MO-ALU

Informatica Industriale - 20 Luglio 2023

GARGARO DAVID | PAPA EMANUELE

Introduzione

Schema Circuitale

Componenti

FSMs

Risultati simulazioni

VHDL

Introduzione

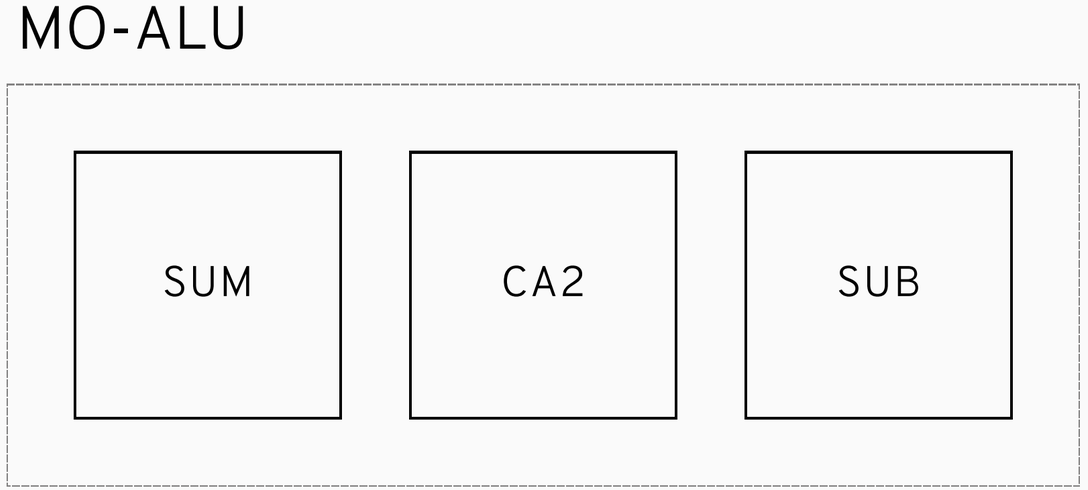
Il sistema Multi-Operation ALU (**MO-ALU**) esegue le seguenti operazioni su due numeri A e B di **Nb** bits:

- SUM : Somma;
- CA2 : Complemento a 2;
- SUB : Sottrazione.

Il sistema opera attraverso una Finite-State-Machine (CU) regolata da un singolo segnale.

Il sistema MO-ALU è dotato di:

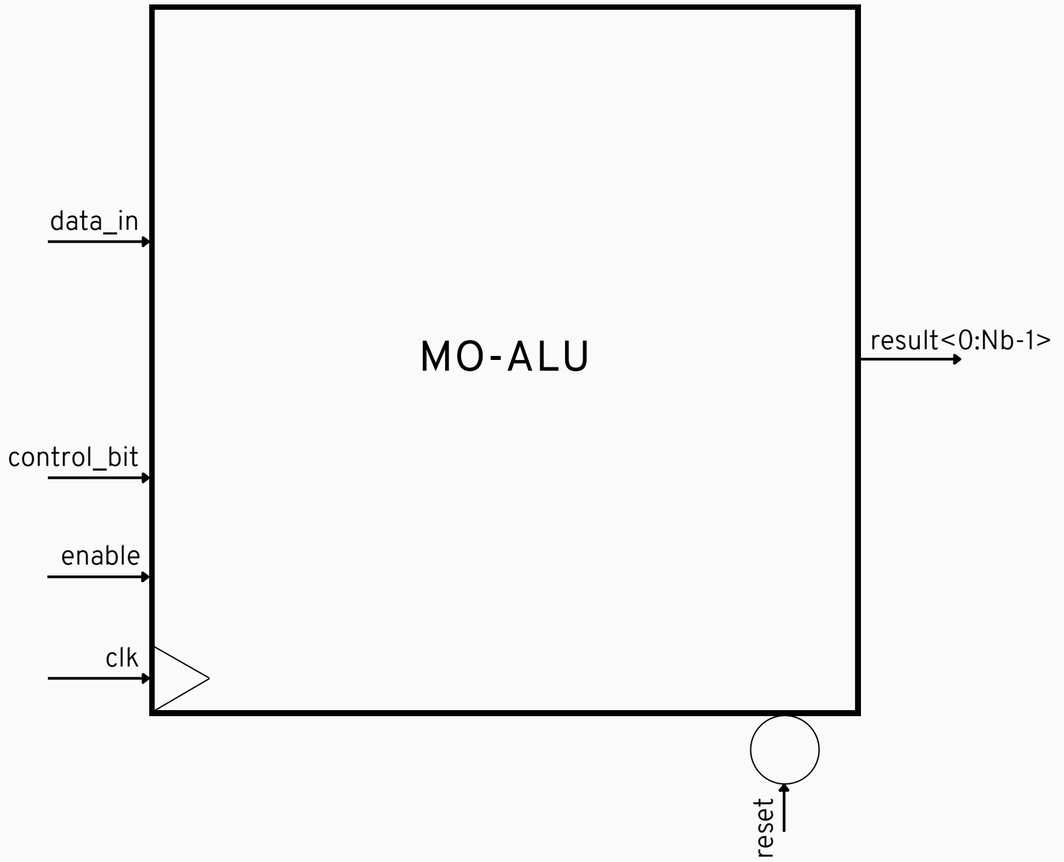
- Reset : Asincrono e attivo basso;
- Enable : Sincrono e attivo alto.



Introduzione

Scelte implementative:

- I numeri passati in ingresso vengono considerati in notazione CA2. Questo implica che, con Nb bits, si possono rappresentare tutti i numeri compresi in $[-2^{Nb-1}, 2^{Nb-1}-1]$;
- È possibile svolgere più operazioni di seguito su gli stessi numeri memorizzati;
- Il sistema è dotato di un controllo per determinare se un'operazione possa causare un Overflow o un Underflow. In queste situazioni, il sistema restituirà 0 come risultato;
- L'operazione da svolgere viene determinata tramite l'unico segnale in ingresso della CU.



Introduzione

Schema Circuitale

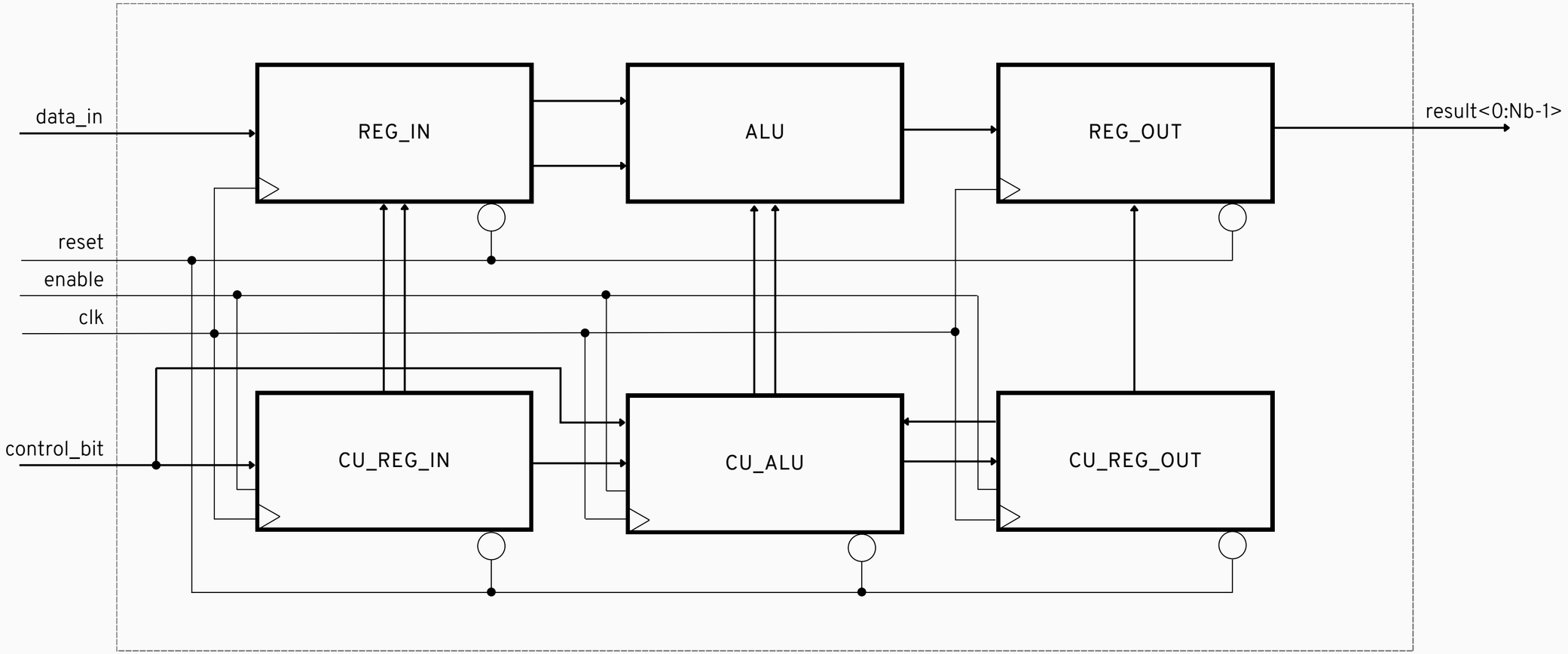
Componenti

FSMs

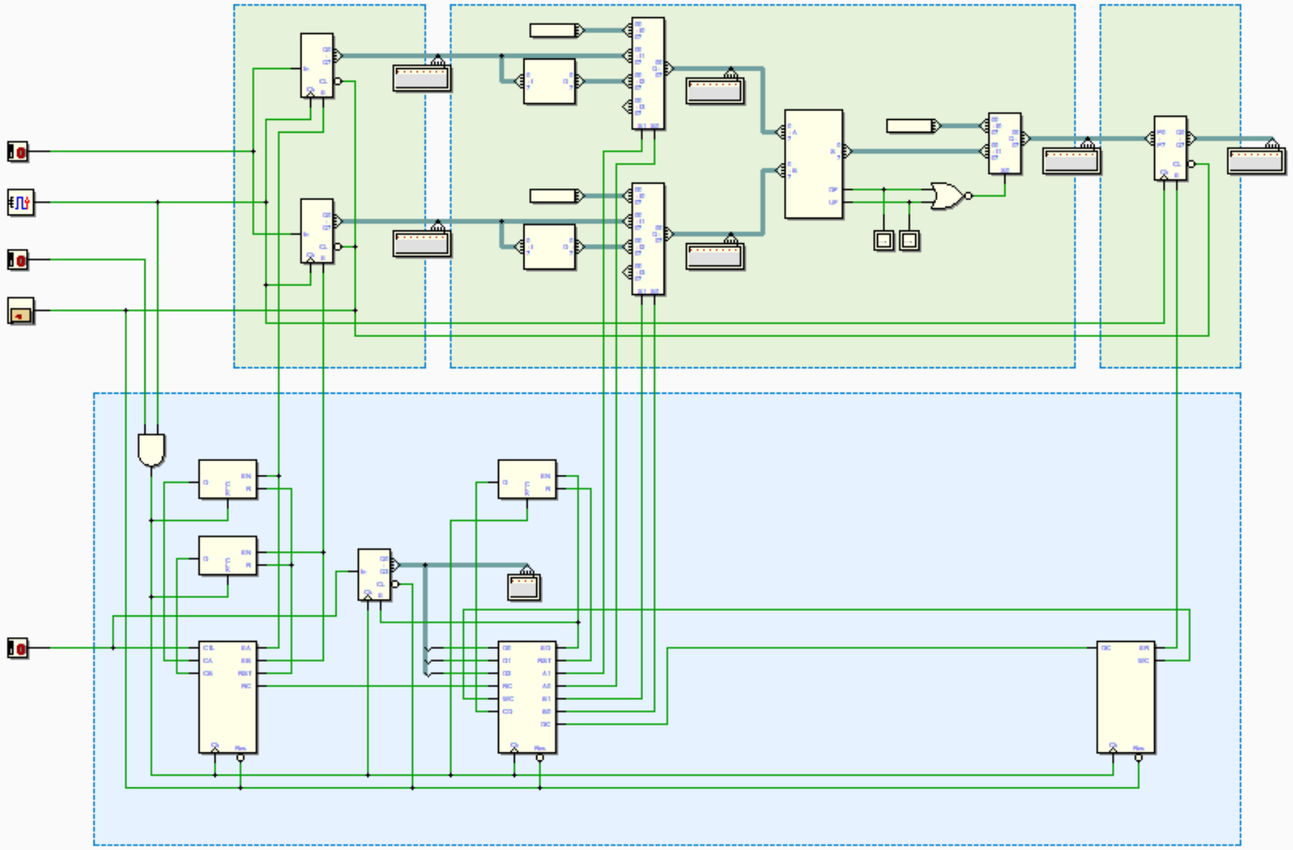
Risultati simulazioni

VHDL

Schema Circuitale



Schema Circuitale



Introduzione

Schema Circuitale

Componenti

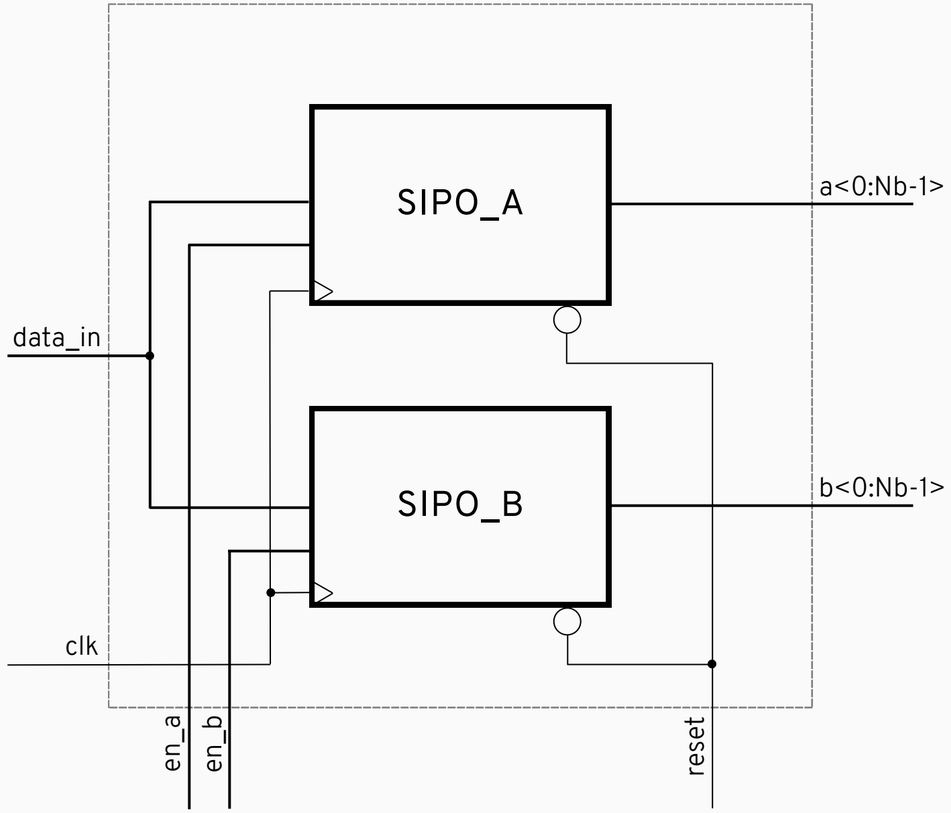
FSMs

Risultati simulazioni

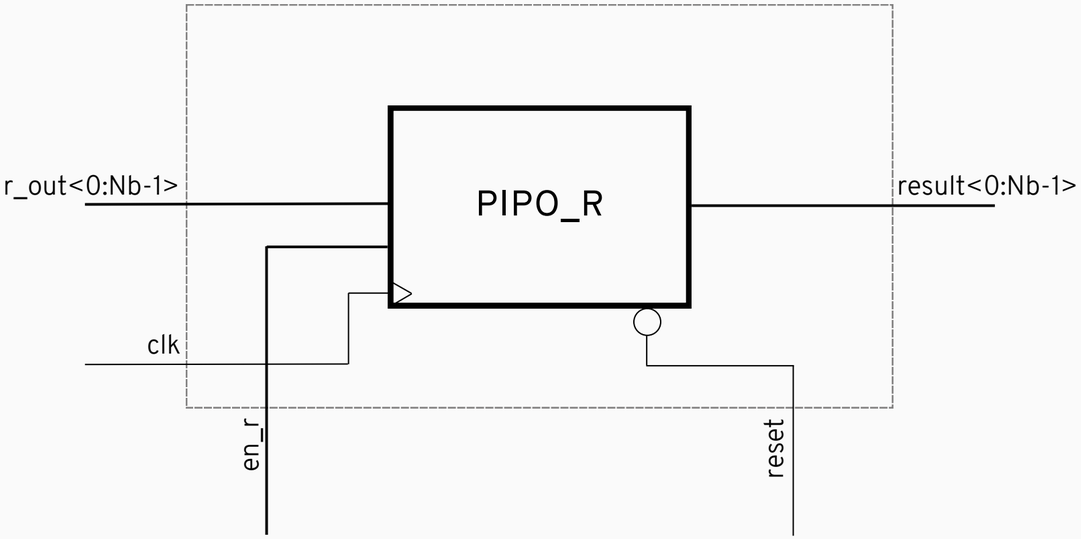
VHDL

Componenti

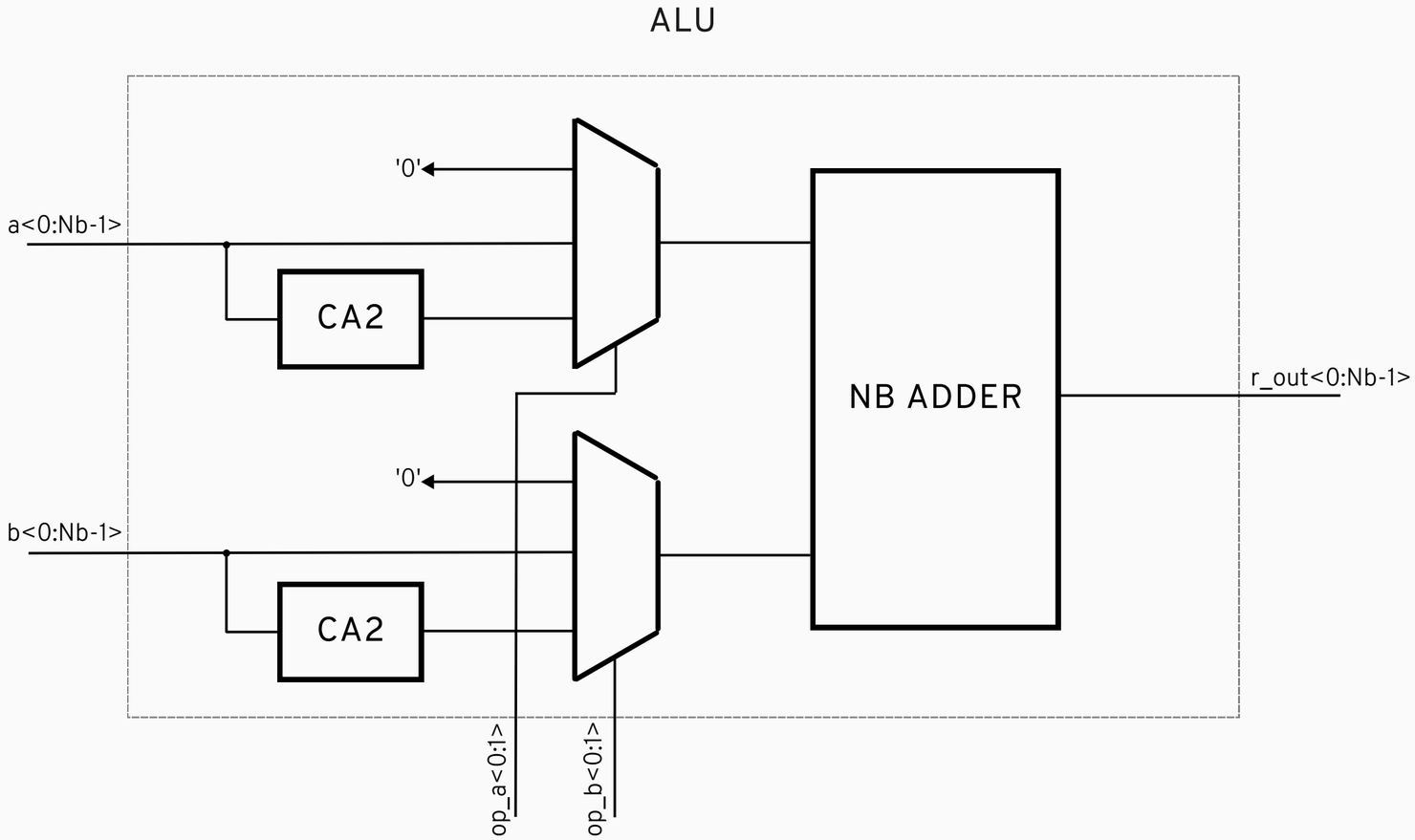
REG_IN



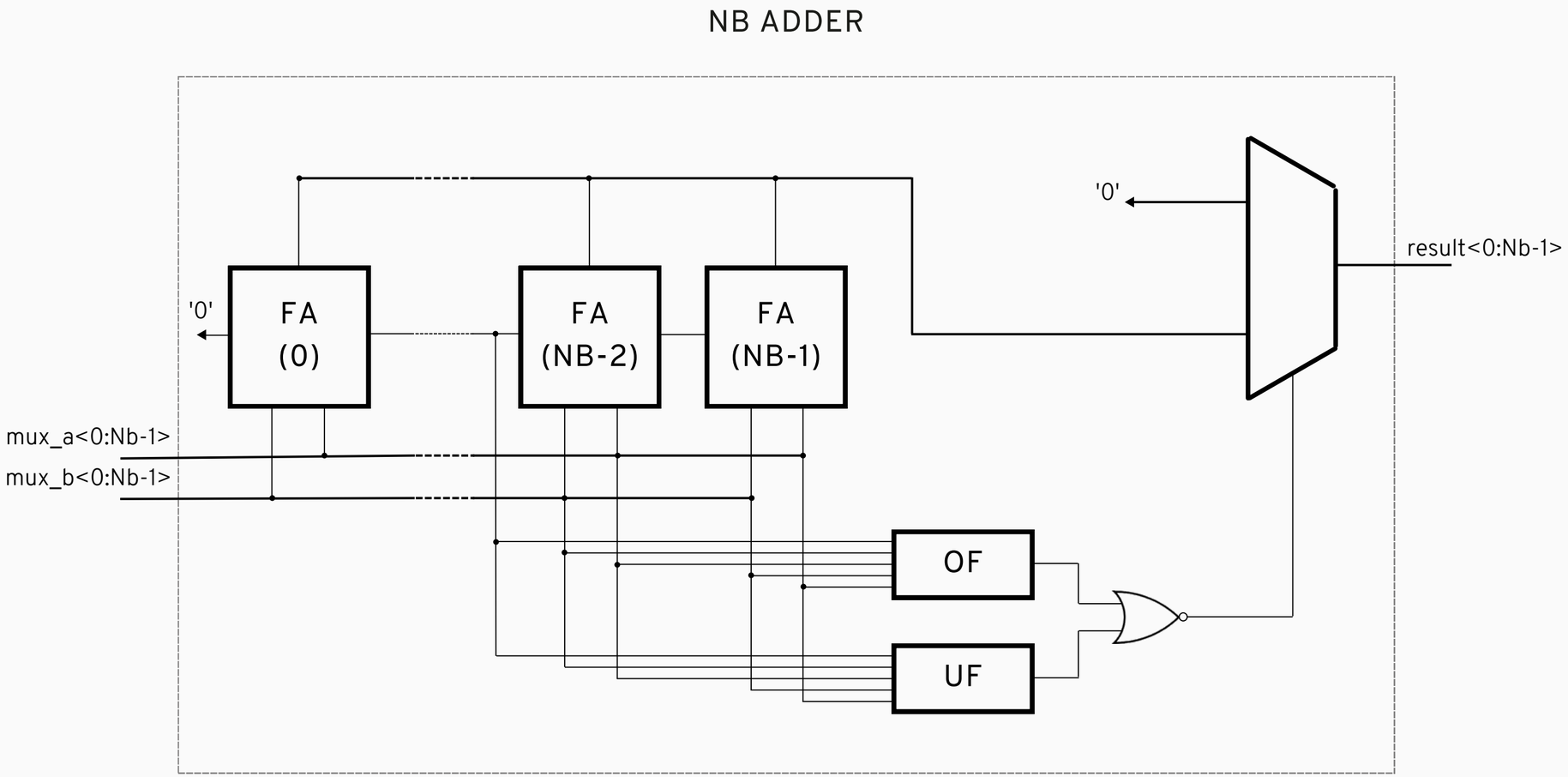
REG_OUT



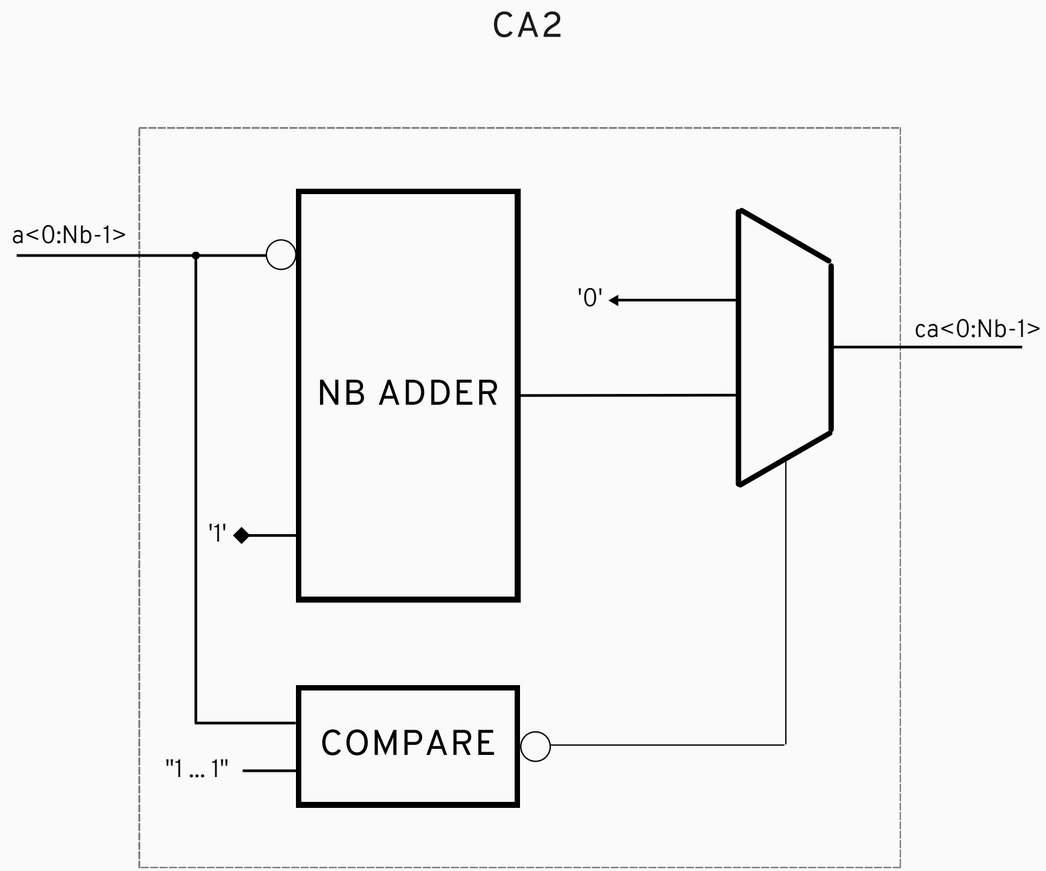
Componenti



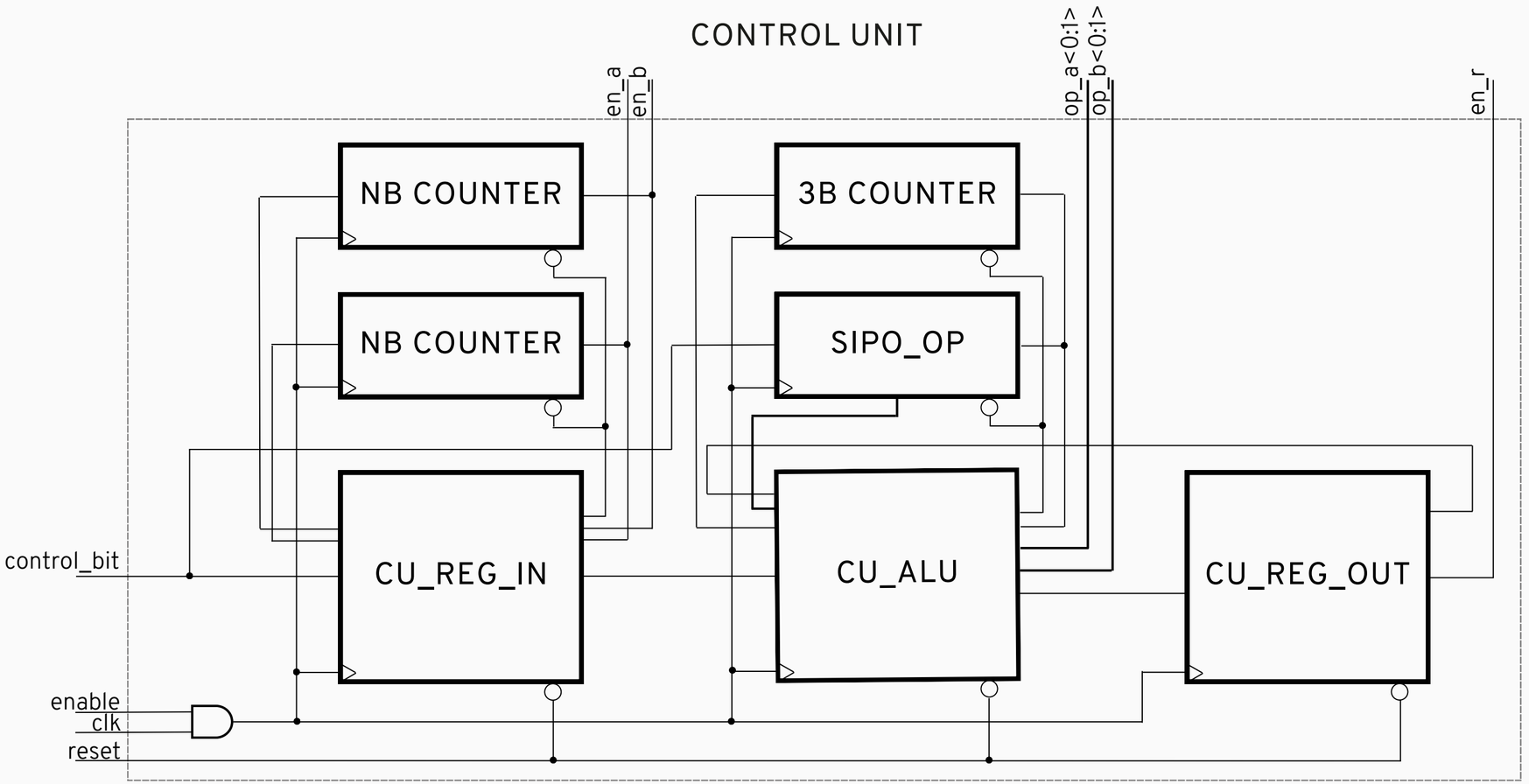
Componenti


NB ADDER
'0'
FA
(0)
FA
(NB-2)
FA
(NB-1)
'0'
result<0:Nb-1>
mux_a<0:Nb-1>
mux_b<0:Nb-1>
OF
UF

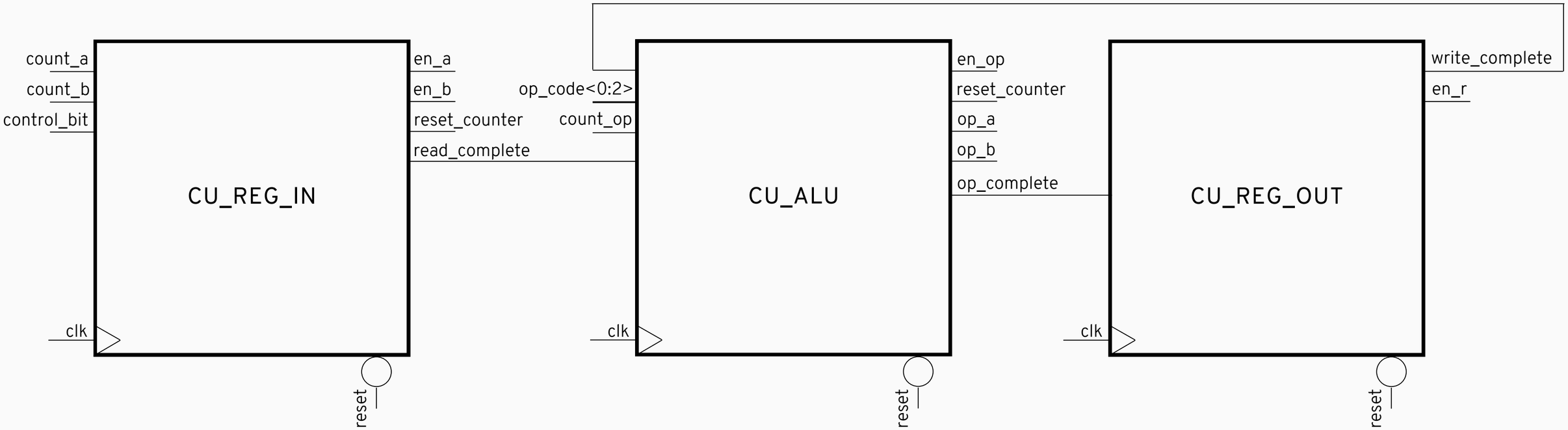
Componenti



Componenti


CONTROL UNIT
en_a
en_b
op_a<0:1>
op_b<0:1>
en_r
NB COUNTER
NB COUNTER
3B COUNTER
SIPO_OP
CU_REG_IN
CU_ALU
CU_REG_OUT
control_bit
enable
clk
reset

Componenti


count_a
count_b
control_bit
clk
reset
CU_REG_IN
en_a
en_b
reset_counter
read_complete
op_code<0:2>
count_op
clk
reset
CU_ALU
en_op
reset_counter
op_a
op_b
op_complete
clk
reset
CU_REG_OUT
write_complete
en_r

Introduzione

Schema Circuitale

Componenti

FSMs

Risultati simulazioni

VHDL

FSMs

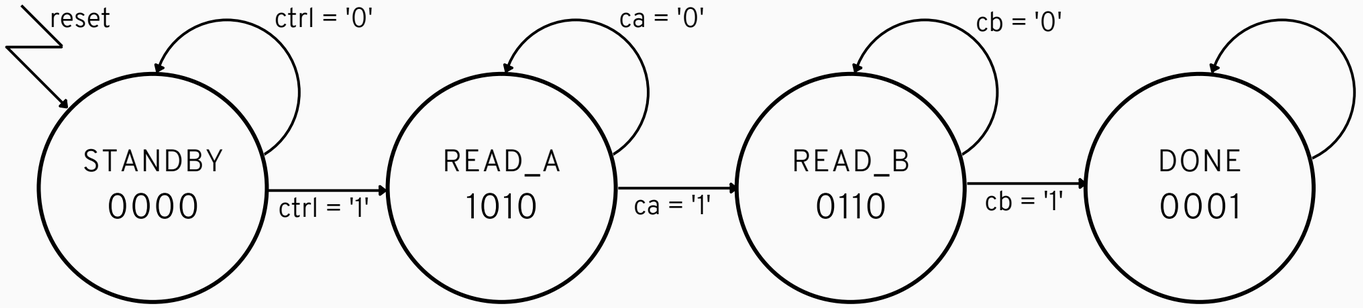
CU_REG_IN gestisce la fase di lettura abilitando i registri dei due numeri da memorizzare.

Input:

- control_bit;
- count_a;
- count_b;

Outputs:

- en_a <= outs(3);
- en_b <= outs(2);
- reset_counter <= outs(1);
- read_complete<= outs(0).



FSMs

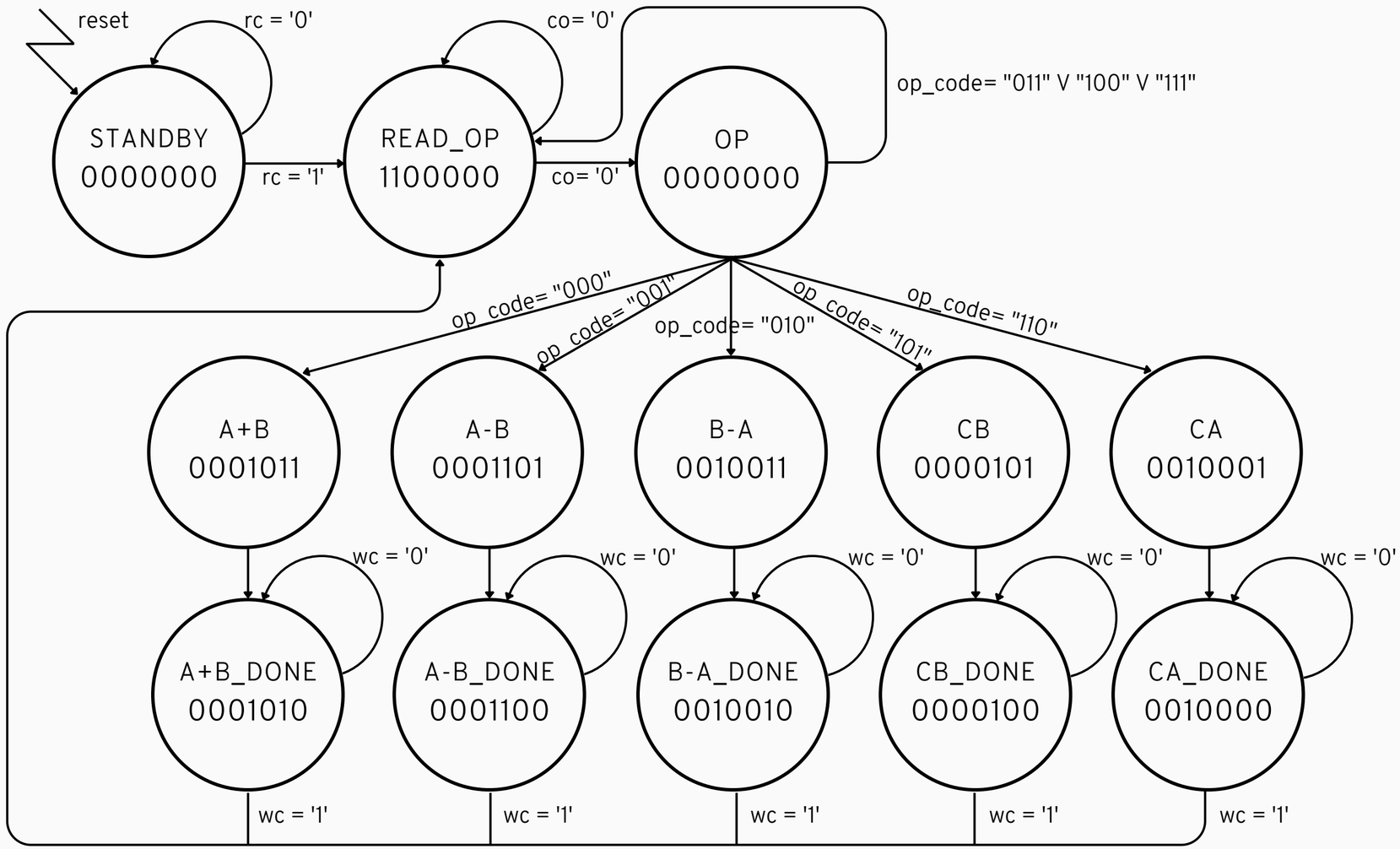
CU_ALU gestisce la fase di operazione selezionando le operazioni da svolgere sui numeri memorizzati.

Input:

- read_complete;
- count_o;
- operation_code(0:2);
- write_complete.

Outputs:

- en_op <= outs(6);
- reset_counter <= outs(5);
- op_a <= outs(3:4);
- op_b <= outs(1:2);
- op_complete<= outs(0).



FSMs

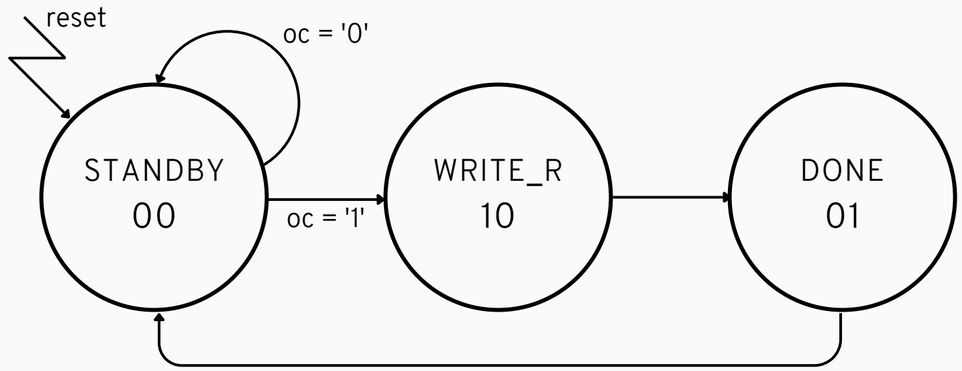
CU_REG_OUT gestisce la fase di scrittura abilitando il registro del risultato ottenuto.

Input:

- op_complete.

Outputs:

- en_r <= outs(1);
- write_complete <= outs(0).



Introduzione

Schema Circuitale

Componenti

FSMs

Risultati simulazioni

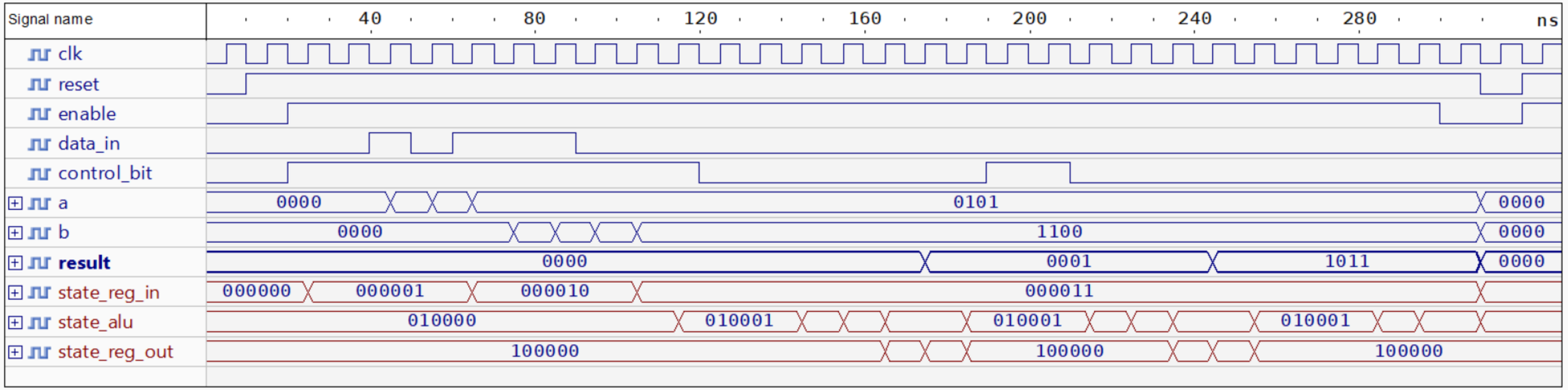
VHDL

Risultati simulazioni

RX → SUM → CA2 → RESET

- A := "0101" = 5
- B := "1100" = -4

Risultato atteso : Result := "0001" → "1011"

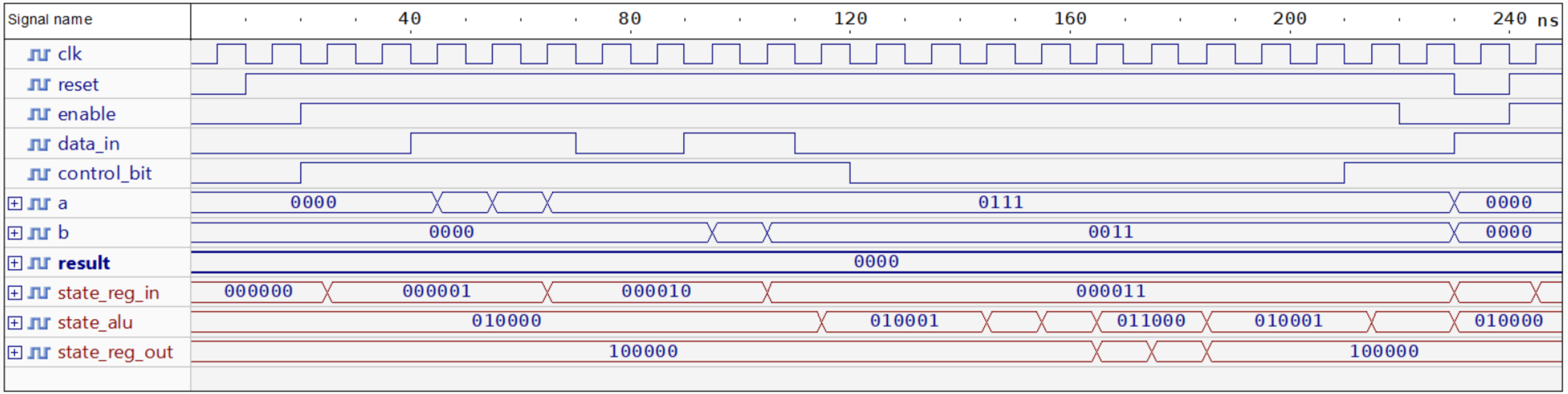


Risultati simulazioni

RX → SUM → RESET

- A := "0111" = 7
- B := "0011" = 3

Risultato atteso : Result := "0000" (Overflow)

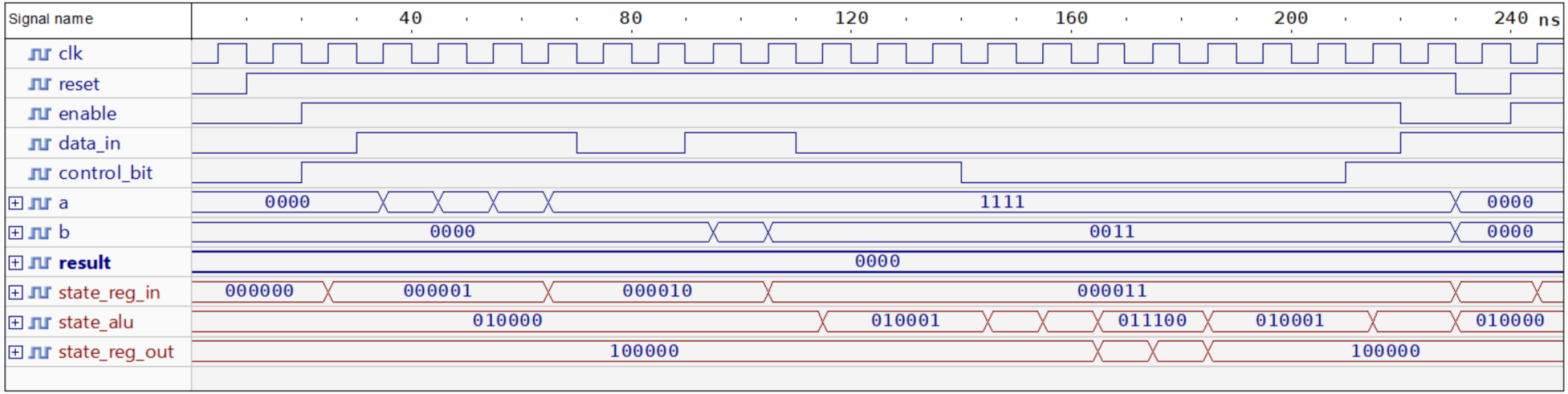


Risultati simulazioni

RX → CA2 → RESET

- A := "1111" = -8
- B := "0011" = 3

Risultato atteso : Result := "0000" (Lowest value)

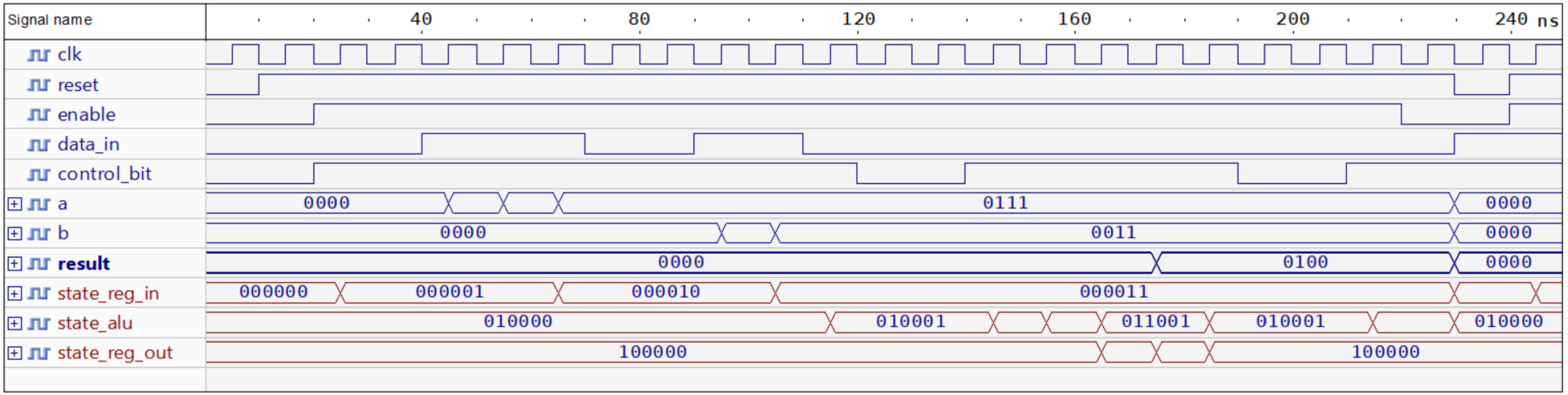


Risultati simulazioni

RX → SUB → RESET

- A := "0111" = 7
- B := "0011" = 3

Risultato atteso : Result := "0100"

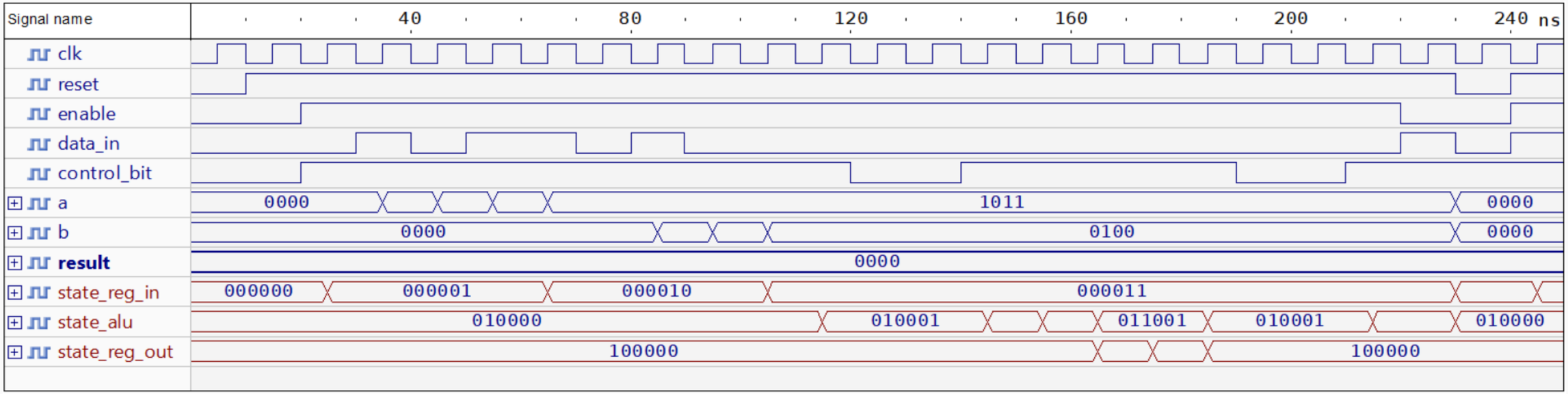


Risultati simulazioni

RX → SUB → RESET

- A := "1011" = -5
- B := "0100" = 4

Risultato atteso : Result := "0000" (Underflow)

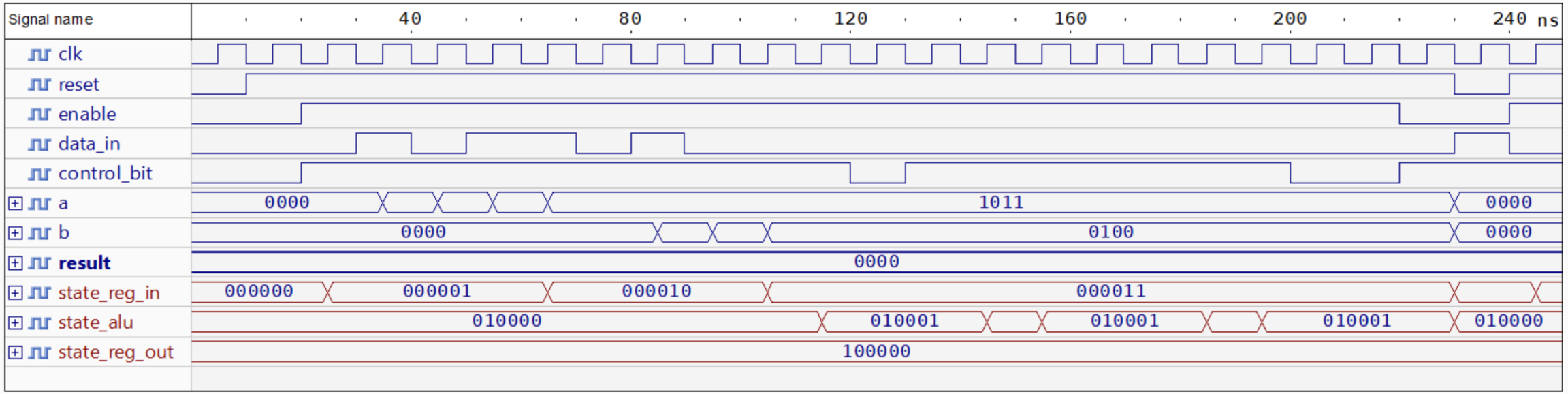


Risultati simulazioni

RX → ??? → RESET

- A := "1011" = -5
- B := "0100" = 4

Risultato atteso : Result := "0000" (Wrong Operation Code)



Introduzione
Schema Circuitale
Componenti
FSMs
Risultati simulazioni
VHDL

VHDL

```vhdl
-------------------------------------------------------------------------------
--
-- Title       : MO_ALU
-- Design      : MO_ALU
-- Author      : e.papa6@campus.unimib.it & d.gargaro@campus.unimib.it
-- Company     : Universita' degli Studi di Milano Bicocca
--
-------------------------------------------------------------------------------
--
-- Description :
--
-------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity MO_ALU is
    generic (Nb : integer);
    port(
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        enable : in STD_LOGIC;
        data_in : in STD_LOGIC;
        control_bit : in STD_LOGIC;
        a : out STD_LOGIC_VECTOR(Nb-1 downto 0);
        b : out STD_LOGIC_VECTOR(Nb-1 downto 0);
        result : out STD_LOGIC_VECTOR(Nb-1 downto 0);
        state_reg_in : out STD_LOGIC_VECTOR(5 downto 0);
        state_alu : out STD_LOGIC_VECTOR(5 downto 0);
        state_reg_out : out STD_LOGIC_VECTOR(5 downto 0)
    );
end MO_ALU;

```

VHDL

```vhdl
architecture MO_ALU_behavior of MO_ALU is

    component REG_IN is
        generic(Nb : integer);
        port(
            clk : in STD_LOGIC;
            reset : in STD_LOGIC;
            enable_sipo_a : in STD_LOGIC;
            enable_sipo_b : in STD_LOGIC;
            data_in : in STD_LOGIC;
            data_out_a : out STD_LOGIC_VECTOR(Nb-1 downto 0);
            data_out_b : out STD_LOGIC_VECTOR(Nb-1 downto 0)
        );
    end component REG_IN;

    component ALU is
        generic (Nb : integer);
        port(
            a : in STD_LOGIC_VECTOR(Nb-1 downto 0);
            b : in STD_LOGIC_VECTOR(Nb-1 downto 0);
            op_a : in STD_LOGIC_VECTOR(1 downto 0);
            op_b : in STD_LOGIC_VECTOR(1 downto 0);
            r : out STD_LOGIC_VECTOR(Nb-1 downto 0)
        );
    end component ALU;

    component REG_OUT is
        generic(Nb : integer);
        port(
            clk : in STD_LOGIC;
            reset : in STD_LOGIC;
            enable_pipo_r : in STD_LOGIC;
            data_in : in STD_LOGIC_VECTOR(Nb-1 downto 0);
            data_out : out STD_LOGIC_VECTOR(Nb-1 downto 0)
        );
    end component REG_OUT;
```

VHDL

```
70
71      component CU is
72          generic (Nb : integer);
73          port(
74              clk : in STD_LOGIC;
75              reset : in STD_LOGIC;
76              enable : in STD_LOGIC;
77              control_bit : in STD_LOGIC;
78              enable_a : out STD_LOGIC;
79              enable_b : out STD_LOGIC;
80              op_a : out STD_LOGIC_VECTOR(1 downto 0);
81              op_b : out STD_LOGIC_VECTOR(1 downto 0);
82              enable_r : out STD_LOGIC;
83              state_reg_in : out STD_LOGIC_VECTOR(5 downto 0);
84              state_alu : out STD_LOGIC_VECTOR(5 downto 0);
85              state_reg_out : out STD_LOGIC_VECTOR(5 downto 0)
86          );
87      end component CU;
88
```

VHDL

```
 97
 98  begin
 99
100      r_in : REG_IN generic map (Nb) port map (clk, reset, enable_a, enable_b, data_in, reg_data_out_a, reg_data_out_b);
101      op_alu : ALU generic map (Nb) port map (reg_data_out_a, reg_data_out_b, op_a, op_b, alu_data_out);
102      r_out : REG_OUT generic map (Nb) port map (clk, reset, enable_r, alu_data_out, result);
103
104      fsm : CU generic map (Nb) port map (clk, reset, enable, control_bit, enable_a, enable_b, op_a, op_b, enable_r, state_reg_in, state_alu, state_reg_out);
105
106      a <= reg_data_out_a;
107      b <= reg_data_out_b;
108
109  end MO_ALU_behavior;
110
```