SINTEZA PROCESOARELOR SPECIALIZATE ÎN BAZA REŢELELOR PETRI HARDWARE FUNCŢIONAL INTERPRETATE

V. Sudacevschi, dr., conf.univ., V. Ababii, dr., conf.univ., E. Guţuleac, prof.univ., dr.hab., M. Podubnîi, drd.
Universitatea Tehnică a Moldovei

INTRODUCERE

Dezvoltarea sistemelor numerice complexe practic nu mai este posibilă fără utilizarea sistemelor de proiectare automată. În prezent majoritatea sistemelor numerice sunt implementate în baza circuitelor ***VLSI*** (Very Large Scale Integration), respectiv, cerinţele impuse faţă de sistemele de proiectare automată au crescut semnificativ, în special faţă de proprietăţile calitative ale sistemelor numerice (performanţă, fiabilitate, convergenţă, etc.). Aceste proprietăţi pot fi asigurate prin includerea a noi metode de verificare funcţională, testare, validare, etc. [1].

Proiectarea cu dispozitive programabile ***FPGA*** (*Field-Programmable Gate Array*) prezintă un interes deosebit pentru dezvoltarea sistemelor numerice complexe de performanţă. Utilizarea circuitelor ***FPGA*** asigură atât o flexibilitate sporită cât şi o performanţă înaltă în procesul de implementare a sistemelor. Circuitele ***FPGA*** constau dintr-o reţea bidimensională de celule logice programabile, interconectate prin comutatoare de rutare, la fel programabile. Domeniile de utilizare a circuitelor ***FPGA*** sunt foarte vaste datorită reducerii semnificative a ciclului de proiectare şi costului relativ redus a acestor dispozitive [2, 3].

Până nu demult, suportul formal pentru sinteza şi modelarea sistemelor digitale era teoria automatelor finite, aplicată în studiul circuitelor logice. Evoluţia rapidă a sistemelor digitale a dus la apariţia unor noi tehnici formale folosite pentru specificarea şi descrierea proceselor, caracteristicile cărora sunt paralelismul, distribuirea fizică şi logică, non-determinismul etc. În prezent descrierea şi modelarea sistemelor complexe este realizată prin intermediul mai multor formalisme matematice, cum ar fi grafurile fluxului de date (***GFD***) şi reţelele Petri (***RP***).

Grafurile fluxului de date sunt utilizate la modelarea sistemelor dominate de date, cum ar fi sistemele transformaţionale, ieşirile cărora sunt determinate de un set de calcule efectuate asupra intrărilor sistemului. ***GFD*** constau dintr-un set de activităţi (transformări) conectate printr-un set de arce care reprezintă fluxul de date.

Pentru reprezentarea sistemelor complexe au fost propuse mai multe modele ***GFD***. Reţelele fluxurilor de date descrise în [4] se utilizează la modelarea sistemelor de procesare a semnalelor. Două cazuri particulare a reţelelor fluxurilor de date, şi anume fluxurile de date sincrone şi fluxurile de date statico-ciclice sunt descrise în [5].

Dezavantajul modelelor ***GFD*** constă în faptul că ele nu conţin nici o informaţie referitoare la metodele sau tehnologiile de implementare sau mapare directă. Din aceste motive, aceste modele sunt utilizate doar în timpul fazei de specificare a sistemelor complexe.

O metodă clasică de verificare funcţională şi validare a sistemelor numerice complexe este formalismul reţelelor Petri [6]. Reţelele Petri au fost definite pentru descrierea sistemelor distribuite în care au loc fenomene de paralelism, sincronizare şi de partajare a resurselor. O reţea Petri reprezintă un caz particular de graf orientat bipartit şi constă din trei tipuri de obiecte. Aceste obiecte sunt poziţiile, tranziţiile şi arcele orientate care conectează poziţiile şi tranziţiile. Arcele sunt etichetate cu ponderile lor care reprezintă valori întregi pozitive. Pentru a studia comportamentul dinamic al sistemului modelat, şi anume stările acestuia şi modificările lor, se utilizează marcajul reţelei. Un marcaj sau o stare atribuie fiecărei poziţii un număr întreg mai mare sau egal cu zero. Marcajul reţelei se poate schimba în conformitate cu regulile de validare şi declanşare a tranziţiilor.

Eficienţa reţelelor Petri în modelarea şi maparea directă a sistemelor numerice este demonstrată în lucrările [7, 8, 9], unde sunt descrise metode şi tehnici de mapare directă a sistemelor de control şi sincronizare fără implicarea fluxurilor de date specifice sistemelor de calcul complexe (procesoare, controlere, etc.).

În lucrarea de faţă se propune o nouă metodă de descriere formală a reţelelor Petri, şi anume reţele Petri funcţional interpretate care asigură modelarea sistemelor de sincronizare şi a fluxurilor de date, asigurând posibilitatea de mapare directă a acestora în arhitecturi reconfigurabile formate din

circuite ***FPGA***. Această abordare permite excluderea mai multor dezavantaje a metodelor de sinteză descrise în lucrările [4, 5, 7, 8, 9] în care lipseşte descrierea şi modelarea fluxurilor de date.

1. REŢELE PETRI FUNCŢIONAL INTERPRETATE

O reţea Petri funcţional interpretată (***RPFI***) este formată din două componente de bază: reţeaua Petri discretă (***RPD***) pentru modelarea procesului de sincronizare şi reţeaua Petri flux de date (***RPF***) pentru modelarea fluxului de date şi a operaţiilor de transformare a datelor (operaţii aritmetice sau logice).

RPD este un 4-tuplu $\left(P^D, T^D, F^D, M_0^D\right)$ [6, 7], unde:

- $P^D = \left\{p_i^D, \forall i = \overline{1, N^D}\right\}$ este o mulţime finită şi nevidă de poziţii discrete;
- $T^D = \left\{t_j^D, \forall j = \overline{1, L^D}\right\}$ este o mulţime finită şi nevidă de tranziţii discrete;
- $F^D \subseteq \left(P^D \times T^D\right) \cup \left(T^D \times P^D\right)$ este o mulţime de arce de conectare a poziţiilor discrete cu tranziţiile discrete şi a tranziţiilor discrete cu poziţiile discrete;
- M_0^D este marcajul iniţial.

Comportamentul dinamic al ***RPD*** este definit de regulile de declanşare a tranziţiilor şi marcajul poziţiilor [6].

RPF este un 3-tuplu $\left(P^F, T^F, F^F\right)$, unde:

- $P^F = \left\{p_i^F, \forall i = \overline{1, N^F}\right\}$ este o mulţime finită de poziţii flux de date;
- $T^F = \left\{t_j^F, \forall j = \overline{1, L^F}\right\}$ este o mulţime finită de tranziţii flux de date;
- $F^F \subseteq \left(P^F \times T^F\right) \cup \left(T^F \times P^F\right)$ este o mulţime de arce de conectare a poziţiilor flux de date cu tranziţiile flux de date şi a tranziţiilor flux de date cu poziţiile flux de date.

Comportamentul dinamic al ***RPF*** este definit de regulile de procesare a datelor şi interacţiunea arcelor (semnalelor) de sincronizare F^{DF} şi stare F^{FD}, unde:

- $F^{DF} \subseteq \left(P^D \times T^F\right)$ este o mulţime de arce de sincronizare de conectare a poziţiilor discrete cu tranziţiile flux de date;
- $F^{FD} \subseteq \left(T^F \times P^D\right)$ este o mulţime de arce de stare de conectare a tranziţiilor flux de date cu poziţiile discrete.

Regulile de validare ale tranziţiilor flux de date sunt definite de starea atributelor arcelor de sincronizare F^{DF}, regulile de funcţionare a elementelor funcţional interpretate (registru, contor, sumator, etc.) şi formulele descriptive ale acestora [10].

2. DESCRIEREA FORMALĂ A MODELELOR DE REŢELE PETRI FUNCŢIONAL INTERPRETATE

Descrierea formală a unei reţele Petri funcţional interpretată este efectuată în limbajul ***XML*** (*Extensible Markup Language*). Regulile de formatare a obiectelor sunt descrise în detaliu în lucrarea [10].

O reţea Petri funcţional interpretată este compusă din trei componente de bază:

- reţeaua Petri discretă;
- elementele funcţionale;
- reţeaua Petri funcţional interpretată.

a) **Descrierea reţelei Petri discrete.**

Formalismul de descriere a modelului reţelei Petri discrete include următoarele obiecte de bază: *TclsNode, TclsTrans, TclsLink* şi *TclsText* care, în dependentă de proprietăţile caracteristice, se divizează în mai multe clase [10].

În continuare sunt prezentate modelele ***XML*** pentru obiectele menţionate.

- Modelul XML al obiectului ***TclsNodeDiscrete*** de descriere a poziţiei discrete:

*<object class="**TclsNodeDiscrete**" **name="p1"** left="95" top="185" width="30" height="30" group="" name="p1" size="osNormal" capacity="9999" markers="1">*
</object>

unde: ***name="p1"*** este numele poziţiei în lista de identificare.

- Modelul XML al obiectului ***TclsTransInstant*** de descriere a tranziţiei discrete netemporizate:

*<object class="**TclsTransInstant**" **name="t1"** left="95" top="267.5" width="30" height="5" group="" size="osNormal" angle="0">*

<guard formula="RefV" Value_Reference="1" Value_Actual="1" />
<priority formula="RefV" Value_Reference="1" Value_Actual="1" />
<speed formula="RefV" Value_Reference="100" Value_Actual="100" />
</object>
unde: ***name="t1"*** este numele tranziţiei în lista de identificare.

- Modelul XML al obiectului ***TclsLinkNormal*** de descriere a arcelor de incrementare, decrementare şi stare a poziţiei:

*<object class="**TclsLinkNormal**" **name="L1"** left="110" top="240.5" width="0" height="26.5" group="" name="L1" **src="p1" dst="t1"**>*
<weight formula="RefV" Value_Reference="1" Value_Actual="1" />
<point x="110" y="214" big="1" />
<point x="110" y="240.5" big="0" />
<point x="110" y="267" big="1" />
</object>
unde: ***name="L1"*** este numele arcului în lista de identificare, ***src="p1"*** este numele obiectului de început al arcului, şi ***dst="t1"*** este numele obiectului de sfârşit al arcului. În cazul dat este un arc de stare şi decrementare care porneşte din poziţia ***p1*** spre tranziţia ***t1***.

- Modelul XML al obiectului ***TclsLinkInhibitor*** de descriere a arcelor de inhibiţie:

*<object class="**TclsLinkIngibitor**" **name="L5"** left="125" top="224" width="27.5" height="91.5" group="" name="L5" **src="p1" dst="t4"**>*
</object>
unde: ***name="L5"*** este numele arcului inhibitor în lista de identificare, ***src="p1"*** este numele obiectului de început al arcului şi ***dst="t4"*** este numele obiectului de sfârşit al arcului.

- Modelul XML al obiectului ***TclsLinkTest*** de descriere a arcelor de test:

*<object class="**TclsLinkTest**" **name="L13"** left="291" top="161" width="3.5" height="82.5" group="" name="L13" **src="p4" dst="t1"**>*
</object>
unde: ***name="L13"*** este numele arcului test în lista de identificare, ***src="p4"*** este numele obiectului de început al arcului şi ***dst="t1"*** este numele obiectului de sfârşit al arcului.

b) **Declararea elementelor funcţionale.**

Declararea elementelor funcţionale este efectuată în obiectul ***Frame*** care include: numele, magistralele şi semnalele de intrare, precum şi magistralele şi semnalele de ieşire.

- Modelul XML al obiectului ***TclsFrame***:

*<object class="**TclsFrame**" **name="Frame1"** left="40" top="15" width="401" height="86" group="" **text="Descrierea Elementelor Functionale"** penstyle="0"> <font name="Arial" size="10" color="000000" style="0" />*
</object>
unde: ***name="Frame1"*** este numele obiectului în lista de identificare, ***text="Descrierea Elementelor Functionale"*** este textul de descriere a obiectului.

- Modelul XML al obiectului ***TclsText*** de descriere a elementelor funcţionale:

** definirea unui registru cu numele **Rg**:*
*<object class="**TclsText**" **name="Text1"** left="47,5" top="22" width="307" height="16" group="" **text="Rg:{Range[8], In:(Data[8], L, CS, OE), Out:(Data[8])};"**> <font name="Arial" size="10" color="000000" style="0" />*
</object>
unde: ***name="Text1"*** este numele obiectului în lista de identificare, ***text="Rg:{Range[8], In:(Data[8], L, CS, OE), Out:(Data[8])};"*** este textul de descriere a obiectului ***Rg***. *Range[8]* – numărul de ranguri (8), *In:(Data[8], L, CS, OE)* – magistrala de date şi semnalele de intrare, *Out:(Data[8])* – magistrala de date şi semnalele de ieşire.
** definirea unui numărător cu numele **Ct**:*
*<object class="**TclsText**" **name="Text2"** left="47,5" top="42" width="304" height="16" group="" **text="Ct:{Range[8], In:(Data[8], L, CS, OE), Out:(Data[8])};"**> <font name="Arial" size="10" color="000000" style="0" />*
</object>
unde: *Range[8]* – numărul de ranguri (8), *In:(Data[8], L, CS, OE)* – magistrala de date şi semnalele de intrare, *Out:(Data[8])* – magistrala de date şi semnalele de ieşire.
** definirea unui sumator cu numele **SM**:*
*<object class="**TclsText**" **name="Text3"** left="47,5" top="62" width="310" height="16" group="" **text="SM:{Range[8], In:(Data1[8], Data2[8]), Out:(Data[8])};"**> <font name="Arial" size="10" color="000000" style="0" />*
</object>
unde: *Range[8]* – numărul de ranguri (8), *In:(Data1[8], Data2[8])* – magistrale de date de intrare, *Out:(Data[8])* – magistrala de date de ieşire.
** definirea unei unităţi logice aritmetice (**ALU**):*
*<object class="**TclsText**" **name="Text4"** left="47,5" top="82" width="389" height="16" group="" **text="ALU:{Range[8], In:(Data1[8], Data2[8], Cop[4]), Out:(Data[8], Owr)};"**> <font name="Arial" size="10" color="000000" style="0" />*
</object>

unde: *Range[8]* – numărul de ranguri (8), *In:(Data1[8], Data2[8], Cop[4]),* – magistrale de date de intrare şi magistrala pentru codul operaţiei, *Out:(Data[8], Owr)* – magistrala de date şi semnale de ieşire.

c) Descrierea reţelei Petri funcţional interpretată.

- Modelul XML de descriere a unui registru ***Reg1*** de tip ***Rg***:

*<object class="**TclsText**" **name="b1"** left="295" top="235" width="54" height="16" group="" **text="Reg1(Rg)"**> <font name="Arial" size="10" color="000000" style="0" />*
</object>

- Modelul XML al obiectului ***TclsNodeDataFlow*** de descriere a poziţiei funcţional interpretate flux de date:

*<object class="**TclsNodeDataFlow**" **name="b1"** left="275" top="205" width="30" height="30" group="" name="b1" size="osNormal" boundmin="0" boundmax="9999" level="8">*
<cost formula="RefV" Value_Reference="1" Value_Actual="1" />
</object>

unde: ***name="b1"*** este numele obiectului în lista de identificare.

Legătura dintre obiectul de descriere a registrului *class="**TclsText**"* şi obiectul de descriere funcţională flux de date *class="**TclsNodeDataFlow**"* este efectuată prin numele comun *"**b1**"*.

- Modelul XML al obiectului ***TclsTransDataFlow*** de descriere a tranziţiei funcţional interpretate flux de date:

*<object **class="TclsTransDataFlow" name="u1"** left="275" top="285" width="30" height="10" group="" size="osNormal" angle="0">*
<guard formula="RefV" Value_Reference="1" Value_Actual="1" />
<priority formula="RefV" Value_Reference="1" Value_Actual="1" />
<expectance formula="RefV" Value_Reference="1" Value_Actual="1" />
<dispersion formula="RefV" Value_Reference="1" Value_Actual="1" />
</object>

unde: ***name="u1"*** este numele obiectului în lista de identificare.

- Modelul XML al obiectului ***TclsLinkDataFlow*** de descriere a arcelor flux de date:

*<object class="**TclsLinkDataFlow**" **name="L7"** left="290" top="259,5" width="0" height="25,5" group="" name="L7" **src="b1" dst="u1">***
<weight formula="RefV" Value_Reference="1" Value_Actual="1" />
<point x="290" y="234" big="1" />
<point x="290" y="259,5" big="0" />
<point x="290" y="285" big="1" />
</object>

unde: ***name="L7"*** este numele obiectului în lista de identificare, ***src="b1"*** este începutul arcului flux de date din poziţia ***"b1"*** şi ***dst="u1"*** sfârşitul arcului în tranziţia ***"u1"***.

3. REŢELE PETRI HARDWARE FUNCŢIONAL INTERPRETATE

O reţea Petri Hardware funcţional interpretată (***RPHFI***) prezintă o mulţime de elemente de procesare concurentă a datelor formată din:

- elemente de procesare hardware $\mathbf{P}^{HD} \equiv \boldsymbol{P}^{D}$ poziţii discrete [7, 8];

- elemente de procesare hardware $\mathbf{T}^{HD} \equiv \boldsymbol{T}^{D}$ tranziţii discrete netemporizate [7, 8];

- conexiuni $\boldsymbol{C}^{+} \equiv \left(\boldsymbol{T}^{D} \times \boldsymbol{P}^{D}\right)$ de incrementare a numărului de jetoane în elementul de procesare poziţie;

- conexiuni $\boldsymbol{C}^{-} \equiv \left(\boldsymbol{P}^{D} \times \boldsymbol{T}^{D}\right)$ de decrementare a numărului de jetoane în elementul de procesare poziţie;

- conexiuni $\boldsymbol{C}^{S} \equiv \left(\boldsymbol{P}^{D} \times \boldsymbol{T}^{D}\right)$ de stare care determină condiţia de validare a elementelor de procesare tranziţie;

- conexiuni $\boldsymbol{C}^{T} \equiv \left(\boldsymbol{P}^{D} \times \boldsymbol{T}^{D}\right)$ de test care determină condiţia de validare a elementelor de procesare tranziţie;

- conexiuni $\boldsymbol{C}^{I} \equiv \left(\boldsymbol{P}^{D} \times \boldsymbol{T}^{D}\right)$ de inhibiţie care determină condiţia de validare a elementului de procesare tranziţie în cazul absenţei jetoanelor în poziţie;

- elemente de procesare hardware $\mathbf{P}^{HF} \equiv \boldsymbol{P}^{F}$ care reprezintă poziţii funcţional interpretate flux de date de tipul registrelor, numărătoarelor, sumatoarelor, ALU, etc.;

- elemente de procesare hardware $\mathbf{T}^{HF} \equiv \boldsymbol{T}^{F}$ care reprezintă tranziţii funcţional interpretate flux de date pentru validarea fluxului de date dintre registre, numărătoare, sumatoare, ALU, etc.;

- magistrale de date $\boldsymbol{B}^{HF} \equiv \left(\boldsymbol{P}^{F} \times \boldsymbol{T}^{F}\right) \cup \left(\boldsymbol{T}^{F} \times \boldsymbol{P}^{F}\right)$ dintre poziţiile

$\mathbf{P}^{HF}$ şi tranziţiile $\mathbf{T}^{HF}$, şi tranziţiile $\mathbf{T}^{HF}$ şi poziţii $\mathbf{P}^{HF}$;

- conexiuni de sincronizare $C^{DF} \equiv \left(P^D \times T^F \right)$ a tranziţiilor $\mathbf{T}^{HF}$;

- conexiuni de stare $C^{FD} \equiv \left(T^F \times P^D \right)$ a poziţiilor $\mathbf{P}^{HF}$.

4. ALGORITMUL DE SINTEZĂ A PROCESOARELOR SPECIALIZATE ÎN BAZA RPHFI

Algoritmul de sinteză a procesoarelor specializate în baza Reţelelor Petri Hardware Funcţional Interpretate (***RPHFI***) sunt prezentate în diagrama bloc din Figura 1.

Iniţial are loc descrierea modelului de reţea Petri [10] cu elemente funcţional interpretate prin intermediul produsului program ***VPNP*** *(Visual Petri Net* +). Astfel sunt efectuate următoarele operaţii:

- introducerea modelului reţelei Petri pentru sincronizare (***RPS***);
- definirea elementelor funcţionale (***DEF***);
- introducerea modelului reţelei Petri pentru descrierea elementelor funcţionale (***RPFI***);

Pentru efectuarea mai eficientă a acestor operaţii se utilizează baza de date cu modele de reţele Petri funcţional interpretate (***MRPFI***).

După procesarea datelor introduse se obţine codul ***XML*** al modelului de reţea Petri funcţional interpretată.

Etapa ***CRPH*** reprezintă compilarea modelului de reţea Petri funcţional interpretată în cod ***HDL*** (Hardware Description Language)**.** În acest scop se îndeplinesc următoarele operaţii:

- identificarea modelului de reţea Petri care îndeplineşte funcţii de sincronizare (***IRPS***);
- identificarea elementelor funcţional interpretate (***IEFI***);
- identificarea modelului de reţea Petri care modelează fluxul de date (***IRPFI***);
- compilarea în cod ***HDL*** a obiectelor identificate din modelul de reţea Petri funcţional interpretate (***CHDL***);

Pentru etapa de compilare se utilizează bazele de date care conţin elemente funcţional interpretate (***EFI***) şi modele de reţele Petri hardware funcţional interpretate (***RPHFI***).

Fişierul rezultant ***HDL*** este procesat în mediul de proiectare ***Quartus II*** [2].

Verificarea şi testarea funcţională a codului de configurare generat pentru reţeaua Petri hardware funcţional interpretată este efectuată pe kit-ul de dezvoltare ***Altera DE0*** [11].

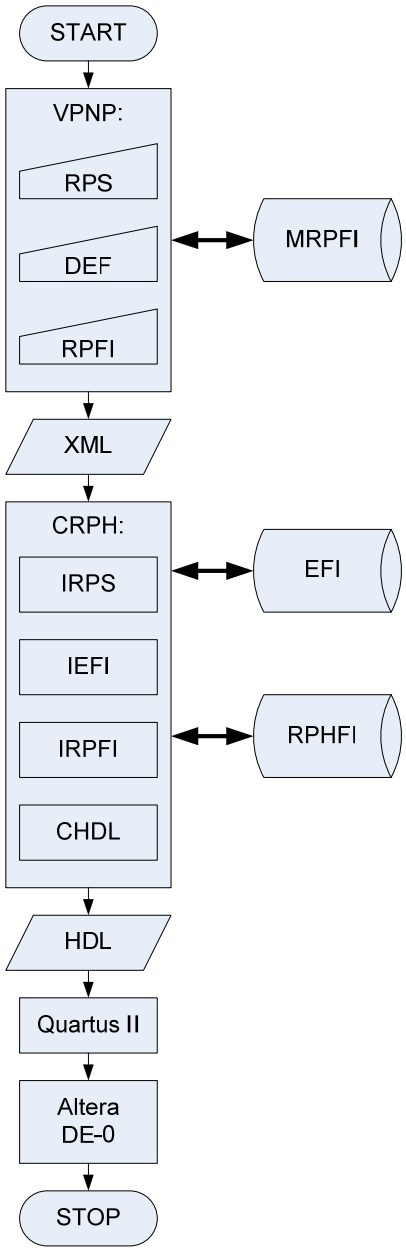

Figura 1. Algoritmul de sinteză a procesoarelor specializate în baza RPHFI.

5. EXEMPLU DE SINTEZĂ ÎN BAZA RPHFI

Sinteza procesorului specializat în baza modelelor ***RPHFI*** se bazează pe utilizarea următoarelor componente:

- elemente de procesare $\mathbf{P}^{HD}$ şi $\mathbf{T}^{HD}$ pentru realizarea blocului de sincronizare;
- elemente funcţionale $\mathbf{P}^{HF}$ pentru memorarea datelor (registre, numărătoare) şi pentru efectuarea operaţiilor aritmetico-logice (sumatoare, ***ALU***).

În Figura 2 este prezentată structura unui procesor specializat, unde: ***B_Syn*** - blocul de sincronizare; ***EF*** - elemente funcţionale (regiştrii ***Rg1***, ***Rg2*** şi ***Rg3***) şi un sumator (***Summ***); **1**, **2** şi **3** – semnale de sincronizare $\boldsymbol{C}^{DF}$ generate de blocul de sincronizare ***B_Syn***; **4**, **5** şi **6** – magistrale de date $\boldsymbol{B}^{F}$.

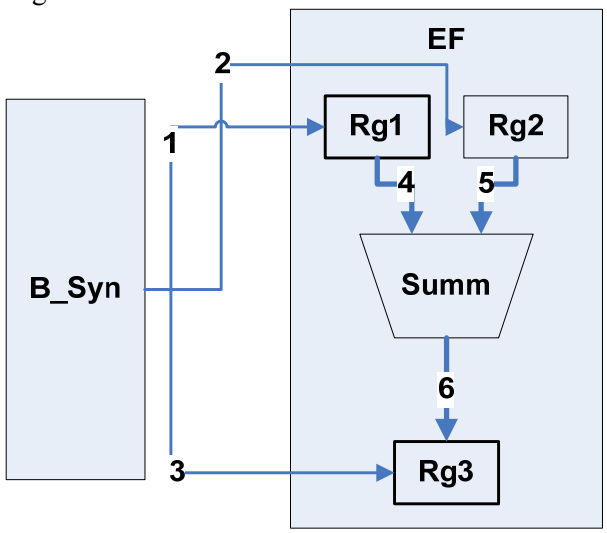

Figura 2. Structura procesorului specializat.

În Figura 3 este prezentat modelul de reţea Petri funcţional interpretată pentru sinteza procesorului specializat.

Procesorul specializat include următoarele elemente funcţionale:

- un registru ***Rg*** pe 8 ranguri, cu magistrala de date de intrare şi ieşire pe 8 ranguri;
- semnale de sincronizare (***CLK*** – încărcare paralelă, ***CLRN*** - resetare, ***PRN*** – setare);
- un sumator ***SM*** pe 8 ranguri, cu două magistrale de date de intrare şi una de ieşire pe 8 ranguri, un semnal de transport de intrare ***CIN*** din rangurile inferioare, un semnal de transport de ieşire ***COUT*** în rangurile superioare.

Descrierea elementelor funcţionale poate include registre, numărătoare, sumatoare, unităţi aritmetico-logice, etc.

- ***Reţeaua pentru Sincronizare*** care include: trei poziţii (**p1, p2, p3**) şi trei tranziţii (**t1, t2, t3**). Prezenţa marcherului în poziţia respectivă determină generarea semnalelor de sincronizare.

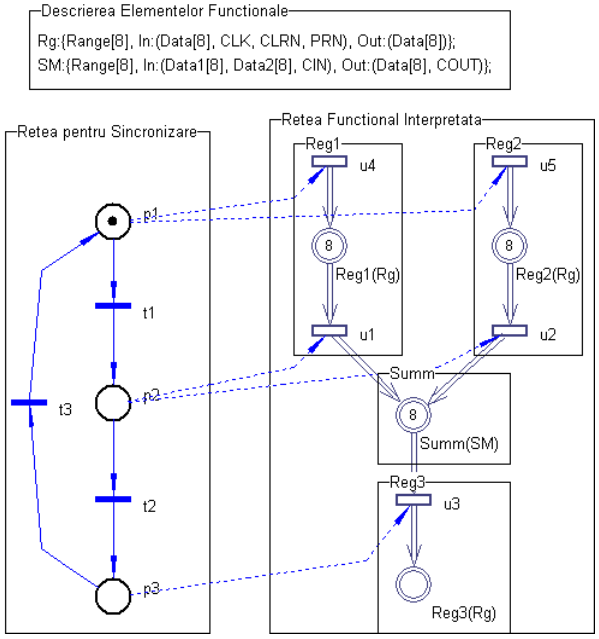

Figura 3. Exemplu ***RPFD*** pentru sinteza procesorului specializat.

- ***Reţeaua Funcţional Interpretată*** care include: trei registre (**Reg1, Reg2, Reg3**) şi un sumator (**Summ**).

Rezultatele prezentate în această lucrare au fost obţinute în cadrul proiectului 15.817.02.28A „Modele, metode şi interfeţe pentru conducerea şi optimizarea sistemelor de fabricaţie inteligente”.

6. CONCLUZII

În lucrare a fost abordată problema proiectării procesoarelor specializate în baza modelelor de reţele Petri. Pentru aceasta a fost propusă o extindere a reţelelor Petri, şi anume reţele Petri funcţional interpretate (***RPFI***) care permit modelarea fluxurilor de date şi a elementelor funcţionale specifice procesoarelor specializate prezente în sistemele de fabricaţie inteligente. Pentru maparea directă în circuite ***FPGA*** a fost elaborată reţeaua Petri Hardware funcţional interpretată (***RPHFI***). Procesul de sinteză a procesoarelor specializate include următoarele etape: descrierea modelului ***RPFI*** prin intermediul produsului program ***VPNP,*** generarea obiectelor ***XML*** necesare pentru modelare, obţinerea modelului ***RPHFI*** al procesorului specializat şi generarea codului ***HDL*** pentru acest model,

verificarea şi testarea funcţională a codului de configurare pe kit-ul de dezvoltare ***Altera DE0.***

Rezultatele ştiinţifice obţinute în această lucrare vor fu utilizate în continuare pentru proiectarea şi implementarea procesoarelor specializate cu aplicarea în sisteme multi-agent [12, 14, 15], sisteme multi-robot [13, 16] şi în reţele de senzori [17] care fac parte din procese de fabricaţie inteligente.

Bibliografie

1. Baruch Z.F. *Contribuţii la proiectarea asistată, Teză de doctorat, Cluj-Napoca, 1999, 247p.*

2. https://www.altera.com *(Accesat 15.08.2015).*

3. https://www.xilinx.com *(Accesat 17.08.2015).*

4. Lee E. A., Parks T. *Dataflow Process Networks. In: Proceedings of IEEE, May 1995, vol. 83, p. 773-799.*

5. Parks T., Pino J. L., Lee E. A. *A Comparison of Synchronous and Cyclo-Static Dataflow, In: Proceedings of the 29th Asilomar Conference on Signals, Systems and Computers, 1995, p. 204-210.*

6. Peterson J.L. *Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1981.*

7. Sudacevschi V. *Sinteza structurilor de procesare concurentă a datelor, Teză de doctor în tehnică, UTM, Chişinău, 2009, 165 p.*

8. Sudacevschi V., Ababii V. *Modelling and Synthezis of Real-Time Control Systems Bazed on Hardware Timed Petri Nets. Buletinul Institutului Politehnic Din Iaşi, Publicat de Universitatea Tehnică „Gheorghe Asachi" din Iaşi, Secţia „Electrotehnică. Energetică. Electronică", Tomul LIX (LXIII), Fasc. 4, 2013, pp. 161-172.*

9. Ababii V. Sudacevschi V., Podubnîi M. Cojuhari I. *Real-time reconfiguration of distributed control system based on hard Petri nets. International Conference on development and application systems 12th Edition, May 15-17, 2014, Suceava, Romania, pp. 21-24, ISSN 1844-5039. DOI: 10.1109/DAAS.2014.6842421.*

10. Guţuleac E., Boşneaga C., Railean A. *VPNP-Software tool for modeling and performance evaluation using generalized stochastic Petri nets. Proceedings of the 6-th International Conference on DAS-2002, 23-25 May 2002, Suceava, România, p. 243-248, ISBN 973-98670-9-X.*

11. http://www.terasic.com.tw*(Accesat 11.08.2015).*

12. Ababii V. Sudacevschi V., Podubnîi M., Negară E. *Mnogoagentnaya assotziativnaya vychislitel'naya sistema // Molodoj uchennyj. 2015. №16 (96). S. 30-36. ISSN: 2072-0297 (Print). ISSN: 2077-8295 (Online).*

13. Ababii V. Sudacevschi V., Cojuhari I., Podubnîi M., Negară E. *Upravlenie dvizheniem avtonomnogo mobil'nogo robota v otnositel'noi sisteme coordinat gravitatzionnogo i magnitnogo polea Zemli // Molodoj uchennyj. 2015. №17 (97). S. 69-74. ISSN: 2072-0297 (Print). ISSN: 2077-8295 (Online).*

14. Podubnîi M., Ababii V., Sudacevschi V., Safonov G. *Mnogoagentnaya sistema na baze NI LabVIEW. Mezhdunarodnyj simpozium „Kompyuternye izmeritel'nye tehnologii - 2015, Moskva, 3.04.2015., pp. 23-25, ISBN 978-5-97060-324-6.*

15. Ababii V. Sudacevschi V., Podubnîi M., Moroshan I. *Assotziativnaya vychislitel'naya set' dlya resheniya slozhnyx zadach na baze ustrojstv s ogranichennymi vychislitel'nymi resursami. Proceeding of the 3rd International Conference "Computational Intelligence (Results, Problems and Perspectives)2015", ComInt-2015, May 12-15, 2015, Cherkasy, Ukraine, pp. 48-49.*

16. Podubnîi M., Ababii V., Sudacevschi V., Safonov G. *Systema mobil'nyx robotov dlya poiska istochnikov ioniziruyusshej radiatzii. Proceeding of the 3rd International Conference "Computational Intelligence (Results, Problems and Perspectives) 2015", ComInt-2015, May 12-15, 2015, Cherkasy, Ukraine, pp. 163-164.*

17. Ababii V., Sudacevschi V., Podubnii M., Roshchiupkin O. *Reţea de Senzori cu Arhitectură Reconfigurabilă. Proceeding of the 5th International Conference "Telecommunications, Electronics and Informatics", ICTEI-2015, may 20-23, 2015, Chişinău, Moldova, pp. 296-299, ISBN 978-9975-45-377-6.*

Recomandat spre publicare: 24.09.2015.