Laborator 9

Proiectarea în VHDL a unui microprocesor pe 8-biți – partea a II-a

1. **Contorul de program**
2. **Indicatorul de stivă**
3. **Selectorul de adrese**
4. **Detectorul de întreruperi**

1. Contorul de program

1.1 Specificații

Contorul de program (PC= Program Counter) are rolul de a păstra adresa locației de memorie din zona PROGRAM, locație de memorie care conține instrucțiunea în curs de executare. Odată procesată instrucțiunea curentă adresa din PC se incrementează, astfel că aceasta va stoca adresa următoarei instrucțiuni ce urmează a fi adusă, decodificată şi executată.

În figura 1 este prezentat circuitul de generare a adreselor. Una dintre componentele acestui circuit este şi contorul de program (modulul)PC0. Din figură se poate observa că acesta poate fi impelemntat utilizând un numărător reîncărcabil pe 8-biți (CB8CLE), luat din lista de componente standard a mediului Xilinx.

Specificațiile de proiectare ale contorului de program sunt următoarele:

- PC-ul trebuie să poate încărca o nouă valoare de pe magistrala internă de date D dacă la o intrarea LD_PC este prezent un „1” logic, iar la intrarea de clock urmează un front crescător. Această operație va fi utilă atunci când au loc salturi în program.
- Pentru executarea unui set de instrucțiuni de la adrese succesive, se menține un „1” logic la o intrarea INC_PC ceea ce va avea ca efect incrementarea conținutului program counterului.
- Valoarea de la ieşirea program counterului trebuie să ajungă pe magistrala D prin intermediul bufferul tristate BUFE8, activând semnalul DRV_D_PC = „1” .Această operație este necesară când se doreşte salvarea valorii PC-ului în memoria stivă.
- Un semnal cu valoarea „1” logic la intrările RESET sau CLR_PC trebuie să aibă ca rezultat ştergerea conținutului program counterului. Acest lucru este necesar înaintea executării unui program, pentru ca acesta să se deruleze de la adresa 0.

De asemenea valoarea din PC mai trebuie reinițializată şi în cazul începerii unei subrutine de tratare a întreruperilor (ISR = Interrupt Service Routine).

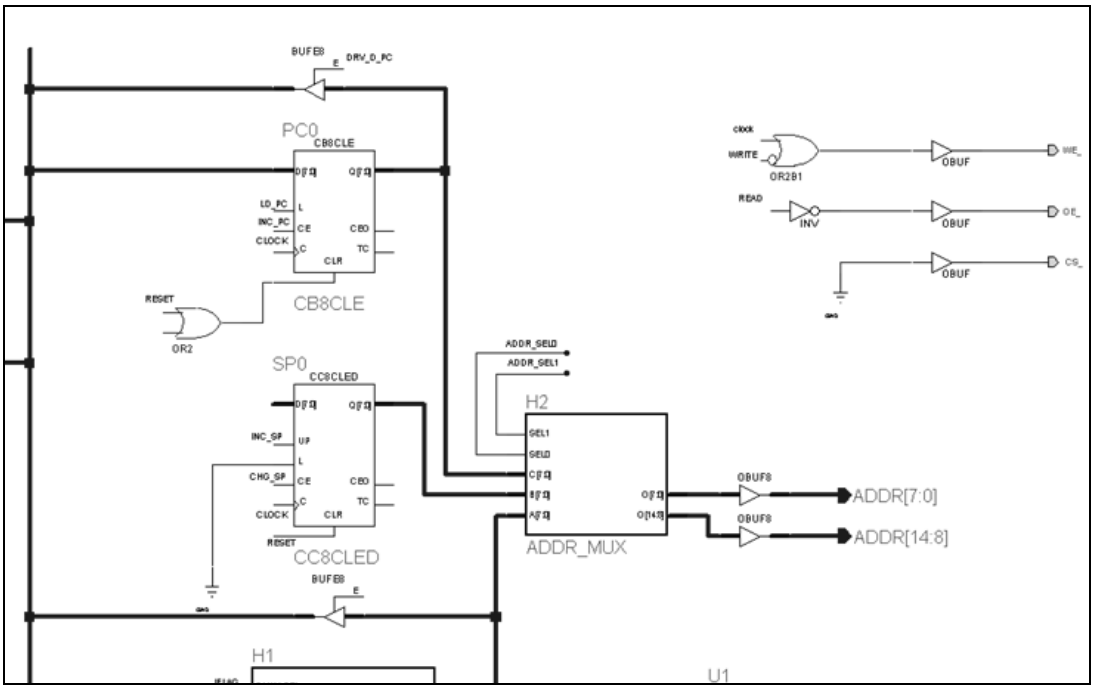

Figura 1 PC, SP, selectorul de adrese şi logica de control a memorie RAM externe

1.2 Descrierea în VHDL a contorului de program

Descrierea în VHDL se va face plecând de la descrierea numărătoarelor din laboratorul 4, modelul din figura 9.

```
entity PC is
port (RESET, LD_PC, ...: in std_logic;
...: in std_logic_vector ( ... downto 0);
...: out std_logic_vector (   ... downto 0 )) ;
end entity PC;

architecture behavior of PC is
signal count : unsigned (     7 downto 0 );
begin
```

```
process(RESET, CLR_PC, CLOCK, count)
begin
     if RESET = '1' or ... = '1' then
          count <= (others=>'0'); --resetare
     elsif CLOCK'event and CLOCK = '1' then
          if LD_PC =... then
               count   <=   unsigned   (D);   --funcția   de
incarcare paralela
          elsif INC_PC =... then
               count <= ...; --incrementeaza
          else null;
          end if;
     end if;
               Q <= conv_std_logic_vector(...,...);
end process;
end architecture behavior;
```

Figura 2. Cod VHDL corespunzător contorului de program

Completați secvențele de cod lipsă ținând cont de specificații şi de blocul corespunzător contorului de program, vezi diagrama din figura 1. Verificați sintaxa şi simulați proiectul.

Un exemplu de stimuli ce pot fi aplicați pentru testarea setului de registre este prezentat în figura 3.

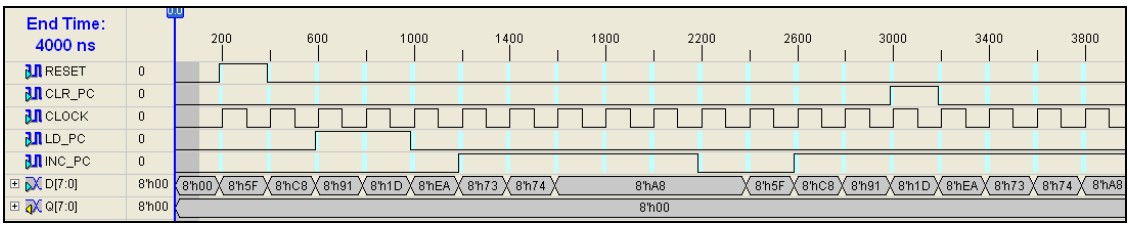

Figura 3. Stimuli pentru verificarea funcționării contorului de program

2. Indicatorul de stivă

2.1 Specificații

Indicatorul de stivă sau stack pointer-ul (SP) păstrează adresele locațiilor de memorie în care se vor stoca conținutul program counterului PC (contor de program) şi a

acumulatorului ACC. În Figura 1 este prezentat indicatorul de stivă (componenta SP0) ca făcând parte din circuitul de generare a adreselor şi poate fi descris şi cu o componentă standard (CC8CLED) .

Specificațiile de proiectare în VHDL ale indicatorului de stivă sunt următoarele:

- Modulul SP0 va fi descris ca şi un circuit numărător reversibil (up/down)**,** reîncărcabil pe 8-biți.
- Un semnal „1" logic la o intrarea CHG_SP va permite SP-ului să-şi modifice conținutul.
- Un semnal „1" logic la o intrarea INC_SP, va permite ca, conținutul indicatorului de stivă să se incrementeze, altfel pentru INC_SP = „0" conținutul SP-ului se va decrementa. Aceste operații de incrementare / decrementare a indicatorului de stivă sunt necesare atunci când sunt salvate valori în memoria stivă (operația *push*) sau când sunt citite (operația *pop*) valori din memoria stivă.
- Pentru o intrae RESET="1" conținutul indicatorului de stivă va fi şters, acest lucru este necesar să se facă în cadrul secvenței de pornire a microprocesorului *µP8*.
- Intrări de date şi de încărcare nu sunt necesare întrucât nu este necesară accesarea aleatoare a memoriei stivă.

2.2 Descrierea în VHDL a indicatorului de stivă

Descrierea în VHDL se va face plecând de la descrierea numărătoarelor din laboratorul 4, modelul din figura 9.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SP is
port (RESET, ...: in std_logic;
...: out std_logic_vector (   7 downto 0 )) ;
end entity SP;

architecture alg of SP is
signal count : unsigned (     7 downto 0 );
begin

process(...)
begin
     if ... = '1' then
```

```
          count <= (others=>'0'); --resetare
     elsif ...'event and ... = '1' then
          if CHG_SP ='1' then
               if ... ='1' then
                    count <= count+1;
               elsif ... ='0' then
                    count <= count-1;
               end if;
          else null;
          end if;
     end if;
               Q <= conv_std_logic_vector(count,8);
end process;
end architecture alg;
```

Figura 4. Cod VHDL corespunzător idicatorului de stivă

Completați secvențele de cod lipsă ținând cont de specificații şi de blocul corespunzător indicatorului de stivă, vezi diagrama din figura 1. Verificați sintaxa şi simulați proiectul.

Un exemplu de stimuli ce pot fi aplicați pentru testarea setului de registre este prezentat în figura 5.

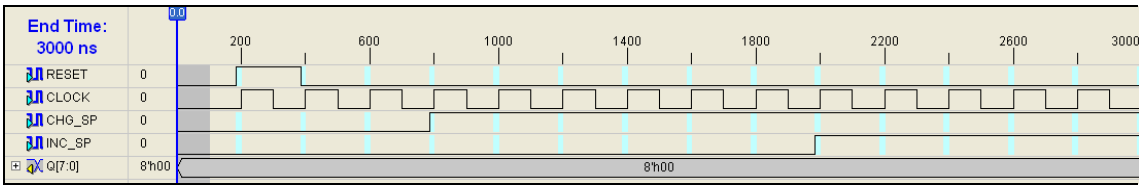

Figura 5. Stimuli pentru verificarea funcționării indicatorului de stivă

3. Selectorul de adrese

3.1 Specificații

Microprocesorul *μP8* împarte memoria externă în trei zone şi anume: 256 octeți memorie de PROGRAM, 256 octeți memorie de DATE şi 256 octeți memorie STIVĂ. Rolul selectorului de adrese este de a activa la un moment dat una din cele trei zone şi de ai trimite o adresă care poate să provină din una din cele trei surse:

- memoria de PROGRAM este adresată de PC (Program Counter);
- memoria STIVĂ este adresată de SP (Stack Pointer);
- memoria de DATE este adresată prin intermediul unei valori provenind din unul din registrele interne.

Blocul de selectare de adrese cu cele trei intrări menționate mai sus este prezentat în Figura 1 (blocul ADDR_MUX).

Specificațiile de proiectare în VHDL ale selectorului de adrese sunt următoarele:
- Prin intermediul unor intrări ADDR_SEL0 şi ADDR_SEL1 se va face selectarea uneia dintre intrările de date (pe 8-biți) din ADDR_MUX şi a celor 7-biți mai semnificativi care se vor concatena cu ceilalți 8 amintiți anterior. Astfel se va obține o adresă validă pe 15-biți.
- În tabelul T.1 sunt prezentate posibilitățile de selectare a celor trei zone din memoria RAM externă. De asemenea în figura 6 este prezentată o „hartă" a memoriei RAM externe.

Tabelul T.1 Împărțirea pe zone a memoriei RAM externe

ADDR_SEL0	ADDR_SEL1	ADDR14...ADDR8	ADDR7...ADDR0	Zona RAM
0	0	1111111(7Fh)	REGB7...REGB0	DATE
0	1	1111110(7Eh)	SP7...SP0	STIVA
1	0	0000001	-	nefolosită
1	1	0000000(00h)	PC7...PC0	PROGRAM

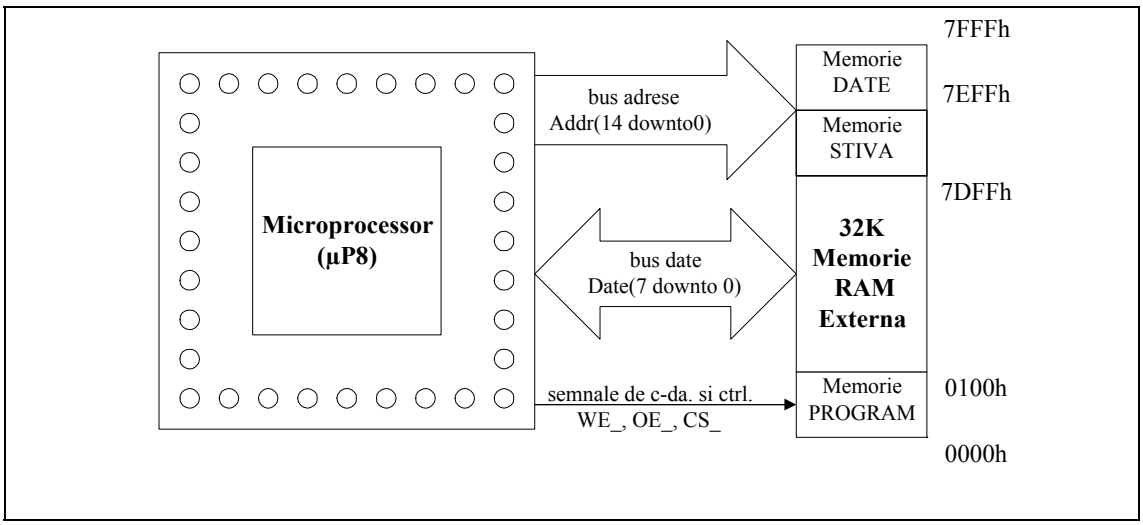

Figura 6 Diagrama bloc a microprocesorul *µP8* şi harta memoriei externe

În Figura 7 sunt prezentate detalii ale circuitului selector de adrese descris cu simboluri în editorul schematic. Se poate observa că, componentele principale sunt două multiplexoare MUX8x2 conectate în serie.

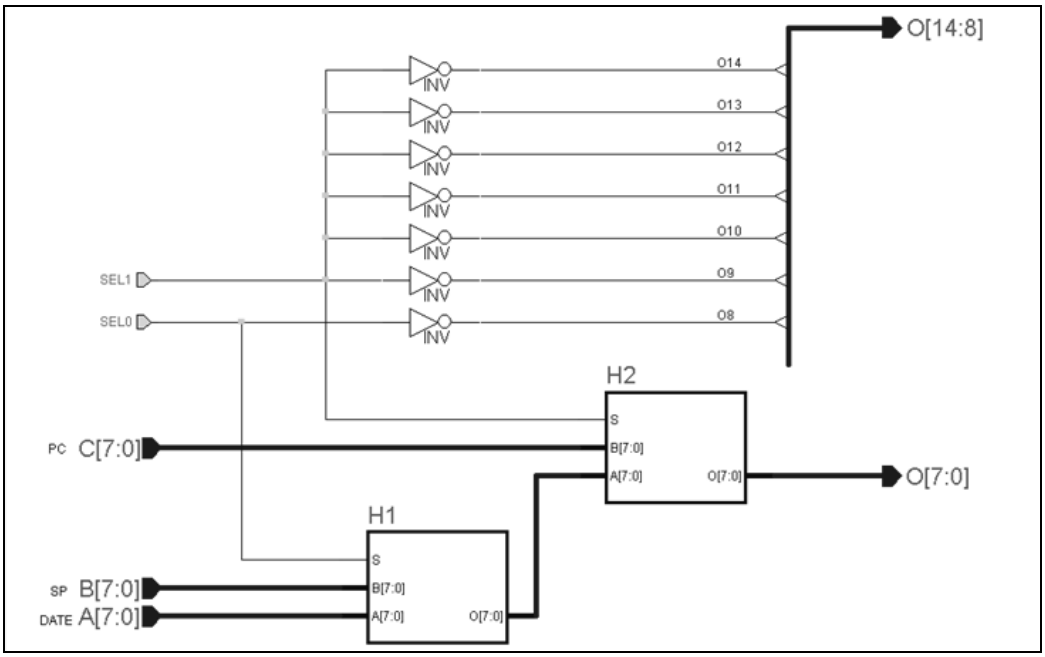

Figura 9 Circuit detailat pentru multiplexorul de adrese

3.2 Descrierea în VHDL a selectorului de adrese

Descrierea în VHDL se va face plecând de la descrierea circuitelor multiplexoare din laboratorul 2, modelul din figura 9.

```
entity adres_sel is
port( ...);
end adres_sel;

architecture Behavioral of adres_sel is

begin
process (...)
variable sel_var: std_logic_vector (1 downto 0);
begin
     sel_var := ...;
     case (sel_var) is
```

```
            when "00" =>  O <= "..."& A;
            when "01" =>  O <= "1111110"& B;
            when "11" =>  O <= "..."& C;
            when others => null;
      end case;
end process;
end Behavioral;
```

Figura 10. Cod VHDL corespunzător selectorului de adrese

Completați secvențele de cod lipsă ținând cont de specificații şi de blocul corespunzător indicatorului de stivă, vezi diagramele din figurile 1, 9. Verificați sintaxa şi simulați proiectul.

Un exemplu de stimuli ce pot fi aplicați pentru testarea setului de registre este prezentat în figura 11.

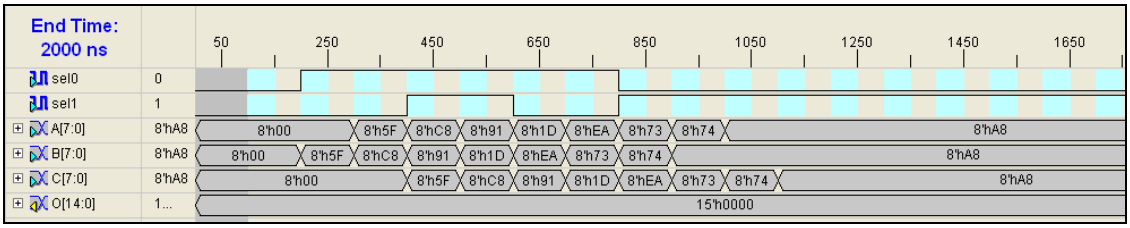

Figura 11. Stimuli pentru verificarea funcționării selectorului de adrese

4. Detectorul de întreruperi

4.1 Specificații

O tranziție a semnalului din „0” logic în „1” logic la pinul INT (întrerupere) al microprocesorului va activa circuitul de sesizare a întreruperilor, care la rândul său va „alerta” microprocesorul obligându-l să-şi schimbe secvența de program.

În figura 12 este prezentată în detaliu descrierea cu simboluri schematice şi porturile corespunzătoare circuitului de sesizare şi înregistrare a unei întreruperi.

Specificațiile de proiectare în VHDL ale detectorului de întreruperi sunt următoarele:

- Un front crescător al semnalului de la o intrarea INT are ca rezultat încărcarea unui registru cu valoarea „1” logic, care se va propaga printr-un alt registru (INTRPT_DFF0, vezi figura 12) şi va apărea la ieşirea acestuia pe următorul front

crescător de clock. În acest fel semnalul de întrerupere extern va fi sincronizat cu semnalul de tact, prevenind astfel apariția stărilor metastabile.

- Un registru suplimentar (IFLAG0) va înregistra apariția unei întreruperi, pentru a indica executarea unei rutine de tratare a întreruperilor.
- Pentru o intrarea SET_IFLAG=”1” registrul IFLAG0 va stoca valoarea „1” logic. Ieşirea IFLAG este folosită pentru a comuta între cele două bancuri ale setului de registre.
- Pentru ambele circuite de detectare şi înregistrare a întreruperilor se vor definii intrările CLR_INTRPT şi CLR_IFLAG active pe „1” logic, astfel registrele vor fi reinițializate cu „0” logic. Reinițializarea acestor registre va avea loc şi în cazul în care se face un reset general al microprocesorului (RESET=”1”).

Ieşirile INTRPT şi IFLAG ale detectorului de întreruperi sunt intrări în decodorul de instrucțiuni.

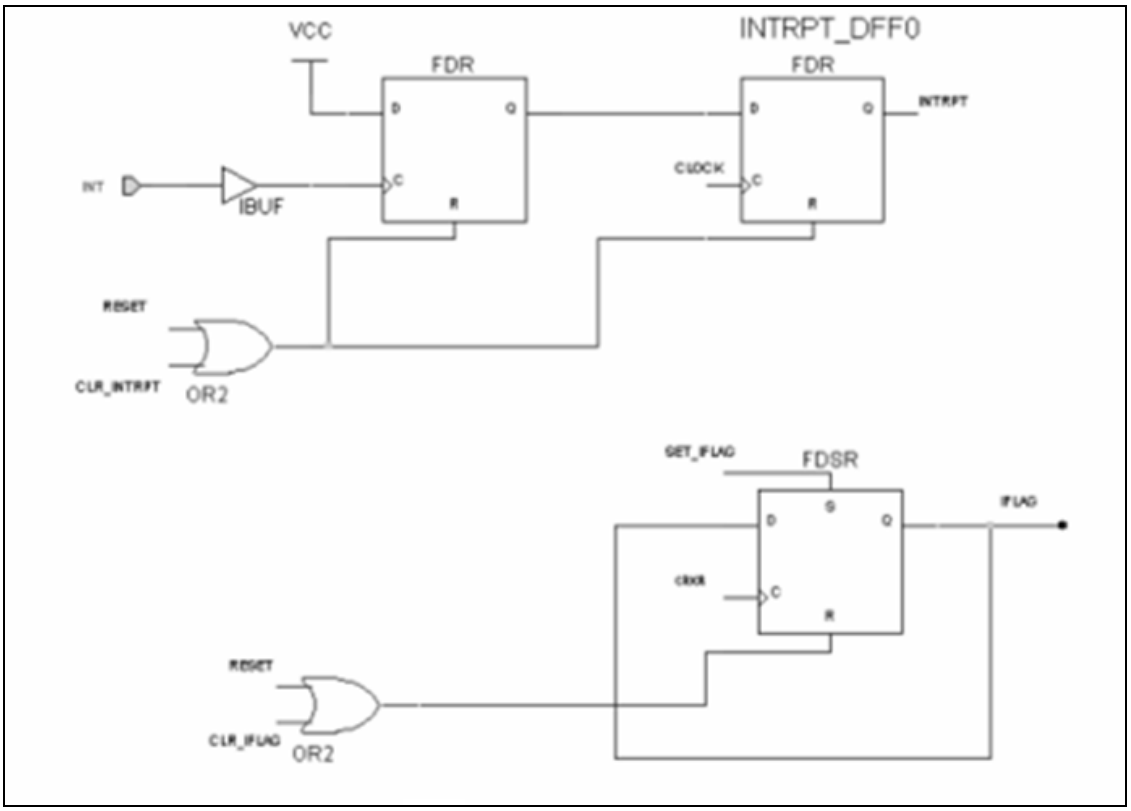

Figura 12. Circuitul de detectare şi înregistrare a întreruperilor

4.2 Descrierea în VHDL a detectorului de întreruperi

Descrierea în VHDL se va face plecând de la descrierea registrelor din laboratorul 4, modelul din figura 6.

```
entity intrp is
port (INT, RESET, CLR_INTRPT, SET_IFLAG, CLR_IFLAG, CLOCK :
in std_logic;
INTRPT, IFLAG: OUT std_logic
);
end intrp;

architecture Behavioral of intrp is
signal intrpt_sig: std_logic;
begin
p1_detectare_intrpt:process (reset, clr_intrpt, int)
begin
     if reset = '1' or ... then
          intrpt_sig <= '0';
     elsif ...'event and int = '1' then
          intrpt_sig <= '1';
     end if;
end process;

p2_sincronizare_intrpt: process (reset, clr_intrpt, clock)
begin
     if ... or clr_intrpt = '1' then
          intrpt <= '0';
     elsif clock'event and clock = '1' then
          intrpt <= intrpt_sig;
     end if;
end process;

p3_inregistrare_intrpt: process (reset, clr_iflag, clock)
begin
     if reset = '1' or ... = '1' then
          ... <= '0';
     elsif clock'event and clock = '1' then
          if set_iflag = ... then
               ... <= ...;
          end if;
```

```
    end if;
end process;

end Behavioral;
```

Figura 13. Cod VHDL corespunzător selectorului de adrese

Codul VHDL din figura 13 cuprinde trei procese, primul pentru detectarea întreruperii, cel de al doilea pentru sincronizarea întreruperii cu semnalul de clock, iar cel de al teilea pentru înregistrarea unei întreruperi care se află ăn curs de tratare.

Completați secvențele de cod lipsă ținând cont de specificații şi de figura 12. Verificați sintaxa, sintetizați şi simulați proiectul.

Un exemplu de stimuli ce pot fi aplicați pentru testarea setului de registre este prezentat în figura 14.

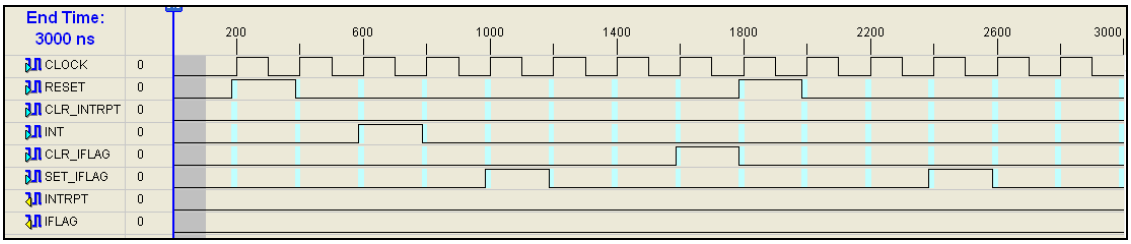

Figura 14. Stimuli pentru verificarea funcționării detectorului de întreruperi