Obsługa wyjątków oraz stanów wznowienia w ramach dualnej specyfikacji

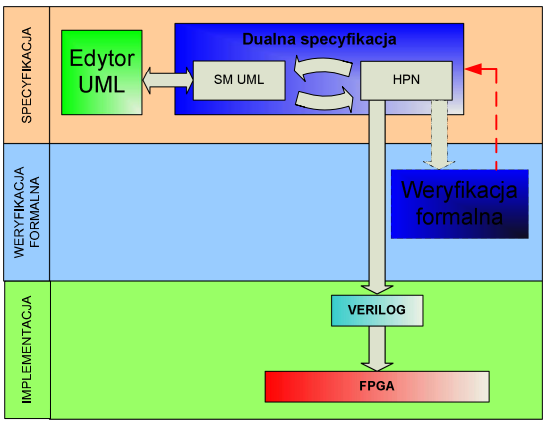
Michał Doligalski, Marian Adamski

Streszczenie: Dualna specyfikacja SM-HPN[4] jest alternatywną metodą specyfikacji sterowników cyfrowych, w odróżnieniu do konkurencji silnie opartą na standardach[7]. Diagram maszyny stanów – będący jej modelem elementarnym – umożliwia przedstawienie obsługi wyjątków oraz stanów wznowienia. Hierarchiczna sieć Petriego – będąca drugim komplementarnym modelem elementarnym – nie posiada takich mechanizmów[3,5]. W artykule omówiono sposób implementacji wznowienia i wywłaszczenia w hierarchicznej sieci Petriego poprzez wprowadzenie miejsc konfiguracyjnych i spoczynkowych.

Słowa kluczowe: dualna specyfikacja, SM-HPN, maszyna stanów UML, hierarchiczna sieć Petriego

1. WPROWADZENIE

Mapy stanów Davida Harela pod postacią diagramów maszyny stanów języka UML pozwalają na elastyczne i transparentne modelowanie systemów sterowania binarnego. Specyfikacja taka jest niezależna od implementacji, która między innymi może być zrealizowana za pomocą języków opisu sprzętu w strukturach reprogramowalnych FPGA[6]. Implementacja może być przeprowadzona bezpośrednio – na podstawie diagramu maszyny stanów, lub pośrednio z zastosowaniem modelu komplementarnego. W ramach dualnej specyfikacji model maszyny stanów UML pełniący role interfejsu użytkownika przekształcany jest na hierarchiczną sieć Petriego. Jej zadaniem jest umożliwienie przeprowadzenia procesu weryfikacji formalnej oraz implementacji układowej[1,2,8].

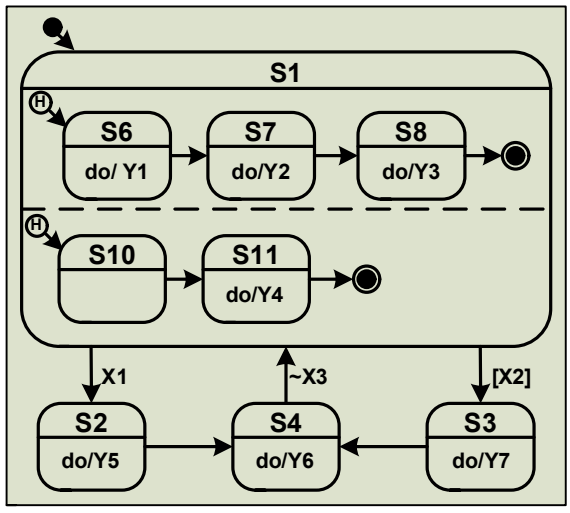


Rys 1. Projektowanie oparte o dualną specyfikację

Proponowaną ścieżkę projektową sterownika cyfrowego wykorzystującą model dualny SM-HPN przedstawiono na Rys.1. W artykule omówiono wybrany aspekt transformacji maszyny stanów UML do ekwiwalentnej hierarchicznej sieci Petriego. W przykładowym sterowniku uwzględniono obsługę sytuacji wyjątkowych za pomocą wywłaszczeń. Dodatkowo, przez przyjęcie odpowiedniej struktury podsieci, umożliwiono wznowienie funkcjonowania po uprzednim wywłaszczeniu.

2. DIAGRAM MASZYNY STANÓW UML

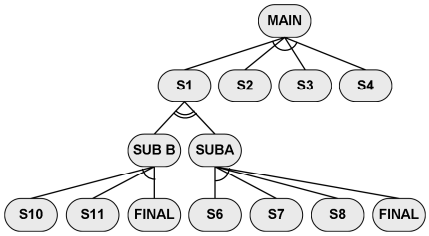
Graficzna specyfikacja behawioralna może być opracowana za pomocą dowolnego edytora graficznego umożliwiającego zapis do pliku XML (XMI, SCXML). Proponuje się zastosowanie formatu SCXML ze względu na uproszczoną składnię.
Specyfikacja sterownika w postaci diagramu maszyny stanów UML została przedstawiona na rysunku 2.



Rys 2. Diagram maszyny stanów

Diagram zawiera na pierwszym poziomie hierarchii stan złożony *S1* składający się z dwóch obszarów współbieżnych. W notacji graficznej diagramów maszyny stanów, nie stosuje się jawnych nazw obszarów współbieżnych.
Z punktu widzenia formatu SCXML zarówno maszyny

stanów i obszary współbieżne otoczone są stanem okalającym, w związku z tym konieczne było nadanie nazw obszarom współbieżnym: *SUB A i SUB B*.



Rys 3. Diagram hierarchii maszyny stanów

Maszyna stanów z rysunku 2, została umieszczona wewnątrz stanu okalającego, któremu została przypisana nazwa *MAIN*. Diagram hierarchii dla maszyny stanów z rysunku 2 uwzględniający stany okalające został przedstawiony na rysunku 3.

Przedstawiona w artykule abstrakcyjna maszyna stanów zawiera cztery typy przejść. Wybrane przejścia zostały przedstawione w tabeli 1.

Tabela 1.
Zestawienie wybranych przejść

Typ	*Stan początkowy*	*Stan docelowy*	*Wyzwalacz*	*Warunek dozoru*
Proste bezwarunkowe	S6	S7	-	-
	S7	S8		
	S10	S11		
Proste warunkowe	S1	S3	-	X2
Wywłaszczenie	S1	S2	X1	-
Proste z wyzwalaczem	S4	S1	~X3	-

Pozornie podobne przejścia: *S1 → S3* oraz S1 → S2, realizują zupełnie odmienny mechanizm. Przejście *S1 → S3* jest przejściem prostym uwarunkowanym spełnieniem warunku dozoru. Realizowane jest w sytuacji gdy osiągnięte zostały wszystkie stany końcowe w złożonym stanie początkowym *(S1) a* warunek dozoru określony przez równanie logiczne przyjmuje wartość *prawda.* Specyfikowanie przejść prosty (bezwarunkowych i warunkowych), których źródłem jest stan złożony ma sens tylko wtedy, gdy w każdym obszarze współbieżnym danego stanu jest przynajmniej jeden stan końcowy.

Odmiennie realizowane jest przejście S1 → S2 mające charakter wywłaszczenia. Jego realizacja następuje w chwili wystąpienia wyzwalacza opisującego przejście. Dopuszcza się określenie wyzwalaczy jako równań logicznych, a wystąpienie wyzwalacza będzie miało miejsce gdy równanie przyjmie wartość *prawda.* Realizacja przejścia z wywłaszczeniem powoduje opuszczenie (deaktywację) wszystkich stanów odpowiadającym w drzewie hierarchii węzom podrzędnym danego stanu źródłowego (*S6, S7, S8, S10, S11).*
W przypadku gdy z jednego stanu jednocześnie możliwa jest realizacja przejść prostych oraz przejść z wywłaszczeniem, wyższy priorytet ma wywłaszczenie. Hierarchia realizacji wynika z faktu iż wywłaszczenie odpowiedzialne jest za realizację sytuacji o szczególnym charakterze (wyjątkowych). Przy czym nie należy ich utożsamiać jedynie z sytuacjami awaryjnymi. Mechanizm wznowienia pozwala na kontynuowanie funkcjonowania wywłaszczonej (wywłaszczonych) maszyn po ponownej aktywacji stanu złożonego.

Tabela 2.
Opis maszyny stanów w formacie SCXML

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<SCXML NAME ="SCXML_FOR_KNWS_X"
XMLNS="HTTP://WWW.W3.ORG/2005/07/SCXML" VERSION="1.0"
INITIAL="MAIN">
  <STATE ID="MAIN">
    <INITIAL>
   <TRANSITION TARGET="S1"/>
  </INITIAL>
   <PARALLEL ID="S1">
         <STATE ID="S1_SUB_A">
                 <HISTORY ID="H_S1_SUBSTATE_A">
                  <TRANSITION TARGET="S6"/>
                 </HISTORY>
                 <STATE ID="S6">
                  <TRANSITION TARGET="S7"/>
                  <ONDO>
                  <SEND EVENT="Y1" />
                  </ONDO>
                 </STATE>
                 <STATE ID="S7">
                  <TRANSITION TARGET="S8"/>
                  <ONDO>
                  <SEND EVENT="Y2" />
                  </ONDO>
                 </STATE>
                 <STATE ID="S8">
                  <TRANSITION TARGET="F_H_S1_SUB_A"/>
                  <ONDO>
                  <SEND EVENT="Y3" />
                  </ONDO>
                 </STATE>
                 <FINAL ID ="F_H_S1_SUB_A"/>
         </STATE>
         <STATE ID="S1_SUB_B">
                 <HISTORY ID="H_S1_SUB_B">
                  <TRANSITION TARGET="S10"/>
                 </HISTORY>
                 <STATE ID="S10">
                  <TRANSITION TARGET="S11"/>
                 </STATE>
                 <STATE ID="S11">
                  <TRANSITION TARGET="F_H_S1_SUB_B"/>
                  <ONDO>
                  <SEND EVENT="Y4" />
                  </ONDO>
                 </STATE>
                 <FINAL ID ="F_H_S1_SUB_B"/>
         </STATE>
         <TRANSITION EVENT="X1" TARGET="S2"/>
         <TRANSITION COND="X2" TARGET="S3"/>
  </PARALLEL>
  <STATE ID="S2">
          <TRANSITION TARGET="S4"/>
          <ONDO>
          <SEND EVENT="Y5" />
          </ONDO>
  </STATE>
  <STATE ID="S3">
          <TRANSITION TARGET="S4"/>
          <ONDO>
          <SEND EVENT="Y7" />
          </ONDO>
  </STATE>
  <STATE ID="S4">
          <TRANSITION EVENT="~X3" TARGET="S1"/>
          <ONDO>
          <SEND EVENT="Y6" />
          </ONDO>
  </STATE>
  <!-- END OF STATE MAIN-->
  </STATE>
</SCXML>
```

Takiej możliwości nie ma w przypadku opuszczenia stanu złożonego przejściem prostym (warunkowym lub bezwarunkowym). Jego realizacja uwarunkowana jest uprzednim osiągnięciem stanów końcowych. W takim przypadku ponowna aktywacja inicjuje jedynie stany początkowe.

Specyfikacja graficzną sterownika logicznego w postaci diagramu maszyny stanów opracowana przy pomocy edytora graficznego winna być, w celu realizacji kolejnych kroków procesu projektowego, zapisana w postaci pliku XML. Ze względu na przejrzystość standardu opisu wybrano format SCXML Tab.2 .

3. HIERARCHICZNA SIEĆ PETRIEGO

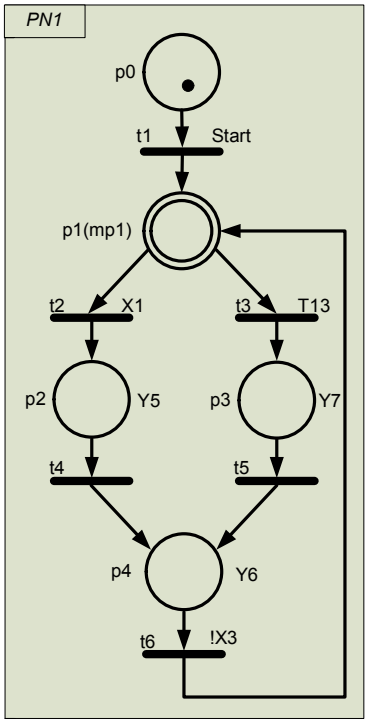
Odzwierciedlenie podstawowych elementów maszyny stanów UML do Hierarchicznej sieci Petriego jest niemalże intuicyjne. W tabeli 4 przedstawiono wzajemną konwersję podstawowych elementów.

Tabela 4.
Odzwierciedlenie elementów podstawowych

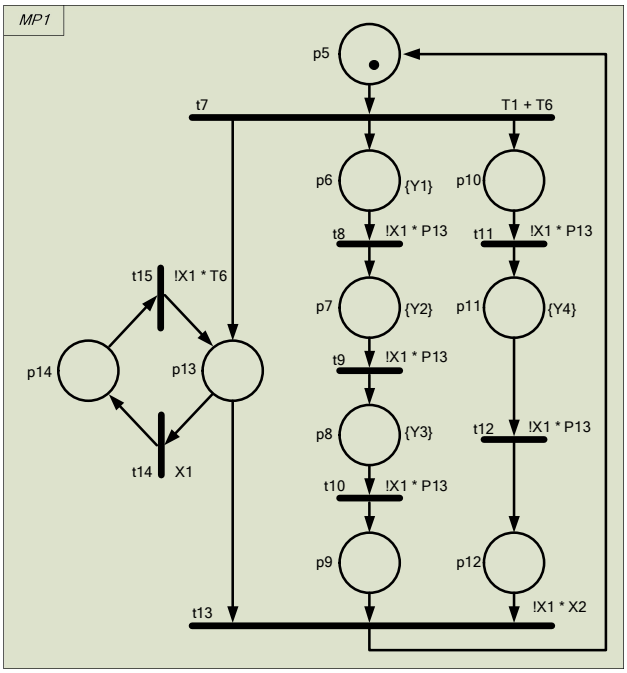
Maszyna stanów UML		Sieć Petriego	
Stan prosty	S1 do/ Y2	Miejsce	p1 Y2
Przejście bezwarunkowe	S2 → S3	Tranzycja bez określonego warunku realizacji	t1
Przejście z warunkiem dozoru	S2 [x1] S3	Tranzycja z określonym warunkiem realizacji	t1 x1
Przejście z wyzwalaczem	S2 x1 S3		
Przejście z wyzwalaczem i warunkiem dozoru	S2 x1[x2] S3	Tranzycja z określonym warunkiem realizacji	t1 x1*x2
Stan oznaczony stanem początkowym	S2	Miejsce oznakowane znacznikiem	p2
Stan końcowy		Miejsce bez określonego przejścia wyjściowego	p3

Sieci Petriego nie dostarczają mechanizmu wywłaszczenia czy teżwznowienia w sposób bezpośredni. Realizacja odbywa się w sposób pośredni, poprzez wprowadzenie dodatkowych miejsc i tranzycji. W kolejnym etapie procesu projektowania sterownika (rys. 1), tworzona jest hierarchiczna sieć Petriego. Oba modele muszą spełniać wymóg wzajemnej konwersji.

Rysunki 4 i 5 przedstawiają odpowiednio sieć główną *PN1* oraz podsieć *MP1* skojarzoną z makromiejscem p1.



Rys 4. Sieć nadrzędna PN1



Rys 5. Sieć podrzędna MP1

W przedstawionych modelach pośrednich uwzględniono zarówno przejścia proste ze stanu złożonego jak i przejście z wywłaszczeniem.

Za realizację przejścia prostego *S1* → *S3* z warunkiem dozoru odpowiadają dwie tranzycje: *t3* i *t13*.

Synchronizacja obu obszarów współbieżnych odbywa się wewnątrz podsieci *MP1*: miejsca *p9* i *p*12 odpowiadają stanom końcowym obszarów współbieżnych stanu złożonego *S1* (Rys. 5). Osiągnięcie miejsc *p9, p12 p13* oraz spełnienie warunku tranzycji powoduje jej realizację i znakowanie miejsca *p5*.

Warunek tranzycji *t13* jest iloczynem afirmacji bądź warunków dozoru przejść oraz negacji wyzwalaczy przejść z wywłaszczeniem z danego złożonego stanu źródłowego. W przedstawionym przypadku jest to iloczyn wyzwalacza *X1* oraz warunku dozoru *X2*.

Realizacja przejścia powoduje również zmiany w znakowaniu sieci nadrzędnej. Tranzycja *t13* jest jednocześnie warunkiem realizacji tranzycji *t3* sieci nadrzędnej.

W przedstawionej hierarchicznej sieci Petriego realizującej mechanizm wywłaszczenia oraz płytkiego wznowienia wyróżnić można trzy typy miejsc:

- Podstawowe: *p5, .., p12*
- Spoczynkowe: *p14*
- Konfiguracyjne: *p13.*

Miejsce konfiguracyjne jest odpowiedzialne za informowanie czy w danej chwili sieć skojarzona z stanem złożonym jest aktywna czy też nie. Jeśli sieć jest aktywna, przejścia pomiędzy miejscami odpowiadającymi stanom wewnątrz danego stanu złożonego są odblokowane. Miejsce spoczynkowe informuje o zaistnieniu wywłaszczenia, jest ono znakowane do czasu ponownego aktywowania uprzednio wywłaszczonego stanu złożonego.

W omawianym przykładzie p13 to miejsce konfiguracyjne podsieci *MP1,* natomiast miejsce *p14* jest miejscem spoczynkowym. Tranzycja spoczynkowa – z miejsca konfiguracyjnego *p13* do spoczynkowego *p14* otrzymuje warunek który jest identyczny z wyzwalaczem. W przypadku gdy dany stan złożony posiada kilka przejść z wywłaszczeniem, tranzycja spoczynkowa przyjmuje warunek będący sumą logiczną wyzwalaczy przejść.

Tranzycja konfiguracyjna – z miejsca spoczynkowego do konfiguracyjnego realizowana jest gdy wywłaszczona sieć jest ponownie aktywowana. Warunkiem realizacji tranzycji spoczynkowej jest odpalenie tranzycji *T6* z sieci nadrzędnej.

Realizacja tranzycji spoczynkowej i przejście żetonu z miejsca konfiguracyjnego do spoczynkowego powodowało zablokowanie przejść pomiędzy miejscami podstawowymi w podsieci. Tym samym aktualna konfiguracja była „zamrażana". Ponowna aktywacja podsieci – czyli realizacja tranzycji konfiguracyjnej – wznawia pracę poprzez odblokowanie możliwości realizacji tranzycji pomiędzy poszczególnymi stanami podstawowymi. Układ zostaje „odmrożony" – stosując terminologię zaczerpniętą z standardu UML, następuje płytkie wznowienie.

Dualna specyfikacja ukierunkowana jest na projektowanie sterowników logicznych implementujących automaty typu Moore'a. Każdy stan prosty maszyny stanów, a więc i miejsce sieci Petriego może mieć przypisaną akcję.

W omawianym przykładzie miejscom *p6, p7, p8, p11* przypisano odpowiednio sygnały wyjściowe *Y1, Y2, Y3, Y4*. Nazwy sygnałów zostały umieszczone w nawiasach klamrowych – w odróżnieniu do miejsc w sieci nadrzędnej – ponieważ wygenerowanie sygnału będzie zależne nie tylko od oznakowania danego miejsca ale również od oznakowania miejsca konfiguracyjnego. Przykładowo dla sygnału *Y2* funkcja wyjścia przyjmie postać *Y2=p7*p13.*

3. REALIZACJA UKŁADOWA STEROWNIKA

Model pośredni w postaci hierarchicznej sieci Petriego wykorzystywany jest w procesie weryfikacji formalnej i implementacji układowej. Ideą dualnej specyfikacji sterownika jest możliwość wykorzystania istniejących narzędzi do weryfikacji formalnej, dlatego proces weryfikacji nie będzie w tym artykule poruszany.

Proponowana ścieżka projektowa przewiduje implementację układową sterownika w strukturach reprogramowalnych FPGA z wykorzystaniem języków HDL. W tabeli 4 i 5 przedstawiono sieć Petriego opisaną językiem Verilog.

Tabela 4.
Opis sieci głównej PN1 w języku Verilog

```
module main_net( CLOCK, reset, X1, X3, T13, T1, T6,
Y5, Y6, Y7);
  input CLOCK, reset, X1, X3, T13;
  output T1, T6;
  output Y5, Y6, Y7;
  reg P0, P1, P2, P3, P4;
  assign Y5 = P2;
  assign Y6 = P4;
  assign Y7 = P3;
  always @(posedge CLOCK)
  begin
        if (reset) P0 <= 1;
        else P0 <= ( P0 & ~( T1 )) ;
  end
  always @(posedge CLOCK)
  begin
        if (reset) P1 <= 0;
        else P1 <= ( P1 & ~( T2 | T3 )) | ( T1 )
                    | ( T6 );
  End
  always @(posedge CLOCK)
  begin
        if (reset) P2 <= 0;
        else P2 <= ( P2 & ~( T4 )) | ( T2 ) ;
  end
  always @(posedge CLOCK)
  begin
        if (reset) P3 <= 0;
        else P3 <= ( P3 & ~ ( T5 )) | ( T3 );
  end
  always @(posedge CLOCK)
  begin
         if (reset) P4 <= 0;
         else P4 <= ( P4 & ~( T6 )) | ( T4 )
                          | ( T5 );
  end
  wire T1, T2, T3, T4, T5, T6;
  assign T1 = P0 ;
  assign T2 = P1 & X1;
  assign T3 = P1 & ~X1 & T13;
  assign T4 = P2 ;
  assign T5 = P3 ;
  assign T6 = P4 & ~X3;
endmodule
```

Sieć implementowana jest jako dwa moduły, komunikujące się za pomocą sygnałów (*T1, T6, T13).* Każde z miejsc w podsieci *MP1* tab. 4, posiada alias globalny w postaci sygnału będącego iloczynem miejsca i miejsca konfiguracyjnego. Na podstawia aliasów globalnych określone są funkcje wyjść.

Prezentowana implementacja układowa zawiera aliasy globalne wszystkich stanów w podsieci. Można z powodzeniem dokonać minimalizacji aliasów poprzez usunięcie tych, które nie są wykorzystywane w opisie funkcji wyjść.

Tabela 5.
Opis sieci makrosieci MP1 w języku Verilog

```
module subnet1( CLOCK, reset, T1, T6, X1, X2, T13,
Y1, Y2, Y3, Y4);
  input CLOCK, reset, X1, X2, T1, T6;
  output T13;
  output Y1, Y2, Y3, Y4;
  assign Y1 = G_P6;
  assign Y2 = G_P7;
  assign Y3 = G_P8;
  assign Y4 = G_P11;
  reg P5, P6, P7, P8, P9, P10, P11, P12, P13, P14;
  always @(posedge CLOCK)
  begin
        if (reset) P5 <= 1;
        else P5 <= ( P5 & ~( T7 )) | ( T13 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P6 <= 0;
        else P6 <= ( P6 & ~( T8 )) | ( T7 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P7 <= 0;
        else P7 <= ( P7 & ~( T9 )) | ( T8 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P8 <= 0;
        else P8 <= ( P8 & ~( T10 )) | ( T9 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P9 <= 0;
        else P9 <= ( P9 & ~( T13 )) | ( T10 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P10 <= 0;
        else P10 <= ( P10 & ~( T11 )) | ( T7 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P11 <= 0;
        else P11 <= ( P11 & ~( T12 )) | ( T11 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P12 <= 0;
        else P12 <= ( P12 & ~( T13 )) | ( T12 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P13 <= 0;
        else P13 <= ( P13 & ~( T13 | T14 )) | ( T7 )
| ( T15 );
  end
  always @(posedge CLOCK)
  begin
        if (reset) P14 <= 0;
        else P14 <= ( P14 & ~( T15 )) | ( T14 );
  end
  wire T7, T8, T9, T10, T11, T12, T13, T14, T15;
  assign T7 = P5 & (T1 | T6);
  assign T8 = P6 & ~X1 & P13;
  assign T9 = P7 & ~X1 & P13;
  assign T10 = P8 & ~X1 & P13;
  assign T11 = P10 & ~X1 & P13;
  assign T12 = P11 & ~X1 & P13;
  assign T13 = P9 & P12 & P13 & ~X1 & X2;
  assign T14 = P13 & X1;
  assign T15 = P14 & ~X1 & T6;
  wire G_P5, G_P6, G_P7, G_P8, G_P9, G_P10, G_P11,
        G_P12, G_P13, G_P14;
  assign G_P5 = P5 & P13;
  assign G_P6 = P6 & P13;
  assign G_P7 = P7 & P13;
  assign G_P8 = P8 & P13;
  assign G_P9 = P9 & P13;
  assign G_P10 = P10 & P13;
  assign G_P11 = P11 & P13;
  assign G_P12 = P12 & P13;
  assign G_P13 = P13;
  assign G_P14 = P14 & P13;
endmodule
```

3. PODSUMOWANIE

W artykule zaprezentowano możliwość modelowania sytuacji wyjątkowych oraz stanów wznowienia w ramach nowej alternatywnej formy specyfikacji sterowników cyfrowych. Dualna specyfikacja, w odróżnieniu do stosowanych obecnie metod, dzięki połączeniu dwóch modeli elementarnych umożliwia zarówno precyzyjne i transparentne modelowanie na poziomie behawioralnym jak również ułatwia proces weryfikacji formalnej czy implementacji układowej.

LITERATURA

[1] Adamski M., Karatkevich A., Węgrzyn M. (red), *Design of embed-ded control systems*, Springer, New York 2005

[2] Andreu D., Souquet G., Gil T., „Petri Net Based Rapid Prototyping of Digital Complex System", *IEEE Computer Society Annual Symposium on VLSI*, IEEE, 2008

[3] Basile F., Chiacchio P., Del Grosso D., „Modelling automation systems by UML and Petri Nets", *Proceedings of the 9th International Workshop on Discrete Event Systems*, Gooteborg, IEEE, 2008

[4] Doligalski M., „Konwersja wybranych elementów maszyny stanów UML w ramach dualnej specyfikacji" *Przegląd Elektrotechniczny* .- 2009, R. 85, nr 7

[5] Gajski D. D., Vahid F., Narayan S., Gong J., *Specification and Design of Embedded Systems*, P T R Prentice Hall, New Jersey 1994

[6] Łabiak G., *Wykorzystanie hierarchicznego modelu współbieżnego automatu w projektowaniu sterowników cyfrowych*, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra 2005

[7] Holvoet T., Verbaeten P., „Petri Charts, an Alternative Technique for Hierarchical Net Construcion", *IEEE Conference on Systems, Man and Cybernetics*, 1995

[8] Węgrzyn A., Symboliczna analiza układów sterowania binarnego z wykorzystaniem wybranych metod analizy sieci Petriego, Oficyna Wydawnicza UZ, Zielona Góra 2003

mgr inż. Michał Doligalski
Uniwersytet Zielonogórski
Wydział Elektrotechniki Informatyki i Telekomunikacji
Instytut Informatyki i

ul. Licealna 9
65-417 Zielona Góra

tel.: (0-68) 328-22-19
e-mail: M.Doligalski@iie.uz.zgora.pl

prof. dr hab. inż. Marian Adamski
Uniwersytet Zielonogórski
Wydział Elektrotechniki Informatyki i Telekomunikacji
Instytut Informatyki i

ul. Licealna 9
65-417 Zielona Góra

tel.: (0-68) 328-22-19
e-mail: M.Adamski@iie.uz.zgora.pl