Grzegorz BAZYDŁO, Marian ADAMSKI, Łukasz STEFANOWICZ
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI,
ul. prof. Szafrana 2, 65-516 Zielona Góra

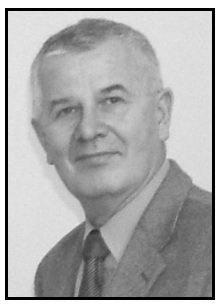
Wykorzystanie pseudostanów historii do modelowania sytuacji awaryjnych w maszynie stanów UML

Dr inż. Grzegorz BAZYDŁO

Absolwent Uniwersytetu Zielonogórskiego, pracę doktorską obronił w 2010 roku. Aktualnie pracuje jako adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania naukowe koncentrują się wokół nowoczesnych metod projektowania specjalistycznych systemów cyfrowych z wykorzystaniem struktur programowalnych FPGA.

e-mail: G.Bazydlo@iie.uz.zgora.pl

Prof. dr hab. inż. Marian ADAMSKI



Profesor zwyczajny, zatrudniony w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych. Członek IEEE, IEE, ACM, PTI oraz PTETiS.

e-mail: M.Adamski@iie.uz.zgora.pl

Streszczenie

W artykule przedstawiono zagadnienia związane z modelowaniem obsługi sytuacji awaryjnych opierając się na metodzie syntezy behawioralnej sterowników logicznych opisanych diagramami maszyny stanowej UML. Szczególną uwagę zwrócono na wykorzystanie pseudostanów historii a także zdarzeń i przejść zakończenia (typu *completion event*), przejść wysokiego poziomu, stanów końcowych i przejść bezwarunkowych. Celem zaproponowanej metody jest takie przekształcenie modelu hierarchicznej maszyny stanów UML, aby otrzymać opis układu w języku opisu sprzętu Verilog. Metoda została poparta stosownym przykładem układu sterowania.

Słowa kluczowe: UML, maszyna stanów, pesudostan historii, sterownik logiczny.

Use of history pseudostates for modeling the emergency situation in a UML state machine

Abstract

The paper presents the design methodology for deriving Verilog descriptions from UML state machine diagrams (Figs. 2, 3) in order to capture the behavioral hierarchy in the array structure of an embedded system. The exception handling is introduced at the top level of the graphical specification. As an intuitive example the interrupt is introduced. It illustrates the case of a system failure, when the control is temporarily transferred to exceptional safe and determined behavior. The precise semantic interpretation of the UML 2.4 state machine diagrams ensures, under the proposed structural design rules, that the Verilog description conserves modular properties of an initial specification. The behavioral hierarchy of the UML state machine is directly mapped into a structural hierarchy inside the designed reconfigurable controller. The tree of properly encapsulated submachines allows independent simulation and modification of particular parts of the behavioral model. In the paper the emphasis is put on the support of modeling an emergency situation with use of history pseudostates, high-level transitions and completion events. The way of hardware implementation of storing the information about the previously active state is also presented (Fig. 5). The most important algorithm of the proposed method is illustrated by an appropriate example (Fig. 1).

Keywords: UML, state machine, history pseudostate, logic controller.

Mgr inż. Łukasz STEFANOWICZ

Absolwent Uniwersytetu Zielonogórskiego, pracę magisterską obronił w 2011 roku. Jest słuchaczem studiów doktoranckich, specjalność informatyka. Członek PTI oraz Uczelnianego Koła Naukowego. Aktywnie uczestniczy w realizacji pokazów naukowych o zasięgu krajowym oraz międzynarodowym. Zainteresowania badawcze obejmują zagadnienia z zakresu teorii grafów oraz hipergrafów w kontekście sterowników logicznych.

e-mail: L.Stefanowicz@weit.uz.zgora.pl

1. Wstęp

Język UML [1] (ang. *Unified Modeling Language*), powszechnie używany w modelowaniu systemów informatycznych, jest coraz częściej wykorzystywany także do specyfikacji behawioralnej systemów cyfrowych, w tym także sterowników logicznych. Opracowano wiele metod syntezy mikrosystemów cyfrowych opartych o wybrane diagramy UML, za pomocą których opisywane jest zachowanie projektowanego układu. Głównym i najczęstszym rodzajem diagramu UML wykorzystywanym do behawioralnego opisu układu jest diagram maszyn stanowych. Jest to zrozumiałe, ponieważ odwołuje się on bezpośrednio do definicji automatu skończonego FSM (ang. *Finite State Machine*), a ponadto wspiera takie cechy projektowanego układu jak hierarchiczność i współbieżność. Inne rodzaje diagramów (np. diagram klas, diagram komponentów [2], diagram sekwencji [3]) pełnią często rolę wspomagającą i nie są bezpośrednio wykorzystywane w głównej ścieżce syntezy układu. W kolejnym kroku większość dotychczas opublikowanych metod generuje na podstawie diagramów UML specyfikację układu w wybranym języku opisu sprzętu. Taki opis może być później wykorzystany do symulacji, syntezy i implementacji układu. Niestety, w przypadku niektórych metod, generowana specyfikacja jest niesyntezowalna, niemodularna [3] lub z zaburzoną strukturą układu [2]. Ponadto opis układu jest zazwyczaj generowany do jednego, wynikowego pliku, co powoduje, że próba modyfikacji jednego stanu na diagramie UML oznacza w praktyce konieczność ponownego wygenerowania całej specyfikacji wynikowej. Także dodanie obsługi sytuacji awaryjnej, polegającej na przykład na wprowadzeniu układu do specjalnego stanu bezpiecznego, wymaga najczęściej przebudowy całego projektu.

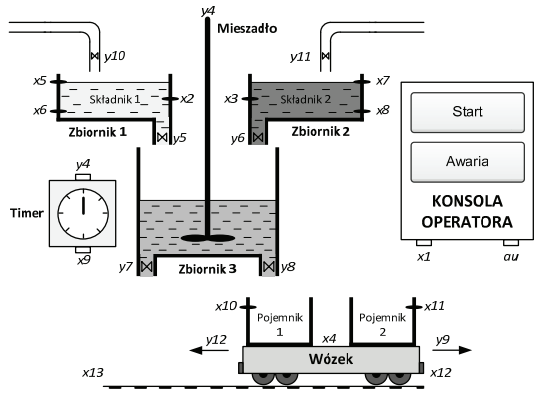
2. Metoda projektowania

Uwzględniając wspomniane w poprzednim rozdziale wady dotychczas opublikowanych metod, w artykule zaproponowano sprawdzoną eksperymentalnie metodę syntezy hierarchiczno-współbieżnej maszyny stanów, wyrażonej diagramem UML, na równoważny mu pod względem semantycznym opis tekstowy w języku opisu sprzętu Verilog. Jako podstawową formę specyfikacji przyjęto diagram maszyny stanowej UML [1]. Model hierarchicznej maszyny stanowej, zapisany w języku XML, dekomponowany jest na pośredni model HCFSM (ang. *Hierarchical Concurrent Finite State Machine*), który w rzeczywistości jest hierarchiczno-współbieżną siecią zdekomponowanych i powiązanych automatów FSM. Następnie dla każdego z poszczególnych automatów FSM dokonywana jest translacja na język opisu sprzętu. Warto zaznaczyć, że wynikiem tego etapu jest zbiór plików tekstowych w syntezowalnym podzbiorze języka Verilog, w których starannie zamodelowana na diagramie UML struktura układu, jest dokładnie odzwierciedlona. Dzięki temu, że wygenerowana specyfikacja jest modularna a poszczególne moduły zawarte są w oddzielnych pli-

kach, możliwa jest ich niezależna weryfikacja, symulacja i optymalizacja. Ostatnim etapem proponowanej metody jest symulacja, synteza i implementacja algorytmu w układzie FPGA.

3. Przykład układu sterowania

Praktyczne wykorzystanie metody zaprezentowano na przykładzie procesu sterowania wytwarzaniem napojów [4, 5]. Przykład ten zmodyfikowano dodając obsługę awarii systemu. Schemat procesu technologicznego przedstawiono na rys. 1. Szczegółowy opis tego systemu, wraz z charakterystyką sygnałów wejściowych znajduje się w [6].



Rys. 1. Schemat procesu technologicznego Mieszalnika
Fig. 1. Diagram of the industrial mixer process

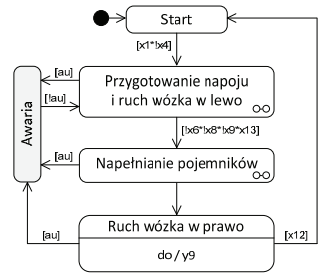
4. Maszyna stanów UML

Język UML [19] to graficzny język modelowania, służący do obrazowania, specyfikowania, tworzenia i dokumentowania nie tylko systemów informatycznych [7], ale także systemów cyfrowych [2, 3] (w tym sterowników logicznych [8]). Obecna wersja tego języka (2.4.1) zawiera 14 rodzajów diagramów, z których tylko jeden używany jest w proponowanej metodzie. Jest to diagram maszyny stanowej, ponieważ jest on graficznym odzwierciedleniem dyskretnego zachowania skończonych systemów typu stan-przejście. Użycie innych rodzajów diagramów jest przedmiotem dalszych prac autorów.

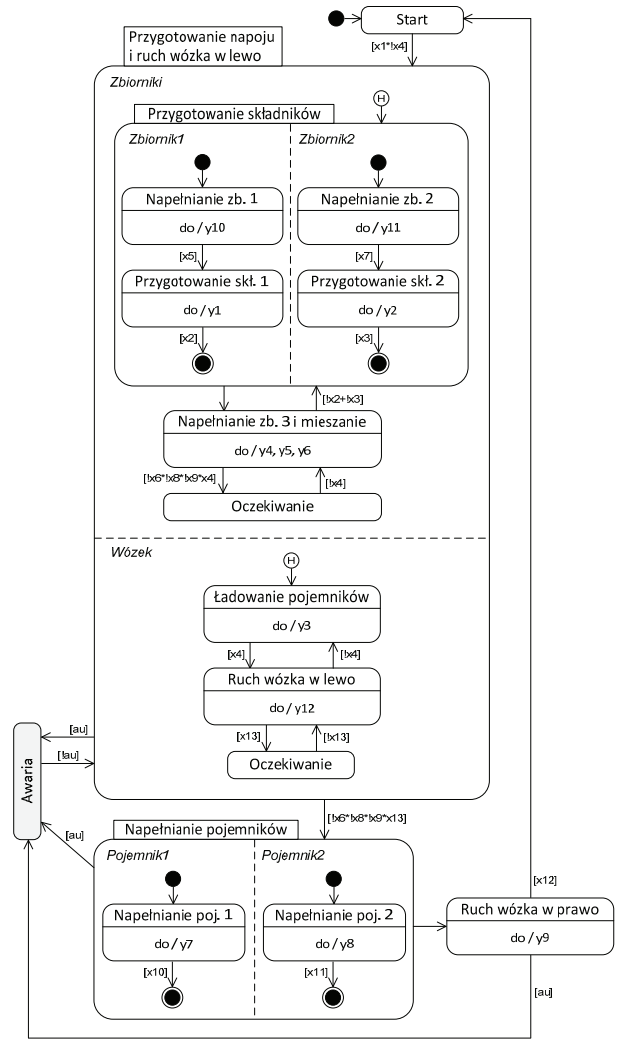
Stany na diagramie maszyny stanowej UML reprezentowane są w postaci prostokątów z zaokrąglonymi rogami, a przejścia oznaczane są strzałkami (rys. 2, 3). Z przejściem może być skojarzone zdarzenie uruchamiające, warunek oraz akcja wykonywana podczas realizacji przejścia. Jednym z założeń proponowanej metody jest docelowa realizacja projektowanego układu jako synchroniczny automat współbieżny z jednym sygnałem zegarowym. Sygnał ten jest także zdarzeniem uruchamiającym przejścia. Aby jednak dane przejście mogło zostać zrealizowane, oprócz zdarzenia uruchamiającego musi być także spełniony warunek, przedstawiany na diagramie jako wyrażenie logiczne składające się z nazw sygnałów i podstawowych operatorów logicznych (np. *x1*!x4*).

Na podkreślenie zasługuje fakt, że diagramy maszyn stanowych wspierają takie cechy układu, jak hierarchia i współbieżność. Pozwala to w łatwy i intuicyjny sposób specyfikować zachowanie złożonych systemów współbieżnych na wybranym poziomie uszczegółowienia. Na rys. 2 przedstawiono diagram maszyny stanowej dla prezentowanego przykładu na najwyższym poziomie hierarchii, a na rys. 3 tę samą maszynę stanową, ale uwzględniając wszystkie detale projektowanego układu.

Warto także zwrócić uwagę na występujące na diagramie przejście ze stanu „Napełnianie pojemników” do stanu „Ruch wózka w prawo”. Mimo, że nie ma ono przypisanego warunku realizacji, nie jest to przejście bezwarunkowe. Zgodnie z notacją UML [1] przejście ze stanu złożonego ma przypisane niejawne zdarzenie zakończenia (ang. *completion event*). W rozpatrywanym przykładzie wystąpi ono np. po przekazaniu sterowania do stanów końcowych we współbieżnie działających regionach współbieżnych (automatach) „Pojemnik1” oraz „Pojemnik2” (rys. 3).



Rys. 2. Diagram maszyny stanowej UML – najwyższy poziom hierarchii
Fig. 2. Diagram of the state machine – the highest hierarchy level



Rys. 3. Diagram maszyny stanowej UML – najniższy poziom hierarchii
Fig. 3. Diagram of the state machine – the lowest hierarchy level

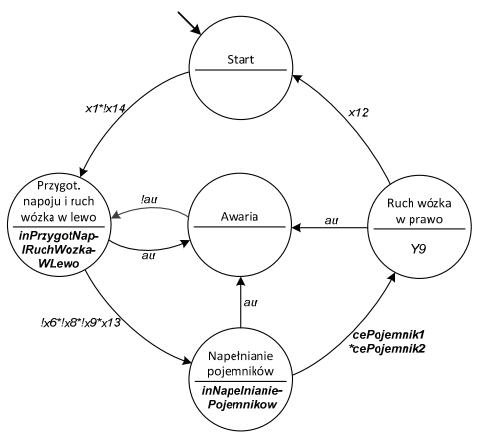
Diagramy maszyny stanowej UML pozwalają na wygodne modelowanie obsługi wyjątków także poprzez użycie przejść wysokiego poziomu (ang. *high-level transitions*). W prezentowanym przykładzie są to np. niektóre przejścia wychodzące i wchodzące do stanu „Awaria” (aktywny sygnał *au*). Zastosowanie przejść wysokiego poziomu pozwala uniknąć tworzenia przejść z każdego podrzędnego stanu do stanu „Awaria”, co wpłynęłoby negatywnie na czytelność modelu.

Zaletą proponowanej metody jest możliwość modelowania UML z wykorzystaniem pseudostanu historii [10] i sposób jego implementacji w rzeczywistym układzie cyfrowym. Występujący

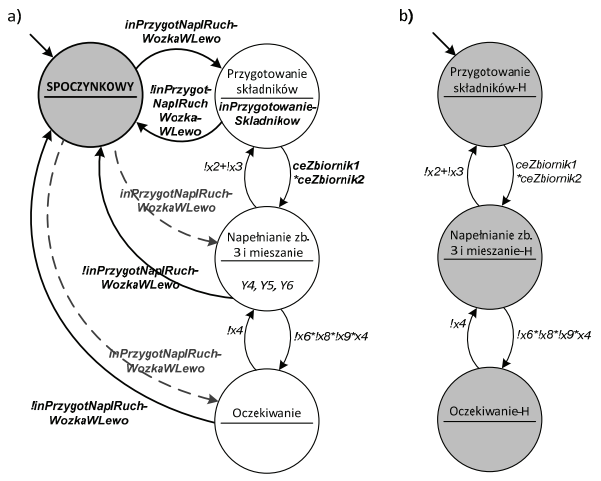
na rys. 3 symbol pseudostanu historii (litera H w okręgu) oznacza, że po uaktywnieniu stanu „Przygotowanie napoju i ruch wózka w lewo" aktywnymi stanami (po jednym w każdym z obszarów współbieżnych) będą te, które były aktywne ostatnio w momencie przekazania sterowania z tego stanu. Jeżeli stan „Przygotowanie napoju i ruch wózka w lewo" staje się aktywnym po raz pierwszy, sterowanie zostanie przekazane do tych stanów, na który jawnie wskazuje pseudostan historii. Zastosowanie pseudostanów historii znacznie upraszcza modelowanie takiego zachowania. Alternatywna konstrukcja oznaczałaby konieczność dodania nadmiarowych przejść lub stanów oraz specjalnych sygnałów.

5. Translacja maszyny stanów UML na język opisu sprzętu

Po opracowaniu diagramu maszyny stanowej UML opisującego zachowanie układu, w kolejnym kroku proponowanej metody następuje dekompozycja maszyny stanów UML na działające współbieżnie hierarchiczne automaty FSM. W prezentowanym przykładzie Mieszalnika diagram UML jest dekomponowany na automat nadrzędny „Proces" (rys. 4) oraz automaty podrzędne: „Wózek", „Zbiorniki" (rys. 5), „Pojemnik1" oraz „Pojemnik2". Dodatkowo automat „Zbiorniki" koordynuje pracę automatów „Zbiornik1" i „Zbiornik2".



Rys. 4. Automat nadrzędny „Proces" z dodanymi sygnałami
Fig. 4. A master FSM "Proces" with added signals



Rys. 5. a) automat podrzędny „Zbiorniki", b) dodatkowy automat „Zbiorniki-H" przechowujący ostatni aktywny stan
Fig. 5. a) a slave FSM "Zbiorniki" , b) an extra FSM "Zbiorniki-H" storing the last active state

W kolejnym kroku automaty FSM pełniące role nadrzędne uzupełniane są o dodatkowe sygnały, powiązane z aktywnością ich automatów podrzędnych (np. *inNapelnianiePojemnikow*). Do każdego podrzędnego automatu FSM dodawany jest także stan spoczynkowy, w którym znajduje się on, gdy powiązany z nim stan w automacie nadrzędnym jest nieaktywny oraz odpowiednie przejścia do stanu spoczynkowego (rys. 5).

Na szczególną uwagę zasługują automaty, które na diagramie UML modelowane były z wykorzystaniem pseudostanów historii. Jednym z nich jest automat „Zbiorniki" (rys. 5a). Występujące na rysunku przejścia ze stanu neutralnego do poprzednio aktywnego stanu oznaczono linią przerywaną. W proponowanej metodzie pseudostany historii implementowane są w docelowym układzie cyfrowym jako specjalne, dodatkowe rejestry, przechowujące informacje o poprzednio aktywnych stanach. Konsekwencją takiego rozwiązania jest utworzenie dla każdego automatu zawierającego pseudostan historii jego uproszczonej kopii (np. „Zbiorniki-H" na rys. 5b), której jedynym zadaniem jest przechowywanie informacji o ostatnio aktywnym stanie. Kiedy sterowanie przekazywane jest ze stanu neutralnego to właśnie ten automat wskazuje, które przejście ma zostać zrealizowane.

Wszystkie zdekomponowane automaty tworzą hierarchiczną strukturę komunikujących się automatów FSM. Automaty na tym samym poziomie hierarchii mogą pracować równolegle, podczas aktywności automatu nadrzędnego. Strukturę tę można potraktować jako syntezowalny model HCFSM.

W ostatnim etapie każdy automat FSM konwertowany jest do języka opisu sprzętu Verilog. W efekcie powstaje zbiór powiązanych ze sobą plików, dzięki czemu modularna struktura HCFSM jest w pełni odwzorowywana w języku opisu sprzętu. Szczegóły implementacyjne proponowanej metody znajdują się w [6].

6. Podsumowanie

W artykule przedstawiono sprawdzoną eksperymentalnie koncepcję odwzorowania hierarchiczno-współbieżnej maszyny stanów, przedstawionej w sposób behawioralny intuicyjnym diagramem UML, na równoważny mu pod względem semantycznym opis tekstowy w języku Verilog. W proponowanej metodzie maszyna stanów jest dekomponowana na uporządkowaną hierarchiczno-współbieżną sieć automatów cyfrowych (HCFSM). Zaletą zastosowanej metody implementacji jest możliwość niezależnej symulacji, syntezy i optymalizacji każdego z modułów (automatu FSM). Wprowadzanie poprawek do modelu najczęściej nie pociąga za sobą konieczności wygenerowania całego układu, lecz tylko tego automatu, w którym dokonano zmiany.

7. Literatura

[1] OMG. Unified Modeling Language. Superstructure. v2.4.1. www.omg.org/spec/UML/2.4.1, 2011.
[2] Wood S., Akehurst D., Uzenkov O., Howells W., McDonald-Maier K.: A Model Driven Development Approach to Mapping UML State Diagrams to Synthesizable VHDL. IEEE Transactions on Computers, Vol. 57, No 10, 2008.
[3] Łabiak G., Karatkevich A..: Metody specyfikacji, syntezy i weryfikacji hierarchicznych diagramów stanów, PAK, nr 7, s. 109-111, 2006.
[4] Valette R.: Etude comparative de deux outils de représentation: Grafcet et Réseaux de Petri. Le Nouvel Automatisme, nr 3, str. 377-382, 1978.
[5] Bazydło G., Adamski M.: Specyfikacja hierarchicznej maszyny stanów UML 2.4 i jej automatyczna implementacja w języku Verilog. Przegląd Elektrotechniczny, nr 11, s. 145-149, 2011.
[6] Bazydło G., Adamski M.: Modelowanie sytuacji awaryjnych w hierarchicznej maszynie stanów UML 2.4 z wykorzystaniem atrybutu historii. Projektowanie, analiza i implementacja systemów czasu rzeczywistego, red. L. Trybus, S. Samolej, Wydaw. Komunikacji i Łączności, s. 41-52, Warszawa, 2011.
[7] Booch G., Rumbaugh J., Jacobson I.: UML przewodnik użytkownika. WNT, Warszawa, 2001.
[8] Bazydło G.: Synteza behawioralna sterowników rekonfigurowalnych na podstawie modelu maszyny stanowej UML. PAK, Nr 7, s. 508-510, 2009.

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*