Verificarea Funcţională a Sistemului Multiprocesor de Bord al Satelitului

Nicolae SECRIERU, Emilian GUŢULEAC, Roman NICU
Universitatea Tehnică a Moldovei
nsecrieru@gmail.com, egutuleac@mail.utm.md, roman@rgg.md

***Abstract* — În lucrare sunt prezentate aspecte de elaborare şi verificare funcţională prin reţele Petri temporizate a sistemului multiprocesor de bord al unui microsatelit.**

***Index Terms* — sistem multiprocesor de bord, procesare concurentă în timp real, reţele Petri, verificare funcţională.**

I. INTRODUCERE

Arhitectura unui sistem de calcul de bord al unui satelit este determinată de misiunea care va fi îndeplinită şi performanţele lui conform sarcinii tehnice [4, 7].

În cazul unui microsatelit (MS), sarcinile ale unui astfel de sistem de calcul sunt individualizate in fiecare caz, dintre care cele principale: orientarea în spaţiu; controlul telemetriei; controlul alimentarii; comunicarea cu staţiile de monitorizare de pe pământ.

Un alt aspect al dezvoltării sistem de calcul de bord constă în necesitatea de a procesa mai multe fluxuri de date concurente şi controlul aplicaţiilor în timp real, ceea ce poate să creeze conflicte care la rândul său poate să ducă la ieşirea din funcţie a întregului sistem. Situaţia fiind şi mai complicată în cazul unui satelit în cazul apariţiei unor defecte în timpul funcţionării sistemului, deoarece practic până când satelitul nu este lansat este imposibil de simulat funcţionarea sistemelor lui în condiţii spaţiului cosmic, de asemenea, există pericolul apariţia defectărilor în urma lansării satelitului, când întregul sistem este supus supraîncărcării din cauza forţelor fizice.

Concepţia unui astfel de sistem de calcul bazat pe paralelism şi cooperarea componentelor de la specificaţia sa iniţială până la verificarea faptului că elaborarea propusă *satisface* cerinţelor de performanţă, necesită un mediu de dezvoltare. În acest context, este de dorit ca într-un astfel de mediu să fie folosit *un singur formalism*, care să aibă capabilităţi suficiente pentru descrierea acţiunilor proceselor cooperante, protocoalelor de comunicaţie, construirea şi validarea modelului sistemului analizat [2].

Reţelele Petri (RP) şi extensiile lor [3, 6] pot fi considerate modele stare-tranziţie care permit modelarea, verificarea funcţională şi evaluarea performanţelor proceselor componente ale unui sistem cu evenimente discrete.

În continuare, succint sunt prezentate unele aspecte de elaborare şi verificare funcţională a sistemului multi-procesor de bord al unui microsatelit prin reţele Petri temporizate.

II. ELABORAREA SISTEMULUI MULTIPROCESOR

Elaborarea unui sistem multiprocesor de bord (SMB) include: definirea cerinţelor produsului; elaborarea specificaţiilor funcţionale; crearea proiectului arhitectural; verificarea funcţională şi evaluarea performanţelor; furnizarea versiunii arhitecturale elaborate.

În Fig. 1 este prezentată diagrama etapelor de proiectare şi dezvoltare a SMB al MS.

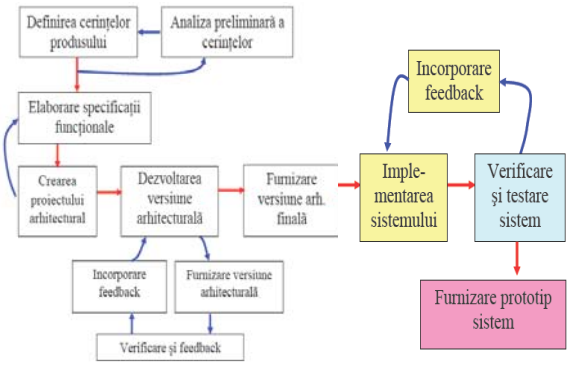

Fig.1. Diagrama etapelor de proiectare şi dezvoltare a SMB.

În baza cerinţelor specificate la începutul proiectului a fost dezvoltat modelul de cerinţe faţă de SMB, folosind formalismul UML [1].

În Fig. 2 este prezentată diagrama generală a modelului cerinţelor faţă de SMB dezvoltat.

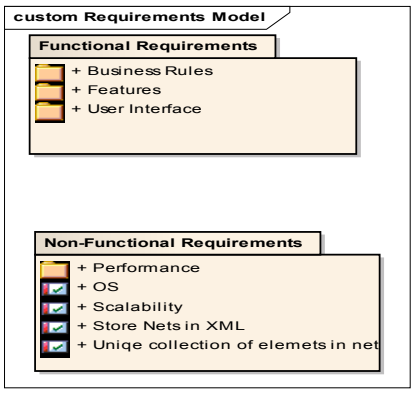

Fig. 2. Modelul cerinţelor.

Cerinţele funcţionale se referă la funcţionalităţile: intrări şi ieşiri (interfaţa cu utilizatorul şi mediul); funcţii şi constrângeri de timp ale sistemului software şi modul de realizare a lor, precum şi specificarea unor algoritmi de prelucrare a datelor.

La cerinţele nefuncţionale se refera cele care nu ţin de funcţionalităţile de bază ale sistemului. Acestea pot să se refere la: performanţele calculatorului; costuri; consumul de

energie ; dimensiunea fizică şi greutate; fiabilitate, siguranţă în funcţionare, mentenabilitate hardware şi software, securitatea datelor, sistemul de operare, cromatica interfeţei grafice, scalabilitatea aplicaţiei, persistenţa datelor, portabilitatea datelor / aplicaţiei, etc.

Există o strânsă legătură între arhitectura sistemului de calcul, nivelul de performanţă şi software-ul care urmează să-i pună în valoare caracteristicile funcţionale.

Pe baza disponibilităţii tehnice şi structurale ale acestor elemente şi a combinaţiilor arhitecturale dintre ele, se obţine mulţimea modelelor, imaginabile teoretic, ale sistemelor de calcul. În realitate, doar o submulţime destul de restrânsă este viabilă sub aspectele fezabilităţii tehnologice şi a acceptabilităţii în raport cu criteriile de performanţă, utilitate şi cost. Un principiu de bază pentru concepţia arhitecturii unui sistem de calcul în scopul îmbunătăţirii calităţii şi al creşterii performanţelor sale este paralelismul şi cooperarea resurselor de calcul concurente folosite, care îşi găseşte aplicabilitatea atât la nivel hardware, cât şi software [4, 6, 7].

Diferenţele între diversele arhitecturi ale sistemelor de calcul de bord, ce se întâlnesc în literatură, pot fi cuantificate cu următorii parametri [6]:

(a) prima cantitate este dimensiunea masivului de date memorat, numărul de procesoare şi puterea de prelucrare a procesorului individual, produsul acestora determinând astfel, până la un punct anumit, puterea de procesare a întregului sistem;

(b) complexitatea suportului de comunicaţie, care va determina flexibilitatea sistemului şi, de aici, dacă puterea de procesare obţinută prin multiplicare poate fi folosită de o clasă mare de probleme;

(c) distribuţia controlului sistemului, adică dacă întregul masiv de procesoare este condus de un procesor central, sau dacă fiecare procesor are propriul său controler;

(d) forma de control a sistemului, care poate fi dedusă dintr-o secvenţă predefinită de instrucţiuni sau o structură de control mai adecvată stilurilor de programare obiect orientate.

La elaborarea modelului arhitectural al SMB au fost luate în consideraţie atât aspecte structurale statice de modularitate sau dinamice de reconfigurabilitate, cât şi cele de funcţionalitate şi de implementare [4, 7, 8].

Arhitectura sistemului SMB pentru MS este prezentată într-o formă simplificată în Fig. 3.

Organizarea şi interacţiunea componentelor SMB.

Pentru a mări fiabilitatea sistemului au fost prevăzute 2 elemente de prelucrare (EP) a datelor. Întregul sistemul este compus din câteva subsisteme MC cu EP locale de prelucrare şi control a datelor, care la rândul lor vor comunica cu microprocesorul (MP) central.

Rolul principal în arhitectura îl joacă 2 MP identice, care în Fig. 3 sunt notate ca ” MAIN”. Ele vor lucra ca coordonatoare de lucru şi supraveghere a proceselor ce au loc pe MS. Unul va lucra în regim normal de funcţionare, iar al doilea va fi în rezervă în starea de repaus. Funcţionarea lor va fi analizată de microcontrolerul ”arbitrul”. În cazul apariţiei unor erori la funcţionarea unuia din microprocesoare sau ieşirea lui din funcţie, ”arbitrul” va activa MP care se află în repaus, comunicându-i adresa instrucţiunii şi datele curente ale MP principal, ce va păstra funcţionalitatea sistemului. Fiecare subsistem lucrează în mod similar. În SMB sunt prevăzute microcontrolere care se conectează la unităţile periferice prin comutatoare care vor juca rolul de ”arbitru”. În calitate de interfaţă pentru comunicarea subsistemelor cu MP central este folosită interfaţa RS485.

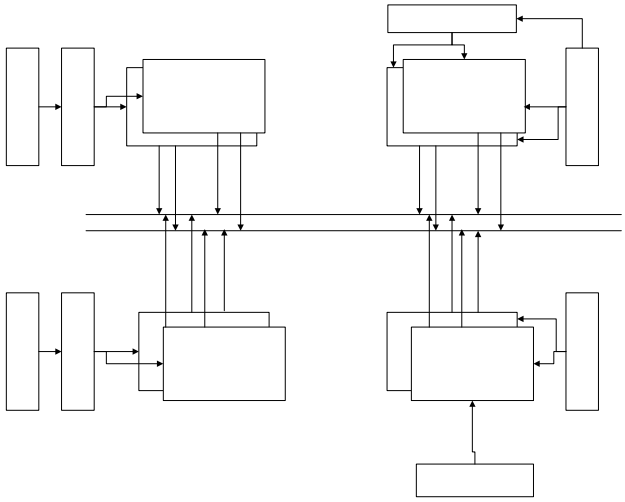

Fig. 3. Arhitectura SMB a microsatelitului.

Funcţia principală a SMB constă în a primi şi transmite datele de la MP *Master* sau senzori, subsisteme MC (dispozitive *Slave*), de a prelucra datele şi ca urmare, de a o transmite *Master*-ului rezultatul lucrului efectuat. Datele pot fi transmise pe una din cele două magistrale comune, dacă aceasta nu este ocupată. În cazul în care ele sunt ocupate, răspunsul trebuie memorizat atât timp până când una din ele nu se va elibera. La fel dispozitivul *Slave* nu primeşte datele şi nu începe prelucrarea lor atât timp, cât el este ocupat de prelucrarea datelor altei probleme. În aşa caz se verifică dacă aceste date pot fi prelucrate de un alt subsistem. Dacă aceasta este posibil, atunci informaţia trece la subsistemul respectiv, unde ea este prelucrată şi răspunsul este transmis la MAIN, care decide ce să facă cu rezultatul obţinut. Dacă datele primite nu pot fi prelucrate de alte subsisteme, aceste date sunt stocate intr-o unitate de memorie, unde îşi aşteaptă rândul.

Pentru schimbul de date între subsisteme este cercetată procedura SDA – transfer date cu confirmare (**S**end **D**ata with **A**cknowledge), similar protocolului standard Profibus [10].

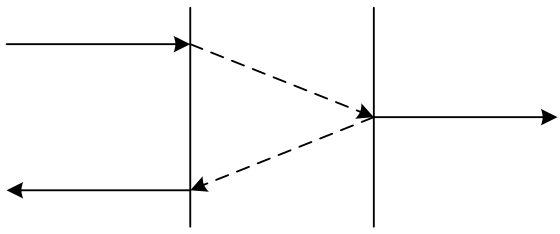

Fig. 4. Schimbul de date în regimul SDA.

Transmiterea datelor cu confirmare ASK permite dispozitivului MASTER de a trimite o comandă sau un set de date (SDA-data) unui dispozitiv SLAVE (Fig. 4). În cazul în care setul de date este transmis dispozitivului SLAVE fără erori, comanda şi setul de date sunt transmise stratului aplicativ. Ca urmare a acestor operaţii, va fi

transmisă o confirmare că datele au ajuns cu succes (confirmarea cmd = command adoptată) la destinaţie. În cazul în care setul de date conţine erori, dispozitivul SLAVE va trimite un mesaj de eroare în procesul schimbului de date (cmd = CMD_ERR, Data = ERR_TX). Dispozitivul MASTER la recepţionarea confirmării, va verifica dacă a fost o eroare în procesul schimbului de date, apoi va repeta procedura de transmitere a copiei datelor la acelaşi subsistem.

Algoritmul programului de prelucrare a datelor. În Fig. 5 este prezentată schema – bloc simplificată a algoritmului programului de prelucrare a datelor.

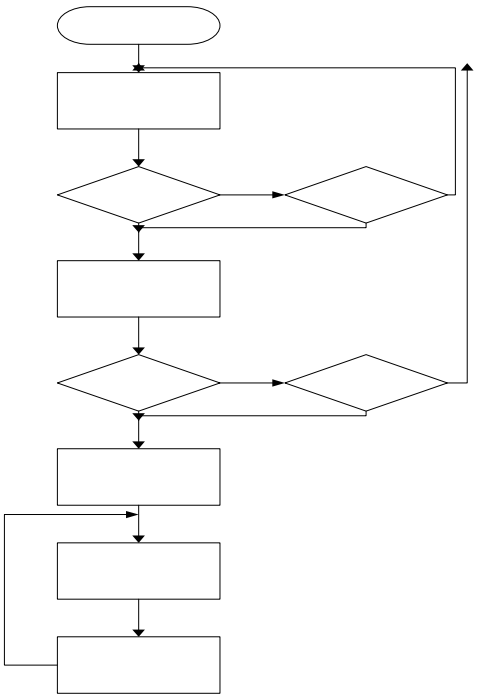

Fig. 5. Schema –bloc simplificată a algoritmului programului de prelucrare a datelor.

Punctul de pornire pentru acest algoritm se consideră momentul ieşirii MS pe orbita stabilită şi lansării sistemului de alimentare, în rezultatul căreia microprocesorul central Main este setat în starea de testarea, iar procesul de calcul este supravegheat de Arbitru. În cazul apariţiei unei erori se recurge la păstrarea datelor despre starea curentă în memoria de date, ceea ce va permite în viitor analiza erorii la staţia de pe pământ şi identificarea cauzei, însă există pericolul că păstrarea datelor nu poate fi posibilă în memoria externă. Pentru acest caz pot fi folosite registrele interne ale MP, însă din cauza că volumul de date poate să depăşească capacitatea memorii, în ea pot fi salvate doar adresele instrucţiunilor. Când datele sunt păstrate, sistemul analizează eroarea. Ca o posibilitate de analiză a erorii pot fi folosite codurile care corespund erorilor ce va facilita identificarea cauzei. În cazul în care eroarea este identificată se recurge la metoda specificată de remediere, însă există posibilitatea că eroarea nu poate fi identificată şi deci ea nu va avea codul propriu. În acest caz Arbitrul va comuta MP care se află în rezervă. Asupra acestuia se vor execută tot aceleaşi proceduri de testare ca şi asupra MP principal. Testarea ambelor MP se va efectua un mod succesiv. Aceasta se face din cauza că Arbitrul analizează într-un interval de timp doar un singur MP, dacă vor fi analizate concomitent două MP şi la ambele vor apărea aceeaşi eroare, ce teoretic este posibil, există probabilitatea că Arbitrul să se blocheze. Aceasta va duce la blocarea întregului sistem, deoarece nu va fi posibil de a executa algoritmii de remediere a erorilor care au apărut în MP.

Cum a fost deja menţionat, MP secundar va repeta executarea aceluiaşi algoritm ca şi la MP primar. În cazul în care la momentul testării şi MP secundar ies din funcţie se va recurge din nou la analiza cauzei. Dacă cauza nu va fi identificată, Arbitrul va trimite semnal pentru a reseta întregul sistem şi îl va restabili la punctul iniţial al algoritmului. Acest proces se va repeta în mod ciclic până ce nu va fi restabilit regimul specificat de funcţionarea al MP în care nu va apărea eroarea dată în procesul testării. Însă, dacă după un interval de timp specificat, sistemul nu poate restabili regimul de bună funcţionare, putem spune că eroarea este cauzată de factori fizici, care nu pot fi înlăturaţi prin intermediul executării programului. Acesta este un caz extrem şi ca consecinţă duce la pierderea MS.

În continuare, vom analiza algoritmul de funcţionare în condiţii când MP primar lucrează stabil şi în urma testării nu au fost depistate erori. După testarea MP primar se recurge la testarea sistemelor periferice. În principiu, algoritmul este identic cu cel de testare al MP primar, însă diferenţa constă în numărul sistemelor periferice şi deci a folosirii metodelor de redundanţă. Ca şi MP primar, subsistemele sunt dublate însă folosind MP identice, cu memoria internă ce are o capacitate mare, se poate de realizat o rezervare mai eficientă, deoarece în memoria unui subsistem poate să fie înscrisă nu numai programul lui dar şi programul unuia sau a mai multor subsisteme, care depind de capacitatea memoriei prevăzute. Avantajele unei astfel de rezervare ar fi fiabilitatea care creşte semnificativ, ceea ce permite de a repartiza funcţiile unui subsistem la altele, în cazul în care dacă acesta este ocupat. Să analizăm cazul în care la etapa testării în unul din subsisteme este depistată o eroare, însă analiza ei nu duce la identificarea cauzei. Acest fapt duce la activarea MP secundar, însă şi la testarea lui tot poate apărea o eroare, care nu poate fi identificată şi deci ea nu poate fi remediată. În acest caz se recurge la restartarea întregului sistem. Dacă după ciclul de restartare cauza erorii nu este înlăturată, MP central ia decizia de a activa în locul susbsistemului defectat unul din MP din subsistemelor care au trecut testul.

După testarea întregului sistem, se va verifica starea curentă a Arbitrului, care a controlat funcţionarea MP central. În cazul în care el a depistat o eroare, sistemul se va restarta. În caz contrar sistemul trece din starea de testare în starea de funcţionarea conform sarcinii tehnice. Trebuie de menţionat că testele nu pot să cuprindă toate aspectele funcţionării sistemului. De aceea este prevăzut şi cazul în care după începutul executării programului pot apărea abateri de la bună funcţionare. De supravegherea acestor procese se ocupă arbitri locali, pe schema-bloc ele sunt indicate ca comutatoare. Apartiţia a astfel de erori poate fi cauzată de mai mulţi factori atât locali, cât şi globali, de aceea se recurge la restartarea întregului sistem. Însă probabilitatea apariţiei a astfel de erori este foarte mică.

Start

Testarea Main

Pentru a efectua verificarea funcţională a acestui sistem multiprocessor a fost elaborate şi validat un model de reţea Petri temporizate.

Modelul de rețea Petri ce descrie interacțiunea dintre „Main-subprogram” și „Main-senzor” este prezentat în Fig. 6.

Pe parcursul transmiterii pe magistrale, se va alege concret care dintre ele va fi alocată, setul de date va fi transmis în paralel doar prin două din ele. Dacă datele au fost recepționate și au fost prelucrate, jetonul din locația p_{17} care răspunde de alegerea magistralei se întoarce înapoi și îi permite altui subsistem de a recepționa datele de la componenta Main.

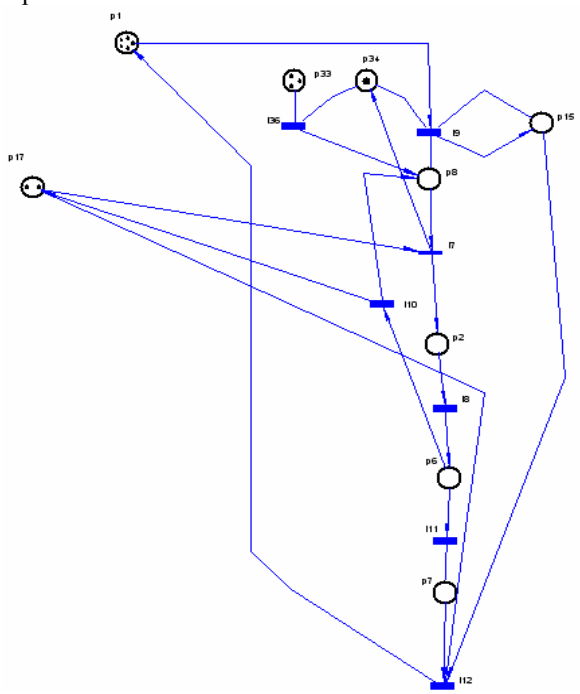

Fig. 6. Modelul de rețea Petri RP1 ce descrie interacțiunea dintre „Main-subprogram” și „Main-senzor” .

În acest timp subsistemele care nu au primit datele de la componenta Main pot lucra în mod autonom cu datele primite de la senzori. Subsistemul nu poate recepționa două sau mai multe instrucțiuni consecutive de la Main, căci aceasta poate provoca erori în timpul funcționării. Din această cauză se introduce un arbitru, care nu permite trecerea altui set de date în subsistemul respective, până când el nu finisează executarea operațiilor recepționate anterior. Oarecum, Main-ul are prioritate și în cazul în care subsistemul este ocupat de prelucrarea datelor primite de la senzori, iar în acest timp sosește o comandă de la dispozitivul MASTER, atunci prelucrarea datelor se întrerupe și rezultatele curente obținute sunt stocate intr-o unitate de memorie, atât timp, cât este necesar pentru executarea comenzii primite de la Main. Acest lucru a fost menționat anterior și este prezentat în Fig. 4, unde Main-ul este locația p_1 cu 5 probleme ce trebuie executate, iar 3 senzori sunt redați de locația p_{33} cu cei 3 jetoane ca marcaj inițial. În Fig. 4 se observă că în locația p_{34} este prezent un jeton, ce indică faptul că arbitrul este liber și decide ce cale are prioritate: de la dispozitivul MASTER sau de la senzori. Tranzițiile temporizate t_9 și t_{36} respectiv redau duratele respective pentru a forma calea prin care sunt transmise datele de la Main sau de la senzori.

Locația p_8 indică așteptarea accesului la o magistrală liberă, iar tranziția imediată t_7 - alocarea unei magistrale pentru a efectua transferul de date. Locația p_8 indică procesarea corectitudinii datelor recepționate, durata căreia este redată de tranziția temporizată t_8. Tranzițiile temporizate t_{10} și t_{11} ce se află în conflict structural redau duratele care corespund respectiv Analizei Erorilor și remedierea lor (t_{10}) sau procesarea datelor corect recepționate(t_{11}).

Pentru verificarea funcțională și validarea modelului RP1 de interacțiune dintre „Main-subprogram” și „Main-senzor” a fost folosit sistemul program instrumental VHPN, descris în [5].

III. CONCLUZIE

În lucrare sunt prezentate unele aspecte de elaborare și verificare funcțională a sistemului multiprocesor al unui microsatelit în baza rețelelor Petri temporizate, care sunt un instrument adecvat de modelare a funcționării proceselor de calcul și permit excluderea conflictelor atât la nivel hardlui, cât și a softului.

BIBLIOGRAFIE

[1] L. Baresi, M Pezzé, Improving “ UML with Petri Net,” *Electronic* Notes in Theorical Computer Science, vol. 44, pp. 1-13, 2001.

[2] M. Calzarossa, R. Marie, “Tools for Performance Evaluation,” Perf. Evaluation, no. 33, pp.1-3, 1998.

[3] C. Ciufudean, A. B. Larionescu,“ Estimation of the Performances of The Discrete Events Systems,” Advances in Electrical and Computer Engineering, No. 2, pp. 30-34, 2003.

[4] L. Corts, P. Eles, Z Peng, “Modeling and Formal Verification of Embedded Systems Based on a Petri Net Representation,” Journal of Systems Architectture, 49(12-15), pp. 571-598, 2003.

[5] E. Guțuleac, I., Țurcanu Em. Guțuleac, D. Odobescu, “ VHPN– software tool for visual discrete-continuous modelling of hybrid system using generalized timed differential Petri nets,” Proceedings of the 8-th International Conf. on D&AS 2006, Suceava, România, pp. 255-262, 25-27 May 2006.

[6] E. Guțuleac, Evaluarea performanțelor sistemelor de calcul prin rețele Petri stochastice. Ed.: „Tehnica-Info”, Chișinău, 2004, - 276 p.

[7] Ahmed A. Jerraya, W. Wolf, “Hardware/software interface codesign for embedded systems,” IEEE Computer, 38(2), pp. 63-69, February 2005.

[8] W. Qin, S. Malik, “Flexible and formal modeling of microprocessors with application to retargetable simulation,” Proceedings of Conference on Design Automation and Test in Europe, pp. 556-561, 2003.