ISSN 1232-3829
tts
10/2013
Technika Transportu Szynowego koleje ▪ tramwaje ▪ metro

**KAWALEC Piotr, RŻYSKO Marcin**

# WERYFIKACJA RÓWNAŃ ZALEŻNOŚCIOWYCH Z WYKORZYSTANIEM SYMULATORÓW LOGICZNYCH NA PRZYKŁADZIE ZASTOSOWANIA PAKIETU ACTIVE-HDL

***Streszczenie***

*W artykule przedstawiona została problematyka weryfikacji logiki zależnościowej współczesnych systemów sterowania ruchem kolejowym. Złożoność zagadnienia rośnie w ostatnich latach w wyniku wielu istotnych czynników, takich jak konieczność zapewnienia interoperacyjności systemów, czy tendencja do obejmowania sterowaniem z jednej nastawni coraz większych obszarów. Utrudniona staje się więc manualna analiza poprawności działania projektowanych systemów. W związku z tym do weryfikacji zaproponowane zostało wykorzystanie nowoczesnego, zintegrowanego pakietu programistycznego Active-HDL. Na przykładzie zestawu równań zależnościowych opracowanego w języku VHDL przedstawione zostały możliwości pakietu w zakresie weryfikacji projektu.*

## WSTĘP

Zasadniczą cechą wyróżniającą systemy sterowania ruchem w transporcie (a w szczególności systemy sterowania ruchem kolejowym) wśród układów automatyki jest wysoki wymagany poziom bezpieczeństwa. Bezpieczeństwo to zapewniane jest poprzez niezawodność elementów, redundancję sprzętową, a także przez odpowiednio skonstruowane algorytmy sterujące pracą systemu. Podstawowymi problemami, przed którymi stają dziś projektanci logiki zależnościowej są nowe, rozbudowane funkcjonalności związane ze zwiększającą się integracją systemów. Brak standardowych, formalnych metod zapisu wymagań dla urządzeń srk sprawia, że weryfikacja poprawności funkcjonowania logiki zależnościowej jest procesem trudnym i czasochłonnym. Poszukiwanie nowych rozwiązań opiera się obecnie o wykorzystanie metod formalnych [1, 2], jak również proponowane jest wykorzystanie nowych technik realizacji układów [5]. Niniejszy artykuł przedstawia część prac nad formalizacją zapisu zależności, ze szczególnym uwzględnieniem nowoczesnych metod weryfikacji równań zależnościowych z wykorzystaniem narzędzi komputerowego wspomagania prac projektowych.

# 1. FORMALIZACJA ZAPISU ALGORYTMÓW DZIAŁANIA URZĄDZEŃ SRK

Zastosowanie metod formalnych jest w tym momencie popularnym kierunkiem prac rozwojowych logiki zależnościowej w systemach srk. Formalizacja zapisu zależności daje wymierne korzyści zarówno w procesie projektowania urządzeń nowych, jak i przy aplikowaniu istniejących rozwiązań do konkretnych układów torowych.

Pierwszym etapem tworzenia opisu formalnego jest identyfikacja danych wchodzących i wychodzących z systemu oraz przypisanie tych danych do odpowiednich wektorów. Typowo przyjmuje się, że system zależnościowy pośredniczy pomiędzy systemem nadrzędnym, obsługiwanym przez personel, a sterownikami obiektowymi, kontrolującymi urządzenia w terenie. Przyjmując, że jako $\boldsymbol{X}$ oznaczono wektor zmiennych wejściowych, a jako $\boldsymbol{Y}$ - wektor zmiennych wyjściowych, wyróżniono następujące składowe tych wektorów.

$$\begin{aligned} X &= \{X_K, X_P\}; \\ Y &= \{Y_S, Y_M\} \end{aligned} \tag{1}$$

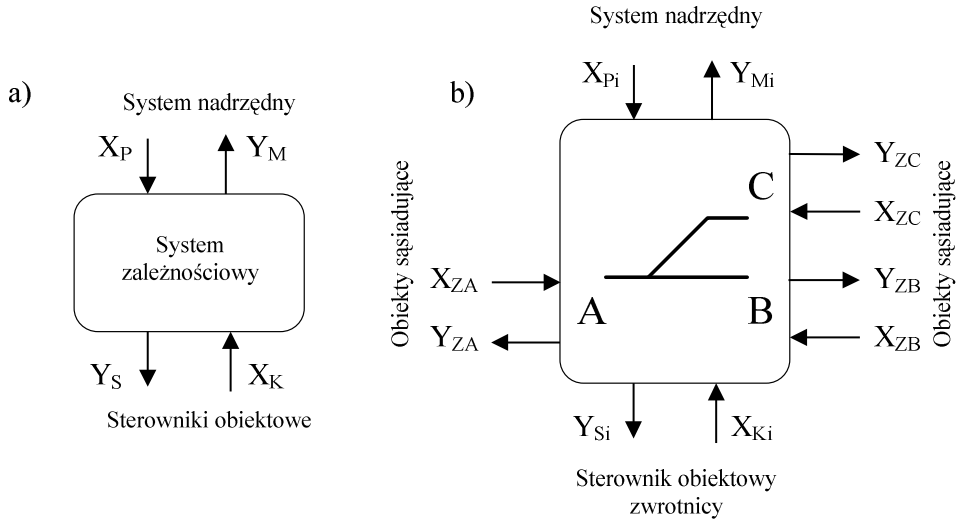
Gdzie:

- $X_K$ - wektor kontroli,
- $X_P$ - wektor poleceń,
- $Y_S$ - wektor sterowań,
- $Y_M$ - wektor meldunków.

Powyższe wektory zostały schematycznie przedstawione na rys. 1a. Po zdefiniowaniu struktury wymiany danych dla systemu jako całości konieczne jest określenie zasad dalszej dekompozycji systemu. Stosowane są dwa podstawowe podejścia do konstruowania logiki zależnościowej: niegeograficzne (ang. *free-wired*) i geograficzne (ang. *geographical*). Wywodzą się one z systemów przekaźnikowych i dziś wykorzystywane są w oprogramowaniu komputerowych systemów srk. W opisywanej metodzie wykorzystane zostało podejście geograficzne, spełniające wymagania kompletnego opisu funkcji zależnościowych zdatnego do wielokrotnego wykorzystania.

Zasada projektowania zależności w systemach o strukturze geograficznej polega na opracowaniu pełnego zestawu równań zależnościowych dla każdego rodzaju elementu (semafor, zwrotnica czy skrzyżowanie torów). Element powinien być wyspecyfikowany w sposób uniwersalny tak, aby jego funkcjonowanie było poprawne niezależnie od układu torowego. Zbudowanie kompletnego opisu elementów pozwala wykorzystywać w aplikacjach gotowe fragmentu kodu, minimalizując czas potrzebny na wykonanie projektu zależności dla zadanego posterunku ruchu.

W oparciu o przedstawione wcześniej alfabety zmiennych dla całego systemu, określone zostały wektory dla pojedynczego elementu systemu geograficznego. Składowe opisanych wcześniej wektorów dla i-tego elementu systemu uzupełnione zostały o wektory zależności, umożliwiające komunikację z obiektami sąsiednimi. Dla przykładowego elementu (modułu zwrotnicy) układ wektorów przedstawia rys. 1b.




**Rys. 1.** Układ wektorów dla systemu zależnościowego (a) oraz obiektu logicznego (b)
Źródło: Opracowanie własne

Równania zależnościowe dla elementów opisanych w niniejszym artykule wykonane zostały w oparciu o algorytmizację działania każdego obiektu logicznego. Na podstawie analizy systemów sterowania ruchem kolejowym wykorzystywanych na sieci PKP PLK (takich jak E, PB, CBP-83, IZH-111, *EBI*Lock 850/950), a także obowiązujących przepisów i wytycznych, określony został zbiór funkcji, które powinny być (w sposób współbieżny) realizowane przez każdy moduł logiczny systemu geograficznego [4, 7, 8]. Do takich funkcji należą między innymi:

– funkcja wykorzystania obiektu w drodze jazdy przebiegu (wybieranie, nastawianie, utwierdzanie i zwalnianie przebiegów),
– funkcja sygnalizacji (wybór obrazu na sygnalizatorze, kontrola przeciwwtórności),
– funkcja dostępności elementu (stopowanie, zamknięcie ruchowe).

Dla przykładowego algorytmu, opisującego funkcję wykorzystania elementu w drodze jazdy, przy wykorzystaniu metody opisanej szerzej w [3, 9] otrzymano formalny zapis w postaci logicznego schematu algorytmu [6], który dla modułu logicznego zwrotnicy wygląda następująco.

$$
\begin{aligned}
ZWRJ = {} & \overset{0}{\downarrow} Y_S X_{swAB} \overset{11}{\uparrow} X_{swAC} \overset{21}{\uparrow} X_{swBA} \overset{31}{\uparrow} X_{swCA} \overset{41}{\uparrow} \omega \overset{0}{\uparrow} \\
& \overset{11}{\downarrow} Y_{WAB} X_{wnAB} \overset{12}{\uparrow} \omega \overset{11}{\uparrow} \overset{21}{\downarrow} Y_{WAC} X_{wnAC} \overset{22}{\uparrow} \omega \overset{21}{\uparrow} \overset{31}{\downarrow} Y_{WBA} X_{wnBA} \overset{32}{\uparrow} \omega \overset{31}{\uparrow} \overset{41}{\downarrow} Y_{WCA} X_{wnCA} \overset{42}{\uparrow} \omega \overset{41}{\uparrow} \\
& \overset{12}{\downarrow} Y_{NAB} X_{nuAB} \overset{13}{\uparrow} \omega \overset{12}{\uparrow} \overset{22}{\downarrow} Y_{NAC} X_{nuAC} \overset{23}{\uparrow} \omega \overset{22}{\uparrow} \overset{32}{\downarrow} Y_{NBA} X_{nuBA} \overset{33}{\uparrow} \omega \overset{32}{\uparrow} \overset{42}{\downarrow} Y_{NCA} X_{nuCA} \overset{43}{\uparrow} \omega \overset{42}{\uparrow} \\
& \overset{13}{\downarrow} Y_{UAB} X_{uzAB} \overset{14}{\uparrow} \omega \overset{13}{\uparrow} \overset{23}{\downarrow} Y_{UAC} X_{uzAC} \overset{24}{\uparrow} \omega \overset{23}{\uparrow} \overset{33}{\downarrow} Y_{UBA} X_{uzBA} \overset{34}{\uparrow} \omega \overset{33}{\uparrow} \overset{43}{\downarrow} Y_{UCA} X_{uzCA} \overset{44}{\uparrow} \omega \overset{43}{\uparrow} \\
& \overset{14}{\downarrow} Y_{ZAB} X_{zsAB} \overset{0}{\uparrow} \omega \overset{14}{\uparrow} \overset{24}{\downarrow} Y_{ZAC} X_{zsAC} \overset{0}{\uparrow} \omega \overset{24}{\uparrow} \overset{34}{\downarrow} Y_{ZBA} X_{zsBA} \overset{0}{\uparrow} \omega \overset{34}{\uparrow} \overset{44}{\downarrow} Y_{ZCA} X_{zsCA} \overset{0}{\uparrow} \omega \overset{44}{\uparrow}
\end{aligned}
\tag{2}
$$

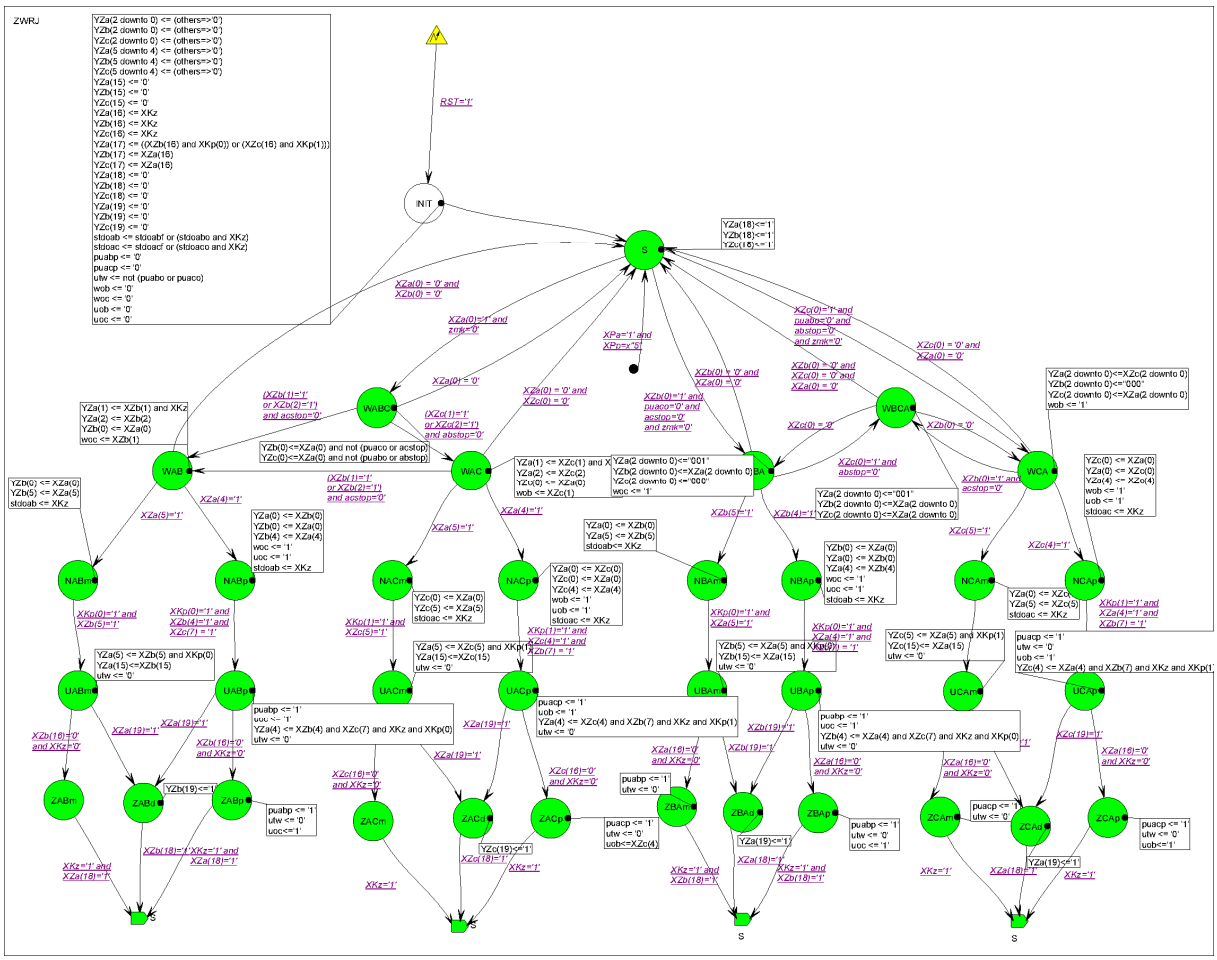
Powyższy zapis, po określeniu wartości poszczególnych wektorów *Y*... i *X*… pozwala zbudować pełny, formalny zapis działania funkcji, który może być wykorzystany do budowy specyfikacji. Opracowanie zapisu działania pozostałych funkcji elementu pozwoliło stworzyć kompletny zapis logiki zależnościowej modułu logicznego zwrotnicy.

W podobny sposób zaprojektowane zostały wszystkie elementy konieczne do realizacji zależności dla dowolnego układu torowego, takie jak semafor, tarcza manewrowa,

skrzyżowanie torów. Tak wykonana formalizacja modelu pozwoliła na wykorzystanie narzędzi wspomagania komputerowego do specyfikacji elementów w języku VHDL oraz weryfikacji poprawności opisu, co przedstawione zostanie w dalszej części artykułu.

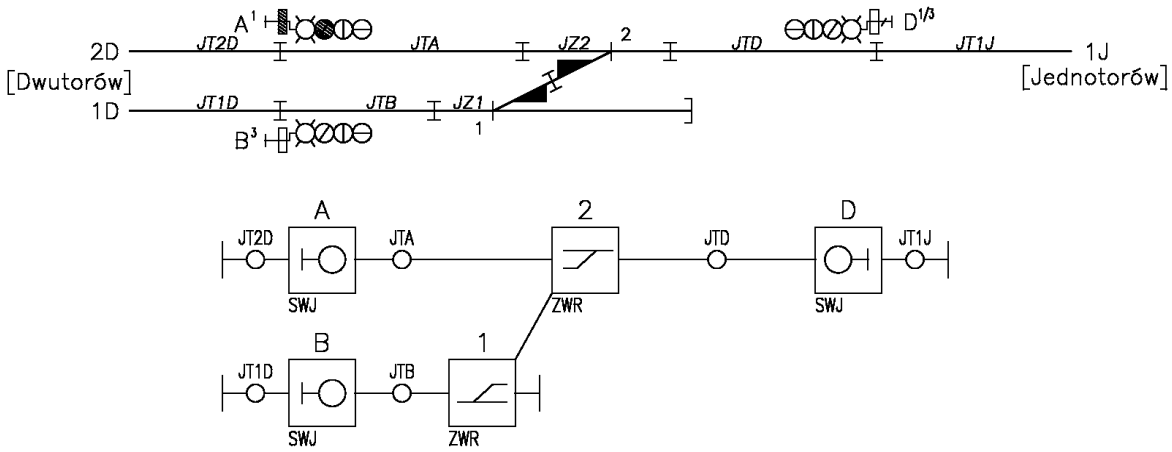
## 2. SPECYFIKACJA ALGORYTMÓW W JĘZYKU OPISU SPRZĘTU

Do wykonania specyfikacji zaprojektowanych algorytmów w języku opisu sprzętu wykorzystany został pakiet *Active-HDL*, będący zintegrowanym środowiskiem projektowym dla języków HDL. Szczególnie istotnym elementem pakietu są edytory graficzne, pozwalające na łatwą specyfikację układów o strukturze sekwencyjnej (w edytorze grafów przejść - FSM) oraz złożonych zależności hierarchicznych (w edytorze schematów blokowych - BDE). W połączeniu z edytorem tekstowym HDE, narzędzia te pozwalają na automatyczne wygenerowanie kodu języka VHDL dla algorytmów zaprojektowanych w procesie formalizacji równań zależnościowych. Proponowana metoda pozwala więc na jednoznaczne przejście od elementarnych warunków w algorytmie działania funkcji do kompletnego kodu stanowiącego zbiór równań zależnościowych obiektu logicznego. Przykładowa specyfikacja algorytmu przedstawiona została na rys. 2.



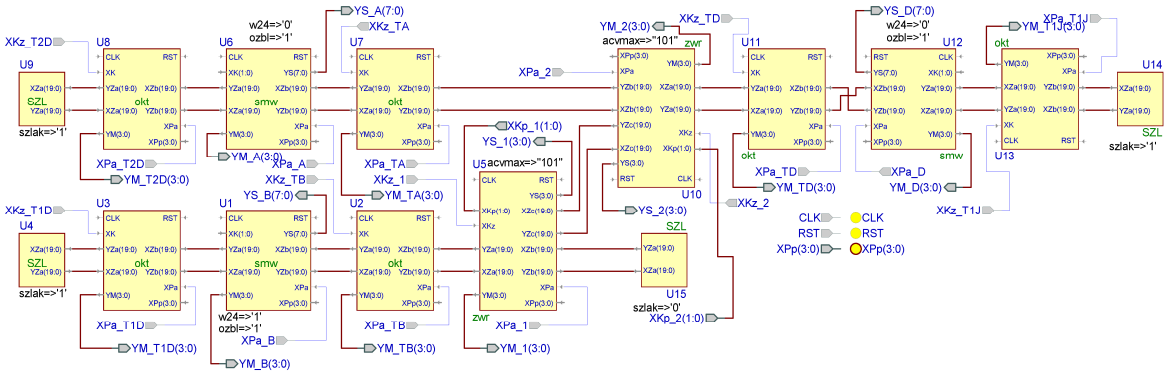

**Rys. 2.** Specyfikacja algorytmu działania zwrotnicy (ZWRJ) w edytorze FSM
Źródło: Opracowanie własne

Każdy element wykonany w edytorze FSM może być następnie przedstawiony w postaci pojedynczego bloku w edytorze BDE. Rys. 3 przedstawia przykładowy układ torowy oraz odpowiadający jemu układ połączeń obiektów logicznych.




**Rys. 3.** Schemat układu torowego i odpowiadający mu układ połączeń obiektów logicznych
Źródło: Opracowanie własne

Dla zadanego układu torowego opracowany został układ połączeń obiektów logicznych, zrealizowany w edytorze BDE (rys. 4).



**Rys. 4.** Połączenia opracowanych modułów logicznych w edytorze BDE
Źródło: Opracowanie własne

Możliwe jest następnie przejście na kolejny poziom hierarchiczny, w którym powyższy układ przedstawiony jest jako jeden element, o wejściach i wyjściach zgodnych z zaproponowanym na początku układem wektorów dla całości systemu zależnościowego. Tak wykonana, wielopoziomowa specyfikacja może być następnie weryfikowana przy użyciu kolejnych narzędzi pakietu *Active-HDL*.

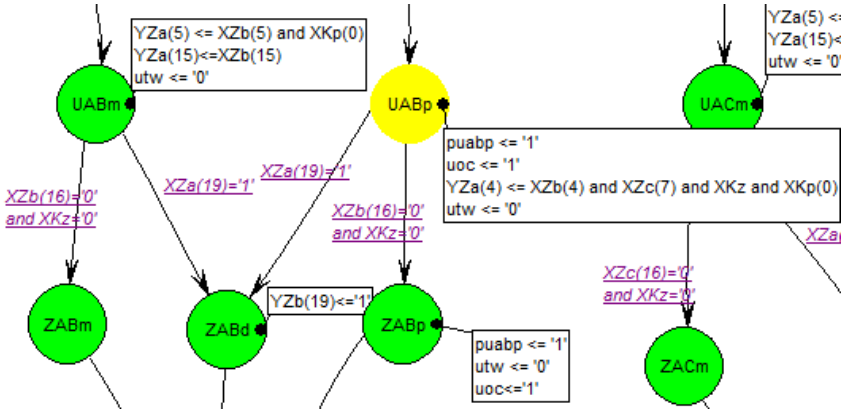
# 3. WYKORZYSTANIE PAKIETU ACTIVE-HDL DO WERYFIKACJI ZAPROJEKTOWANYCH ALGORYTMÓW

Wykorzystanie do specyfikacji zaprojektowanych algorytmów zintegrowanego pakietu programistycznego umożliwia nie tylko automatyczne generowanie kodu języka HDL na podstawie zapisu graficznego. Pozwala także na znaczące uproszczenie procesu projektowania, ponieważ możliwa jest bezpośrednia weryfikacja poprawności projektu bez potrzeby zmiany platformy projektowej. Pakiet *Active-HDL* oferuje rozbudowane narzędzia służące do analizy działania wykonanej specyfikacji.

## 3.1. Metody obserwacji przebiegu symulacji

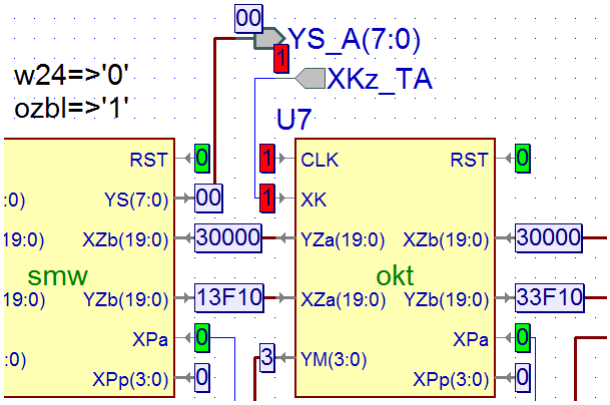
Obserwacja przebiegu symulacji w pakiecie *Active-HDL* może odbywać się zarówno z wykorzystaniem narzędzi projektowych (podgląd w edytorach FSM i BDE), jak

i w stworzonych specjalnie do tego celu edytorach. Wykorzystanie do tego celu edytorów graficznych pozwala w pełni wykorzystać integralność pakietu, symulacja odbywa się bowiem w tym samym środowisku, dając pełny wgląd w funkcjonowanie elementów. Fragmenty projektu wykonane w edytorze FSM umożliwiają wizualizację działania bezpośrednio na grafie, poprzez zaznaczenie aktywnego stanu w każdym obiekcie (rys. 5).




**Rys. 5.** Podgląd aktywnego stanu w edytorze FSM
Źródło: Opracowanie własne

Kolejny z edytorów - BDE, umożliwia obserwację wartości zmiennych na wejściach, wyjściach, oraz liniach transmisyjnych łączących obiekty (rys. 6).




**Rys. 6.** Obserwacja przebiegu symulacji w edytorze BDE
Źródło: Opracowanie własne

Pakiet zawiera również narzędzia dedykowane rejestracji przebiegu symulacji, takie jak *Waveform Editor*, w którym stan każdej zmiennej przedstawiony jest w postaci wykresu od czasu. Edytor ten umożliwia również automatyczne porównywanie dwóch zapisów, wyszczególniając różnice w przebiegach zmiennych. Dostępny jest również widok listy, przedstawiający wartości zmiennych i kroki symulacji w formie tabelarycznej.

## 3.2. Metody kontroli symulacji

Kontrola wartości wektora zmiennych wejściowych może odbywać poprzez przypisanie do każdej zmiennej stymulatora. Stymulatorem może być:
– stała, wymuszona wartości (z zakresu Std_Logic),
– klawisz typu *Hotkey*,
– zegar predefiniowany lub zaprogramowany ręcznie,
– wartość losowa u zadanych parametrach.

Stymulatory umożliwiają pełną kontrolę nad przebiegiem symulacji, pozwalając wygenerować dowolne sekwencje wymuszeń. Analiza wykonywana przy pomocy ręcznego przełączania sygnałów wejściowych przydatna jest do testowania pojedynczych funkcji projektowanego algorytmu.

Z powodu wysokiej złożoności równań zależnościowych, istotna staje się automatyzacja procesu symulacji. Pakiet Active-HDL umożliwia wykorzystanie makr, które tworzone są w przygotowanym do tego celu języku *Active-HDL Macro Language*. Jest to prosty język skryptowy, umożliwiający obsługę wielu funkcji pakietu *Active-HDL* z poziomu terminala. Stworzone w nim makra zawierają instrukcje, takie jak zmiana wartości sygnałów, długość kroku symulacji, czy polecenia kompilacji i eksportu danych do pliku. Z uwagi na prostą składnię makr, możliwe jest łatwe konstruowanie własnych, realizujących założone funkcje testowe. Możliwe jest również automatyczne generowanie makra przez edytor FSM, powstający w ten sposób tak zwany *Testbench* zawiera instrukcje dla symulacji grafu przejść zgodnie z wybranym scenariuszem (na przykład przejście przez wszystkie tranzycje). Rys. 7 przedstawia widok edytora tekstowego pakietu i fragment makra.

```
44  @force XPa_A 1
45  @force poc 1
46  @run 500ns
47  @force XPa_A 0
48  @force poc 0
49  @run 2000ns
50  @if %YS_A = 07
51  @   echo "WARUNEK YS_A=07 ZOSTAL SPELNIONY"
52  @else
53  @   echo "WARUNEK YS_A=07 NIE ZOSTAL SPELNIONY"
54  @endif
55  @run 1000ns
```

**Rys. 7.** Edycja makra w edytorze HDE
Źródło: Opracowanie własne

Wszystkie wyżej opisane metody wykorzystane zostały do weryfikacji opracowanych funkcji zależnościowych dla geograficznego systemu srk.

## 3.3. Weryfikacja zaprojektowanych równań zależnościowych

Dzięki zastosowaniu opisanej metody powstający opis systemu składa się z kilku warstw. Wyróżnić można poziomy hierarchiczne, na których możliwe jest testowanie wykonanej specyfikacji:

– poziom pojedynczego algorytmu,
– poziom połączeń obiektów geograficznych,
– poziom kompletnej aplikacji.

Weryfikacja na poziomie grafu przejść ma na celu analizę poprawności zapisu warunków opisanych w algorytmie działania elementu. Istotą tej części testów jest wyeliminowanie błędów powstałych w trakcie tworzenia grafu przejść w edytorze FSM. Wydajnym narzędziem do wykonania tej części weryfikacji jest *Testbench*, pozwalający wygenerować zapis przejścia przez wszystkie tranzycje wraz analizą wartości zmiennych wyjściowych.

Testowanie połączeń zależnościowych pomiędzy obiektami logicznymi ma na celu analizę współpracy modułów logicznych, a w szczególności pozwala na obserwację wartości wektora zależności dla każdego połączenia geograficznego. Do tego celu wykorzystany został podgląd w edytorze BDE. Ten etap weryfikacji pozwala potwierdzić poprawne działanie poszczególnych algorytmów działania w obiektach logicznych w połączeniu z obiektami sąsiednimi.

Poprawny wynik powyższych symulacji stanowi punkt wyjścia do testów kompletnej aplikacji dla zadanego układu torowego. Podstawowym dokumentem opisującym wymagane zachowanie systemu jest tablica zależności, będąca integralnym elementem każdego projektu posterunku ruchu. Testowanie zgodności z tablicą zależności dotyczyło najwyższego poziomu hierarchicznego projektu.

| L.p. | Sygnał | Przebieg | Wykluczenia | | | | | Zwr. | | Odc. kontrolowane |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $A^1$ | $B^3$ | | $D^3$ | $D^1$ | 1 | 2 | |
| 001 | $A^1$ | Do st. Jednotorów po torze 1J z toru 2D | — | + | | + | + | o+ | + | 2, TA, TD, T1J |
| 002 | $B^3$ | Do st. Jednotorów po torze 1J z toru 1D | + | — | | + | + | — | — | 1, 2, TB, TD, T1J |
| | | | | | | | | | | |
| 003 | $D^3$ | Do st. Dwutorów po torze 1D z toru 1J | + | + | | — | + | — | — | 1, 2, TB, TD, T1D |
| 004 | $D^1$ | Do st. Dwutorów po torze 2D z toru 1J | + | + | | + | — | o+ | + | 2, TA, TD, T2D |

**Rys. 8.** Tablica zależności dla przykładowego układu torowego
Źródło: Opracowanie własne

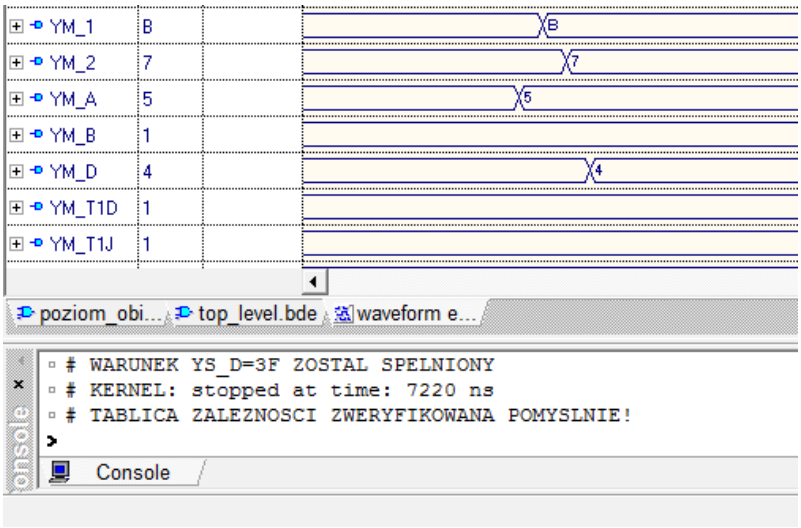
Do wykonania tej części konieczne jest przygotowanie makra zawierającego instrukcje testujące dla każdego pola tablicy. Makro zostało przygotowane w oparciu o tablicę zależności dla przedstawionego wcześniej przykładowego posterunku ruchu (rys. 3), przedstawioną na rys. 8. Dla każdego wiersza (przebiegu) w tablicy makro zawiera instrukcje ustawienia tego przebiegu, a następnie wykonania czynności zgodnie z tab. 1.

**Tab. 1.** Warunki dla badania zgodności z tablicą zależności

| Część | Zależność | Procedura badania |
|---|---|---|
| wykluczenia | przebieg niesprzeczny | ustawienie przebiegu i kontrola poprawności realizacji polecenia, zwolnienie doraźne przebiegu niesprzecznego |
| | przebieg sprzeczny | próba ustawienia przebiegu, kontrola odrzucenia polecenia |
| zwrotnice | zwrotnica nie zamknięta w przebiegu | przestawienie zwrotnicy do przeciwnego położenia i ponownie do podstawowego oraz kontrola wykonania |
| | zwrotnica zamknięta w przebiegu | polecenie przestawienia zwrotnicy do przeciwnego położenia i kontrola odrzucenia polecenia |
| odcinki kontrolowane | odcinek nie kontrolowany w przebiegu | zajęcie odcinka, kontrola świecenia sygału zezwalającego na semaforze początkowym badanego przebiegu |
| | odcinek kontrolowany w przebiegu | zajęcie odcinka, kontrola wygaszenia sygnału zezwalającego, ponowne ustawienie badanego przebiegu |

Źródło: Opracowanie własne na podstawie [7]

Makro testujące wygenerowane zostało za pomocą skryptu stworzonego w języku Python na podstawie tablicy zależności zapisanej w formacie XML i w oparciu o opisane wyżej założenia. Rys. 9 przedstawia widok konsoli programu *Active-HDL* w trakcie wykonywania symulacji.




**Rys. 9.** Automatyczna weryfikacja
Źródło: Opracowanie własne

Pozytywny wynik symulacji potwierdził zgodność pracy algorytmów z tablicą zależności. Możliwość wykorzystania makr jest niezwykle ważną funkcjonalnością pakietu *Active-HDL*, pozwalającą na automatyczną weryfikację zarówno pojedynczych funkcji, jak i kompletnych zależności, co staje się szczególnie istotne przy rozbudowanych układach torowych.

## PODSUMOWANIE

Formalizacja algorytmów działania elementów w geograficznym systemie srk pozwoliła na wykorzystanie narzędzi wspomagania komputerowego do weryfikacji opracowanych zależności. Na przykładzie prac nad formalną metodą specyfikacji i weryfikacji równań zależnościowych w systemach srk zaprezentowane możliwości symulacyjne pakietu *Active-HDL*. Pozytywny wynik badań symulacyjnych, obejmujących różne poziomy hierarchiczne zaprojektowanego pakietu logiki zależnościowej, potwierdził poprawność specyfikacji algorytmów i stanowi punkt wyjścia do implementacji równań zgodnie z wybraną techniką realizacji. Przedstawiona metoda projektowania logiki zależnościowej może znaleźć zastosowanie przy tworzeniu nowoczesnych systemów sterowania ruchem na kolei.

## BIBLIOGRAFIA

1. Kanso K., Moller F., Setzer A., *Automated Verification of Signalling Principles in Railway Interlocking Systems*. Electronic Notes in Theoretical Computer Science, AVoCS 2008.
2. Koliński D., *Formalny opis funkcji zależnościowych systemów srk dla współczesnych posterunków ruchu*. Prace Naukowe - Transport z.86, Warszawa 2012, s.35-52.
3. Kawalec P., Rżysko M., *Komputerowo wspomagana specyfikacja funkcji zależnościowych urządzeń srk w językach opisu sprzętu*. Technika Transportu Szynowego, nr 9/2012, s.1605-1614.
4. Dąbrowa-Bajon M., *Podstawy sterowania ruchem kolejowym*. OWPW, Warszawa 2007.
5. Borecký J., Kubalík P., Kubátová H., *Reliable Railway Station System based on Regular Structure implemented in FPGA*. 12th Euromicro Conf. on Digital System Design, Los Alamitos: IEEE Computer Society (2009), s. 348-354.
6. Traczyk W., *Układy cyfrowe. Podstawy teoretyczne i metody syntezy*. Wydawnictwa Naukowo-Techniczne, Warszawa 1982.

7. *Wytyczne techniczne budowy urządzeń sterowania ruchem kolejowym (WTB-E10).* PKP PLK S.A., Warszawa 1997.
8. *Wytyczne w zakresie zobrazowania, wprowadzania poleceń oraz rejestracji zdarzeń dla komputerowych stanowisk obsługi urządzeń sterowania ruchem kolejowym (Ie-104).* PKP PLK S.A., Warszawa 2012.
9. Kawalec P., Rżysko M., *Zastosowanie grafów przejść automatów skończonych do opisu algorytmów działania urządzeń srk.* Prace Naukowe - Transport z.95, Warszawa 2013, s.221-230.

# VERIFICATION OF INTERLOCKING EQUATIONS USING LOGIC SIMULATORS IN ACTIVE-HDL ENVIRONMENT

***Abstract***

*This paper presents a formal method for railway interlocking logic verification. After decomposing the railway control system into objects and algorithms, interlocking functions were described using formal methods. This allowed the use of integrated design environment - Active-HDL, for verification of the whole design and building VHDL code as a final description. Various methods of manual and automated simulation are presented, showing the capabilities of the described design and verification method.*

***Autorzy**:*
prof. nzw. dr hab. inż. **Piotr Kawalec** – Politechnika Warszawska, Wydział Transportu
inż. **Marcin Rżysko** – Bombardier Transportation (Rail Engineering) Polska