

Circuite VLSI - Proiect

Circuit secvențial de împărțire fără restaurare

numere pozitive

**Autor:**

Pârvan Andrei Leonard

Calculatoare anul 4

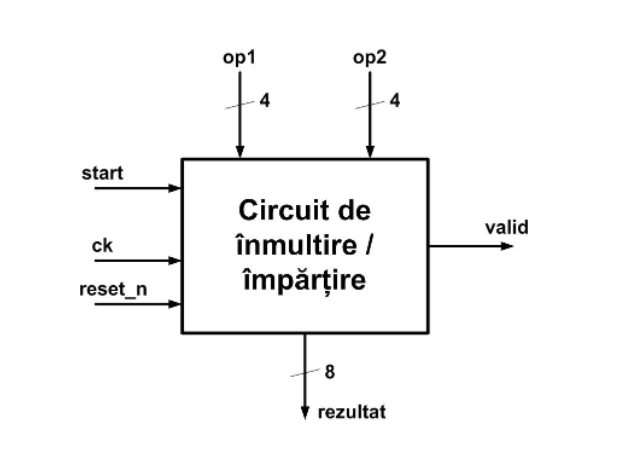
Grupa 4LF781

[andrei.parvan@student.unitbv.ro](mailto:andrei.parvan@student.unitbv.ro)

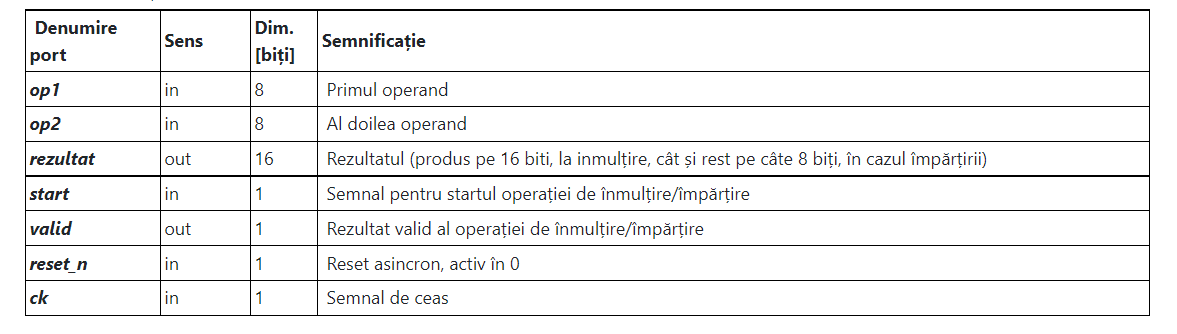
BRAȘOV 2022

Specificații:

* Schema bloc a circuitului:



* Descrierea porturilor:



* Algoritmul utilizat:

*Algoritm de împărțire fără restaurare (numere pozitive)*

- resetează P (n+1 biți)

- încarcă deîmpărțitul în A (n biți)

- încarcă împărțitorul în B (n biți)

- repeta de n ori

- dacă P este negativ (MSB=1) atunci

- deplasează cu o poziție stânga P (LSB P = MSB A)

- P <= P + B

altfel

- deplasează cu o poziție stânga P (LSB P = MSB A)

- P <= P + (-B)

- deplasează cu o poziție stânga A (LSB A = not MSB P)

- dacă P este negativ (MSB=1) atunci

- P <= P + B

- P conține REST

- A conține CÂT

* Constrângeri:

Sistemul se va modela ca un sistem riguros sincron. Semnalul ***ck*** va fi primit de către toate registrele din sistem.

Se vor evita circuitele de divizare a frecvenței semnalului de tact.

Se vor folosi, eventual, circuite de generare a semnalelor de activare (*enable*).

Descrierea sistemului se va face în întregime în VHDL. În descrierea de nivel înalt se vor instanția două componente:

* cale de date
* cale de control

*Exemplu:*

75 : 10 = 7 rest 5 85 : 10 = 8 rest 5

75 = 0 0100 1011 85 = 0 0101 0101

10 = 0 0000 1010 10 = 0 0000 1010

-10 = 1 1111 0110 -10 = 1 1111 0110

Reg. P Reg. A Reg. P Reg. A

----------- --------- ----------- ---------

0 0000 0000 0100 1011 0 0000 0000 0101 0101 - inițial

0 0000 0000 0 0000 0000

1 1111 0110 1 1111 0110

----------- -----------

1 1111 0110 1001 0110 1 1111 0110 1010 1010 - iterația 1

1 1110 1101 1 1110 1101

0 0000 1010 0 0000 1010

----------- -----------

1 1111 0111 0010 1100 1 1111 0111 0101 0100 - iterația 2

1 1110 1110 1 1110 1110

0 0000 1010 0 0000 1010

----------- -----------

1 1111 1000 0101 1000 1 1111 1000 1010 1000 - iterația 3

1 1111 0000 1 1111 0001

0 0000 1010 0 0000 1010

----------- -----------

1 1111 1010 1011 0000 1 1111 1011 0101 0000 - iterația 4

1 1111 0101 1 1111 0110

0 0000 1010 0 0000 1010

----------- -----------

1 1111 1111 0110 0000 0 0000 0000 1010 0001 - iterația 5

1 1111 1110 0 0000 0001

0 0000 1010 1 1111 0110

----------- -----------

0 0000 1000 1100 0001 1 1111 0111 0100 0010 - iterația 6

0 0001 0001 1 1110 1110

1 1111 0110 0 0000 1010

----------- -----------

0 0000 0111 1000 0011 1 1111 1000 1000 0100 - iterația 7

0 0000 1111 1 1111 0001

1 1111 0110 0 0000 1010

----------- -----------

0 0000 0101 0000 0111 1 1111 1011 0000 1000 - iterația 8

1 1111 1011 - final

0 0000 1010

-----------

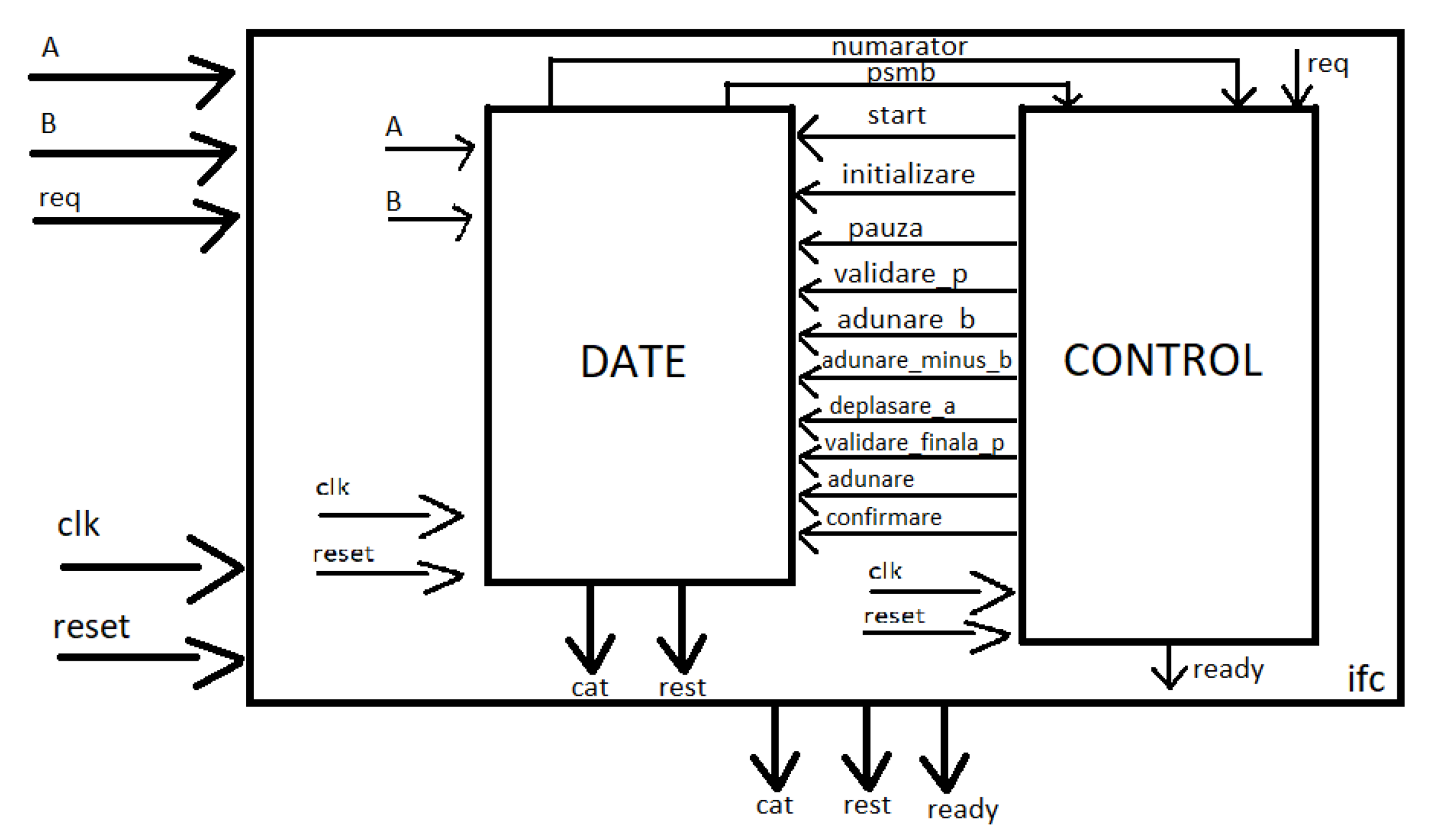
0 0000 0101

0000 0101 0000 0111 0000 0101 0000 1000

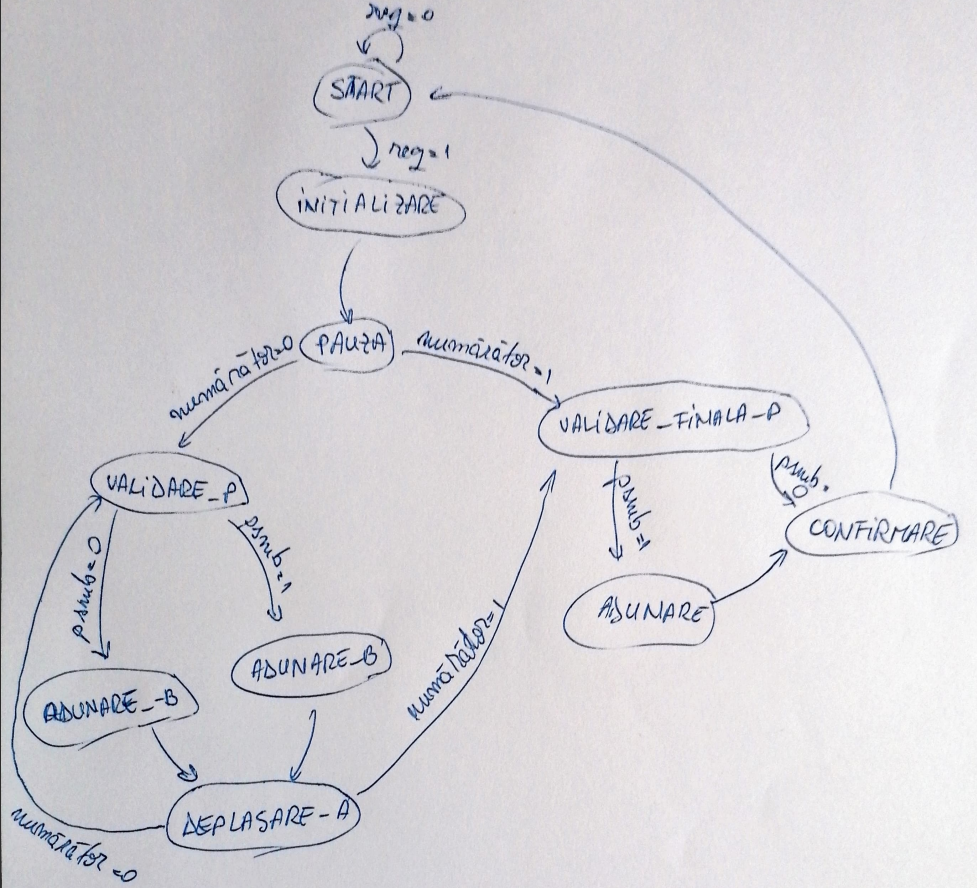
--------- --------- --------- ---------

REST = 5 CÂT = 7 REST = 5 CÂT = 8

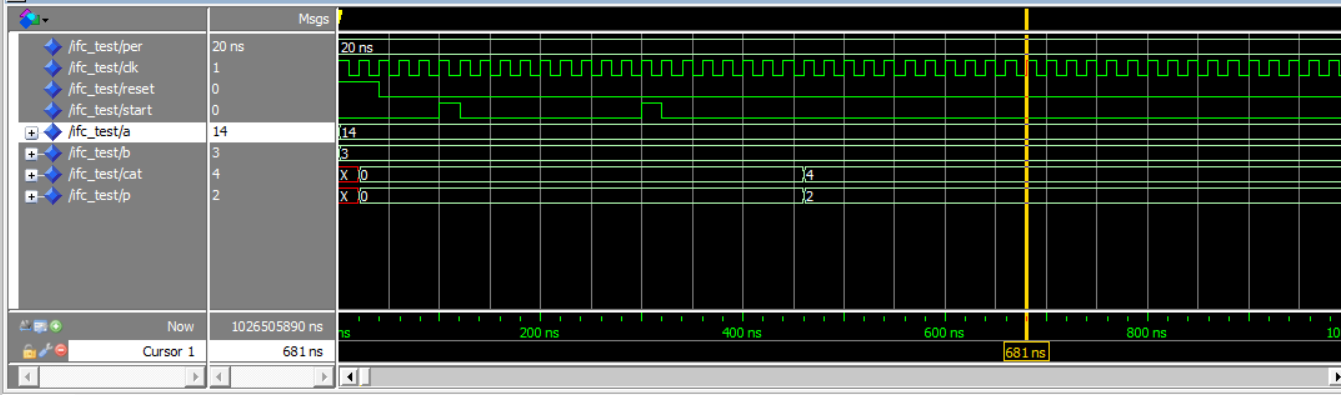
**Reprezentare grafică a structurii de nivel înalt**

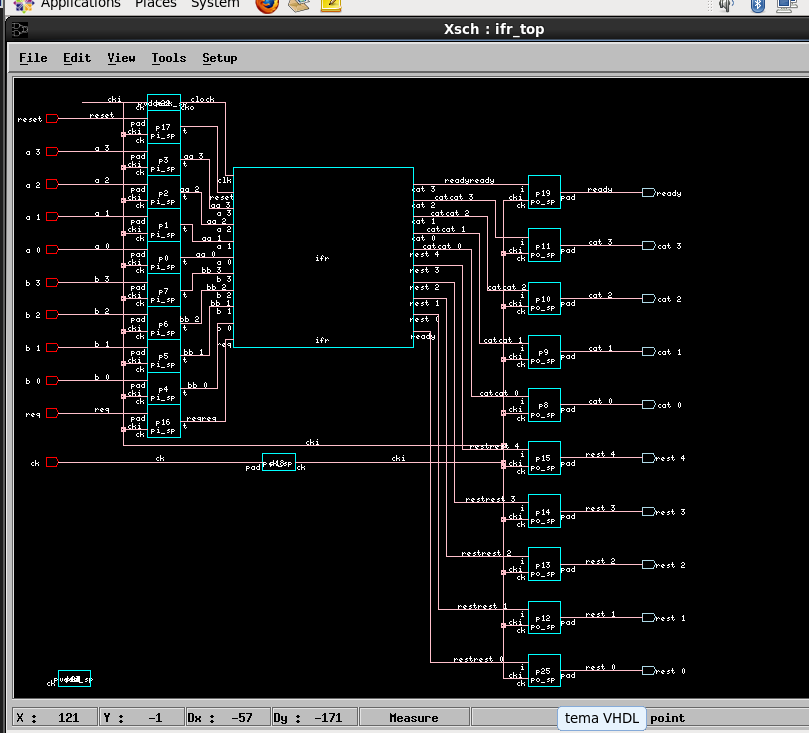


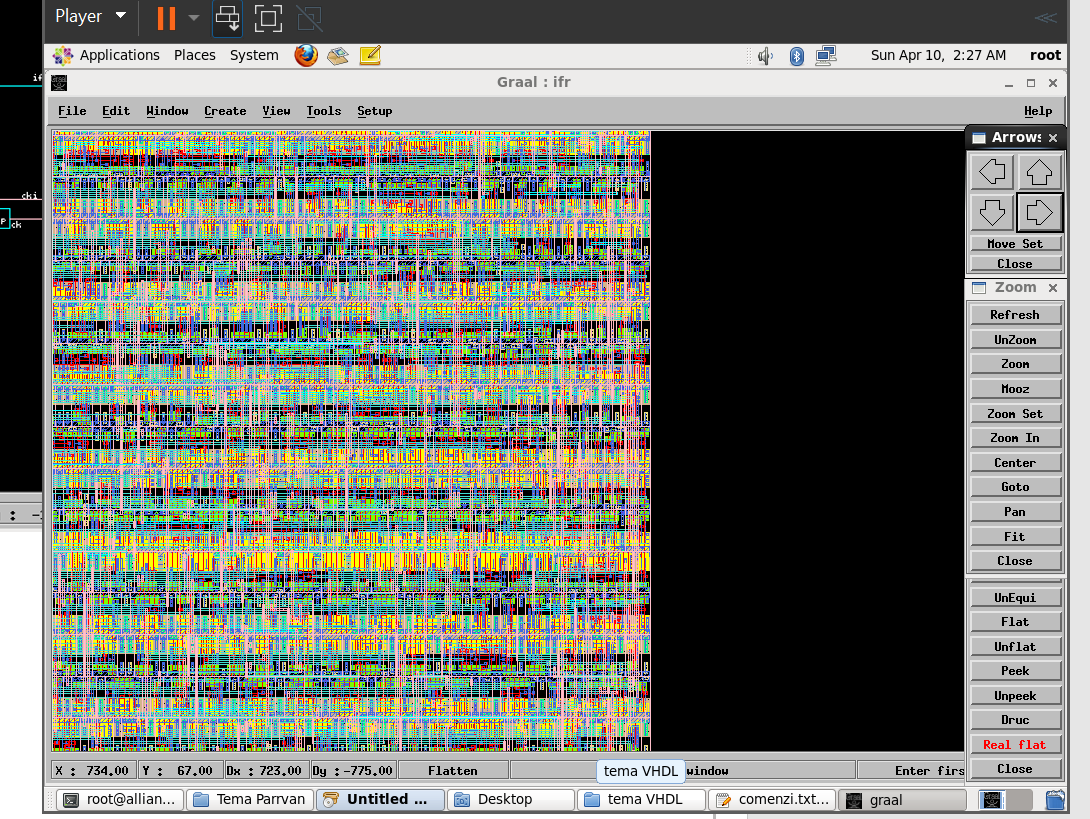
**Graficul de stari**

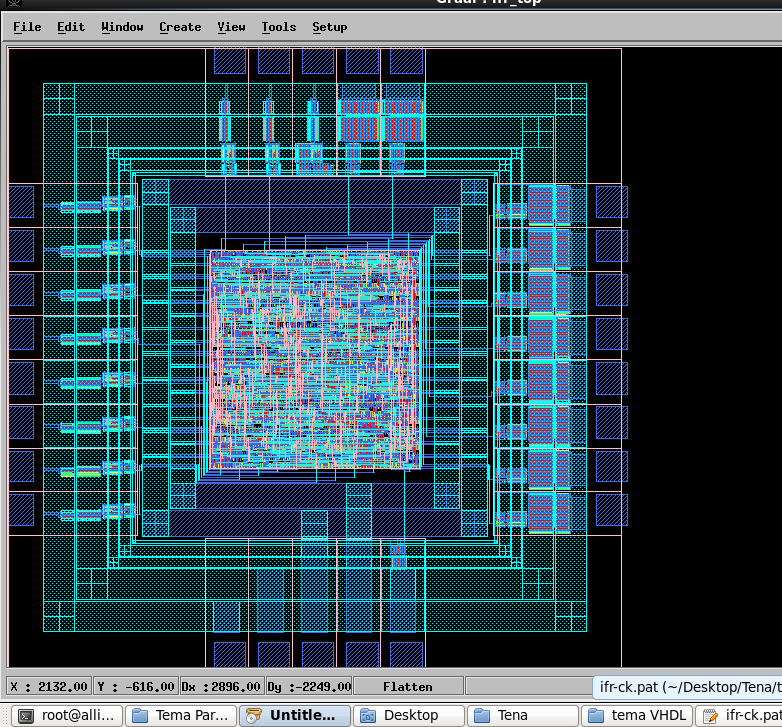
****

**Simulare circuit în ModelSIM**

****

****

****

****

**Fișier date.vhd**

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

USE IEEE.std\_logic\_arith.ALL;

USE IEEE.std\_logic\_unsigned.ALL;

ENTITY date IS

PORT (

    clk           : IN  std\_logic;                        --clock

    reset         : IN  std\_logic;                        --reset

    a             : IN  std\_logic\_vector(3 DOWNTO 0);     --operandul A

    b             : IN  std\_logic\_vector(3 DOWNTO 0);     --operandul B

    start             : IN std\_logic;                     -- semnale pentru stari

    initizlizare      : IN std\_logic;                     -- --||--

    pauza             : IN std\_logic;                     -- --||--

    validare\_p        : IN std\_logic;                     -- --||--

    adunare\_b         : IN std\_logic;                     -- --||--

    adunare\_minus\_b   : IN std\_logic;                     -- --||--

    deplasare\_a       : IN std\_logic;                     -- --||--

    validare\_finala\_p : IN std\_logic;                     -- --||--

    adunare           : IN std\_logic;                     -- --||--

    confirmare        : IN std\_logic;                     -- --||--

  cat                 : OUT std\_logic\_vector(3 DOWNTO 0); --catul

  rest                : OUT std\_logic\_vector(4 DOWNTO 0); --restul

    numarator         : OUT std\_logic; --necesar pentru Calea de control

    pmsb              : OUT std\_logic --necesar pentru Calea de control

);

END date;

ARCHITECTURE date OF date IS

  SIGNAL reg\_a      : std\_logic\_vector(3 DOWNTO 0);      -- a pe n biti

  SIGNAL reg\_b      : std\_logic\_vector(3 DOWNTO 0);      -- b pe n biti

  SIGNAL reg\_p      : std\_logic\_vector(4 DOWNTO 0);      -- p pe n+1 biti

  SIGNAL count      : std\_logic\_vector(2 DOWNTO 0);      -- ce e asta

BEGIN

--reg A

PROCESS(clk)

  BEGIN

    IF (clk'EVENT AND clk = '1') THEN                                                 -- la fiecare event de clock

      IF  (reset = '1')     THEN reg\_a <= (others => '0');                            -- daca reset e 1 atunci reg\_a primeste 0?

      ELSIF (initizlizare = '1')THEN reg\_a <= a;                                      -- daca init activ reg\_a primeste a

    ELSIF (deplasare\_a = '1') THEN reg\_a <= reg\_a(2 DOWNTO 0) & ( not reg\_p(4));      -- daca deplasare a activ reg este shiftat la stanga cu un bit

      END IF;

    END IF;

END PROCESS;

--reg B

PROCESS(clk)

  BEGIN

    IF (clk'EVENT AND clk = '1') THEN                           -- la fiecare event de clock

      IF  (reset = '1')     THEN reg\_b <= (others => '0');      -- reg\_b primeste 0 la reset activ

      ELSIF (initizlizare = '1')THEN reg\_b <= b;                -- daca init activ reg\_b primeste b

      END IF;

    END IF;

END PROCESS;

--reg P

PROCESS(clk)

  BEGIN

    IF (clk'EVENT AND clk = '1') THEN                                                         -- la fiecare event de clock

      IF  (reset = '1')       THEN reg\_p <= (others => '0');                                  -- reset activ reg\_p primeste 0

      ELSIF (initizlizare = '1')  THEN reg\_p <= (others => '0');                              -- initializarea reg\_p primeste 0

    ELSIF (adunare = '1')     THEN reg\_p <= reg\_p + reg\_b;                                    -- cand adunare este activ se aduna reg\_P cu reg\_b

      ELSIF (adunare\_b = '1')     THEN reg\_p <= (reg\_p(3 DOWNTO 0) & reg\_a(3)) + reg\_b  ;     -- se aduna fie B fie (-B)

    ELSIF (adunare\_minus\_b = '1') THEN reg\_p <= (reg\_p(3 DOWNTO 0) & reg\_a(3)) - reg\_b ;

      END IF;

    END IF;

END PROCESS;

--reg count

PROCESS(clk)

  BEGIN

    IF (clk'EVENT AND clk = '1') THEN

      IF  ( reset = '1' )     THEN count <= (others => '0');                                  -- initializare la reset sau init

    ELSIF ( initizlizare = '1' )  THEN count <= (others => '0');

      ELSIF ( validare\_p = '1' )  THEN count <= count + "001";                                -- se incrementeaza cu 1 cand valid\_p activ

      END IF;

    END IF;

END PROCESS;

--cat

PROCESS(clk)

  BEGIN

  IF (clk'EVENT AND clk = '1') THEN

      IF  ( reset = '1' )   THEN cat <= (others => '0');                    -- initializare cat

    ELSIF (confirmare = '1')  THEN cat <= reg\_a;                            -- atribuie catul care se afla in reg\_a la starea de confirmare

    END IF;

  END IF;

END PROCESS;

--rest

PROCESS(clk)

  BEGIN

  IF (clk'EVENT AND clk = '1') THEN

      IF  ( reset = '1' )   THEN rest <= (others => '0');                   -- initalizare rest

    ELSIF (confirmare = '1')  THEN rest <= reg\_p;                           -- i se atribuie restul din reg\_p conform pdf-ului

    END IF;

  END IF;

END PROCESS;

pmsb <= reg\_p(4);                                           -- bitul cel mai semnificativ al lui p

numarator <='1' WHEN count = "100" else '0';                -- numaratorul

END date;

**Fișier control.vhd**

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

ENTITY control IS

PORT (

    clk             : IN  std\_logic;

    reset           : IN  std\_logic;

    --FSM inputs

    req             : IN  std\_logic; --request

    numarator       : IN  std\_logic; -- cand se termina while 'numarator' <=1 altfel =0

    pmsb            : IN  std\_logic;

    --FSM outputs

    start             : OUT std\_logic;

    initizlizare      : OUT std\_logic;

    pauza             : OUT std\_logic;

    validare\_p        : OUT std\_logic;

    adunare\_b         : OUT std\_logic;

    adunare\_minus\_b   : OUT std\_logic;

    deplasare\_a       : OUT std\_logic;

    validare\_finala\_p : OUT std\_logic;

    adunare           : OUT std\_logic;

    confirmare        : OUT std\_logic;

    ready             : OUT std\_logic  --valid (BUSY)

);

END control;

ARCHITECTURE control OF control IS

  TYPE state IS (S\_START,S\_INITIALIZARE,S\_PAUZA,S\_VALIDARE\_P,S\_ADUNARE\_B,S\_ADUNARE\_MINUS\_B,S\_DEPLASARE\_A,S\_VALIDARE\_FINALA\_P,S\_ADUNARE,S\_CONFIRMARE);

  SIGNAL currentState : state;

  SIGNAL nextState    : state;

  SIGNAL ld\_r\_ready   : std\_logic;

BEGIN

CLC: PROCESS ( currentState, req, numarator, pmsb ) BEGIN

  CASE currentState IS

    WHEN S\_START            => IF ( req   = '1' )     THEN

                                    nextState <= S\_INITIALIZARE;

                               ELSE

                                    nextState <= S\_START;

                               END IF;

    WHEN S\_INITIALIZARE     =>      nextState <= S\_PAUZA;

    WHEN S\_PAUZA            => IF ( numarator = '1' ) THEN

                                    nextState <= S\_VALIDARE\_FINALA\_P;

                               ELSE

                                    nextState <= S\_VALIDARE\_P;

                               END IF;

    WHEN S\_VALIDARE\_P       => IF ( pmsb = '1' )      THEN

                                    nextState <= S\_ADUNARE\_B;

                               ELSE

                                    nextState <= S\_ADUNARE\_MINUS\_B;

                               END IF;

    WHEN S\_ADUNARE\_B        =>      nextState <= S\_DEPLASARE\_A;

    WHEN S\_ADUNARE\_MINUS\_B  =>      nextState <= S\_DEPLASARE\_A;

    WHEN S\_DEPLASARE\_A      => IF ( numarator = '1' )  THEN

                                    nextState <= S\_VALIDARE\_FINALA\_P;

                               ELSE

                                    nextState <= S\_VALIDARE\_P;

                               END IF;

    WHEN S\_VALIDARE\_FINALA\_P=> IF ( pmsb = '1' )       THEN

                                    nextState <= S\_ADUNARE;

                               ELSE

                                    nextState <= S\_CONFIRMARE;

                               END IF;

    WHEN S\_ADUNARE          =>      nextState <= S\_CONFIRMARE;

    WHEN S\_CONFIRMARE       =>      nextState <= S\_START;

  END CASE;

END PROCESS CLC;

REG: PROCESS ( clk ) BEGIN

  IF (clk'EVENT AND clk = '1') THEN

    IF (reset = '1') THEN

        currentState <= S\_START;

    ELSE

        currentState <= nextState;

    END IF;

  END IF;

END PROCESS REG;

--FSM outputs

start             <= '1' WHEN (currentState = S\_START             ) ELSE '0';

initizlizare      <= '1' WHEN (currentState = S\_INITIALIZARE      ) ELSE '0';

pauza             <= '1' WHEN (currentState = S\_PAUZA             ) ELSE '0';

validare\_p        <= '1' WHEN (currentState = S\_VALIDARE\_P        ) ELSE '0';

adunare\_b         <= '1' WHEN (currentState = S\_ADUNARE\_B         ) ELSE '0';

adunare\_minus\_b   <= '1' WHEN (currentState = S\_ADUNARE\_MINUS\_B   ) ELSE '0';

deplasare\_a       <= '1' WHEN (currentState = S\_DEPLASARE\_A       ) ELSE '0';

validare\_finala\_p <= '1' WHEN (currentState = S\_VALIDARE\_FINALA\_P ) ELSE '0';

adunare           <= '1' WHEN (currentState = S\_ADUNARE           ) ELSE '0';

confirmare        <= '1' WHEN (currentState = S\_CONFIRMARE        ) ELSE '0';

ld\_r\_ready        <= '1' WHEN (currentState = S\_START             ) ELSE '0';

-- ready is ld\_r delayed

PROCESS ( clk ) BEGIN

  IF (clk'EVENT AND clk = '1') THEN

    IF (reset = '1') THEN

        ready <= '0';

    ELSE

        ready <= ld\_r\_ready;

    END IF;

  END IF;

END PROCESS;

END control;

**Fișier ifc.vhd**

LIBRARY IEEE;

USE IEEE.std\_logic\_1164.ALL;

ENTITY ifc IS

PORT (

    clk     : IN  std\_logic;

    reset   : IN  std\_logic;

    a       : IN  std\_logic\_vector(3 DOWNTO 0);

    b       : IN  std\_logic\_vector(3 DOWNTO 0);

    req     : IN  std\_logic;

    cat     : OUT std\_logic\_vector(3 DOWNTO 0);

    rest    : OUT std\_logic\_vector(4 DOWNTO 0);

    ready   : OUT  std\_logic

);

END ifc;

ARCHITECTURE ifc OF ifc IS

    COMPONENT date

        PORT (

                clk                 : IN  std\_logic;

                reset               : IN  std\_logic;

                a                   : IN  std\_logic\_vector(3 DOWNTO 0);

                b                   : IN  std\_logic\_vector(3 DOWNTO 0);

                start               : IN std\_logic;

                initizlizare        : IN std\_logic;

                pauza               : IN std\_logic;

                validare\_p          : IN std\_logic;

                adunare\_b           : IN std\_logic;

                adunare\_minus\_b     : IN std\_logic;

                deplasare\_a         : IN std\_logic;

                validare\_finala\_p   : IN std\_logic;

                adunare             : IN std\_logic;

                confirmare          : IN std\_logic;

                cat                 : OUT std\_logic\_vector(3 DOWNTO 0);

                rest                : OUT std\_logic\_vector(4 DOWNTO 0);

                numarator           : OUT std\_logic; --necesar pentru Calea de control

                pmsb                : OUT std\_logic --necesar pentru Calea de control

        );

    END COMPONENT;

    COMPONENT control

        PORT (

            clk                 : IN  std\_logic;

            reset               : IN  std\_logic;

            --FSM inputs

            req                 : IN  std\_logic; --request

            numarator           : IN  std\_logic; -- cand se termina while 'numarator' <=1 altfel =0

            pmsb                : IN  std\_logic;

            --FSM outputs

            start               : OUT std\_logic;

            initizlizare        : OUT std\_logic;

            pauza               : OUT std\_logic;

            validare\_p          : OUT std\_logic;

            adunare\_b           : OUT std\_logic;

            adunare\_minus\_b     : OUT std\_logic;

            deplasare\_a         : OUT std\_logic;

            validare\_finala\_p   : OUT std\_logic;

            adunare             : OUT std\_logic;

            confirmare          : OUT std\_logic;

            ready               : OUT std\_logic  --valid (BUSY)

        );

     END COMPONENT;

-- signals to connect date/control

SIGNAL start,initizlizare,pauza,validare\_p,adunare\_b,adunare\_minus\_b,deplasare\_a,validare\_finala\_p,adunare,confirmare,numarator,pmsb : std\_logic;

BEGIN

  i\_date: date

  PORT MAP(

        clk                 => clk,

        reset               => reset,

        a                   => a,

        b                   => b,

        start               => start,

        initizlizare        => initizlizare,

        pauza               => pauza,

        validare\_p          => validare\_p,

        adunare\_b           => adunare\_b,

        adunare\_minus\_b     => adunare\_minus\_b,

        deplasare\_a         => deplasare\_a,

        validare\_finala\_p   => validare\_finala\_p,

        adunare             => adunare,

        confirmare          => confirmare,

        cat                 => cat,

        rest                => rest,

        numarator           => numarator,

        pmsb                => pmsb

   );

  i\_ctrl: control

    PORT MAP(

        clk                 => clk,

        reset               => reset,

        req                 => req,

        numarator           => numarator,

        pmsb                => pmsb,

        start               => start,

        initizlizare        => initizlizare,

        pauza               => pauza,

        validare\_p          => validare\_p,

        adunare\_b           => adunare\_b,

        adunare\_minus\_b     => adunare\_minus\_b,

        deplasare\_a         => deplasare\_a,

        validare\_finala\_p   => validare\_finala\_p,

        adunare             => adunare,

        confirmare          => confirmare,

        ready               => ready

   );

END ifc;

**Fișier .c**

#include <genlib.h>

main()

{

int i;

   GENLIB\_DEF\_LOFIG("ifr\_top");

   GENLIB\_LOCON("vdd",           IN,        "vdd");

   GENLIB\_LOCON("vss",           IN,        "vss");

   GENLIB\_LOCON("vdde",          IN,       "vdde");

   GENLIB\_LOCON("vsse",          IN,       "vsse");

   GENLIB\_LOCON("ck",            IN,       "ck");

   GENLIB\_LOCON("reset",         IN,      "reset");

   GENLIB\_LOCON("a[3:0]",        IN,     "a[3:0]");

   GENLIB\_LOCON("b[3:0]",        IN,     "b[3:0]");

   GENLIB\_LOCON("req",         IN,      "req");

   GENLIB\_LOCON("ready",        OUT,      "ready");

   GENLIB\_LOCON("cat[3:0]",    OUT,    "cat[3:0]");

   GENLIB\_LOCON("rest[4:0]",    OUT,  "rest[4:0]");

   GENLIB\_LOINS ("pvsse\_sp", "p20", "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS ("pvdde\_sp", "p21", "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS ("pvddeck\_sp", "p22", "clock", "cki", "vdde", "vdd", "vsse", "vss",0);

   GENLIB\_LOINS ("pvssi\_sp", "p23", "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS ("pvddi\_sp", "p24", "cki", "vdde", "vdd", "vsse", "vss", 0);

   for (i = 0; i < 4; i++)

    GENLIB\_LOINS("pi\_sp", GENLIB\_NAME("p%d", i),

          GENLIB\_NAME("a[%d]", i), GENLIB\_NAME("aa[%d]", i),

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   for (i = 0; i < 4; i++)

    GENLIB\_LOINS("pi\_sp", GENLIB\_NAME("p%d", i + 4),

          GENLIB\_NAME("b[%d]", i), GENLIB\_NAME("bb[%d]", i),

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   for (i = 0; i < 4; i++)

    GENLIB\_LOINS("po\_sp", GENLIB\_NAME("p%d", i + 8),

          GENLIB\_NAME("catcat[%d]", i), GENLIB\_NAME("cat[%d]", i),

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS("po\_sp", "p25",

         "restrest[0]", "rest[0]",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

GENLIB\_LOINS("po\_sp", "p12",

         "restrest[1]", "rest[1]",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

GENLIB\_LOINS("po\_sp", "p13",

         "restrest[2]", "rest[2]",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

GENLIB\_LOINS("po\_sp", "p14",

         "restrest[3]", "rest[3]",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

GENLIB\_LOINS("po\_sp", "p15",

         "restrest[4]", "rest[4]",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS("pi\_sp", "p16",

         "req", "reqreq",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS("pi\_sp", "p17",

        "reset", "resetreset",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS("pck\_sp", "p18",

         "ck",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

   GENLIB\_LOINS("po\_sp", "p19",

         "readyready", "ready",

         "cki", "vdde", "vdd", "vsse", "vss", 0);

     GENLIB\_LOINS("ifr", "ifr","clock", "resetreset",

           "aa[3:0]", "bb[3:0]","reqreq","readyready",

           "catcat[3:0]","restrest[4:0]",

         "vdd", "vss", 0);

   GENLIB\_SAVE\_LOFIG();

   exit(0);

}

**Fișier .pat**

**in a(3 DOWNTO 0);;;**

**in b(3 DOWNTO 0);;;**

**in clk;;;**

**in reset;;;**

**in req;;;**

**in vdd;**

**in vss;;;**

**out ready;;;**

**out cat(3 DOWNTO 0);;**

**out rest(4 DOWNTO 0);;**

**BEGIN**

**-- Pattern description :**

**-- a b c r r vv r c r**

**-- l e e ds e a e**

**-- k s q ds a t s**

**-- e d t**

**-- t y**

**pat\_0 : 1111 0011 1 1 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_1 : 1111 0011 0 1 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_2 : 1111 0011 1 1 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_3 : 1111 0011 0 1 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_4 : 1111 0011 1 0 1 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_5 : 1111 0011 0 0 1 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_6 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_7 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_8 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_9 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_10 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_11 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_13 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_14 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_15 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_16 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_17 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_18 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_19 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_20 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_21 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_22 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_23 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_24 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_25 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_26 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_27 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_28 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_29 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_30 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_31 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_32 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_33 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_34 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_35 : 1111 0011 1 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**pat\_36 : 1111 0011 0 0 0 10 ?\* ?\*\*\*\* ?\*\*\*\*\* ;**

**end;**

**Fișier .rin**

**#file: ifr.rin**

**width (vdd 100 vss 100)**

**west (p0 p1 p2 p3 p4 p5 p6 p7)**

**north (p16 p17 p18 p19 p25)**

**east (p8 p9 p10 p11 p12 p13 p14 p15)**

**south (p20 p21 p23 p24 p22)**