Grzegorz BAZYDŁO
UNIWERSYTET ZIELONOGÓRSKI

Synteza behawioralna sterowników rekonfigurowalnych na podstawie modelu maszyny stanowej UML

Mgr inż. Grzegorz BAZYDŁO

Jest absolwentem Uniwersytetu Zielonogórskiego (2004). Ukończył studia na specjalności Inżynieria Komputerowa. Obecnie jest doktorantem na Wydziale Elektrotechniki, Informatyki i Telekomunikacji UZ. Zainteresowania naukowe koncentrują się wokół nowoczesnych metod projektowania specjalistycznych układów cyfrowych.

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

Streszczenie

W pracy przedstawiono nową metodę projektowania sterowników logicznych realizowanych w sposób układowy w strukturach FPGA z wykorzystaniem języka Verilog. Modelem behawioralnym programu sterownika jest diagram maszyny stanowej UML 2.1.2. Formalnym modelem strukturalnym jest hierarchiczna sieć współpracujących ze sobą automatów cyfrowych. Wynikiem jest modularny opis modelowanego systemu w języku opisu sprzętu Verilog. Taka specyfikacja tekstowa może być następnie poddana symulacji i syntezie w zewnętrznych systemach.

Słowa kluczowe: UML, rekonfigurowalny sterownik, Verilog, FPGA.

Behavioural synthesis of reconfigurable controllers based on UML state machine model

Abstract

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting artifacts of software systems [9], as well as for business modelling and other non-software, for example reactive, systems [1, 8, 10]. The UML represents a collection of the best engineering practices that have proven successful in modelling large and complex systems [14]. The current version of the language is 2.1.2 [6]. One of the UML diagrams is a state machine diagram that defines a set of concepts that can be used for modelling discrete behavior through finite state transition systems. The paper presents a new design method for reconfigurable logic controllers implemented as digital circuit in Field Programmable Gate Arrays (FPGA) by means of hardware description language Verilog. The UML state machine diagram is used as an initial behavioural model [5]. It is worth mentioning that state machine diagrams support various features of the modelling systems such as hierarchy and orthogonality [12]. Figure 2 shows a state machine diagram for the exemplary model of two trolleys control process (Fig. 1) [2]. The formal structured design model is based on the hierarchical network of collaborated Finite State Machines [3, 15]. The specification in Verilog can be simulated and synthesized in professional tools, e.g. Active HDL or Xilinx ISE. To verify presented method a special CAD system UML-XML2Verilog was designed. This system allows automating the translation process from UML diagrams (described in XML) to behavioural, synthesized specification in Verilog. As for future research, the use of other diagrams from UML is going to be investigated.

Keywords: UML, reconfigurable controller, Verilog, FPGA.

1. Wstęp

Opracowano wiele metod projektowania układów cyfrowych, jednak przy modelowaniu złożonego zachowania, tradycyjne metody okazują się być często niewystarczające. Brak pojęć abstrakcyjnych zmusza projektanta do operowania dużą liczbą szczegółów, co sprawia, że specyfikacja jest mało czytelna, a samo projektowanie trudniejsze. Ponadto niektóre z metod nie odpowiadają już dzisiejszym potrzebom w zakresie wykorzystania dostępnych zasobów sprzętowych, w tym zwłaszcza mikrosystemów cyfrowych SoC (ang. *System on Chip*). Dostawcy technologii oferują projektantom układy programowalne zawierające nawet 10 milionów bramek. Coraz większego znaczenia nabiera więc modelowanie abstrakcyjne (na poziomie systemu), którego zaletą jest zwiększenie możliwości modelowania funkcjonowania, a projektowane urządzenia mogą charakteryzować się większymi i szerszymi możliwościami zastosowań. Podejście abstrakcyjne ma też pewne ograniczenia mające wpływ na etapy projektowania i na funkcjonalność projektowanego układu [1], które jednak w porównaniu z zaletami (modularność i możliwość wykorzystania już wcześniej przetestowanych bloków), wydają się być mniej istotne.

Istnieje wiele metod pozwalających w sposób graficzny modelować zachowanie sterownika (np. sieci SFC [2], sieci Petriego [3], diagramy stanów [4]), Są one bardziej intuicyjne i łatwiejsze w zrozumieniu niż metody specyfikacji tekstowej. Te drugie, w tym języki HDL, lepiej nadają się do dalszego przetwarzania. Format tekstowy jest też często jedyną akceptowalną specyfikacją dla zaawansowanych systemów do symulacji, syntezy i implementacji sterowników logicznych. Niestety brak jest uniwersalnej metody łączącej zalety metody graficznej i tekstowej.

Zagadnienia syntezy diagramów UML (ang. *Unified Modeling Language*) w postaci struktur cyfrowych są tematem aktualnym [5], dopiero pojawiającym się w literaturze przedmiotu. W artykule zaprezentowano koncepcję syntezy behawioralnej modeli opisanych diagramami maszyny stanowej UML [6]. Wynikiem jest modularny opis modelowanego systemu w języku opisu sprzętu Verilog. Taka specyfikacja tekstowa może być następnie poddana symulacji i syntezie w zewnętrznych środowiskach. W celu praktycznej weryfikacji metody zaprojektowano system CAD umożliwiający automatyczną translację modeli zapisanych w języku UML do języka opisu sprzętu Verilog.

2. Rekonfigurowalny sterownik logiczny

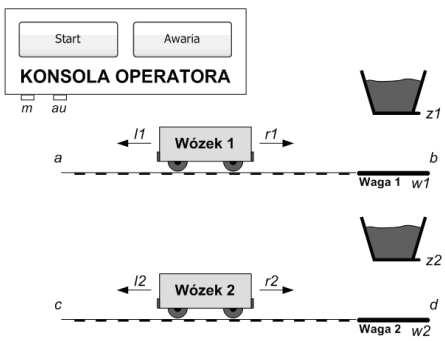
Rekonfigurowalny sterownik logiczny [7] można potraktować jako przykład systemu reaktywnego [8]. W systemach tych przyjmuje się, że dane wejściowe mogą pojawiać się w dowolnym momencie, co więcej, oczekuje się natychmiastowej reakcji systemu na te zdarzenia. Ponadto systemy reaktywne mogą być sterowane zdarzeniami, prowadzą stałą interakcję z otoczeniem (z użyciem sygnałów i przerwań), zmieniają swój stan w zależności od bieżącego i przeszłego zachowania oraz są to często systemy współbieżne [1]. Zaproponowana w pracy metoda projektowania jest szczególnie przydatna dla implementacji układowych w rekonfigurowalnych sterownikach logicznych (RLC – ang. *Reconfigurable Logic Controller*) z wbudowaną matrycą makrokomórek CLB (ang. *Configurable Logic Block*) np. układach FPGA (ang. *Field Programmable Gate Array*). Nie mniej jednak istnieje możliwość zastosowania proponowanej koncepcji także do symulacji i weryfikacji programów dla sterowników typu PLC (ang. *Programmable Logic Controller*). W tym celu pośredni model zostanie potraktowany jako modularna struktura współpracujących ze sobą bloków funkcyjnych FBD (ang. *Function Block Diagram*) i przedstawiony zgodnie ze standardem IEC 1131-3 [2].

Projektowanie formalne sterowników logicznych z wykorzystaniem proponowanej metody składa się z następujących etapów:

a) specyfikacja behawioralna z wykorzystaniem diagramów maszyny stanowej UML 2.1.2,
b) opis diagramów w języku znaczników XML (ang. *Extensible Markup Language*),
c) stworzenie pośredniego modelu funkcjonowania sterownika jako struktury hierarchiczno-współbieżnej złożonej z klasycznych maszyn stanów,
d) dekompozycja hierarchiczno-współbieżnej maszyny stanów na współpracujące podmaszyny,

e) konstrukcja sekwencyjnego układu koordynującego funkcjonowanie podmaszyn stanowych,
f) opis podmaszyn oraz układu koordynującego w języku opisu sprzętu Verilog,
g) symulacja w systemie CAD,
h) prototypowanie w strukturze FPGA.

Praktyczne wykorzystanie metody zaprezentowano na przykładzie procesu sterowania ruchem dwóch wózków, wzorowanym na przykładach zamieszczonych w książce [2]. Wybierając przykład zwrócono uwagę na takie elementy jak modularność, współbieżność, sekwencyjność, hierarchia i obsługa wyjątków. Schemat tego procesu przedstawiono na rys. 1. Proces rozpoczyna się po naciśnięciu przycisku *Start* (sygnał *m*). Wózki 1 i 2 startują z położenia *a* oraz *c* i poruszają się w prawo. Jeżeli w trakcie ruchu w prawo wystąpi awaria (wartość logiczna sygnału *au* wynosi „1"), praca systemu jest zatrzymywana. Po usunięciu awarii (wartość sygnału *au* wynosi „0" oznaczona jako *!au*) proces jest kontynuowany. Po dojechaniu do punktów *b* i *d*, następuje otwarcie zsypów *z1* oraz *z2* i załadowanie wózków. Po załadowaniu powrót wózków następuje w kolejności: najpierw wraca wózek 1, a gdy ten dojedzie do punktu *a*, wówczas wraca wózek 2.



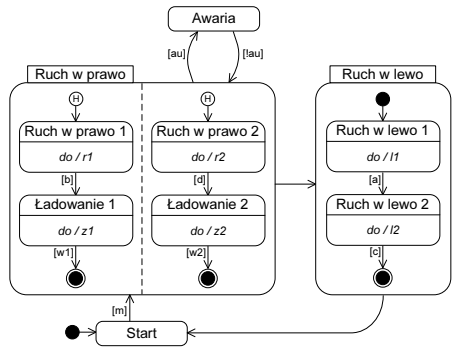
Rys. 1. Schemat przykładowego procesu technologicznego
Fig. 1. Case study of technological discrete process

3. Modelowanie sterownika w języku UML

Pierwszym krokiem proponowanej metody jest zamodelowanie w języku UML zachowania projektowanego sterownika logicznego. Język UML (wersja 2.1.2 [6]) to język modelowania wizualnego, służący do obrazowania, specyfikowania, tworzenia i dokumentowania szeroko pojętych systemów informatycznych [9], a także tych nie związanych bezpośrednio z oprogramowaniem [10]. Podstawowym środkiem oferowanym przez język UML są diagramy, które można traktować jako szkic funkcjonowania systemu. Najbardziej przydatne z punktu widzenia wykorzystania do specyfikacji behawioralnej sterowników logicznych wydają się być diagramy maszyny stanowej, ponieważ są one graficznym odzwierciedleniem dyskretnego, skokowego zachowania skończonych systemów typu stan–przejście.

Na rys. 2 przedstawiono diagram maszyny stanowej dla procesu sterowania ruchem wózków. Stany na diagramie reprezentowane są w postaci prostokątów z zaokrąglonymi rogami, a przejścia oznaczane są strzałkami. Z przejściem może być skojarzone zdarzenie uruchamiające, warunek oraz akcja wykonywana podczas realizacji przejścia. Ponieważ projektowany sterownik realizowany jest jako synchroniczny automat współbieżny z jednym sygnałem zegarowym przyjęto, że zdarzeniami uruchamiającymi są impulsy synchronizujące [11]. W związku z tym o realizacji przejścia między stanami decydują warunki, przedstawiane jako wyrażenia logiczne zbudowane z nazw sygnałów i operatorów logicznych (iloczyn i negacja), np. *[x*!w]* oznacza, że przejście będzie zrealizowane jeżeli w momencie pojawienia się impulsu zegarowego sygnał *x* będzie miał wartość logiczną „1", a *w* wartość „0".

Warto zwrócić uwagę na wspieranie przez diagramy maszyny stanowej takich cech układu jak hierarchiczność, współbieżność oraz obsługę wyjątków [12] (np. poprzez atrybut historii [6]). Pozwala to w sposób intuicyjny i czytelny specyfikować zachowanie złożonych systemów współbieżnych na wybranym poziomie uszczegółowienia [13]. W prezentowanym przykładzie stany *Ruch w prawo* i *Ruch w lewo* to stany złożone. Występujący na rysunku symbol pseudostanu historii płytkiej [14] (litera *H* w okręgu) oznacza, że po uaktywnieniu stanu *Ruch w prawo* pierwszym aktywnym stanem będzie ten, który był aktywny ostatnio w momencie przekazania sterowania. Jeżeli stan *Ruch w prawo* staje się aktywnym po raz pierwszy, sterowanie zostanie przekazane do tego stanu, na który wskazuje pseudostan historii.

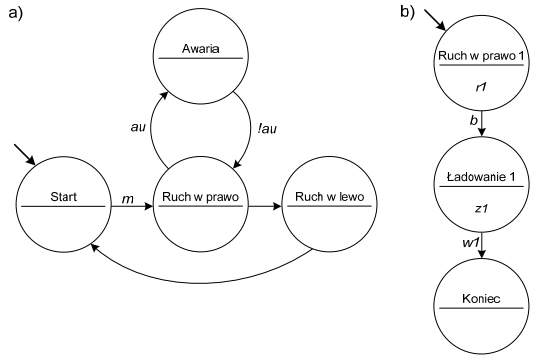


Rys. 2. Diagram maszyny stanowej
Fig. 2. State machine diagram

Na rynku istnieje wiele narzędzi do modelowania w języku UML. Zaletą proponowanej metody jest możliwość wykorzystania praktycznie dowolnego oprogramowania UML, także darmowego, co pozwala zmniejszyć koszty projektowania. Ważne tylko, aby wybrane narzędzie dysponowało możliwością generowania specyfikacji w języku XML, gdyż właśnie ten format wybrano jako wejściowy do zaprojektowanej aplikacji *UML-XML2Verilog*.

4. Opis maszyny stanowej w języku Verilog

Kiedy sterownik logiczny zostanie zamodelowany za pomocą diagramów UML, kolejnym etapem jest translacja jego graficznej reprezentacji na opis w języku opisu sprzętu Verilog (syntezowalny podzbiór). Opracowana metoda syntezy bazuje na metodach opisanych w [3] oraz [15], których najważniejszym elementem jest takie przekształcenie modelu, aby otrzymać hierarchiczną strukturę połączonych modułów (automatów FSM). Ze względów praktycznych modelowanie zawężono do diagramów modularnych to znaczy takich, w których brak jest przejść pomiędzy stanami na różnym poziomie hierarchii. Przekształcenia diagramów niemodularnych są przedmiotem dalszych badań autora.

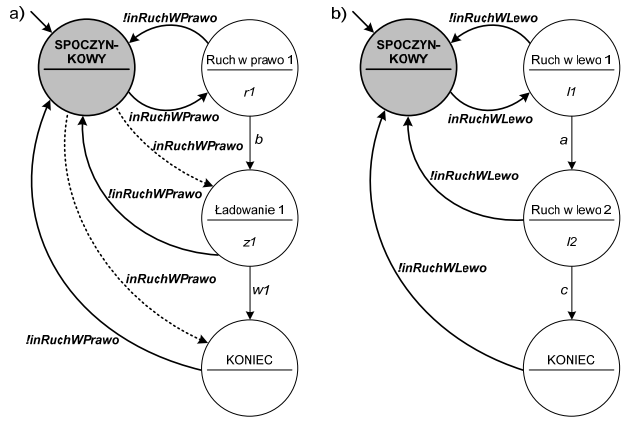


Rys. 3. Podział diagramu maszyny stanowej na automaty FSM: a) automat nadrzędny, b) przykładowy automat podrzędny
Fig. 3. Modular decomposition of UML state machine diagram: a) top FSM, b) exemplary lower level FSM

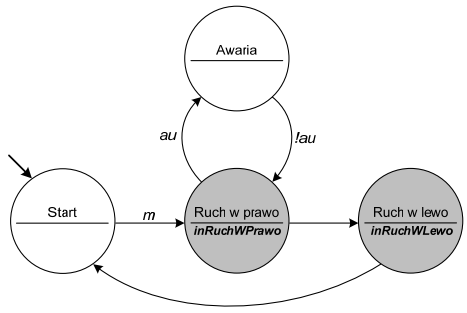
Pierwszym krokiem etapu translacji jest podział modelowanej maszyny stanowej (diagram UML) na automaty FSM (rys. 3). Następnie do każdego podrzędnego automatu FSM dodawany jest stan spoczynkowy (neutralny), do którego przekazywane jest sterowanie w czasie, kiedy powiązany automat nadrzędny jest nieaktywny. Oprócz stanu spoczynkowego dodawane są także specjalne przejścia do tego stanu. Zmodyfikowane przykładowe

automaty FSM przedstawia rys. 4. Występujące na rys. 4a przejścia oznaczone linią przerywaną związane są z realizacją atrybutu historii (z rys. 2) w automacie FSM i oznaczają alternatywne przekazanie sterowania w zależności od poprzedniej aktywności automatu. Sposób realizacji historii w automacie FSM w języku Verilog szczegółowo przedstawiono w pracy [16].

Kolejnym krokiem jest uzupełnienie nadrzędnych automatów FSM o dodatkowe sygnały związane z aktywnością poszczególnych automatów podrzędnych (rys. 5). Jeżeli np. automat nadrzędny znajduje się w stanie złożonym *Ruch w prawo* generowany jest sygnał *inRuchWPrawo*, co z kolei powoduje aktywność wszystkich automatów podrzędnych, wrażliwych na zmianę tego sygnału: *Ruch wózków w prawo 1* i *Ruch wózków w prawo 2*.

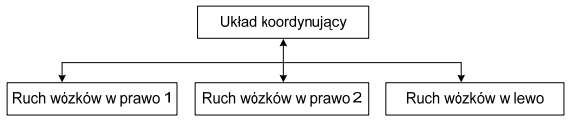


Rys. 4. Wybrane automaty FSM z dodanymi przejściami i stanami bezczynności
Fig. 4. Some of FSMs with additional special transitions and idle states



Rys. 5. Nadrzędny automat FSM z uzupełnionymi sygnałami
Fig. 5. Top FSM with additional special signals

Wyszczególnione przekształcenia prowadzą do stworzenia specjalnego modelu, w którym poszczególne moduły tworzą hierarchiczną strukturę komunikujących się automatów FSM (rys. 6). Automaty na tym samym poziomie hierarchii mogą pracować równolegle, oczywiście przy aktywności automatu nadrzędnego. Strukturę tę można więc potraktować jako model HCFSM (ang. *Hierarchical Concurrent Finite State Machine*) [3, 12]. Dla prezentowanego przykładu struktura ta posiada tylko dwa poziomy hierarchii, co nie oznacza, że opracowana metoda nie może być stosowana dla układów z bardziej rozbudowaną hierarchią.



Rys. 6. Hierarchiczna struktura połączonych automatów FSM
Fig. 6. Hierarchical structure of linked FSMs

W ostatnim etapie każdy automat FSM konwertowany jest do języka opisu sprzętu Verilog. Translacja dokonywana jest w oparciu o opracowane w języku Verilog specjalne szablony, szczegółowo przedstawione w pracy [16]. Specyfikację w języku Verilog można poddać symulacji i syntezie w zewnętrznych środowiskach np. *Active-HDL* firmy Aldec lub *Xilinx ISE* firmy Xilinx.

5. Podsumowanie i kierunki dalszych prac

Zaproponowana metoda bazuje na graficznej reprezentacji sterownika logicznego. Zastosowanie diagramów UML pozwala z jednej strony w sposób stosunkowo łatwy i intuicyjny na specyfikację behawioralną układu, z drugiej natomiast otwiera możliwość wykorzystania do tego celu, często darmowego, oprogramowania UML (obniżenie kosztów projektowania). Diagramy maszyny stanowej umożliwiają kompletne i jednoznaczne specyfikowanie zachowania projektowanego systemu. Specyfikacja behawioralna algorytmu sterowania binarnego, przedstawiona w postaci użytecznego, nienadmiarowego i zwartego podzbioru diagramów maszyny stanowej, może być potraktowana jako abstrakcyjna forma programu dla sterownika logicznego.

Kierunki dalszych prac skupiają się wokół pełnej automatyzacji procesu projektowania sterowników rekonfigurowalnych, opisanych podzbiorem diagramów maszyny stanowej UML. Zaprojektowano i zrealizowano system CAD (*UML-XML2Verilog*), w którym diagramy maszyny stanowej UML (zunifikowany zapis XML) są najpierw przekształcane na wewnętrzny, strukturalny model symulacyjny, a następnie na jego podstawie generowany jest opis układu w języku Verilog. Ponieważ wybrano syntezowalny podzbiór tego języka, akceptowalne wyniki symulacji mogą stanowić podstawę do automatycznej syntezy. Obecnie prowadzone prace dotyczą także możliwości wykorzystania innych diagramów UML (np. przypadków użycia, czynności, klas).

6. Literatura

[1] G. Łabiak: Wykorzystanie hierarchicznego modelu współbieżnego automatu w projektowaniu sterowników cyfrowych. Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra, 2005.
[2] M. Adamski, M. Chodań: Modelowanie układów sterowania dyskretnego z wykorzystaniem sieci SFC. Wydawnictwo Politechniki Zielonogórskiej, Zielona Góra, 2000.
[3] M. Adamski: Petri Nets in ASIC Design. Applied Mathematics and Computer Science, vol. 3, WSI w Zielonej Górze, 1993.
[4] D. Harel: Statecharts, A visual formalism for complex Systems. Science of Computer Programming, Vol. 8, 1987.
[5] S. Wood, D. Akehurst, O. Uzenkov, W. Howells, K. McDonald-Maier: A Model Driven Development Approach to Mapping UML State Diagrams to Synthesizable VHDL. IEEE Transactions on Computers, vol. 57, Nr 10, 2008.
[6] OMG: OMG Unified Modeling Language, Superstructure, V2.1.2. http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF, 2007.
[7] Adamski M.: Logic design of reconfigurable logic controllers, IEEE Second International Symposium on Industrial Embedded Systems, SIES'07, Lizbona, 2007, str. 373-376.
[8] D. Harel, M. Politi: Modeling Reactive Systems With Statecharts: The Statemate Approach. McGraw Hill Text, 1998.
[9] G. Booch, J. Rumbaugh, I. Jacobson: UML przewodnik użytkownika. WNT, Warszawa, 2001.
[10] G. Bazydło, M. Adamski: Graficzna specyfikacja programów dla sterowników logicznych z wykorzystaniem języka UML, Materiały VII Krajowej Konferencji Naukowej Reprogramowalne Układy Cyfrowe 2005 – RUC'05, Szczecin, 2005.
[11] P. Minns, I. Elliott: FSM based Digital Design using Verilog HDL, John Wiley & Sons Ltd, Chichester, Anglia, 2008.
[12] D. Gajski, F. Vahid, S. Narayan, J. Gong: Specification and Design of Embedded Systems. PTR Prentice Hall, New Jersey, USA, 1994.
[13] G. Bazydło: Specyfikacja behawioralna dla rekonfigurowalnych sterowników logicznych z wykorzystaniem diagramów maszyny stanowej z języka UML 2.0. Pomiary Automatyka Kontrola, 5'2007, Vol. 53, 2007.
[14] W. Dąbrowski, A. Stasiak, M. Wolski: Modelowanie systemów informatycznych w języku UML 2.1. Wydawnictwo Naukowe PWN, Warszawa, 2007.
[15] D. Drusinsky, D. Harel: Using Statecharts for Hardware Description and Synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 8, 1989.
[16] G. Bazydło, M. Adamski: Projektowanie sterowników logicznych opisanych diagramami maszyny stanowej UML. Czasopismo Techniczne, seria Informatyka, Politechnika Krakowska, 1-I/2008, 2008.

Artykuł recenzowany