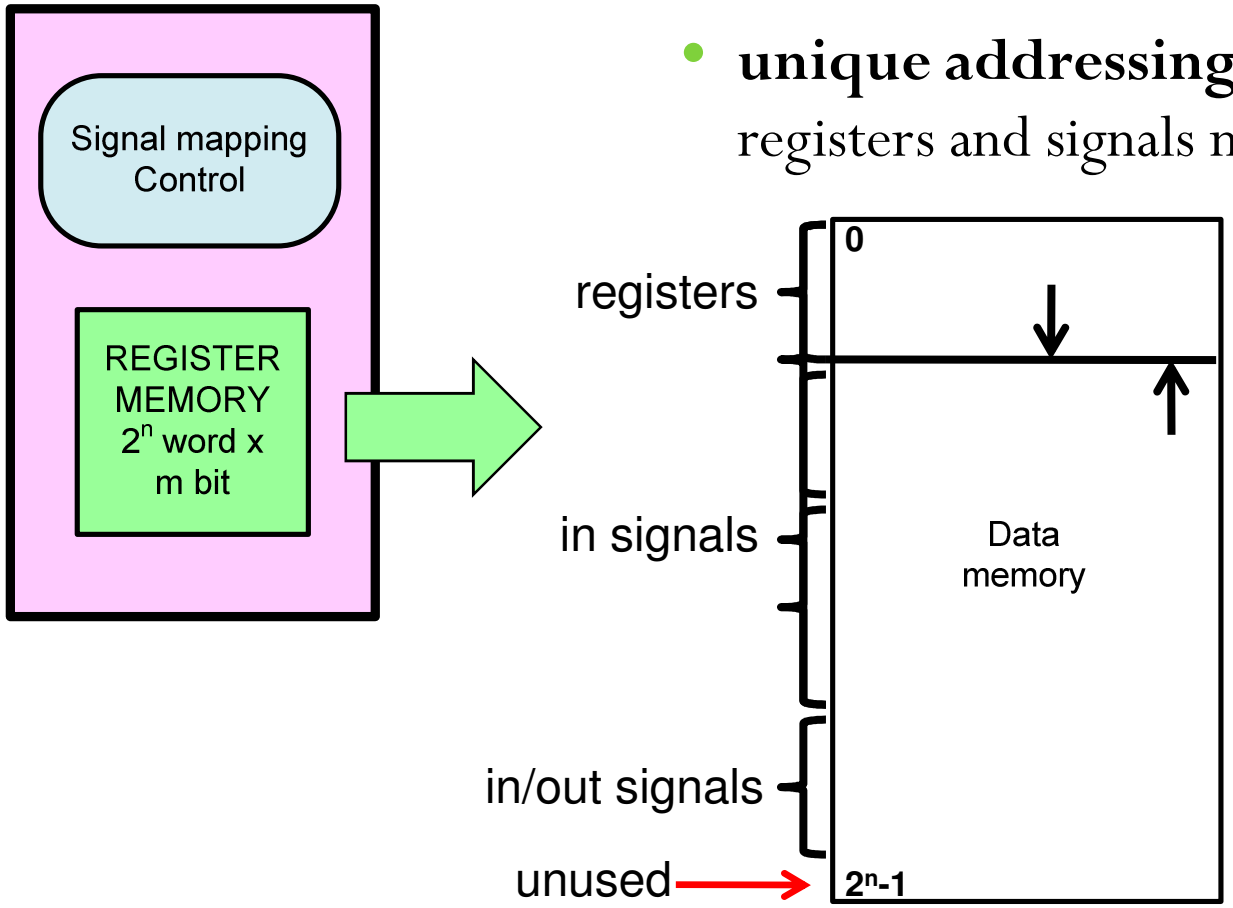
- code is parametric

Register Unit



- code is parametric

Register Memory addressing space

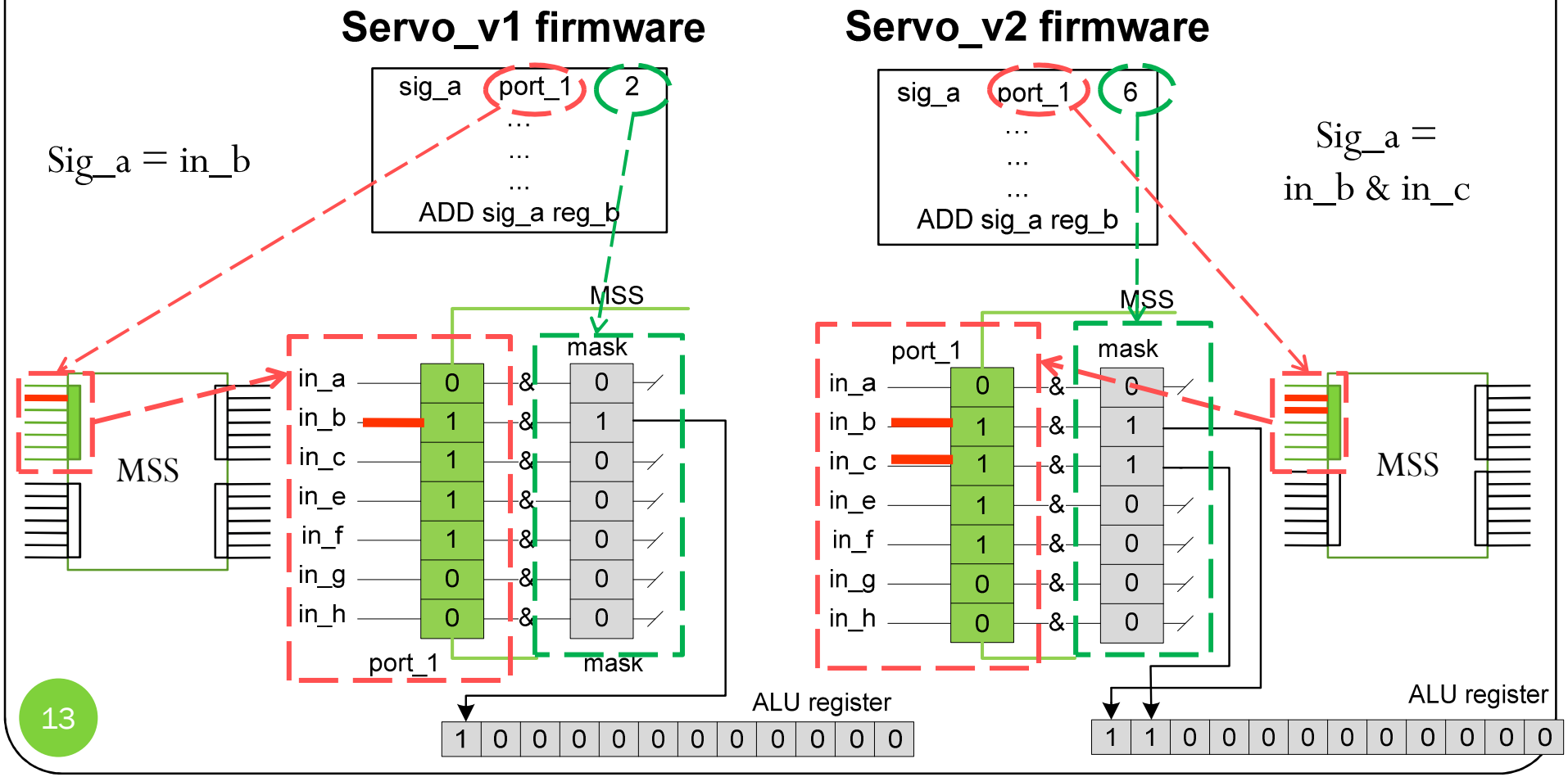


- **unique addressing space** for internal registers and signals mapping data

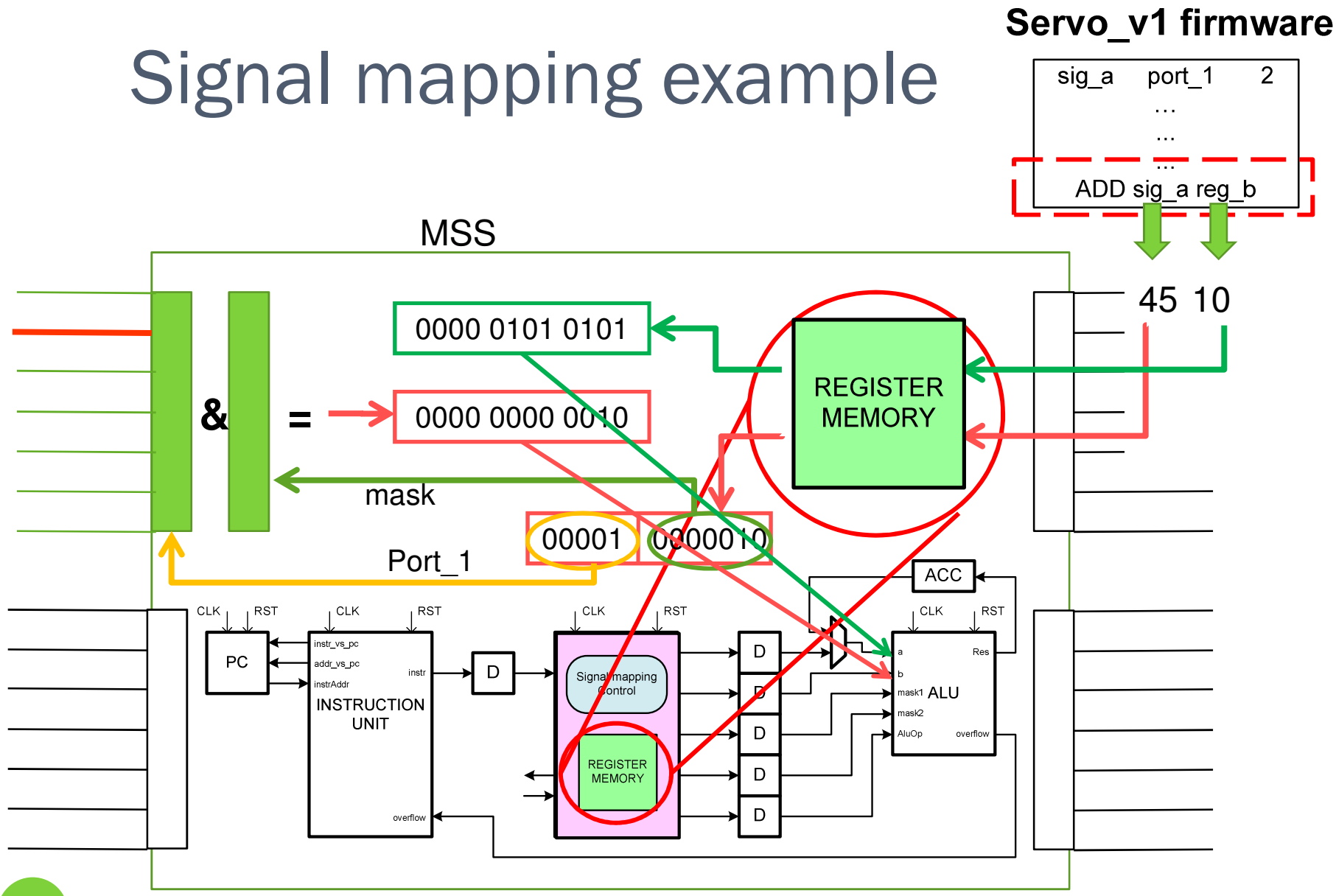
Signal mapping

correspondence between each signal and a firmware label:

- port address, 5 bit wide, and mask information, 7 bit wide

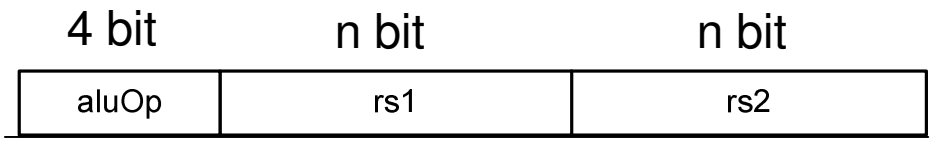


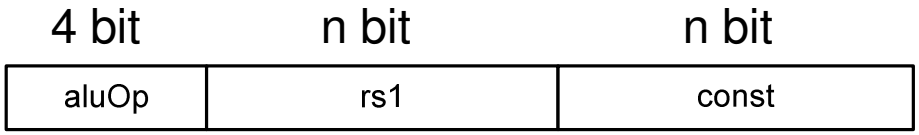
Signal mapping example

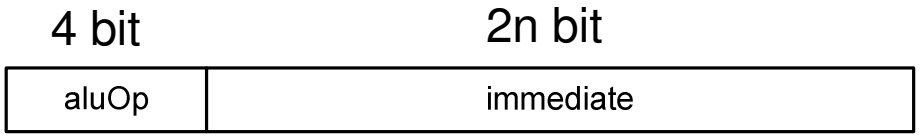


Instruction set

4+2n bit instruction length







	aluOp	Type	Operazion
R	00 00	AND	*rd =*rs1&*rs2
	00 01	OR	*rd =*rs1 | *rs2
	00 11	ADD (signed)	if *rd =*rs1+*rs2 overflow => ovf = 2
	01 11	CONCAT	*rs1 & *rs2
	11 10	COUNTER	generic counter
	01 00	SET	*rd = segnale
	11 11	NOT	Not (rs1)
I	10 00	COMPI	If *rs1 = const then ovf = 2
	01 01	WAIT	ctrl SG, HALT
	11 00	COMPNI	If *rs1 != const then ovf = 2
	11 01	SLLR	*rs1<<imm
	00 10	STORE	*rd = acc
	10 01	SET_V	*rd = const
J	10 10	JUMP	branch unconditioned
	10 11	NOP	No op
	01 10	JUMPCS	branch conditioned

Customized instructions

- WAIT
  - ASIP idle until the signal *sig1* assumes *value* value

```
WAIT        sig1    value
```

  - configuration for the activation of the Servo Gate control

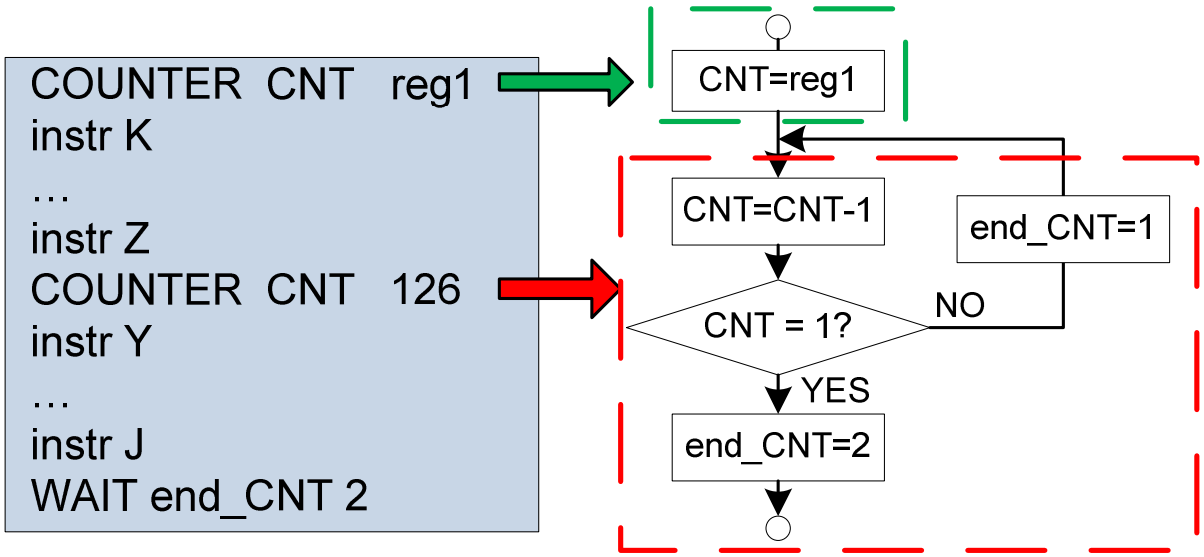
```
WAIT          SG          0
JUMPCS    failure_state
```

- SLLR
  - SLL and SLR substituted with a single instructions
  - Shifts right if the value is negative and left if the value is positive

```
SLLR        sig2     -5
```

Customized instructions

- COUNTER
  - COUNTER instruction manages the hardware programmable counter *CNT*



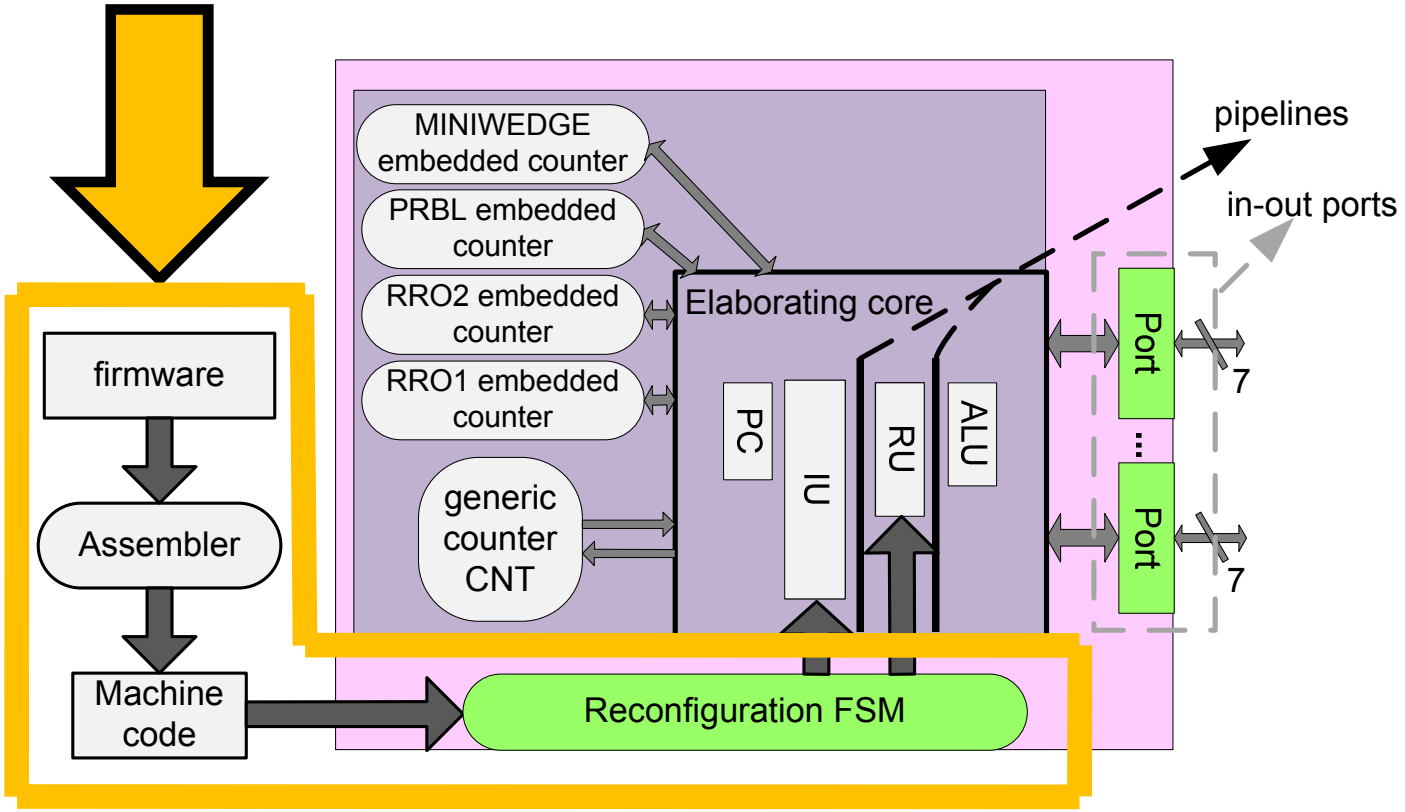
Customized instructions

- JUMPCS
  - branch realized with the sequence of 2 instructions: the first provides the overflow and the second is the JUMPCS

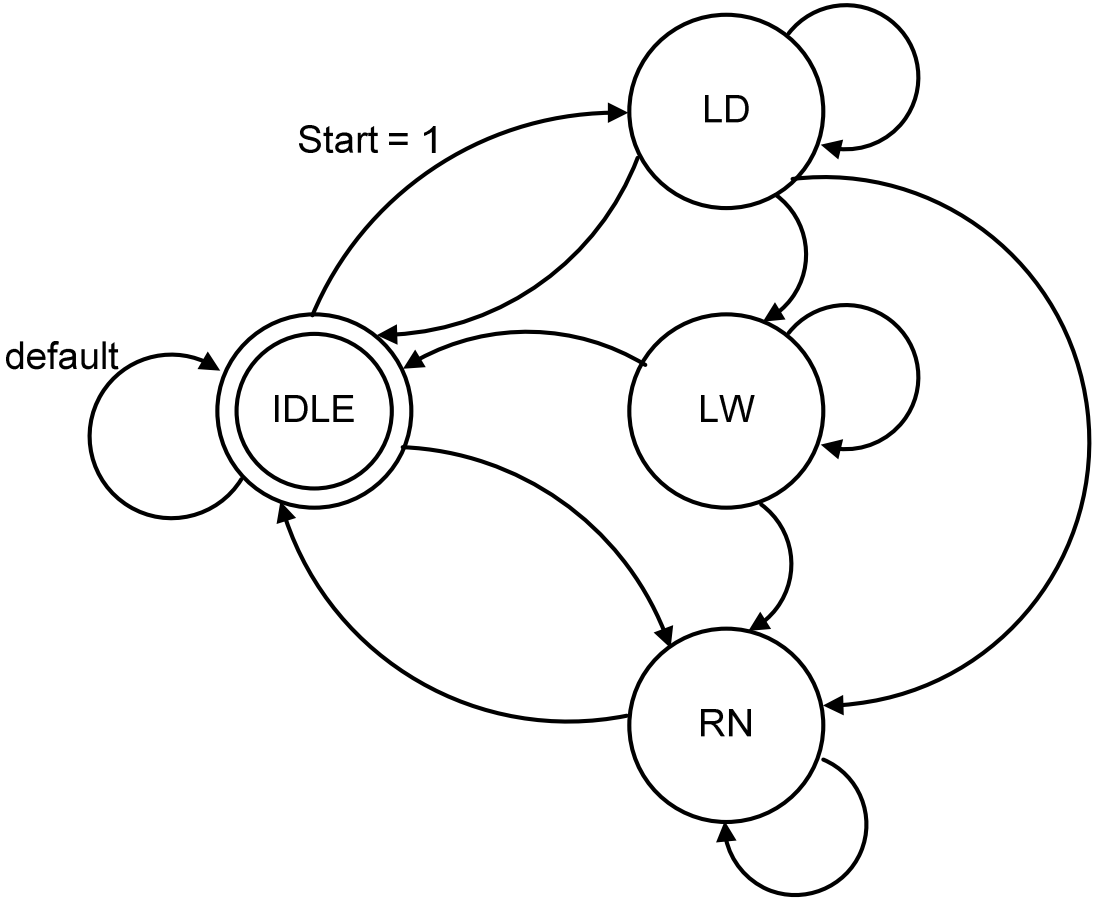
```
COMPI      sig1          0
JUMPCS     failure_state
```

  - prefetch technique to reduce execution time from 3 to 2 clock cycles:
    - branch destination instruction and the next instruction are read before that the overflow information is ready
    - when the overflow is ready the correct instruction is executed

Reconfiguration process

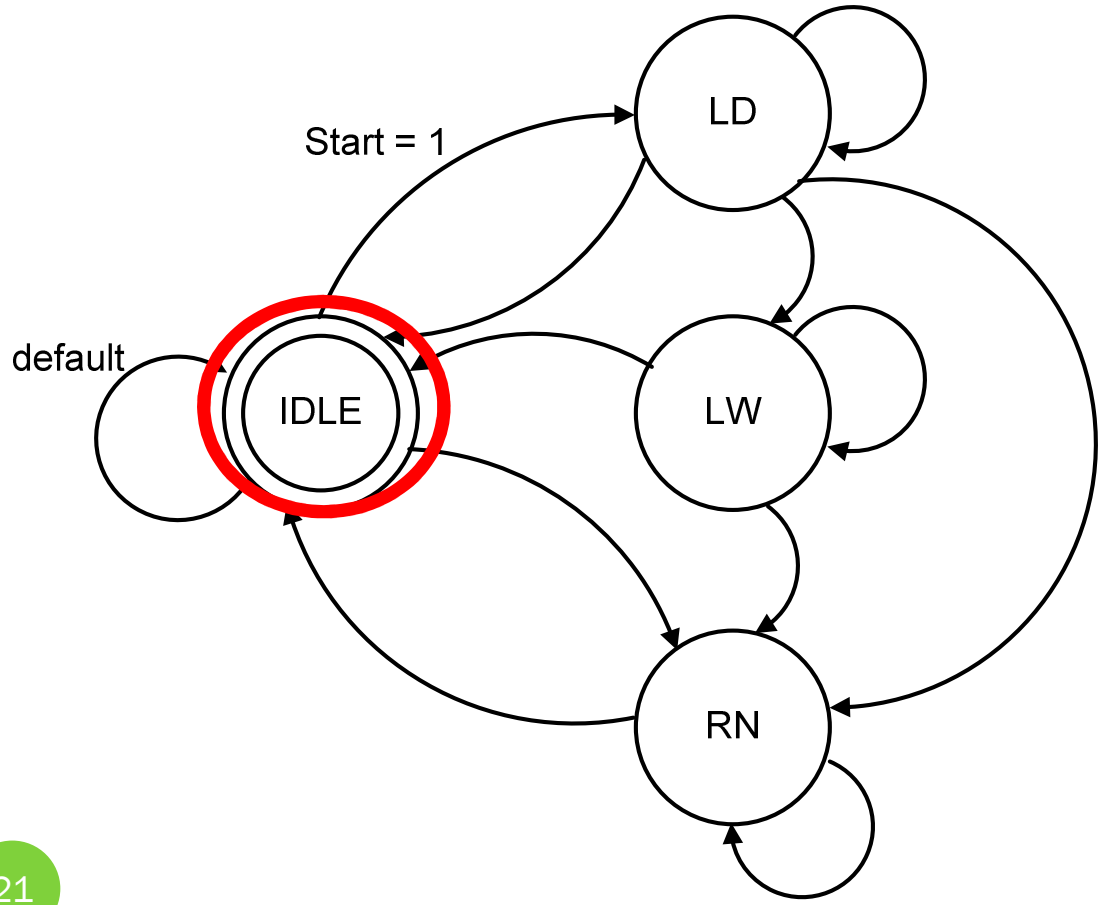


Reconfiguration FSM



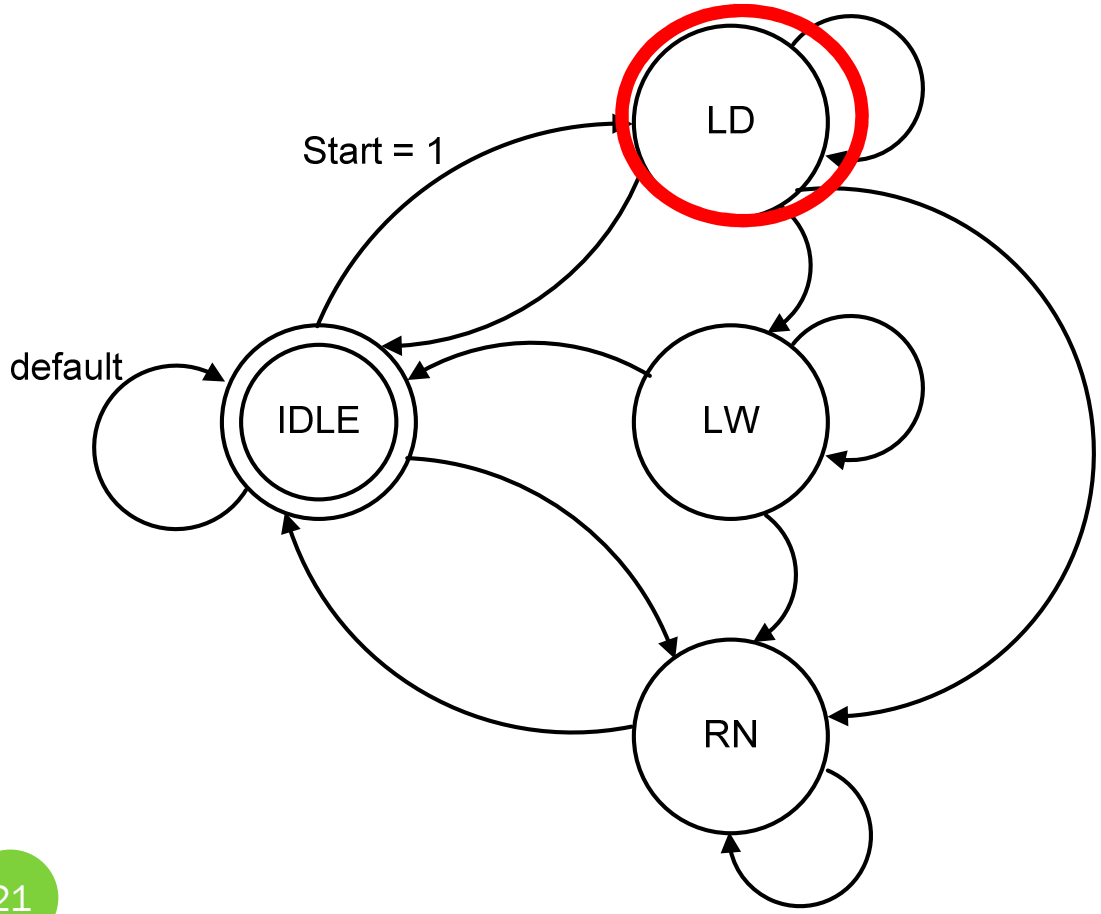
IDLE

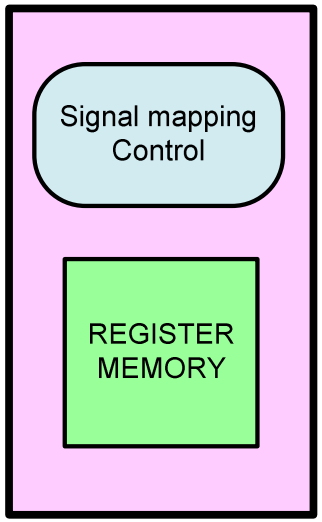
- Startup



Register Memory initialization

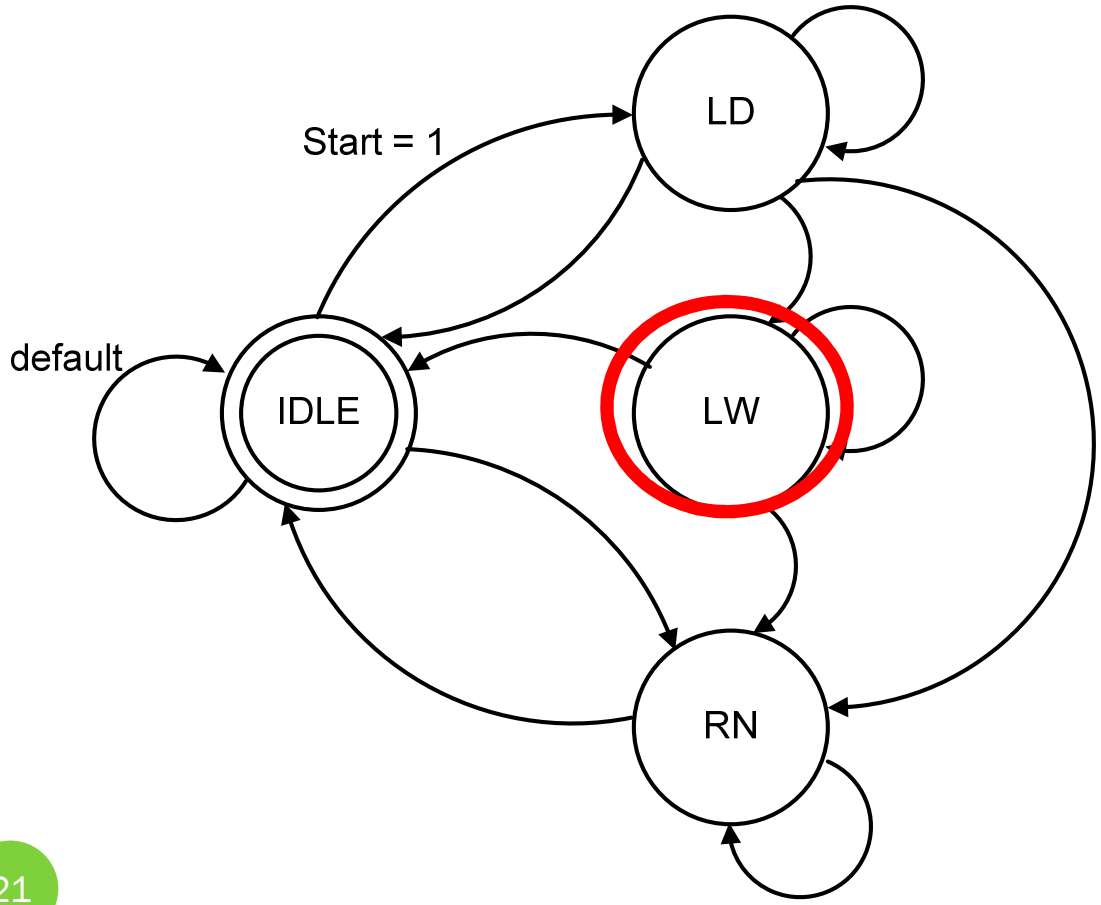
- Registers values and signal mapping information

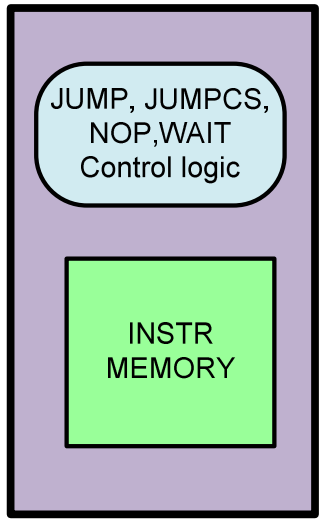




Instruction Memory initialization

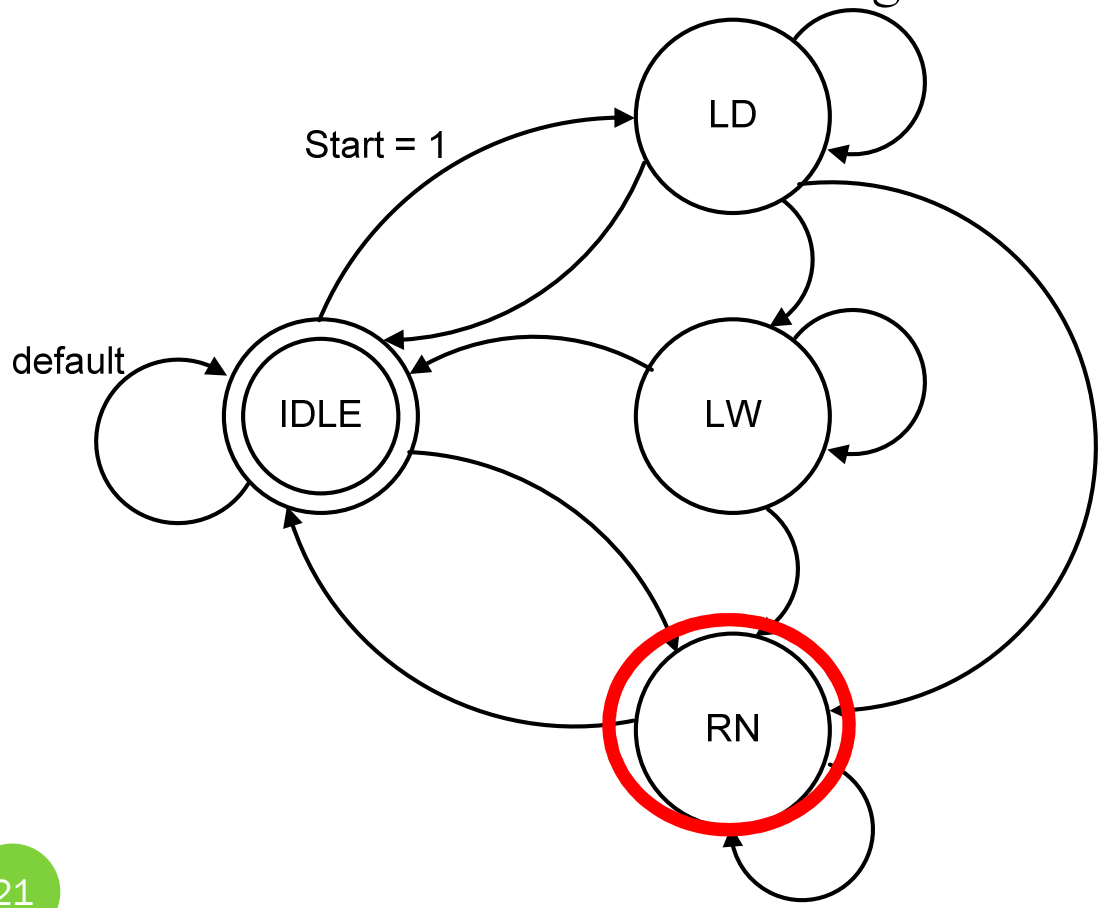
- Firmware instruction



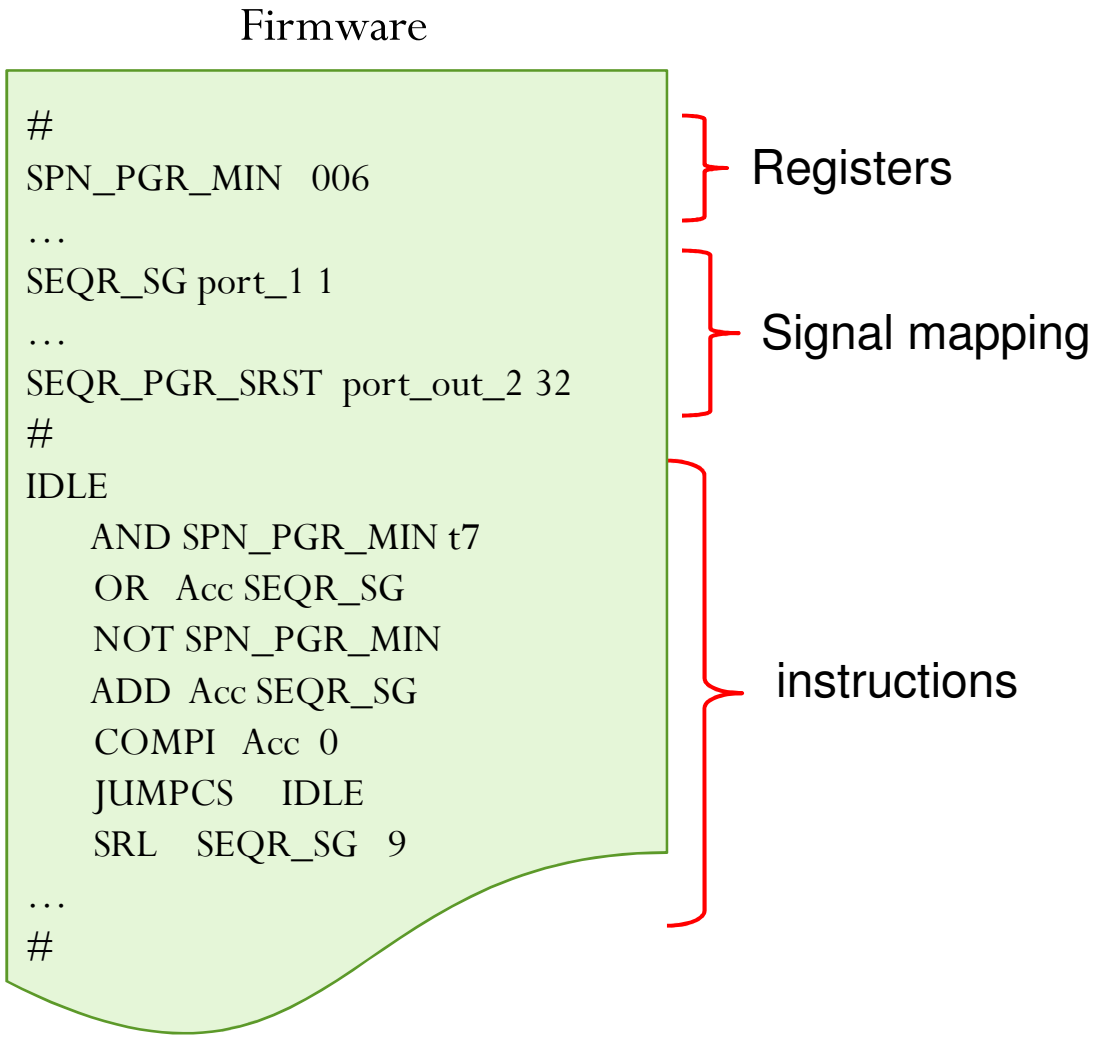


Execution (RN)

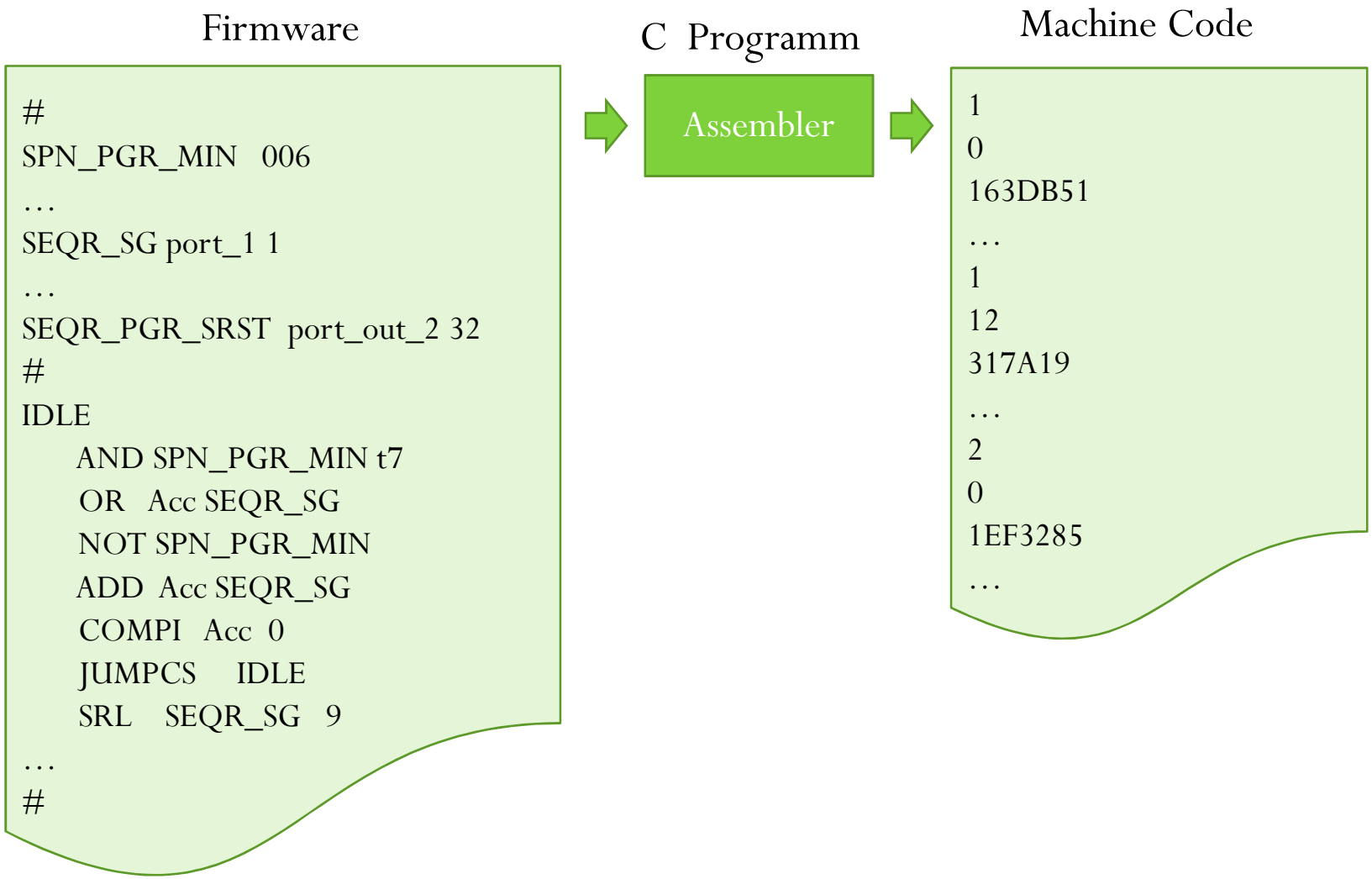
- No initialization
- Firmware execution during Servo data reading



Firmware


Firmware
#
SPN_PGR_MIN 006
…
SEQR_SG port_1 1
…
SEQR_PGR_SRST port_out_2 32
#
IDLE
AND SPN_PGR_MIN t7
OR Acc SEQR_SG
NOT SPN_PGR_MIN
ADD Acc SEQR_SG
COMPI Acc 0
JUMPCS IDLE
SRL SEQR_SG 9
…
#
Registers
Signal mapping
instructions

Assembler


Firmware
C Programm
Machine Code
#
SPN_PGR_MIN 006
…
SEQR_SG port_1 1
…
SEQR_PGR_SRST port_out_2 32
#
IDLE
AND SPN_PGR_MIN t7
OR Acc SEQR_SG
NOT SPN_PGR_MIN
ADD Acc SEQR_SG
COMPI Acc 0
JUMPCS IDLE
SRL SEQR_SG 9
…
#
Assembler
1
0
163DB51
…
1
12
317A19
…
2
0
1EF3285
…

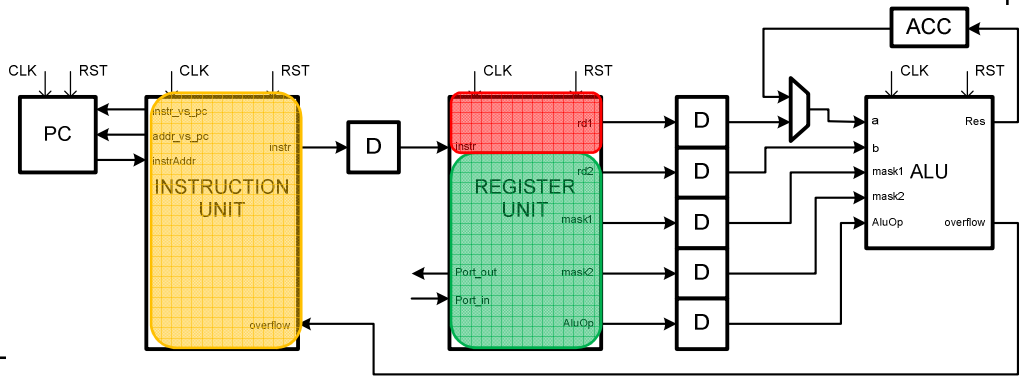
Assembler

Firmware

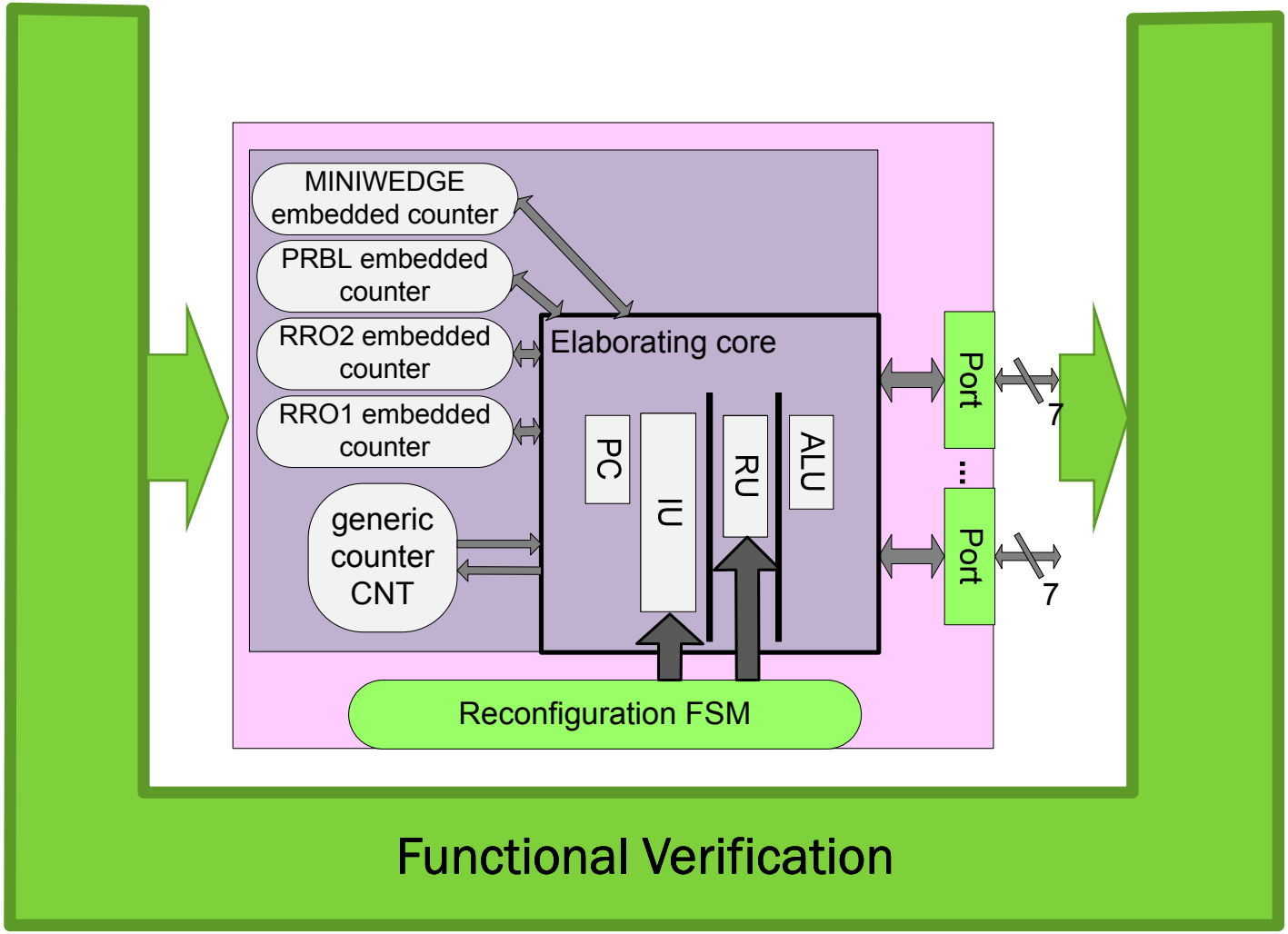
```
#
SPN_PGR_MIN   006
…
SEQR_SG port_1 1
…
SEQR_PGR_SRST  port_out_2 32
#
IDLE
      AND SPN_PGR_MIN t7
      OR   Acc SEQR_SG
      NOT SPN_PGR_MIN
      ADD  Acc SEQR_SG
      COMPI   Acc  0
      JUMPCS     IDLE
      SRL    SEQR_SG   9
…
#
```

Machine Code

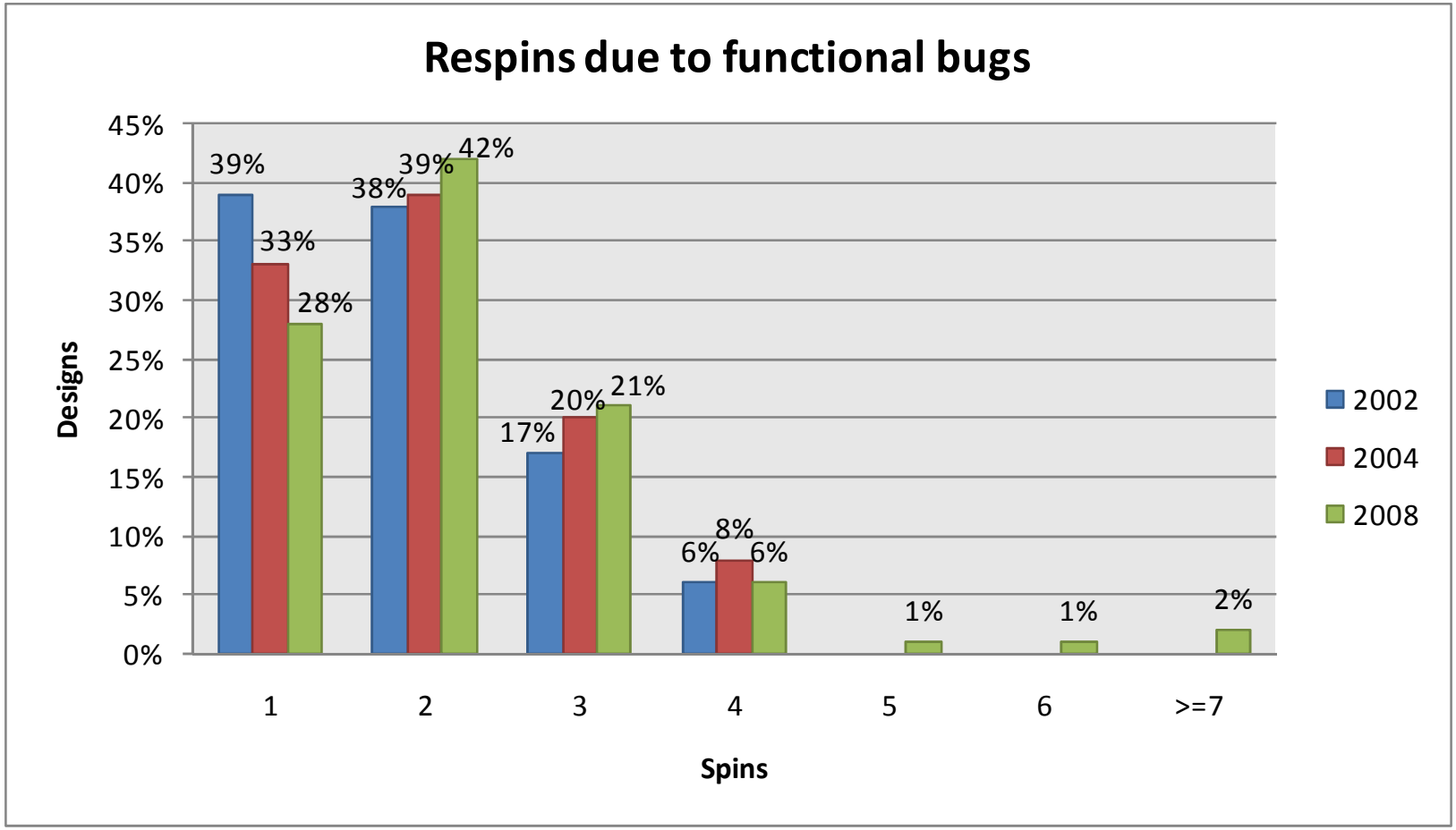
```
1
0
163DB51
…
1
12
317A19
…
2
0
1EF3285
…
```



Functional verification

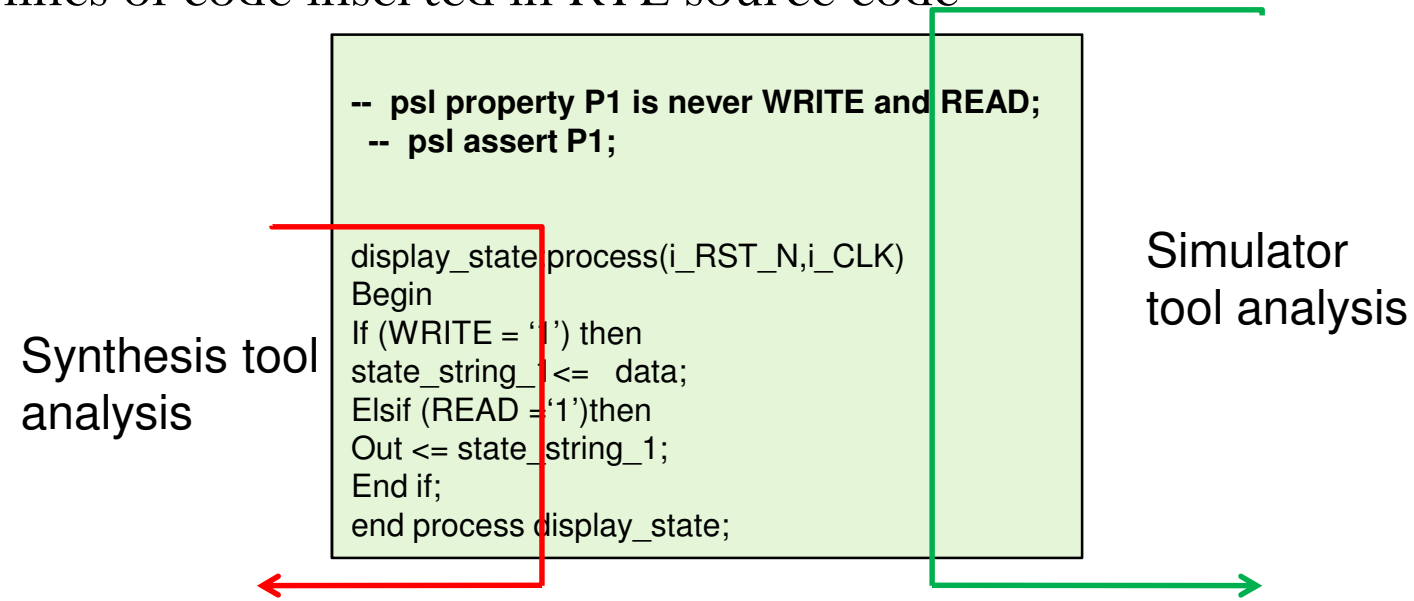


Why verification is so important?



Assertion Based Verification

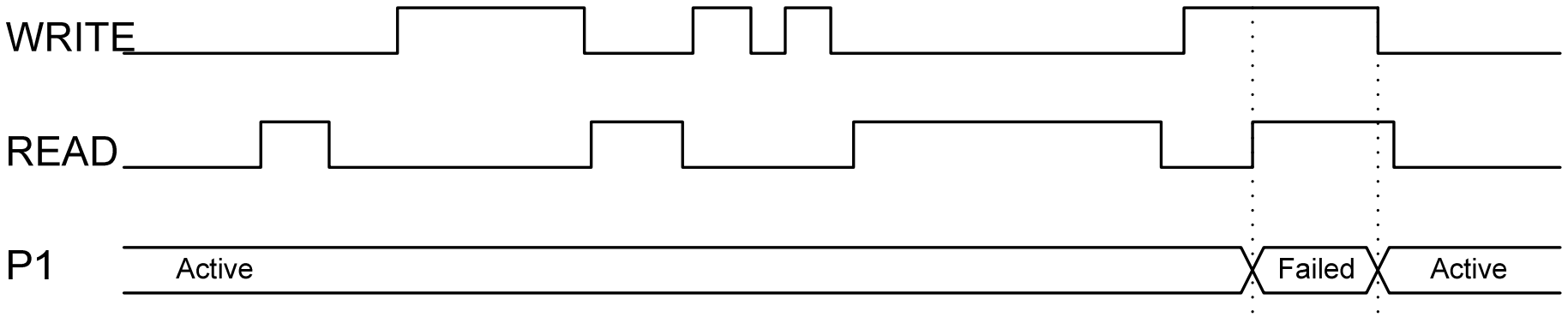
- lines of code inserted in RTL source code



- Assertions describe low level and high level design functionality
- Advantages:
  - reusability characteristic
  - design observability increasing

Assertion behavior

```
-- psl property P1 is never WRITE and READ;
-- psl assert P1;
```

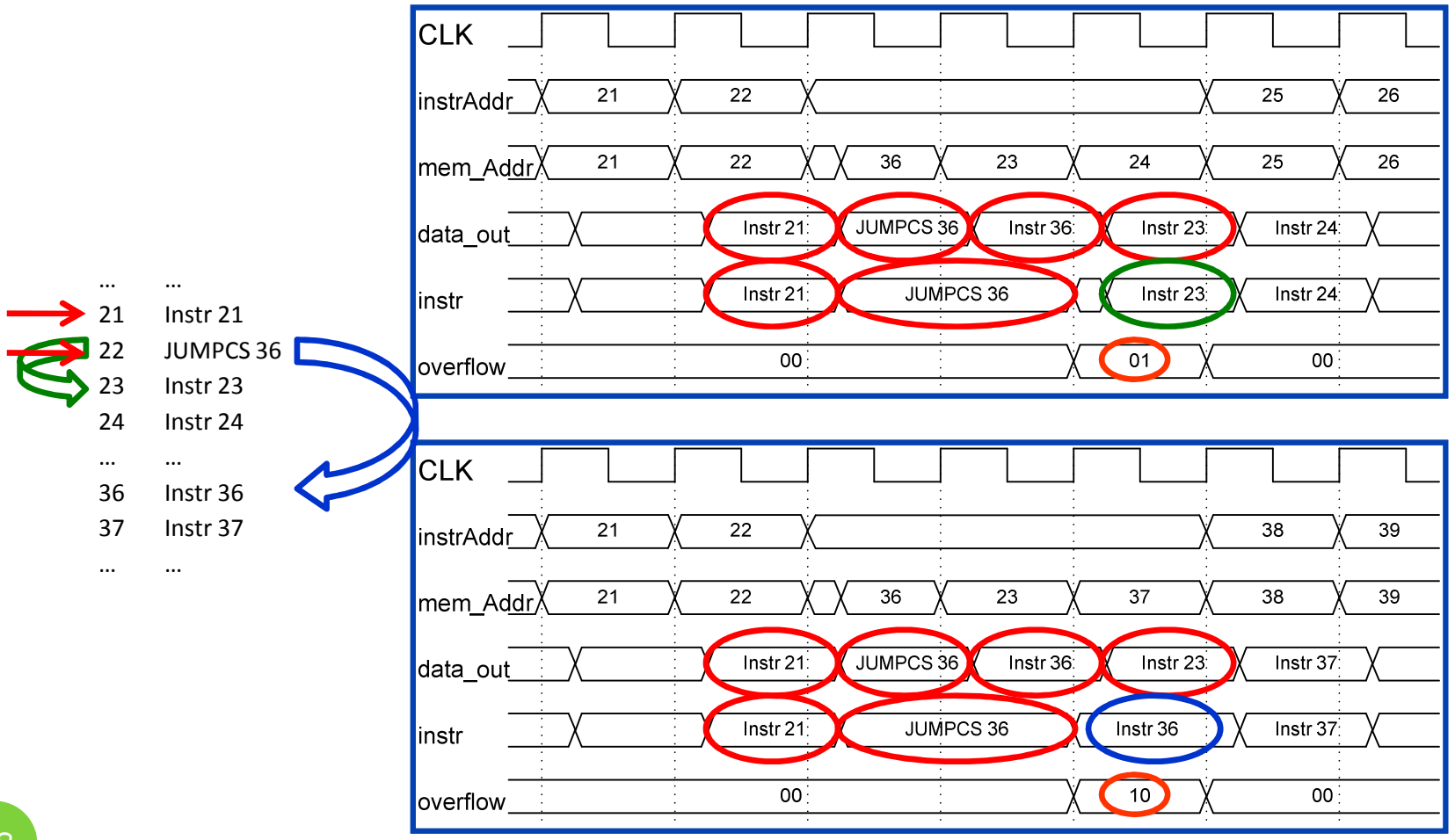


Assertion Based Verification

- Property Specification Language (PSL) - IEEE 1850

- complex MSS functionality verified with PSL assertion:
  - JUMP, branch uncoditioned
  - JUMPCS, branch conditioned
  - NOP, no operation (idle condition)
  - WAIT, idle condition till a signal value change
  - Signal Mapping

JUMPCS verification

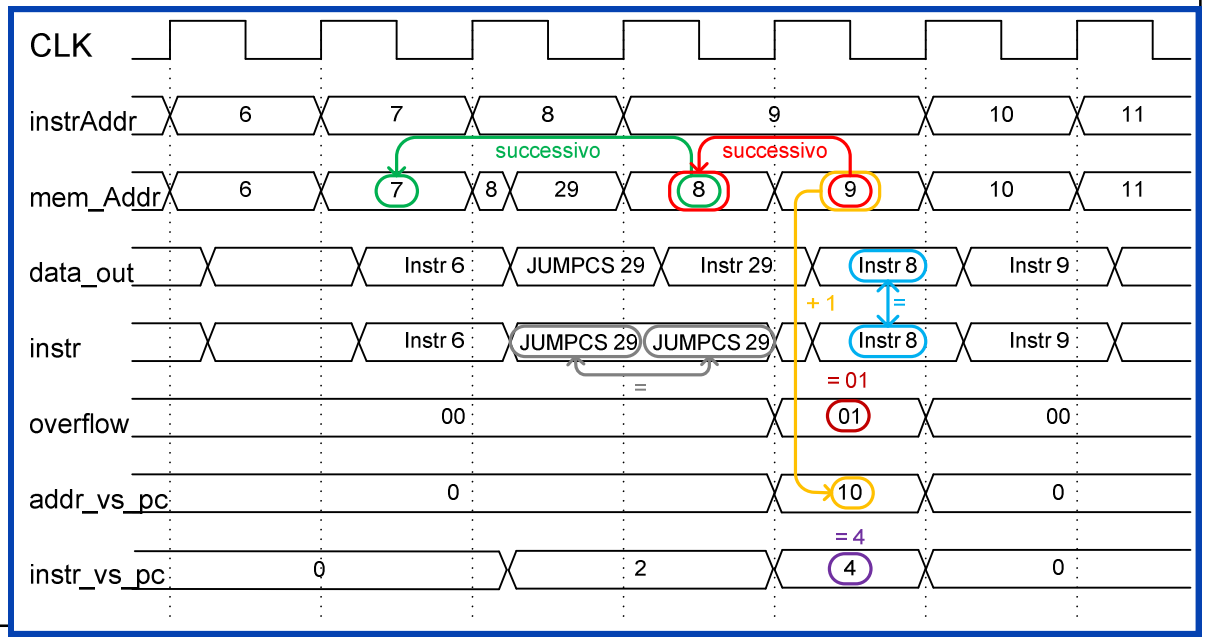


```
-- psl Property p_jumpcs_no is
 -- always s_jumpcs_no |->
  -- (instr= data_out
  --  or (data_out = WAIT_126_0
  --   and instr = NOP_0))
  -- and ((successivo(mem_addr, prev(mem_addr,1), nInstr))
   -- or (((instr(li-1 downto li-lc) = "1010")
    -- and (mem_addr = instr(li-lc-4 downto li-lc-limm_j)))))
  -- and instr_vs_pc = 4
  -- and successivo(prev(mem_addr,1), prev(mem_addr,3), nInstr)
  -- and ((addr_vs_pc = mem_addr+1)
   -- or (addr_vs_pc = mem_addr
    -- and instr(li-1 downto li-lc) = "1011"
    -- and not(instr(li-lc-4 downto li-lc-limm_j)=2)
    -- and not (instr(li-lc-4 downto li-lc-limm_j)=1)));
```

```
-- psl Sequence s_jumpcs_no is
 -- {instr(li-1 downto li-lc)="0110"[*2]; overflow
= “01”};
```

```
-- psl Property p_jumpcs is
 -- always {not(instr = NOP0);
 -- rose(instr(li-1 downto li-lc)="0110")} |=>
-- {stable(instr);
  -- (overflow = “01”) or (overflow = “10”)};
-- psl Assert p_jumpcs;
```

...	...
6	Instr 6
7	JUMPCS 29
8	Instr 8
9	Instr 9
...	...
29	Instr29
...	...



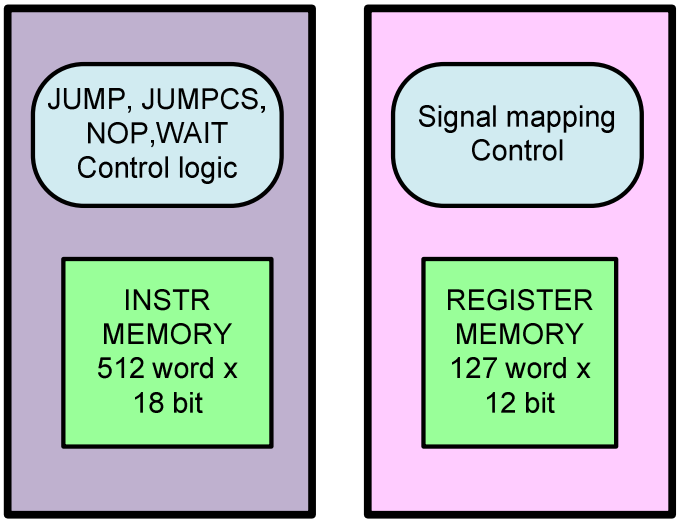
Assertion Based Verification

- Different firmware used to point out functional bugs

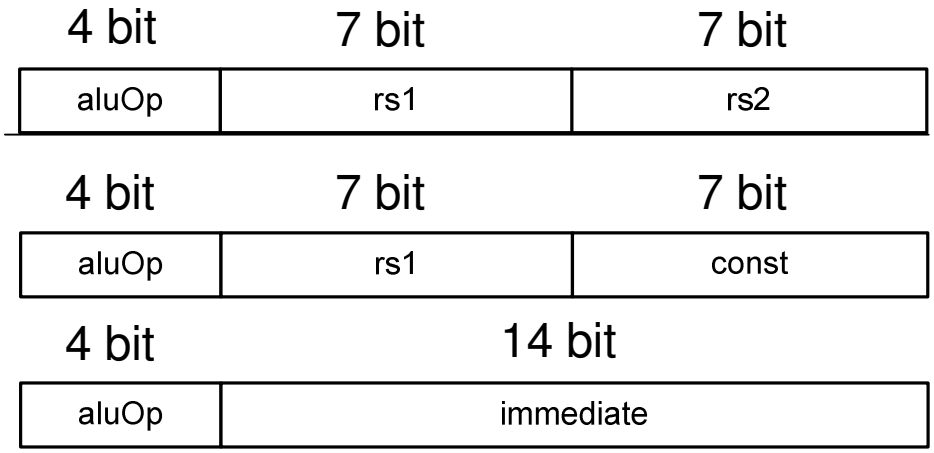
STORE	reg0,1	STORE	reg0,1	STORE	reg0,1
COMPI	reg0,1	COMPI	reg0,1	COMPI	reg0,1
JUMPCS	A	JUMPCS	B	JUMPCS	C
NOT	reg1	NOT	reg1	NOT	reg1
A:		B:		C:	
NOP	1	NOP	2	NOP	1023
AND	reg0,reg1	AND	reg0,reg1	AND	reg0,reg1
STORE	reg0,1	STORE	reg0,1	STORE	reg0,1
COMPI	reg0,0	COMPI	reg0,0	COMPI	reg0,0
JUMPCS	C	JUMPCS	C	JUMPCS	C
NOP	1	NOP	2	NOP	1023
D:		E:		F:	
AND	reg0,reg1	AND	reg0,reg1	AND	Reg0, reg1

Servo Sequencer case study

- Servo Sequencer is a control logic of RWC servo subsystem:
  - Elaborating frequency: 300 MHz
  - CMOS Technology 65 nm
  - Industrial product
- Microprogrammed Servo Sequencer



•20 7 bit I/O ports



Synthesis results

Device	Library	clock [ns]	freq [MHz]	Area [µm2]	Power [mW]	Leakage [µW]
MSS (*)	PVT (process SLOW, 1.1V, 125C) LVT,HVT,SVT	3.33	300	36,7 x 10^3	3.24	1234
RSS		3.33	300	4.2 x 10^3	1.4	232

Global Cell Area		Local Cell Area					
Total [μm^2]	%	Combinational [μm^2]	%	Non combinational [μm^2]	%	Black boxes [μm^2] (*)	%
36770	100	9390	~25	4326	~12	23055	~63

- CMOS Technology 65 nm

(*) Instruction Memory -> single port RAM in 1.1 V, 65 nm CMOS technology with standard voltage threshold

(*) Register Unit -> dual port RAM in 1.1 V, 65 nm CMOS technology with standard voltage threshold

Publications

- Baldrighi, P.; Maggi, M. M.; Castellano, M.; Vacchi, C.; Crespi, D.; Bonifacino, P.; "Implementation of Microprogrammed Hard Disk Drive Servo Sequencer," Proceedings of DSD 2008 (11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN) – 3-5 September 2008
- Baldrighi, P.; Castellano, M.; Vacchi, C.; Canina, D.; Golzi, P.; Ferrante, G.; "Digital Nuclear Magnetic Resonance acquisition channel," Proceedings of DSD 2008 (11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN) – 3-5 September 2008
- Castellano, M.; Baldrighi, P.; Vacchi, C.; Natuzzi, M.; Furlan A.,; "Custom Formal Verification Technique For Partial Product Reduction Tree", Proceedings of ICM 2008 (20th IEEE International Conference on Microelectronics) – 14-17 December 2008
- Castellano, M.; Baldrighi, P.; Vacchi, C.; "Pipelined Inner Product Function Generator", Proceedings of ESSDERC-ESSCIRC 2008-poster Session – 15-19 September 2008
- Castellano, M.; Baldrighi, P.; Vacchi, C.; Natuzzi, M.; "Algorithm and Architecture for High Speed Merged Arithmetic FIR Filter Generation", Proceedings of ISCCSP 2008 (3rd International Symposium on Communications, Control and Signal Processing) ;12-14 March 2008;
- Baldrighi, P.; Vacchi, C.; Reconfigurable I/O ports Application Specific Processor: Servo Sequencer case study **submitted** to IEEE Magnetic Letters