ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2013
Seria: TRANSPORT z. 80 Nr kol. 1895

Krzysztof FIRLĄG[1], Piotr KAWALEC[2]

BUDOWA MODELI SPECJALIZOWANYCH STEROWNIKÓW RUCHU DROGOWEGO W JĘZYKACH OPISU SPRZĘTU

Streszczenie. W artykule przedstawiono metodę budowy modeli specjalizowanych sterowników ruchu drogowego realizowanych w języku VHDL. Rozwiązaniem problemu braku reprezentacji graficznej i konieczności bardzo dobrej znajomości języka VHDL przez projektanta jest wykorzystanie oprogramowania narzędziowego CAD, pozwalającego na projektowanie urządzeń srd w sposób przyjazny i intuicyjny dla inżyniera sterowania ruchem.

W artykule zaproponowano metodę budowy sterownika opierając się na modelach specyfikacji formalnej mającej graficzną reprezentację. Pierwszym etapem jest zamiana algorytmów sterowania ruchem drogowym w sieć działań GSA. Przedstawiono metodę konwersji dla wszystkich rodzajów klatek algorytmu sterowania. Następnie pokazano sposób konwersji sieci GSA w graf przejść automatu skończonego FSM, gdzie w celu zwiększenia czytelności grafu sterowania zaproponowano wprowadzenie stanów hierarchicznych, dla obsługi przejść międzyfazowych i faz ruchu. Wykorzystując hierarchiczny graf sterowania zaproponowano uniwersalną strukturę logicznego sterownika ruchu drogowego. Sterownik ten wyspecyfikowano w programie Active-HDL, który wygenerował model sterownika logicznego w języku VHDL.

Słowa kluczowe. Algorytmy sterowania ruchem drogowym, specjalizowane sterowniki ruchu drogowego, układy programowalne

CONSTRUCTION OF SPECIALIZED MODELS OF ROAD TRAFFIC CONTROLLERS IN HARDWARE DESCRIPTION LANGUAGES

Summary. The paper presents the construction method of specialized models of road traffic controllers realized within VHDL language. The designer solves the problem of lack of graphic representation and the necessity of a very good command of VHDL language by using utility software CAD allowing for designing traffic control devices that would be pleasant and intuitive in use for the traffic control engineer.

The construction method of a controller has been proposed in the article on the basis of formal specification models having intuitive graphic representations. The first stage consists in changing the algorithms of road traffic control into a network of generalized stochastic automata (GSA) activities. The method of conversion has been presented for all types of control algorithm frames. Afterwards, the way of conversion of GSA network into the transition graph of finite state machine (FSM) was presented, where in order to increase

[1] Zakład Sterowania Ruchem, Wydział Transportu, Politechnika Warszawska, ul. Koszykowa 75, 00-662 Warszawa, tel. (+48 22) 234758, kfr@it.pw.edu.pl
[2] Zakład Sterowania Ruchem, Wydział Transportu, Politechnika Warszawska, ul. Koszykowa 75, 00-662 Warszawa, tel. (+48 22) 234758, pka@it.pw.edu.pl

the clarity of control graph, it was proposed to introduce hierarchical states for interstage transitions as well as traffic phases. With the use of hierarchical graph of control, universal structure of logic road traffic controller has been proposed. This controller has been specified within Active-HDL program which generated a model of logic controller in VHDL language.

Keywords. Traffic Control Algorithms, Specialized Traffic Controllers, Programmable Devices

1. WPROWADZENIE

We współczesnych systemach sterowania ruchem drogowym coraz większą rolę odgrywają inteligentne systemy transportowe ITS, które skupiają się głównie na sterowaniu obszarowym, integrując w tym procesie wiele podsystemów. Jednak pomimo intensywnego rozwoju ITS, nadal główne decyzje dotyczące sterowania i zapewnienia bezpieczeństwa uczestnikom ruchu zapadają na lokalnym poziomie systemów sterowania.

Urządzeniami sterującymi w systemach sterowania ruchem, niezależnie od złożoności systemu, są zwykle sterowniki lokalne ruchu drogowego. Podstawową funkcją tych sterowników jest realizacja bezpośredniego sterowania ruchem w obrębie skrzyżowania.

Starsze sterowniki pracujące w otwartym układzie sterowania realizowały proste sterowania cykliczne. Współczesne sterowniki ruchu drogowego pracują głównie w zamkniętych układach sterowania, realizując sterowanie zależne od ruchu, adaptacyjne. Sterowniki te, poza bezpośrednim sterowaniem, mają wiele innych funkcji – przetwarzają i archiwizują dane o ruchu pojazdów, za pomocą wielu czujników (detektorów ruchu) monitorują i analizują sytuacje na skrzyżowaniu, odpowiadają za bezpieczeństwo procesu transportowego. Większość tych zadań realizowana jest w czasie rzeczywistym.

Idea sterowania ruchem we współczesnych sterownikach lokalnych ruchu drogowego opiera się na realizacji przez sterownik algorytmu sterowania ruchem, czyli skończonego, uporządkowanego ciągu jasno zdefiniowanych czynności, koniecznych do wykonania założonej funkcji sterowania ruchem. Algorytm powinien być opracowany stosownie do celu sterowania na podstawie formalnie sprecyzowanych informacji o obiekcie, znanych zakłóceniach i z zastosowaniem racjonalnych metod sterowania.

Sterowniki ruchu, początkowo budowane z elementów dyskretnych, realizowały algorytm sterowania w sposób sprzętowy, algorytm wynikał z budowy sterownika. Współcześnie stosowane sterowniki to głownie rozwiązania mikroprocesorowe, realizujące algorytm sterowania programowo [6]. Analiza rozwiązań sterowników, zarówno starszych rozwiązań sprzętowych, jak i nowszych procesorowych, wykazuje wiele problemów związanych z tymi rozwiązaniami.

Alternatywą jest połączenie obydwu sposobów realizacji algorytmów, czyli powrót do sprzętowej realizacji algorytmów sterowania, z wykorzystaniem techniki umożliwiającej programową rekonfigurację i zmianę programów sterownika. Możliwości takie daje wykorzystanie programowalnych struktur logicznych przy budowie układów sterowników ruchu drogowego [5]. Układy programowalne pozwalają na budowę szybkich i wydajnych urządzeń sterowania, współbieżnie realizujących algorytmy sterowania wraz z obsługą w czasie rzeczywistym wielu portów wejściowych. Urządzenia te są całkowicie sprzętowe, jednak wyjątkowo łatwo rekonfigurowalne. Współczesne reprogramowalne układy cyfrowe PLD, CPLD, FPGA dzięki swojej konstrukcji umożliwiają budowę kompletnych systemów cyfrowych w postaci jednego układu scalonego [7]. Rozwiązania takie mogą egzystować

samodzielnie lub jako systemy osadzone, czyli samodzielne jednostki zadaniowe wewnątrz innych systemów.

Narzędzia syntezy do opisu urządzeń realizowanych w układach programowalnych wymagają języków opisu sprzętu. Fakt ten narzuca model opisu projektowanego układu, czyli jeden z języków opisu sprzętu. Języki opisu sprzętu, jak VHDL, pozwalają modelować zarówno strukturą układu (opis strukturalny), jak i funkcje układu (opis behawioralny). Jednak języki te nie mają reprezentacji graficznej, przez co znacznie maleje przejrzystość modelowanego za ich pomocą systemu. Modelowanie urządzeń sterowania za pomocą VHDL wymaga bardzo dobrej znajomości tego języka, praktycznie ograniczając krąg projektantów do programistów, przez co wykorzystanie języków VHDL do modelowania urządzeń srd jest ograniczone.

Rozwiązaniem problemu jest oprogramowanie narzędziowe, pozwalające na projektowanie urządzeń srd w sposób przyjazny i intuicyjny dla inżyniera sterowania ruchem. Oprogramowanie takie powinno pozwalać na specyfikację algorytmów sterowania ruchem drogowym za pomocą przejrzystego modelu graficznego, a następnie automatycznie dokonywać konwersji w model oparty na języku opisu sprzętu, niezbędny na etapie syntezy i implementacji urządzeń.

W procesie budowy sterownika logicznego, niezbędna jest analiza form zapisu algorytmów adaptacyjnego sterowania ruchem na skrzyżowaniu w celu określenia uniwersalnej formy zapisu algorytmów. Niezbędna jest również analiza modeli specyfikacji formalnej sterowników cyfrowych i wybór korzystnej dla tego przypadku metody. Ostatecznie konieczna jest uniwersalna metoda zapisu algorytmów sterownia ruchem w formie wybranego modelu specyfikacji sterowników cyfrowych.

Istotną sprawą jest intuicyjność budowy modelu, w związku z tym nacisk położono na wykorzystanie modeli mających graficzne reprezentację i specjalistyczne narzędzia CAD, umożliwiające specyfikację graficznych forma modeli i ułatwiające otrzymanie opisu urządzenia w języku VHDL.

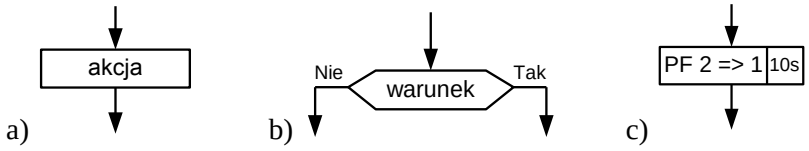
2. METODY ZAPISU ALGORYTMÓW STEROWANIA RUCHEM DROGOWYM

Większość współcześnie stosowanych lokalnych sterowników ruchu drogowego realizuje sterowanie adaptacyjne do warunków ruchu na skrzyżowaniu (sterowanie w układzie zamkniętym). Projektowanie sygnalizacji adaptacyjnej dla skrzyżowania jest procesem żmudnym i wieloetapowym. Wytyczne, co do projektowania sygnalizacji zawiera Rozporządzenie ministra infrastruktury z dnia 3 lipca 2003 r. [2]. Określa ono wiele etapów, które musi zawierać projekt sygnalizacji na skrzyżowaniu. W przypadku sygnalizacji adaptacyjnej efektem prac projektowych jest algorytm sterowania ruchem na skrzyżowaniu, opisujący warunki zakończenia i rozpoczęcia poszczególnych faz ruchu, opracowany stosownie do celu sterowania i z zastosowaniem racjonalnych metod sterowania.

W powyższym rozporządzeniu brak jest wytycznych dotyczących sformalizowania sposobu przedstawienia algorytmu sterowania adaptacyjnego. W związku z tym, w projektach sygnalizacji świetlnej spotyka się różne formy przedstawiania tego algorytmu:

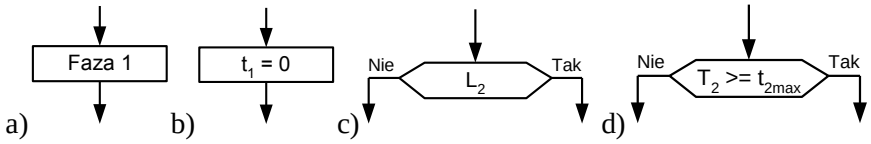
- opisowe, w postaci kolejnych punktów, przedstawienie sekwencji i warunków zmiany faz ruchu, np.: faza F1 po czasie t2 przejdzie w fazę F2 itp.,
- w postaci tabeli, gdzie w wierszach i kolumnach znajdują się poszczególne fazy ruchu, natomiast na ich przecięciu opisane są warunki przejścia pomiędzy nimi,
- w postaci przypominającej sieć działań (graf algorytmiczny AMS, graficzny schemat algorytmu GSA).

Pomimo że wszystkie opisy są akceptowalne, najbardziej czytelną i intuicyjną jest trzecia forma opisu. Podobnie jak sieć działań zawiera ona dwa podstawowe bloki – blok stanu (rys. 1a) i blok warunku (rys. 1b).



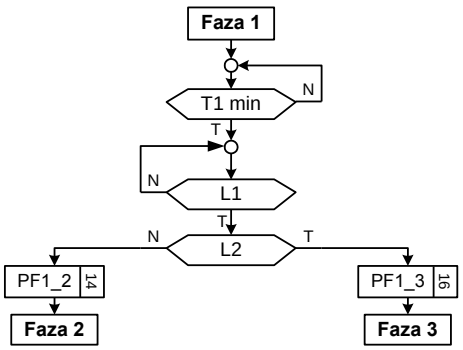
Rys. 1. Bloki stosowane w algorytmicznym zapisie programu sterowania ruchem: a) blok stanu, b) blok warunku, c) blok przejścia międzyfazowego
Fig. 1. Blocks used in writing the traffic control: a) action block, b) conditional block, c) interstage block

W zapisie algorytmu sterowania występuje również trzeci rodzaj bloków (rys. 1c), pełniący rolę bloku stanu, wywołującego sekwencję zmian stanów sygnalizacji nazywaną przejściem międzyfazowym, która ma na celu zmianę w bezpieczny sposób jednej fazy ruchu w drugą. Sposób realizacji przedziałów międzyfazowych jest wyznaczany na wcześniejszym etapie projektu sygnalizacji i jest stały dla każdej pary faz ruchu. W związku z tym nie uwzględnia się w algorytmie konkretnego przebiegu zmian stanu sygnalizacji, sygnalizując jedynie, że przejście międzyfazowe występuje w tym miejscu algorytmu i trwa określony czas. Przykłady stosowania poszczególnych bloków w algorytmie sterowania przedstawione są na rys. 2.



Rys. 2. Przykłady stosowania bloków stanu i warunkowych: a) rozpoczęcie fazy nr 1, b) wyzerowanie czasu t1, c) sprawdzenie warunku logicznego L2, d) sprawdzenie wartości czasu T2
Fig. 2. Examples of action and conditional blocks: a) start of phase 1, b) reset the time t_1, c) check the logical condition L2, d) check the time T2

Przedstawiony sposób zapisu algorytmów sterowania w zasadzie bazuje na niemieckich wytycznych zawartych w RiLSA [3]. Przykład opracowanego w ten sposób algorytmu sterowania adaptacyjnego ruchem na skrzyżowaniu przedstawiono na rys. 3. Fragment algorytmu opisuje warunki przejścia z fazy 1 do fazy 2 lub fazy 3.



Rys. 3. Fragment algorytmu sterowania ruchem na skrzyżowaniu
Fig. 3. Piece of traffic control algorithm

Budowa modelu specjalizowanego sterownika ruchu drogowego w języku VHDL na podstawie algorytmu sterowania adaptacyjnego wymaga przekształcenia algorytmów sterowania ruchem w jeden z formalnych modeli stosowanych w projektowaniu sterowników cyfrowych.

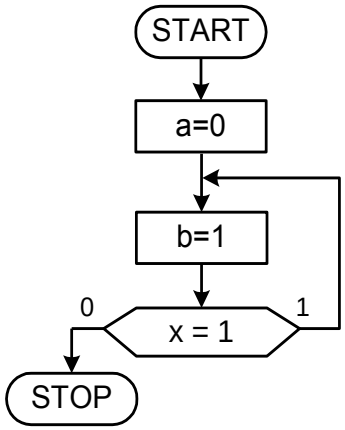
3. MODELE SPECYFIKACJI FORMALNEJ STEROWNIKÓW CYFROWYCH

Ogólnie przy projektowaniu systemów cyfrowych dzieli się je na dwie części, układ sterujący US oraz układ przetwarzania danych UO. Klasyfikacja modeli opisu formalnego dzieli systemy cyfrowe, ze względu na możliwości ich zastosowania do opisu części sterującej lub przetwarzania danych [1, 4] na:

- zorientowane na stan – modele behawioralne, zalecane do opisu układów sterowania (ASM, GSA, FSM, CFSM, HCFSM, sieci Petriego),
- zorientowane na aktywność (DFG, CFG),
- zorientowane na strukturę (schematy blokowe, ideowe, RTL),
- zorientowane na dane,
- heterogeniczne – łączące cechy poszczególnych modeli (CDFG).

Pożądaną cechą modelu jest wspieranie przez niego opisu behawioralnego działania systemu oraz reprezentacja graficzna, ponieważ umożliwia opisywanie układu na poziomie jego zachowań bez konieczności znajomości szczegółów realizacyjnych. Analiza powyższych modeli zawęziła wybór do kilku, które mają pożądane cechy.

Sieć działań GSA służy do opisu algorytmu sterowania i jest grafem skierowanym. Zawiera klatki operacyjne i warunkowe. Wierzchołki operacyjne służą do modelowania akcji wyjściowych, a decyzyjne do wprowadzania warunków wejściowych. Może zawierać również klatki START i STOP (rys. 4).



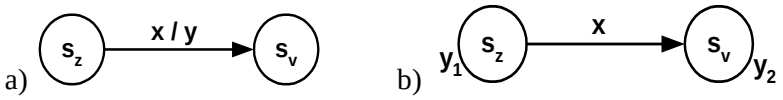
Rys. 4. Sieć działań GSA
Fig. 4. GSA

Automat skończony z pamięcią FSM (Finie State Machine) jest przykładem modelu zorientowanego na stany. Jest najbardziej popularnym modelem opisu układów sterowania. Składa się ze zbioru stanów, przejść pomiędzy nimi oraz zbioru akcji przypisanych do tranzycji lub stanów. Formalnie automat opisany jest uporządkowaną piątką [4]:

$$A = \langle S, X, Y, \delta, \lambda \rangle, \tag{1}$$

gdzie:
$X = \{x_1,\ldots,x_i\}$ – skończony niepusty zbiór wejść (alfabet wejściowy),
$S = \{s_1,\ldots,s_j\}$ – skończony niepusty zbiór stanów (alfabet wewnętrzny),
$Y = \{y_1,\ldots,y_k\}$ – skończony niepusty zbiór wyjść (alfabet wyjściowy),
δ – funkcja przejść, taka że δ: $D_\delta \rightarrow S$, przy czym $D_\delta \subseteq X \times S$,
λ – funkcja wyjść, taka że λ: $D_\lambda \rightarrow Y$, przy czym $D_\lambda \subseteq X \times S$ dla automatu Mealy'ego albo $D_\lambda \subseteq S$ dla automatu Moore'a.

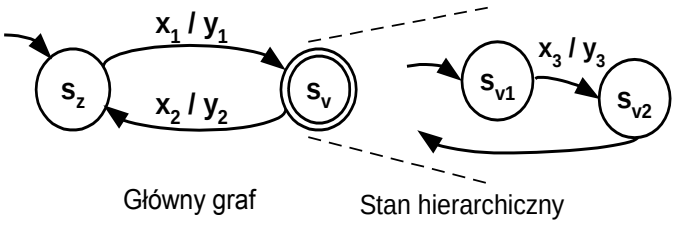
Graficznie FSM przedstawia się jako skierowany graf, zawierający stany wewnętrzne automatu w wierzchołkach i stany wejść na tranzycjach. Wyjścia na grafach opisuje się przy wierzchołkach dla automatu Moore'a (rys. 5a) lub na tranzycjach dla automatu Mealy'ego (rys. 5b).



Rys. 5. Graficzna reprezentacja: a) automatu Mealy'ego, gdzie: $\delta(x, s_z) = s_v$, $\lambda(x, s_z) = y$, b) automatu Moore'a, gdzie: $\delta(x, s_z) = s_v$, $\lambda(s_z) = y_1$, $\lambda(s_v) = y_2$
Fig. 5. Graphical specification of: a) Mealy FSM, where: $\delta(x, s_z) = s_v$, $\lambda(x, s_z) = y$, b) Moore FSM, where: $\delta(x, s_z) = s_v$, $\lambda(s_z) = y_1$, $\lambda(s_v) = y_2$

Hierarchiczny automat stanów jest modelem bazującym na automacie skończonym FSM. Jako wadę FSM uważa się to, że jest to model płaski. Powoduje to, że projektant może mieć trudności ze specyfikacją takiego modelu na wymaganym poziomie szczegółowości, ponieważ zmuszony jest operować najdrobniejszymi szczegółami specyfikacji.

Rozwiązaniem problemu przejrzystości modelu jest wprowadzenie hierarchii do FSM. W zasadzie model taki różni się od FSM jedynie w reprezentacji graficznej, zawierając fragmenty grafu „zwinięte" w jeden stan hierarchiczny (rys. 6). Synteza tego modelu nie różni się znacząco od syntezy FSM.



Rys. 6. Graficzna reprezentacja hierarchicznego FSM
Fig. 6. Graphical specification of hierarchical FSM

Przy projektowaniu systemów cyfrowych do modelowania można wykorzystać również języki opisu sprzętu. Dwa najczęściej wykorzystywane to VHDL i Verilog. Języki te pozwalają na modelowanie zarówno behawioralne, jak i strukturalne systemu. Wadą języków opisu, jako modeli specyfikacji jest to, że bez odpowiedniego wspomagania komputerowego nie oferują wsparcia graficznego, będącego bardzo ważną cechą dobrego modelu. Zaletą jest uniwersalność, pozwalająca zrealizować tak zamodelowane urządzenie na większości dostępnych platformach sprzętowych.

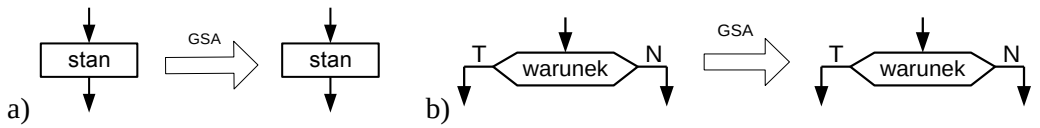
4. BUDOWA MODELU SPECJALIZOWANEGO STEROWNIKA RUCHU DROGOWGO

Bazując na wcześniejszych założeniach i przeanalizowanych modelach sterowników cyfrowych, wykorzystano autorską metodę przekształcenia algorytmu sterowania ruchem na skrzyżowaniu w model sterownika logicznego, zapisany w języku opisu sprzętu VHDL. Pośrednio wykorzystano dwa modele specyfikacji sterowników cyfrowych mające czytelne reprezentacje graficzne: GSA i FSM. Ostatecznie model graficzny wykorzystano przy specyfikacji sterownika w programie Active-HDL, który umożliwił uzyskanie modelu w języku VHDL.

4.1. Budowa formalnego modelu graficznego bloku logicznego sterownika

Popularna forma opisu algorytmów sterowania ruchem drogowym w naturalny sposób nadaje się do specyfikacji za pomocą graficznego schematu algorytmu GSA. Budowa GSA, reprezentującego funkcję sterowania na krzyżowaniu, ma na celu budowę formalnego opisu algorytmu sterowania ruchem drogowym. GSA jest postacią formalną, niezbędną do otrzymania docelowego modelu sterowania, którym jest graf sterowania automatu skończonego. Algorytm sterowania ruchem na skrzyżowaniu (rys. 3), pomimo że wizualnie przypomina sieć działań, zawiera klatki, których działanie nie mieści się w standardzie i formacie opisu GSA. Dlatego dla wszystkich możliwych zachowań zaproponowano sposób modelowania klatek GSA.

Dla klatek operacyjnych i warunkowych algorytmu sterowania metoda modelowania w sieć działań zakłada utworzenie tożsamych klatek GSA (rys. 7).



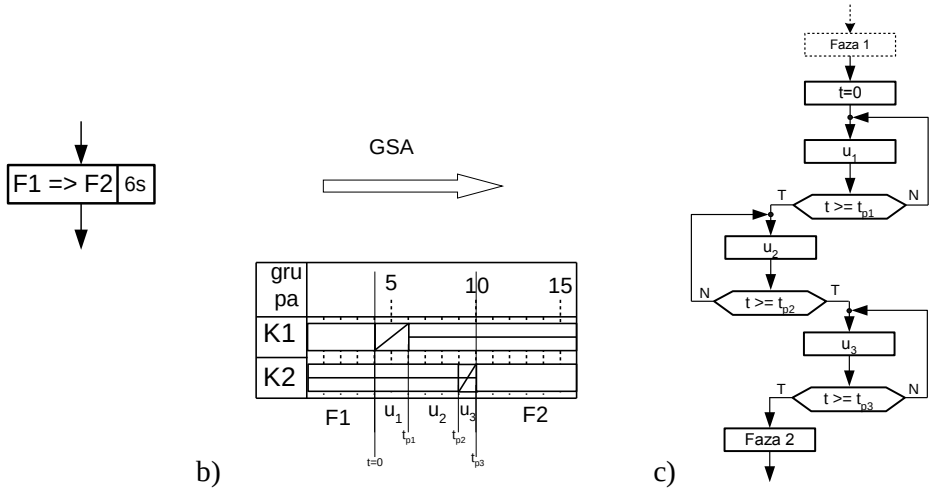
Rys. 7. Zamiana klatek algorytmu srd na GSA: a) klatka operacyjna, b) klatka warunkowa
Fig. 7. Conversion from algorithm to GSA: a) action block, b) conditional block

W przypadku klatek przejścia międzyfazowego sytuacja jest trudniejsza. Przejście międzyfazowe definiowane jest jako sekwencja zmian wektora sterowania, w bezpieczny sposób przechodząca z jednej fazy do drugiej. Czas przejścia międzyfazowego może wynosić od 0 do kilkudziesięciu sekund. W związku z tym, przejście międzyfazowe nie jest jedną mikrooperacją, lecz zbiorem mikrooperacji.

W przedstawionej na rys. 8 zamianie klatki przejścia międzyfazowego na sieć GSA, jednej klatce algorytmu srd (rys. 8a) odpowiada kilka lub kilkanaście klatek GSA (rys. 8c). Dla przeprowadzenia przekształcenia niezbędny jest program przejścia międzyfazowego, którego zamiana dotyczy (rys. 8b).

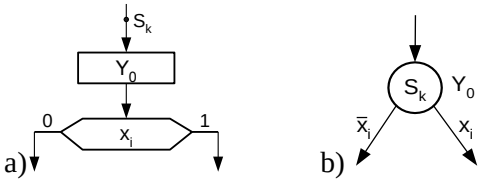
Przedstawiona metoda budowy algorytmów GSA pozwala na zamianę dowolnego adaptacyjnego algorytmu sterowania ruchem drogowym w sieć działań GSA, będącą formalnym modelem opisu procesu sterowania ruchem drogowym.

Posiadając GSA, który opisuje proces sterowania ruchem, możliwa jest budowa ekwiwalentnego grafu sterowania automatu skończonego Moore'a. Po oznakowaniu sieci działań, konwersja sieci w graf automatu Morea'a przedstawiona została na rys. 9.



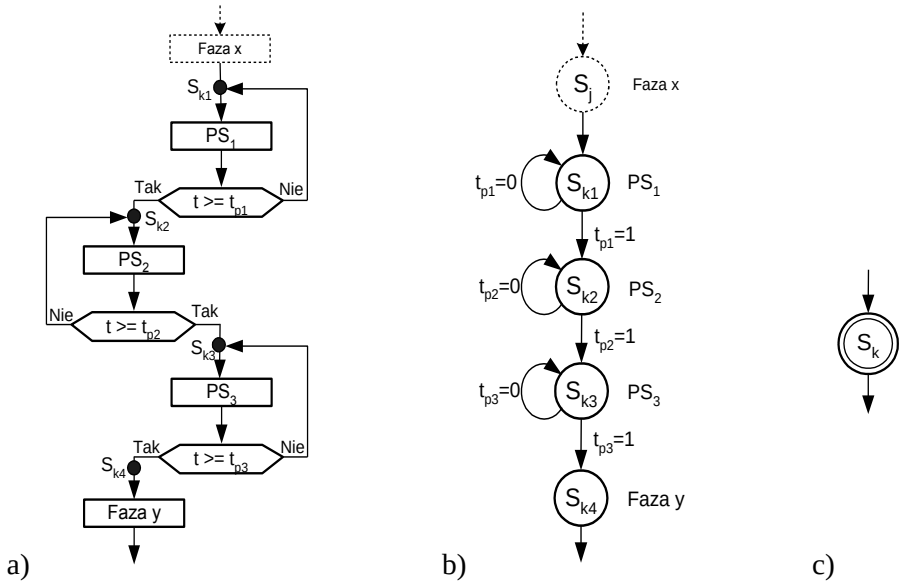
Rys. 8. Zamiana klatki przejścia międzyfazowego w GSA: a) klatka przejścia międzyfazowego, b) program przejścia międzyfazowego, c) sieć GSA przejścia międzyfazowego

Fig. 8. Conversion from interstage to GSA: a) interstage, b) interstage program, c) interstage GSA



Rys. 9. Zamiana sieci działań w graf automatu Moore'a: a) oznakowany fragment sieci działań, b) graf stanów automatu Moore'a odpowiadający oznakowanej sieci

Fig. 9. Conversion from GSA to Moore FSM: a) marked GSA, b) equivalent Moore FSM



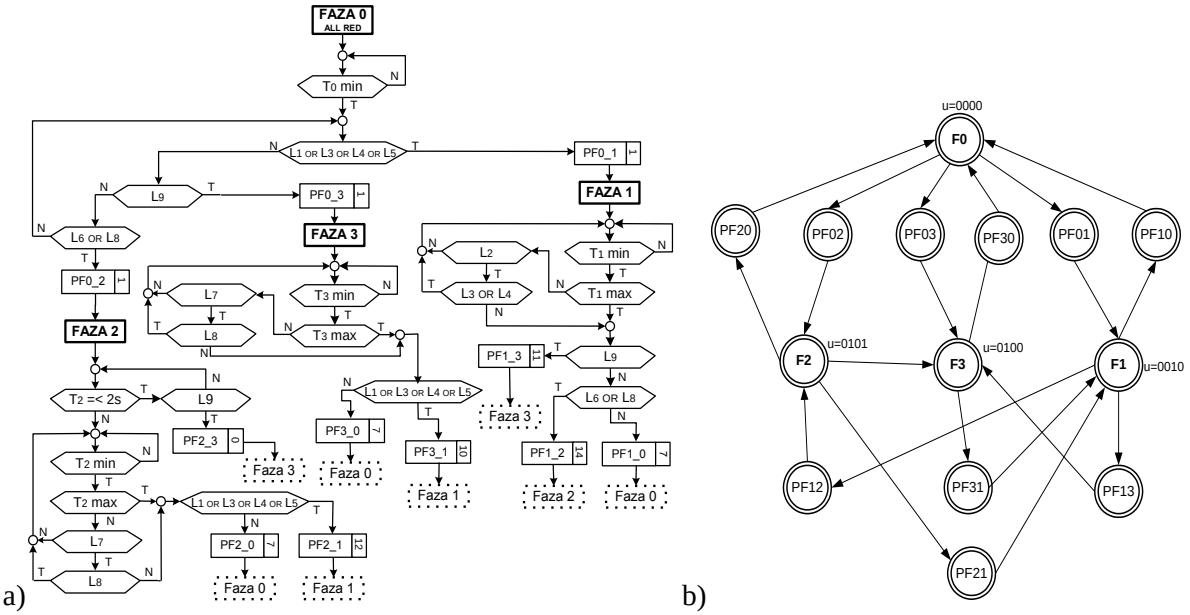
Rys. 10. Realizacja stanu hierarchicznego dla przejścia międzyfazowego: a) GSA przejścia międzyfazowego, b) graf sterowania przejścia międzyfazowego, c) stan hierarchiczny

Fig. 10. Realization of a hierarchical state for interstage: a) interstage GSA, b) interstage FSM, c) hierarchical state

Dla zbudowanego grafu sterowania ruchem na skrzyżowaniu, jako eliminację płaskości modelu, a także zwiększenie czytelności, wykorzystano model automatu hierarchicznego. Hierarchizacja grafu sterowania obejmuje:

- budowę wierzchołków hierarchicznych dla przejść międzyfazowych (rys. 10),
- budowę wierzchołków hierarchicznych realizujących obsługę faz ruchu.

Końcowym efektem budowy grafu przejść automatu skończonego, dla przykładowego algorytmu srd (rys. 11a), jest graf hierarchiczny zawierający wierzchołki hierarchiczne odpowiadające fazom sygnalizacji i przejściom międzyfazowym (rys. 11b).



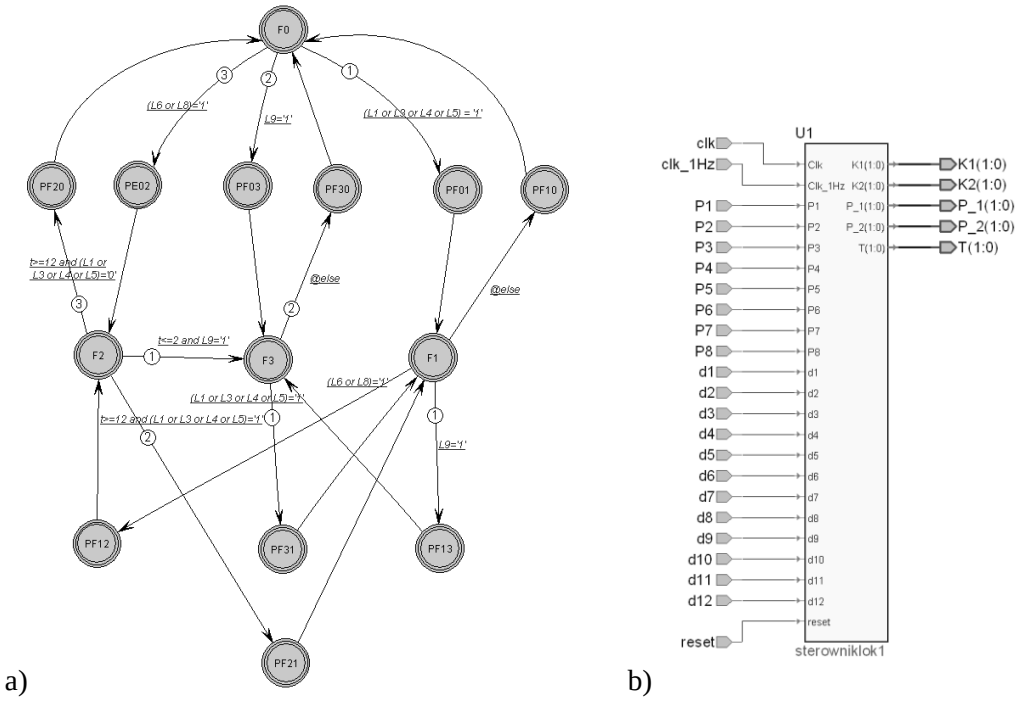
Rys. 11. Modele sterowania ruchem drogowym na skrzyżowaniu: a) algorytm sterowania ruchem, b) hierarchiczny graf sterowania ruchem

Fig. 11. Models of traffic control at the intersection: a) traffic control algorithm, b) hierarchical graph of traffic control

4.2. Specyfikacja bloku logicznego sterownika specjalizowanego

Jednym z kryteriów wyboru modelu specyfikacji formalnej sterownika było wsparcie tego modelu graficznego przez narzędzia CAD, służące do specyfikacji urządzeń cyfrowych. Wykorzystano oprogramowanie Active-HDL firmy Aldec. Program ten ma edytor pozwalający na specyfikację hierarchicznych grafów przejść automatów skończonych (rys. 12a). Opracowano uniwersalną architekturę bloku logicznego sterownika ruchu drogowego, w której poza grafem sterowania znajdują się moduły liczników czasu, obsługi logiki detektorowej itp. Cały blok logiczny sterownika ruchu wyspecyfikowano w formie modelu strukturalnego w edytorze BDE programu Active-HDL (rys. 12b).

Taka graficzna specyfikacja bloków sterownika pozwala na automatyczną generację kodu VHDL, przez pakiet Active-HDL. W ten sposób otrzymano model realizowanego sterownika w języku opisu sprzętu, pozwalający na realizację urządzenia na dowolnej platformie sprzętowej.



Rys. 12. Specyfikacja w Active-HDL: a) hierarchiczny graf sterowania ruchem na skrzyżowaniu, b) model strukturalny

Fig. 12. Specification in Active-HDL: a) hierarchical graph of the intersection traffic control, b) struktural model

5. PODSUMOWANIE

Przedstawiono metodę pozwalającą, na podstawie typowego zapisu algorytmów adaptacyjnego sterowania ruchem na skrzyżowaniu, na budowę modelu bloku logicznego sterownika ruchu drogowego w języku VHDL. Przy czym metoda ta, wykorzystując formalne modele specyfikacji sterowników cyfrowych, pozwala na etapie projektowania wykorzystywać intuicyjne graficzne reprezentacje tych modeli. Efektem końcowym jest model sterownika w postaci hierarchicznego grafu przejść automatu skończonego.

Wykorzystanie wspomagania komputerowego pozwala na prostą specyfikację otrzymanych modeli graficznych w odpowiednich środowiskach, zarówno grafów przejść automatów skończonych, jak i modeli strukturalnych. Na tym etapie możliwa jest automatyczna zamiana modeli graficznych na kod w języku opisu sprzętu, będący wyjściową platformą do realizacji urządzeń w układach programowalnych.

Kolejnym, niezwykle ważnym etapem jest weryfikacja zbudowanych modeli. Opracowano procedurę weryfikacji tych modeli, wykorzystując w tym procesie narzędzia wspomagania komputerowego oraz metody weryfikacji funkcjonalnej, a także strukturalnej.

Bibliografia

1. De Micheli G.: Synteza i optymalizacja układów cyfrowych, Wydawnictwo Naukowo-Techniczne, Warszawa 1998.
2. Dz. U. z 2003 r. Nr 220, poz. 2181, z późn. zm. Rozporządzenie ministra infrastruktury z dnia 3 lipca 2003 r. w sprawie szczegółowych warunków technicznych dla znaków i sygnałów drogowych oraz urządzeń bezpieczeństwa ruchu drogowego i warunków ich umieszczania na drogach.
3. FGSV – Forschungsgesellschaft fuer Strassen- und Verkehrswesen (2003a). Guidelines for Traffic Signals. English Version of Richtlinien fuer Lichtsignalanlagen RiLSA. Edition 1992 (with minor modifications), Translation 2003. FGSV 321/S, FGSV-Verlag, Cologne 2003.
4. Gajski D.: Principles of digital design, Prentice Hall International, 1997.
5. Kawalec P.: Analiza i synteza specjalizowanych układów modelowania i sterowania ruchem w transporcie, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2009.
6. Leśko M., Guzik J.: Sterowanie ruchem drogowym. Sterowniki i systemy sterowania i nadzoru ruchu, Wydawnictwo Politechniki Śląskiej, Gliwice 2000.
7. Pasierbiński J., Zbysiński P.: Układy programowalne w praktyce, WKŁ, Warszawa 2001.