Proiectarea Sistemelor de Transfer Bidirecţional de Date în Baza Reţelelor Petri Hard

Viorica SUDACEVSCHI, Victor ABABII
Technical University of Moldova
svm@mail.utm.md, avv@mail.utm.md

***Abstract* - Lucrarea de faţă este dedicată proiectării sistemelor de transfer bidirecţional de date, unde, ca unitate de control a transferului de date se utilizează reţele Petri hard. Structura reţelei Petri hard corespunde funcţional şi structural modelului de reţea Petri. Expunerea metodologiei de proiectare s-a efectuat în baza unui exemplu de transfer de date. În lucrare sunt descrise: modul de funcţionare a sistemului de transfer de date, modelul reţelei Petri a unităţii de control, metodologia de implementare a reţelei Petri hard şi rezultatul implementării unităţii de control în cod AHDL. Rezultatele experimentale conţin diagramele de timp obţinute în rezultatul simulării unităţii de control în mediul MAX + plus II.**

***Cuvinte cheie* - Reţele Petri, Reţele Petri Hard, unitate de control, transfer bidirecţional de date, cod AHDL.**

I. DESCRIEREA SISTEMULUI DE TRANSFER BIDIRECŢIONAL DE DATE

Una din operaţiile de bază în sistemele de calcul este transferul de date. În acest scop în arhitectura sistemului sunt prevăzute dispozitive speciale pentru organizarea transferului de date (unităţi de control, controlere programabile, etc.). Vom analiza un sistem bidirecţional de transfer de date (figura 1).

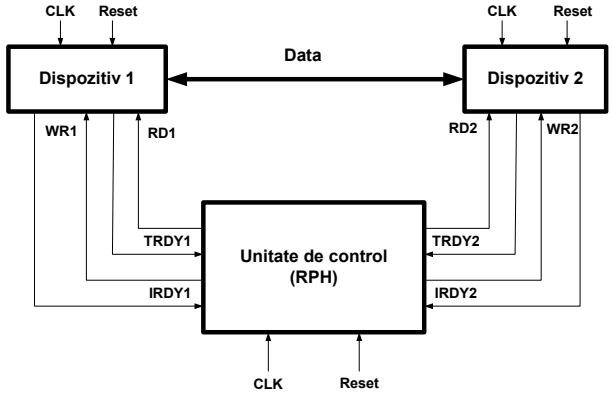

Figura 1. Sistem de transfer bidirecţional de date.

Sistemul este compus din:
- sursele de date (***Dispozitivul 1*** şi ***Dispozitivul 2***);
- unitatea de control realizată în baza reţelei Petri hard (***RPH***);
- magistrala de date bidirecţională ***Data***.

Sistemul este dirijat de semnalele externe ***CLK*** (semnal global de sincronizare) şi ***Reset*** (semnal de resetare a sistemului). Transferul de date este controlat de următoarele semnale: ***IRDY1*** şi ***IRDY2*** (semnale de iniţiere a operaţiei de transfer de date pentru ***Dispozitivul 1*** şi ***Dispozitivul 2***); ***TRDY1 şi TPDY2*** (semnale de confirmare a pregătirii dispozitivelor pentru transferul datelor); ***RD1*** şi ***RD2*** (citirea datelor din dispozitivele respective); ***WR1*** şi ***WR2*** (înscrierea datelor în dispozitivele respective).

Modul de funcţionare a sistemului de transfer bidirecţional de date este următorul: pentru a efectua transferul de date, dispozitivul sursă generează semnalul ***IRDY***. În cazul, în care semnalul ***TRDY*** al dispozitivului receptor este activ, unitatea de control ***RPH (reţea Petri Hard)*** generează semnalul ***RD*** pentru dispozitivul sursă. Acesta efectuează capturarea magistralei şi transmite datele citite. Pentru înscrierea datelor în dispozitivul receptor, ***RPH*** generează pentru acesta semnalul ***WR***. Dacă semnalele ***IRDY*** sunt active pentru ambele dispozitive, prioritatea este acordată ***Dispozitivului 1***. ***Dispozitivul 2*** va începe operaţia de transfer de date numai după finisarea operaţiei de către ***Dispozitivul 1***.

II. MODELAREA UNITĂŢII DE CONPROL ÎN BAZA REŢELELOR PETRI

Modelul reţelei Petri [1, 2] pentru unitatea de control a operaţiei de transfer de date între două dispozitive pe o magistrală bidirecţională este prezentat în figura 2.

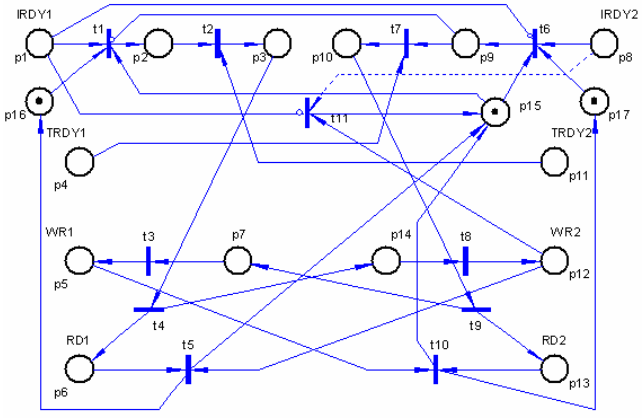

Figura 2 Modelul reţelei Petri pentru unitatea de control a operaţiei de transfer de date.

Specificarea poziţiilor şi tranziţiilor:

p1 (IRDY1)/p8 (IRDY2) – iniţierea operaţiei de transfer de date de către ***Dispozitivul 1/ Dispozitivul 2;***

p2/p9 – operaţia iniţiată este în curs de execuţie;

p3/p10 – operaţia este acceptată de receptor;

p4 (TRDY1)/p11 (TRDY2) – confirmare a pregătirii dispozitivelor pentru transferul datelor;

p5 (WR1)/p12 (WR2) – înscrierea datelor în ***Dispozitivul 1/ Dispozitivul 2***;

p6 (RD1)/p13 (RD2) – citirea datelor din ***Dispozitivul 1/ Dispozitivul 2***;

p7/p14 – sincronizarea operațiilor de citire-înscriere pentru ***Dispozitivul 1/ Dispozitivul 2***;

p16/p17 – permite declanşarea unui nou ciclu de transfer după finisarea celui precedent pentru ***Dispozitivul 1/ Dispozitivul 2***;

p15 – permite acordarea priorității pentru ***Dispozitivul 1***, în cazul când ambele dispozitive solicită transferul de date;

t1/t6 – începutul operației de deservire a ***Dispozitivului 1/ Dispozitivului 2*** şi analiza concurenței la deservire ;

t2/t7 – dispozitivul receptor este gata pentru primirea datelor;

t3/t8 – începutul operației de înscriere a datelor în dispozitivul receptor;

t4/t9 – începutul operației de citire a datelor din dispozitivul emițător;

t5/t10 – sfârşitul operațiilor de citire şi înscriere a datelor;

t11 – acordarea priorității pentru ***Dispozitivul 1***, în cazul inițierii operației de transfer de date de către ambele dispozitive.

Modelul rețelei Petri a fost analizat utilizând produsul program VPNP. În urma analizei s-a stabilit că rețeaua este sigură (1-mărginită), viabilă (indiferent de marcajul care a fost atins pornind din M_0, este posibil ca, în continuare, să fie executată orice tranziție T a rețelei) şi reversibilă (din orice marcaj M se poate ajunge în marcajul inițial M_0).

III. SINTEZA UNITĂȚII DE CONTROL

Pentru efectuarea sintezei unității de control se obține modelul matematic RPH [3, 6] al unității de control, în baza căruia se generează codul AHDL, utilizând produsul program HPNS.

Aplicând algoritmul de translatare a codului XML, de descriere a modelului de rețea Petri, obținem mulțimile $M_0, M_{\max}, P, T, P^{In}, P^{Out}, A^+, A^-, A^I, A^T$ si A^S de descriere matematică a modelului RPH [4].

Codul AHDL, prezentat în figura 3, a fost generat în urma aplicării algoritmului de translatare a modelului matematic în cod AHDL.

```
include "p_obj.inc";
include "t_obj.inc";
subdesign Figura4_2_TD3(
        clock, set, reset : input;
        P1_In : Input;          P4_In : Input;
        P8_In : Input;          P11_In : Input;
        P5_Out : Output;        P6_Out : Output;
        P12_Out : Output;       P13_Out : Output;
            )
  variable
P1 : P_Obj;       P2 : P_Obj;       P3 : P_Obj;
P4 : P_Obj;       P5 : P_Obj;       P6 : P_Obj;
P7 : P_Obj;       P8 : P_Obj;       P9 : P_Obj;
P10 : P_Obj;      P11 : P_Obj;      P12 : P_Obj;
P13 : P_Obj;      P14 : P_Obj;      P15 : P_Obj;
P16 : P_Obj;      P17 : P_Obj;      T1 : T_Obj;
T2 : T_Obj;       T3 : T_Obj;       T4 : T_Obj;
T5 : T_Obj;       T6 : T_Obj;       T7 : T_Obj;
T8 : T_Obj;       T9 : T_Obj;       T10 : T_Obj;
T11 : T_Obj;
  begin
-- Setarea si resetarea  P
  P1.Reset = Reset;   P2.Reset = Reset;
  P3.Reset = Reset;   P4.Reset = Reset;
  P5.Reset = Reset;   P6.Reset = Reset;
  P7.Reset = Reset;   P8.Reset = Reset;
  P9.Reset = Reset;   P10.Reset = Reset;
  P11.Reset = Reset;   P12.Reset = Reset;
  P13.Reset = Reset;   P14.Reset = Reset;
  P15.Set = Set;   P16.Set = Set;
  P17.Set = Set;
-- Resetarea T
  T1.Reset = Reset;   T2.Reset = Reset;
  T3.Reset = Reset;   T4.Reset = Reset;
  T5.Reset = Reset;   T6.Reset = Reset;
  T7.Reset = Reset;   T8.Reset = Reset;
  T9.Reset = Reset;   T10.Reset = Reset;
  T11.Reset = Reset;
-- Sincronizarea P si T
  P1.Clk = Clock;   P2.Clk = Clock;
  P3.Clk = Clock;   P4.Clk = Clock;
  P5.Clk = Clock;   P6.Clk = Clock;
  P7.Clk = Clock;   P8.Clk = Clock;
  P9.Clk = Clock;   P10.Clk = Clock;
  P11.Clk = Clock;   P12.Clk = Clock;
  P13.Clk = Clock;   P14.Clk = Clock;
  P15.Clk = Clock;   P16.Clk = Clock;
  P17.Clk = Clock;   T1.Clk = !Clock;
  T2.Clk = !Clock;   T3.Clk = !Clock;
  T4.Clk = !Clock;   T5.Clk = !Clock;
  T6.Clk = !Clock;   T7.Clk = !Clock;
  T8.Clk = !Clock;   T9.Clk = !Clock;
  T10.Clk = !Clock;   T11.Clk = !Clock;
-- Conectarea intrărilor si ieşirilor P
    P1.Pinc0=P1_In;     P4.Pinc0=P4_In;
    P8.Pinc0=P8_In;     P11.Pinc0=P11_In;
    P5_Out=P5.Pout;     P6_Out=P6.Pout;
    P12_Out=P12.Pout;     P13_Out=P13.Pout;
-- Intrări de incrementare in poziții
    P2.Pinc1=T1.Tout;     P3.Pinc1=T2.Tout;
    P5.Pinc1=T3.Tout;     P6.Pinc1=T4.Tout;
    P7.Pinc1=T9.Tout;     P9.Pinc1=T6.Tout;
    P10.Pinc1=T7.Tout;     P12.Pinc1=T8.Tout;
    P13.Pinc1=T9.Tout;     P14.Pinc1=T4.Tout;
    P15.Pinc1=T5.Tout;     P15.Pinc2=T10.Tout;
    P15.Pinc3=T11.Tout;     P16.Pinc1=T5.Tout;
    P17.Pinc1=T10.Tout;
-- Intrări Test, Inhibiție si Stare în  tranziții
    T1.Tin0=P1.Pout;     T1.Tin1=!P9.Pout;
    T1.Tin2=P15.Pout;     T1.Tin3=P16.Pout;
    T2.Tin0=P2.Pout;     T2.Tin1=P11.Pout;
    T3.Tin0=P7.Pout;     T4.Tin0=P3.Pout;
    T5.Tin0=P6.Pout;     T5.Tin1=P12.Pout;
    T6.Tin0=!P1.Pout;     T6.Tin1=P8.Pout;
    T6.Tin2=P15.Pout;     T6.Tin3=P17.Pout;
    T7.Tin0=P4.Pout;     T7.Tin1=P9.Pout;
    T8.Tin0=P14.Pout;     T9.Tin0=P10.Pout;
    T10.Tin0=P5.Pout;     T10.Tin1=P13.Pout;
    T11.Tin0=!P1.Pout;     T11.Tin1=P8.Pout;
    T11.Tin2=P12.Pout;
-- Intrari de decrementare în poziții
    P1.Pdec0=T1.Tout;     P2.Pdec0=T2.Tout;
    P3.Pdec0=T4.Tout;     P4.Pdec0=T7.Tout;
    P5.Pdec0=T10.Tout;     P6.Pdec0=T5.Tout;
    P7.Pdec0=T3.Tout;     P8.Pdec0=T6.Tout;
    P9.Pdec0=T7.Tout;     P10.Pdec0=T9.Tout;
    P11.Pdec0=T2.Tout;     P12.Pdec0=T5.Tout;
    P12.Pdec1=T11.Tout;     P13.Pdec0=T10.Tout;
    P14.Pdec0=T8.Tout;     P15.Pdec0=T1.Tout;
    P15.Pdec1=T6.Tout;     P16.Pdec0=T1.Tout;
    P17.Pdec0=T6.Tout;
    end;
```

Figura 3. Codul AHDL al unității de control.

IV. REZULTATE EXPERIMENTALE

În rezultatul compilării şi simulării codului AHDL în pachetul de proiectare MAX+PLUS II [5] a fost generat codul de configurare pentru circuitul FPGA FLEX10K (EPF10K10LC84-3). Rezultatele simulării (figura 4) includ diagramele de timp pentru trei cicluri de transfer de date. Au fost simulate situaţiile de iniţiere a transferului de date de către ***Dispozitivul 1*** (0-200ns), ***Dispozitivul 2*** (200-400ns) şi ambele dispozitive (400-700ns). Rezultatele obţinute confirmă corectitudinea funcţionării sistemului de transfer de date.

Complexitatea unităţii de control este de 29 celule logice, ceea ce constituie 5% din complexitatea totală a circuitului EPF10K10LC84-3.

V. CONCLUZII

Una din operaţiile de bază în arhitectura unui sistem de calcul poate fi considerată transferul de date. În cele mai dese cazuri acest transfer de date este bidirecţional, ceia ce aduce la conflicte. Scopul lucrării a fost dezvoltarea metodologiei de implementare a unităţilor de control în modele de reţele Petri hard. Această metodologie s-a bazat pe păstrarea funcţionalităţii şi structurii modelelor de reţele Petri la implementare în dispozitive hardware. Rezultatele obţinute dovedesc corectitudinea funcţionării unităţii de control ceia ce confirmă concepţiile metodologice ale utilizării modelelor de reţele Petri ca metode de descriere formală a unităţilor de control.

În lucrarea de faţă, pentru implementare, s-au utilizat modele de reţele Petri sigure, însă un interes major poate fi utilizarea modelelor de reţele Petri ordinare pentru implementarea unităţilor de control.

REFERINŢE

[1]. J. Peterson, *Petri Net theory and modeling of systems.* New York, 1984.

[2]. T. Murata, *Petri Nets: properties, analysis and applications* Proceedings of IEEE, vol. 77, pp. 541-580, Apr. 1989.

[3]. V. Ababii, V. Sudacevschi, *Modele analitice pentru sisteme CAD în baza reţelelor Petri*, Conferinţa Jubiliară Tehnico-Stiinţifică a Colaboratorilor, Doctoranzilor şi Studenţilor consacrată celei de-a 40-a Aniversări a Doctoranturei UTM, 17-18 Noiembrie 2006, Chişinău 2006.

[4]. V. Ababii, V. Sudacevschi, *Compilator pentru implementarea modelelor de reţele Petri în hard,* Conferinţa Internaţională ICMCS-2007, Chişinău, Moldova, Septembrie 19-21, 2007, Vol. 2, pp. 81-86.

[5]. http://www.altera.com.

[6]. V. Sudacevschi, V. Ababii, V. Negura, *A Hardware Implementation Of Safe Petri Net Models,* Advances in Electrical and Computer Engineering, 2006, vol. 6(13). pp. 54-58.

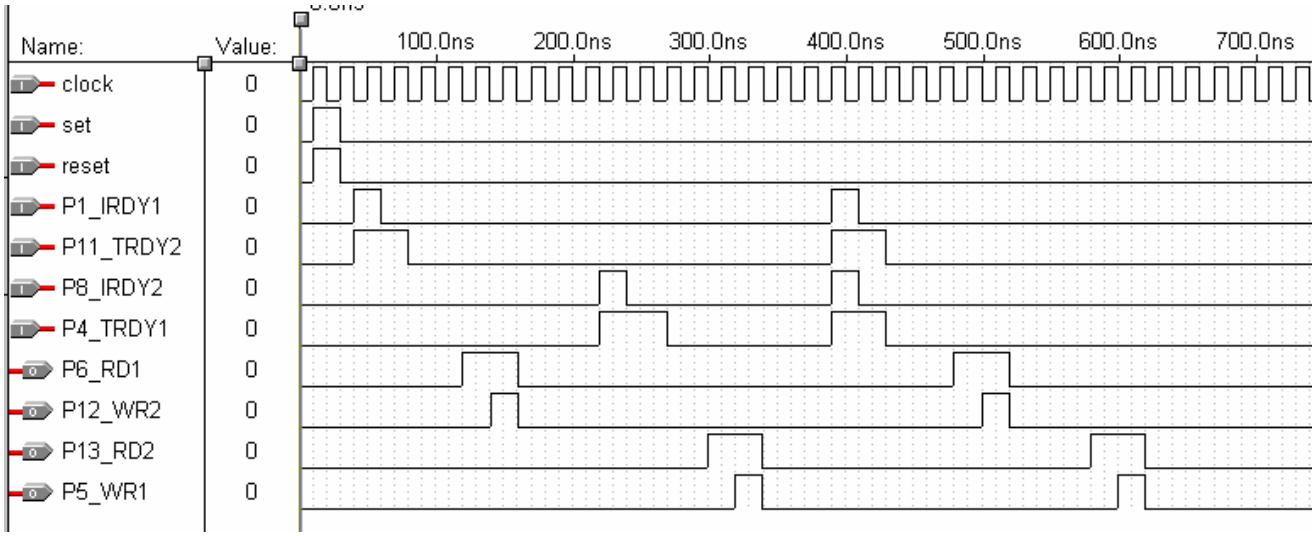

Figura 4. Diagramele de timp ale simulării unităţii de control.