Organizarea Transferului de Date Concurent în Baza Modelelor RPH

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

***Adnotare* — În lucrare se propune o metodă de organizare a transferului de date în baza modelelor de reţele Petri Hard pentru sistemele de calcul cu magistrală comună. S-a propus un sistem de calcul care conţine o mulţime de dispozitive ce comunică între ele prin magistrală comună. Fiecare dispozitiv este asigurat cu un sistem de sincronizare a transferului de date elaborat în baza unui model de reţea Petri Hard. Funcţionalitatea sistemului de control s-a validat utilizând modelul de reţea Petri a acestuia. Sistemul de control se obţine în rezultatul compilării modelului de reţea Petri în cod AHDL care în continuare se utilizează pentru configurarea circuitului FPGA care îndeplineşte funcţia de sistem de control.**

***Cuvinte Cheie* — reţele Petri, XML, AHDL, sistem de control, transfer concurent de date.**

I. INTRODUCERE

Problema gestionării transferului de date concurent prin magistrale comune este foarte actuală, luând în consideraţie arhitectura multe-procesor a sistemelor de calcul [1]. Această problemă persistă şi în arhitecturele mono-procesor, deoarece concurenţa transferului de date apare în momentul accesării simultane a magistralei de date de către diferite dispozitive (procesor, HDD, CD-ROM, interfeţe Ethernet, USB, etc.) [2]. Din aceste considerente, elaborarea unor metode noi de modelare, validare şi implementare a sistemelor de gestionare a procesului de sincronizare a transferului de date concurent este foarte importantă.

În lucrarea de faţă se propune o metodă de modelare, validare şi implementare a sistemelor de sincronizare a transferului de date concurent în baza structurilor de reţele Petri Hard (RPH) [3, 4, 5, 10] care satisfac tuturor criteriilor şi proprietăţilor comportamentale şi structurale ale modelelor de reţele Petri [6].

II. CONCEPTUL PROBLEMEI DE TRANSFER DE DATE CONCURENT PRIN MAGISTRALA COMUNĂ

Structura sistemului de transfer de date concurent este prezentată în figura 1. Sistemul conţine ***Procesorul principal*** care gestionează toate operaţiile de transfer de date prin intermediul magistralelor de ***Control (MC)***, ***Adresă (MA)*** şi ***Date (MD)***. ***Dispozitivele 1 … M*** sunt destinate pentru stocarea sau achiziţia datelor. Fiecare dispozitiv poate funcţiona în două regimuri: master, fiind emiţător de date, şi regim slave, fiind receptor de date. Pentru fiecare dispozitiv sunt prevăzute sisteme de sincronizare a transferului de date concurent ***RPH 1 – RPH M***. Transferul de date este sincronizat de semnalele: ***STBo*** (generat de dispozitivul master) - sincronizează transmiterea datelor, ***STBi*** (recepţionat de dispozitivul slave) - sincronizează primirea datelor, ***ACKo*** (generat de dispozitivul slave) - confirmă primirea datelor, şi ***ACKi*** (recepţionat de dispozitivul master) - confirmă primirea datelor.

Setarea regimului de funcţionare a dispozitivelor este efectuată prin semnalele ***MS*** – regim master şi ***SV*** – regim slave. Adresa dispozitivului pentru setarea regimului este generată prin ***magistrala de adresă (MA)*** de către ***Procesorul principal***.

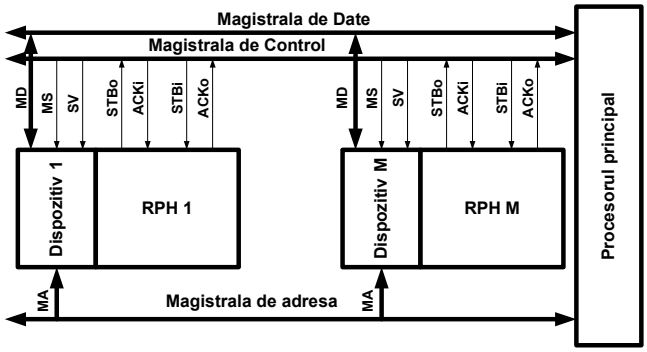

Figura 1. Structura sistemului de transfer de date concurent

III. MDELAREA TRANSFERULUI DE DATE CONCURENT

Verificarea şi validarea corectitudinii funcţionării sistemului de gestionare a transferului de date concurent a fost efectuată în baza modelului reţelei Petri, prezentat în figura 2, unde:

P1 (MS) – regim master pentru dispozitivul respectiv;
P2 (SV) – regim slave pentru dispozitivul respectiv;
P3 (ACKi) - starea semnalului de intrare ACK in blocul de control;
P4 (DRDY) – datele sunt citite din dispozitivul master şi transmise la magistrala de date;
P5 (STBo) – starea semnalului de ieşire STB din blocul de control al dispozitivului master;
P6 – starea blocului de control a dispozitivului master;
P7 (STBi) – starea semnalului de intrare STB în blocul de control;
P8 (WR Data) – semnal de înscriere a datelor în dispozitivul slave;
P9 (ACKo) – starea semnalului de ieşire ACK din blocul de control a dispozitivului slave;
P10 – starea blocului de control a dispozitivului slave.

Validarea modelului rețelei Petri s-a efectuat utilizând produsul program VPNP [9].

În modelul rețelei Petri arcele $T6 \rightarrow P3$ şi $T2 \rightarrow P7$ sunt incluse pentru modelarea sistemului de control. La implementarea sistemului în arhitectură hard aceste arce sunt excluse.

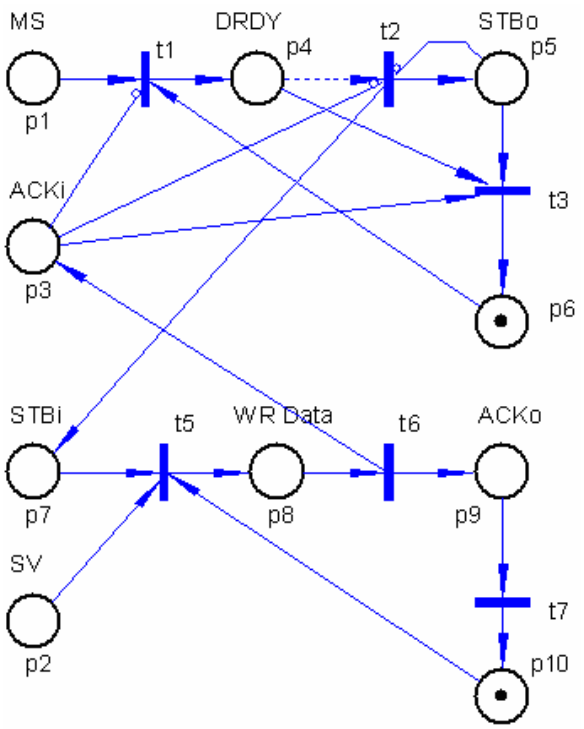

Figura 2. Modelul rețelei Petri

IV. IMPLEMENTAREA SISTEMULUI DE GESTIUNE A TRANSFERULUI DE DATE CONCURENT

Algoritmul de implementare a modelului rețelei Petri pentru gestionarea transferului de date concurent se bazează pe translatarea acestuia din cod XML în cod AHDL, Codul AHDL este utilizat pentru implementarea în hard a sistemului de sincronizare. Această operație este efectuată automat de produsul program, elaborat de autori. Codul AHDL, obținut este prezentat în figura 3.

```
-- Codul AHDL al modelului de Rețea Petri;
include "p_obj.inc";
include "t_obj.inc";
subdesign Figura_2(
        clock, set, reset : input;
        P1_In , P2_In, P3_In, P7_In : Input;
        P4_Out, P5_Out, P8_Out, P9_Out : Output;
              )
  variable
        P1,  P2,  P3,  P4 ,  P5,  P6,  P7,  P8,  P9,  P10 : P_Obj;
        T1,  T2,  T3,  T4,  T5,  T6 : T_Obj;
  begin
-- Codul Setarea si resetarea P
   P1.Reset = Reset;
   P2.Reset = Reset;
   P3.Reset = Reset;
   P4.Reset = Reset;
   P5.Reset = Reset;
   P6.Set = Set;
   P7.Reset = Reset;
   P8.Reset = Reset;
   P9.Reset = Reset;
   P10.Set = Set;
-- Codul resetarea T
   T1.Reset = Reset;
   T2.Reset = Reset;
   T3.Reset = Reset;
   T4.Reset = Reset;
   T5.Reset = Reset;
   T6.Reset = Reset;
-- Codul 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;
   T1.Clk = !Clock;
   T2.Clk = !Clock;
   T3.Clk = !Clock;
   T4.Clk = !Clock;
   T5.Clk = !Clock;
   T6.Clk = !Clock;
    P2.Pinc0=P2_In;
     P3.Pinc0=P3_In;
     P7.Pinc0=P7_In;
     P4_Out=P4.Pout;
     P5_Out=P5.Pout;
     P8_Out=P8.Pout;
     P9_Out=P9.Pout;
-- Codul de conectare a intrărilor de incrementate in poziții P
     P4.Pinc1=T1.Tout;
     P5.Pinc1=T2.Tout;
     P6.Pinc1=T3.Tout;
     P8.Pinc1=T4.Tout;
     P9.Pinc1=T5.Tout;
     P10.Pinc1=T6.Tout;
-- Codul de conectare a intrărilor Test, Inhibitorii si Stare in tranziție T
     T1.Tin0=P1.Pout;
     T1.Tin1=!P3.Pout;
     T1.Tin2=P6.Pout;
     T2.Tin0=!P3.Pout;
     T2.Tin1=P4.Pout;
     T3.Tin0=P3.Pout;
     T3.Tin1=P4.Pout;
     T3.Tin2=P5.Pout;
     T4.Tin0=P2.Pout;
     T4.Tin1=P7.Pout;
     T4.Tin2=P10.Pout;
     T5.Tin0=P8.Pout;
     T6.Tin0=P9.Pout;
-- Codul de conectare a intrărilor de Decrementare a Poziției P
     P1.Pdec0=T1.Tout;
     P2.Pdec0=T4.Tout;
     P3.Pdec0=T3.Tout;
     P4.Pdec0=T3.Tout;
     P5.Pdec0=T3.Tout;
     P6.Pdec0=T1.Tout;
     P7.Pdec0=T4.Tout;
     P8.Pdec0=T5.Tout;
     P9.Pdec0=T6.Tout;
     P10.Pdec0=T4.Tout;
   end;
```

Figura 3. Codul AHDL pentru implementarea modelului rețelei Petri Hard

Funcționalitatea codului se bazează pe utilizarea a două elemente de procesare ***T_Obj.inc*** şi ***P_Obj.inc***. Elementul de procesare ***T_Obj.inc*** este definit de modelul matematic (1) şi respectiv ***P_Obj.inc*** este definit de modelul matematic (2) [7].

$$\boldsymbol{Tout} = \left\{ \prod_{i=1}^{N_j^S} \left(A_{ji}^S \right),\ j = \overline{1, L} \right\} \qquad (1)$$

$$\boldsymbol{Pout} = \begin{cases} 1, \left| \boldsymbol{m}_i = 1 \right. \\ 0, \left| \boldsymbol{m}_i = 0 \right. \end{cases}$$

$$\boldsymbol{m}_i^{k+1} = \begin{cases} 1 \ \left| \ \sum_{j=1}^{L_i^+} (A^+_{ij}) = 1,\ \forall\ \boldsymbol{m}_i^k = 0; \right. \\ 0 \ \left| \ \sum_{j=1}^{L_i^-} (A^-_{ij}) = 1\ \forall \boldsymbol{m}_i^k = 1; \right. \\ \boldsymbol{m}_i^k \ \left| \ \sum_{j=1}^{L_i^+} (A^+_{ij}) = 0\ \&\ \sum_{j=1}^{L_i^-} (A^-_{ij}) = 0; \right. \\ \boldsymbol{m}_i^k \ \left| \ \sum_{j=1}^{L_i^+} (A^+_{ij}) = 1\ \&\ \sum_{j=1}^{L_i^-} (A^-_{ij}) = 1; \right. \end{cases} \quad , \boldsymbol{i} = \overline{1, N} \qquad (2)$$

Specificarea variabilelor: ***Pout*** - valoarea logică a elementului de procesare ***P_Obj.inc***; $\boldsymbol{m}_i^k$ - numărul de marcheri în poziția $\boldsymbol{i}$ în momentul de timp $\boldsymbol{k}$; $\boldsymbol{m}_i^{k+1}$ - numărul de marcheri în poziția $\boldsymbol{i}$ în momentul de timp $\boldsymbol{k+1}$; A_{ij}^+ - intrările de incrementare a poziției $\boldsymbol{i}$; A_{ij}^- - intrările de decrementare a poziției $\boldsymbol{i}$; ***Tout*** - valoarea logică a elementului de procesare tranziție ***T_Obj.inc***; A_{ji}^S - intrările de stare globală pentru tranziția $\boldsymbol{j}$.

V. REZULTATE EXPERIMENTALE

Diagramele de timp obținute în rezultatul simulării modelului rețelei Petri Hard în mediul MAX+PLUS II 10.2 [8] sunt prezentate în figura 4.

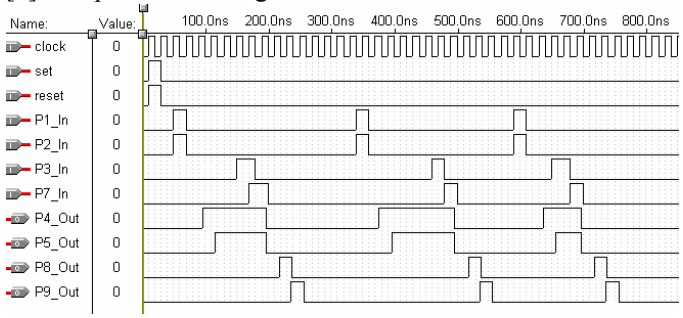

Figura 4. Diagramele de timp pentru modelul rețelei Petri

Diagramele de timp obținute confirmă corectitudinea funcționării sistemului de gestionare a transferului de date concurent. A fost simulată situația de transmitere a trei cuvinte de la dispozitivul master ***MS*** la dispozitivul slave ***SV***.

Complexitatea sistemului este de 16 celule logice, ceea ce constituie 1% din complexitatea totală a circuitului FPGA ***EPF6010ATC100-1***. Numărul celulelor logice coincide cu numărului de elemente funcționale din modelul RPH: 10 poziții ***P*** și 6 tranziții ***T***. Acest fapt demonstrează că complexitatea elementelor funcționale ***P*** și ***T*** este minimă, ceea ce constituie o celulă logică pentru un element funcțional.

VI. CONCLUZII

Rezultatele obținute în lucrarea de față confirmă încă o dată metoda, propusă de autori, de utilizare a modelelor de rețele Petri în calitate de modele de descriere formală a sistemelor de control şi utilizarea acestor modele ca sursă de implementare automată a sistemelor de control în arhitectura hard. Programul elaborat permite de a converti modelul rețelei Petri (format XML) în descriere logică a sistemului de control (format AHDL). Utilizând produse tools Altera codul AHDL obținut este compilat în cod de configurare a dispozitivului FPGA care în continuare se utilizează în calitate de sistem de control.

REFERINȚE

[1] L. Petrea: *Sisteme de prelucrare distribuita*, Editura „Gh.Asachi" Iaşi, 2002.

[2] Baruch, Z. F., *Arhitectura calculatoarelor.* Editura TODESCO, Cluj-Napoca, 2000, ISBN 973-99780-7-x.

[3] V. Ababii, V. Sudacevschi, *Safe Petri Nets Models Mapping into FPGA Using HDL Code.* International Symposium on Systems Theory, SINTES 12, October 20-22, 2005, Craiova, România, pp. 697-699.

[4] V. Ababii, V. Sudacevschi, *FPGA-Based Implementation of Safe Petri Nets Models.* Proceeding of the 4th International Conference on Microelectronics and Computer Science, September 15-17, 2005, Chisinau, Moldova, pp.226-229.

[5] V. Sudacevschi, V. Ababii, V. Negura, *Reconfigurable Control Systems Modeling and Design Based on Petri Nets,* The 9th International Conference on Development and Application Systems, 22-25 May, 2008 Suceava, România, ISSN 1844-5020, pp. 46-49.

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

[7] V. Ababii, V. Sudacevschi, *Modele analitice pentru sisteme CAD în baza rețelelor Petri,* Conferința Jubiliară Tehnico-Ştiințifică a Colaboratorilor, Doctoranzilor şi Studenților consacrată celei de-a 40-a Aniversări a Doctoranturei UTM, 17-18 Noiembrie 2006.

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

[9] E. Gutuleac, C. Bosneaga, A. Reilean. *VPNP – Software Tool for Modeling and Performance Evaluation Using Generalized Stochastic Petri Nets.* 6th International Conference on DAS – 2002, Suceava, România, May 23-25, 2002.

[10] 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.