PROBLEMY KOLEJNICTWA
RAILWAY REPORT Zeszyt 186 (marzec 2020)
INSTYTUT KOLEJNICTWA
DOI: 10.36137/1867P
ISSN 0552-2145 (druk) ISSN 2544-9451 (on-line)

Bezpieczeństwo jednokanałowych urządzeń sterowania ruchem kolejowym

Roman PNIEWSKI[1]

Streszczenie
Bramki rewersyjne pozwalają na tworzenie układów cyfrowych odpornych na uszkodzenie. Zastosowanie tych bramek umożliwia realizację bezpiecznych układów sterowania. Obecnie, w systemach srk, zamiast układów przekaźnikowych stosuje się systemy mikroprocesorowe. Zastosowanie techniki komputerowej umożliwia konstrukcję bezpiecznych systemów, jednak ze względu na dużą liczbę elementów, maleje niezawodność. Rozwiązaniem alternatywnym jest wykorzystanie logiki rewersyjnej w syntezie systemów cyfrowych. W artykule przedstawiono podstawowe bramki rewersyjne oraz przykład ich wykorzystania w syntezie systemów cyfrowych. Zaletą logiki odwracalnej jest możliwość syntezy układów samotestujących i odpornych na błędy. Wykorzystanie tych układów umożliwi konstrukcję bezpiecznych systemów sterowania. W artykule przedstawiono propozycję zastosowania logiki rewersyjnej w prostych układach sterowania. Pokazano sposób modelowania i symulacji układu opartego na bramkach rewersyjnych.

Słowa kluczowe: logika rewersyjna, systemy srk, bezpieczeństwo

1. Wprowadzenie

W latach siedemdziesiątych ubiegłego wieku do sterowania ruchem na kolei (urządzenia srk) zaczęto wykorzystywać układy elektroniczne. Układy elektroniczne, a w szczególności systemy cyfrowe, zaczęły wypierać stosowane wcześniej systemy kluczowe i przekaźnikowe. Zwiększenie stopnia integracji w układach scalonych umożliwiło budowę urządzeń srk realizujących coraz bardziej rozbudowane funkcje. Pojawienie się sterowników przemysłowych i przemysłowych wersji komputera PC (wraz z wykorzystaniem systemów operacyjnych czasu rzeczywistego) umożliwiło zastosowanie rozwiązań programowych do realizacji algorytmów działania urządzeń srk. We współczesnych, cyfrowych systemach srk, algorytmy sterowania, przetwarzania i przechowywania danych są realizowane głównie w sposób programowy, zwykle w układach mikroprocesorowych, w których przeprowadzenie zadanego algorytmu odbywa się zgodnie z przechowywanym w pamięci programem.

Przekaźnikowe systemy srk były projektowane jako systemy bezpieczne, oparte na regule *fail-safe*. Oznaczało to, że żadne pojedyncze uszkodzenie nie może prowadzić do błędnego wysterowania urządzeń zewnętrznych (sygnalizatora, zwrotnicy). Z tego wynika, że w przekaźnikowych urządzeniach, pojedyncze uszkodzenie musi wymuszać zmianę stanu systemu na taki, który jest zdefiniowany jako stan bezpieczny. W urządzeniach komputerowych stosowana jest redundancja. Rozwiązania zastosowane w komputerowych systemach sterowania ruchem, wykorzystują zwykle dwa komputery realizujące ten sam algorytm sterowania i kontrolujące nawzajem swoje działanie [9, 11].

Alternatywą dla rozwiązań komputerowych (które ze względu na znaczną rozbudowę powodują zmniejszenie niezawodności) może być powrót do rozwiązań sprzętowych (elektronicznych), bądź sprzętowo-programowych (układy SOC – *System On Chip*) [5, 8], uwzględniających rozwój technologii specjalizowanych układów scalonych. We współczesnych systemach automatyki kolejowej (srk) coraz powszechniej są stosowane specjalizowane układy cyfrowe. Nadrzędnym celem układów srk jest zapewnienie bezpieczeństwa, dlatego metody projektowania tych systemów odbiegają od powszechnie stosowanej metodologii syntezy systemów cyfrowych. Przy projektowaniu układów cyfrowych największy nacisk kładzie się na minimalizację funkcji logicznych, opisujących system. W systemach srk najistotniejszym

---

[1] Dr hab. inż., prof. UTH Rad.; Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu, Wydział Transportu i Elektrotechniki; e-mail: r.pniewski@uthrad.pl.

jest określenie sposobu działania układu w sposób zdeterminowany, a projektant powinien przewidzieć, jak zadziała układ w każdej możliwej sytuacji. Niezależnie od sposobu realizacji algorytmów sterowania, współczesne urządzenia i systemy srk muszą spełniać odpowiednie normy bezpieczeństwa. Dla nowych systemów muszą być spełnione wymagania ujęte w następujących normach:

1. PN-EN 50126: Zastosowania kolejowe – Specyfikacja niezawodności, dostępności, podatności utrzymaniowej i bezpieczeństwa;
2. PN-EN 50128: Zastosowania kolejowe – Łączność, sygnalizacja i systemy sterowania – Programy dla kolejowych systemów sterowania i zabezpieczeń;
3. PN-EN 50129: Zastosowania kolejowe – Łączność, sygnalizacja i systemy sterowania – elektroniczne systemy sygnalizacji związane z bezpieczeństwem.

Normy te definiują większość wymagań dotyczących rozwiązań sprzętowych, programowych i sprzętowo-programowych. I tak sprzęt stosowany w urządzeniach i systemach srk powinien spełniać wymagania norm PN-EN 50126 i PN-EN 50129, natomiast programy realizujące funkcje sterowania powinny być zgodne z wymaganiami normy PN-EN 50128.

W przytoczonych normach nie przedstawiono wymagań dotyczących wspomaganego komputerowo procesu specyfikacji bądź wytwarzania układów scalonych, przeznaczonych do systemów srk. Alternatywą dla klasycznych układów cyfrowych może być zastosowanie bramek rewersyjnych, które umożliwiają kontrolę stanu systemu cyfrowego. Rozwiązanie to pozwala na tworzenie jednokanałowych, bezpiecznych systemów zależnościowych dla kolei.

2. Bramki rewersyjne

Konwencjonalne komputery wykorzystują dwuwartościową logikę Boole’a. Funkcje opisujące układ cyfrowy wykorzystują najczęściej dwa operatory AND i OR. Te dwie operacje, posiadają kilka bitów wejściowych i jeden wyjściowy, co powoduje zmniejszenie informacji na wyjściu. Gdy układ ma mniej dostępnych stanów, to jego entropia staje się mniejsza [12], ponieważ zgodnie z drugą zasadą termodynamiki, entropia w zamkniętym układzie jest stała, to zmniejszenie entropii w jednym miejscu musi być skompensowane generacją entropii w innym miejscu. Generowana entropia wskutek skasowania bitu informacji wynosi:

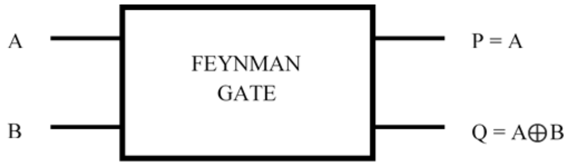
$$\Delta S = k_b T \cdot \ln 2 .$$

Zatem komputery pracujące na podstawie algebry Boole’a są zawsze urządzeniami rozpraszającymi energię nie mniejszą niż $k_b T \cdot \ln 2$.

Ta generacja ciepła w ciągu procesu obliczeniowego stanowi ograniczenie możliwej szybkości komputera ze względu na ilość generowanego ciepła. Fredkin i Toffoli udowodnili, że rewersyjne bramki logiczne mogą tworzyć podstawę dla komputera uniwersalnego. Kolejną zaletą (oprócz mniejszej mocy rozpraszanej) bramek rewersyjnych jest możliwość konstruowania z nich układów samotestujących i układów odpornych na uszkodzenia.

W literaturze można znaleźć wiele przykładów bramek odwracalnych (rewersyjnych) – od podstawowych, opracowanych wiele lat temu, po ich nowsze odmiany (uwzględniające kontrolę parzystości wejść i wyjść) [2, 3, 4]. W niniejszym artykule zamieszczono przykłady podstawowych bramek rewersyjnych wraz z tablicami prawdy i kodem w języku VHDL.

Bramka Feynmana należy do bramek dwukubitowych. Symbol bramki przedstawiono na rysunku 1, w tablicy 1 zaś pokazano funkcję przejścia bramki oraz zamieszczono opis bramki w języku VHDL.



Rys. 1. Bramka Feynmana; opracowanie własne na podstawie [3]

Tablica 1

Funkcja przejść bramki Feynmana

A	**B**	**P = A**	**Q = A⊕B**
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

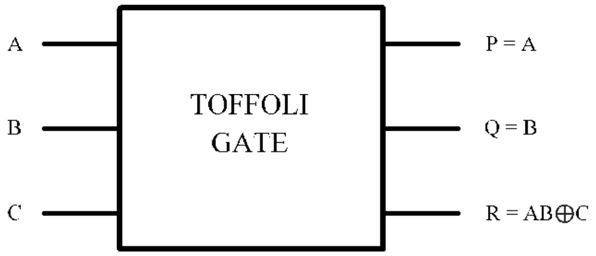
[Opracowanie własne].

```
Library ieee[2];
Use ieee std_logic.1164..all;
Entity feynmang is
Port(A, B: in std_logic;
P, Q: out std_logic);
end feynmang;
architecture ckt of feynmang is
begin
P<= A;
Q<= A xor B;
End ckt;
```

---

[2] Standardowa biblioteka typów danych.

Bramka Toffoliego to trzykubitowa bramka kwantowa zwana podwójnie sterowaną negacją. Symbol bramki pokazano na rysunku 2, w tablicy 2 zaś przedstawiono funkcję przejść bramki i opis bramki w języku VHDL.



Rys. 2. Bramka Toffoliego; opracowanie własne na podstawie [3]

Tablica 2

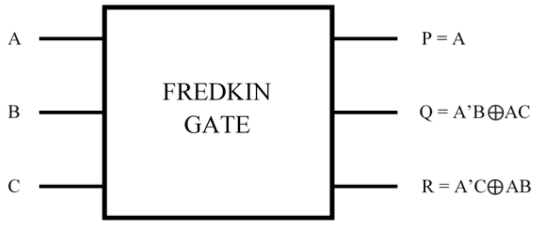
Funkcja przejść bramki Toffoliego

A	B	C	$P = A$	$Q = B$	$R = AB\oplus C$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

[Opracowanie własne].

```
Library ieee;
Use ieee std_logic.1164..all;
Entity toffolig is
Port(A, B, C : in std_logic;
P, Q, R : out std_logic);
end toffolig;
architecture ckt of toffolig is
signal s1 : std_logic;
begin
P<= A;
Q<= B;
S1<=A and B;
R<= S1 xor C;
End ckt;
```

W odróżnieniu od bramki Toffoliego, która ma dwa bity kontrolne i jeden bit celowy, bramka Fredkina (rys. 3, tabl. 3) ma jeden kontrolny kubit i dwa bity celowe. Celowe bity wymieniają się, jeżeli kontrolny bit jest równy 1, w przeciwnym wypadku pozostają one bez zmian.



Rys. 3. Bramka Fredkina; opracowanie własne na podstawie [3]

Tablica 3

Funkcja przejść bramki Fredkina

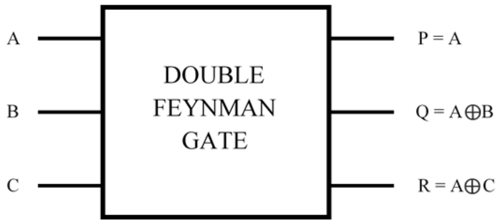
A	B	C	$P = A$	$Q = A\oplus B$	$R = AB\oplus C$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

[Opracowanie własne].

```
Library ieee;
Use ieee std_logic.1164..all;
Entity fredking is
Port(A, B, C : in std_logic;
P, Q, R : out std_logic);
end fredking;
architecture ckt of fredking is
signal Abar, S1, S2, S3, S4 : std_logic;
begin
P<= A;
Abar<= not A;
S1<=Abar and B;
S2<= A and C;
Q<= S1 xor S2;
S3<= Abar and C;
S4<= A and B;
R<= S3 xor S4;
End ckt;
```

2.1. Bramki zachowujące parzystość

Istnieje wiele znanych od dawna bramek odwracalnych, zachowujących parzystość (jak np. podwójna bramka Feynmana lub Fredkina), jednak w ostatnich latach pojawiły się nowe koncepcje (jak np. nowa bramka odporna na błędy lub bramka Islama) [1, 6, 7]. W dalszej części artykułu, przy tworzeniu systemu odpornego na błędy, wykorzystano nową bramkę odporną na błędy (ang. *New Fault Tolerant*). Na rysunku 4 przedstawiono symbol podwójnej bramki Feynmana w tablicy 4 zaś jej funkcje przejść, a na rysunku 5 symbol bramki NFT i tablice prawdy. W tablicy 5 zamieszczono funkcje przejść.



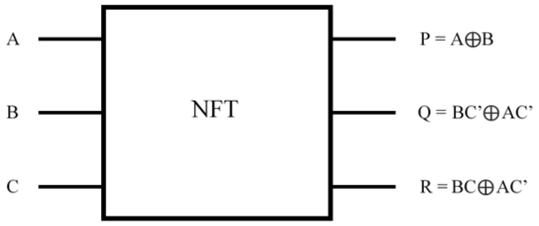
Rys. 4. Podwójna bramka Feynmana; opracowanie własne na podstawie [1]

Tablica 4

Funkcja przejść podwójnej bramki Feynmana

A	B	C	P = A	$Q = A \oplus B$	$R = A \oplus C$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

[Opracowanie własne].



Rys. 5. Bramka NFT; opracowanie własne na podstawie [1]

Tablica 5

Funkcja przejść bramki NFT

A	B	C	$P = A \oplus B$	$Q = BC' \oplus AC'$	$R = BC \oplus AC'$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	1

[Opracowanie własne].

W tablicy prawdy nie trudno zauważyć, że wybrana bramka spełnia kryterium zachowania parzystości. Wykonując operację alternatywy rozłącznej dla wszystkich wartości na wejściu bramki ($A \oplus B \oplus C$) i porównując ją z takim samym działaniem dla wyjść ($P \oplus Q \oplus R$), zawsze otrzymuje się równość. Zależność ta okaże się bardzo pomocna przy określaniu poprawności działania poszczególnych bramek.

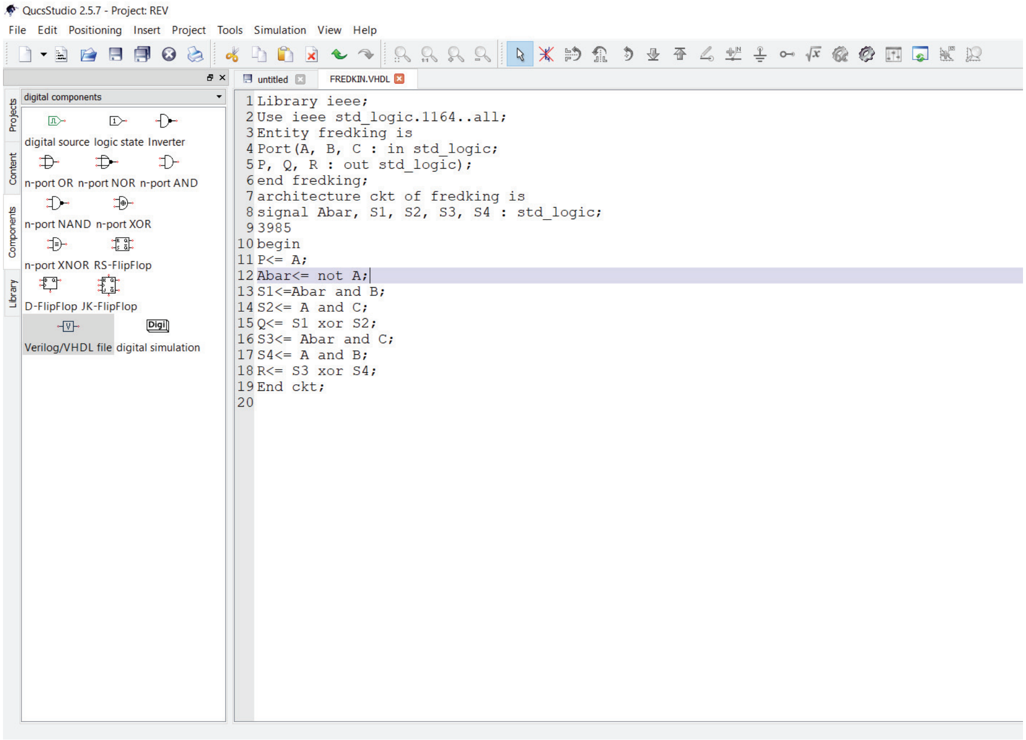
3. Symulacja układów z bramkami rewersyjnymi

Analiza bezpieczeństwa systemów srk wymaga sprawdzenia ich działania w różnych sytuacjach, ponieważ konieczne jest sprawdzenie reakcji układu na występujące błędy i zakłócenia. Nie wszystkie stany układu można wymusić w warunkach laboratoryjnych, dlatego przy opracowywaniu dowodu bezpieczeństwa często stosuje się symulacje komputerowe. W Zakładzie Systemów Sterowania w Transporcie UTH opracowano modele symulacyjne bramek rewersyjnych dla dwóch symulatorów układów cyfrowych QUCS i Multisim.

Program QUCS (*Quite Universal Circuit Simulator*) jest darmowym symulatorem układów elektronicznych, analogowych i cyfrowych. Pierwsza wersja programu była przeznaczona dla systemu Linux, aktualne wersje programu pracują w systemach operacyjnych Linux i Windows. Oprogramowanie jest w pełni darmowe, dostępne są kody źródłowe programu, co umożliwia jego modyfikację. Do symulacji układów analogowych wykorzystano algorytm SPICE, natomiast symulacja cyfrowa układów przebiega wieloetapowo. Na podstawie schematu ideowego, program generuje listę połączeń i zapisuje w pliku netlist.txt. Lista jest zapisana w języku VHDL (*VHSIC Hardware Description Language*), następnie dokonywana jest konwersja na język C (przy wykorzystaniu środowiska FreHDL). Do kompilacji otrzymanego kodu źródłowego zastosowano kompilator Mingw. Procesem symulacji w QUCS „steruje" plik wsadowy „qucsdigi.bat". Modyfikacja tego pliku pozwala na dowolne sterowanie procesem symulacji. Dzięki takiemu rozwiązaniu jest możliwe dołączenie do symulatora własnych programów działających „wsadowo". Program umożliwia również włączanie do schematu plików opisujących moduły w językach Verilog i VHDL. Symbol dołączonego pliku tworzony jest automatycznie na podstawie opisu interfejsu. Na rysunku 6 pokazano zrzut ekranu z projektowania modelu bramki rewersyjnej.

Program Multisim jest narzędziem do symulacji i analizy układów elektronicznych oraz projektowania obwodów drukowanych. Jest produktem firmy Electronics Workbench wchodzącej w skład korporacji *National Instruments*. Multisim jest kompletnym systemem narzędzi projektowych, obejmującym między innymi:

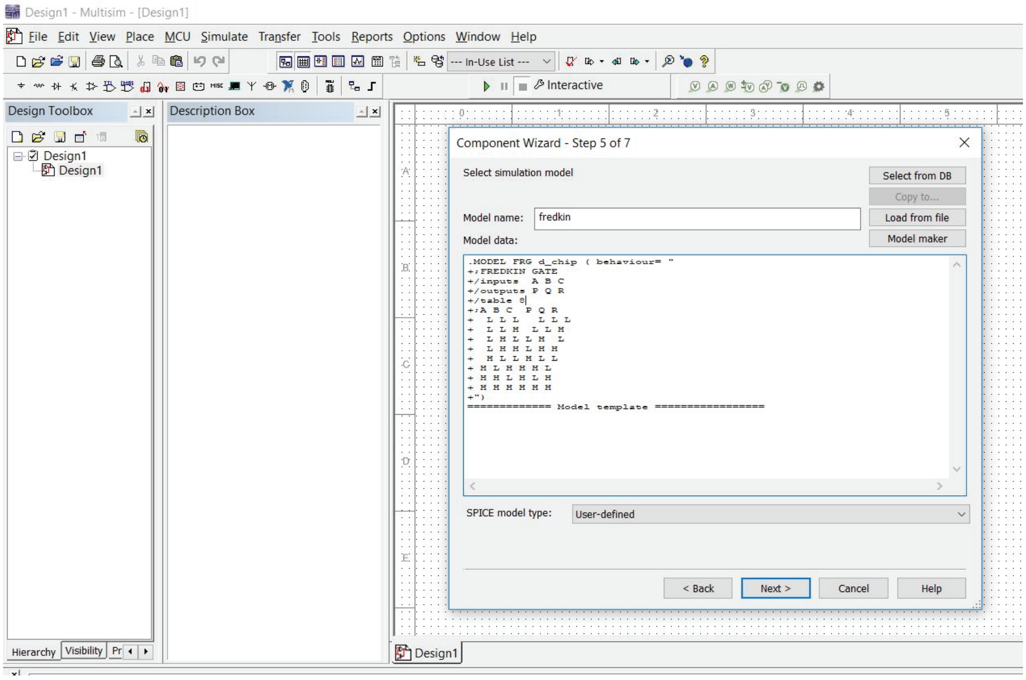
- graficzną edycję schematów,
- bogatą bazę elementów,
- symulację układów analogowych,
- symulację układów cyfrowych



```
Library ieee;
Use ieee std_logic.1164..all;
Entity fredking is
Port(A, B, C : in std_logic;
P, Q, R : out std_logic);
end fredking;
architecture ckt of fredking is
signal Abar, S1, S2, S3, S4 : std_logic;
3985
begin
P<= A;
Abar<= not A;
S1<=Abar and B;
S2<= A and C;
Q<= S1 xor S2;
S3<= Abar and C;
S4<= A and B;
R<= S3 xor S4;
End ckt;
```

Rys. 6. Wprowadzanie modelu bramki Fredkina w programie QUCS [opracowanie własne]

Do układu dołączane są przyrządy pomiarowe z grupy wyświetlaczy i wskaźników lub z listwy przyrządów. W pierwszym przypadku są to woltomierze i amperomierze, natomiast w drugim: multimetr, generator sygnałowy, oscyloskop, ploter Bodego, generator słów cyfrowych i temu podobne. Umożliwia to pomiar i obserwację sygnałów elektrycznych w interesujących punktach obserwowanego układu elektronicznego. W programie, w prosty sposób, można definiować nowe elementy. Po wybraniu z menu opcji Tools/Component Wizard, program prowadzi przez cały proces projektowania (łącznie z symbolem). Na rysunku 7 pokazano przykład tworzenia modelu dla bramki Fredkina.



Select simulation model

Model name: fredkin

Model data:

```
.MODEL FRG d_chip ( behaviour= "
+;FREDKIN GATE
+/inputs  A B C
+/outputs P Q R
+/table 8
+;A B C  P Q R
+  L L L  L L L
+  L L H  L L H
+  L H L L H  L
+  L H H L H H
+  H L L H L L
+ H L H H H L
+ H H L H L H
+ H H H H H H
+")
============= Model template =================
```

SPICE model type: User-defined

Rys. 7. Definiowanie modelu bramki Fredkina w programie Multisim [opracowanie własne]

4. Bezpieczeństwo układów

Podstawowym działaniem, umożliwiającym zapewnienie bezpieczeństwa, jest kontrola parzystości na wszystkich wyprowadzeniach (wejściach i wyjściach) bramek rewersyjnych (rys. 8). W założeniach jest to prosta metoda, jednak wymaga znacznego nadmiaru sprzętowego. Do każdego wejścia i wyjścia będą podłączone bramki kontrolowanej negacji (CNOT), które będą wykonywały operację – dla wejść i – dla wyjść. Wynik działania jest przechowywany na kolejnej linii sygnałowej, która zaczyna się od wartości „0". Jeżeli po przejściu przez wszystkie dodatkowe bramki sygnał testowy pozostanie zerem, wówczas sprawdzana bramka działa poprawnie. W przeciwnym wypadku, szybko można zidentyfikować wadliwą bramkę i dokonać wymiany jedynie uszkodzonego elementu. Nie jest to jednak rozwiązanie idealne – jeżeli któraś kontrolowana negacja zawiedzie (np. ostatnia), nawet w przypadku awarii nie będzie szansy szybko namierzyć przyczyny.

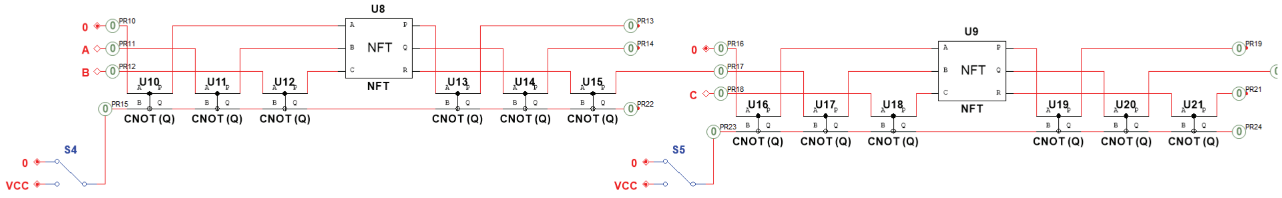
Znając potencjalne zagrożenie, jakim może być niepoprawnie funkcjonująca bramka w linii testowania, rozszerzono metodę klasyczną. Wiedząc, że przy zachowanej parzystości, wielokrotne wykonanie operacji XOR na linii sygnałowej nie zmieni wartości początkowej, zmieniono stałą wartość „0" na sygnał, którego wartość jest zmieniana w czasie. W takiej sytuacji, zmieniając okresowo logiczne „0" na logiczne „1" lub wysyłając impulsy testowe można sprawdzić, czy elementy CNOT działają poprawnie. Jest to znaczący skok w kierunku budowy niezawodnie działającego układu.

Jak już wspomniano, do zachowania bezpieczeństwa i niezawodności działania układu sterowania stosuje się zupełnie odmienną technikę. Zwykle sygnały wejściowe trafiają do dwóch niezależnych kanałów przetwarzania (dwóch odrębnych układów realizujących tę samą funkcję). Na wyjściach tych układów zastosowano aktywny komparator, który porównuje ich wyniki. Jeżeli obie wartości są identyczne, układy działają poprawnie. Kiedy jednak pojawi się różnica, jest to znak, że do któregoś z układów wkradł się błąd lub nastąpiła awaria. Jest znacznie mniejsze prawdopodobieństwo, że oba układy ulegną uszkodzeniu w tej samej chwili (i w ten sam sposób). W przypadku logiki odwracalnej wiadomo, że można monitorować poprawność działania każdego elementu z osobna, a dzięki poprawkom wprowadzonym w linii testowej, istnieje możliwość sprawdzenia, czy jej elementy funkcjonują bezbłędnie. Choć przedstawione metody wymagają sporych nakładów sprzętowych, pozwalają znacznie przyspieszyć i usprawnić identyfikację usterki. Mając na uwadze cechy, które wynikają z odwracalności, można w jednym kanale przetwarzania osiągnąć efekt podobny do tego, który obecnie jest osiągany na dwóch kanałach. Po stworzeniu układu z elementów zachowujących parzystość (i uzupełnieniu go o testowanie), można szeregowo podłączyć do niego ten sam układ w odwrotną stronę. Pokazano to na rysunku 9.

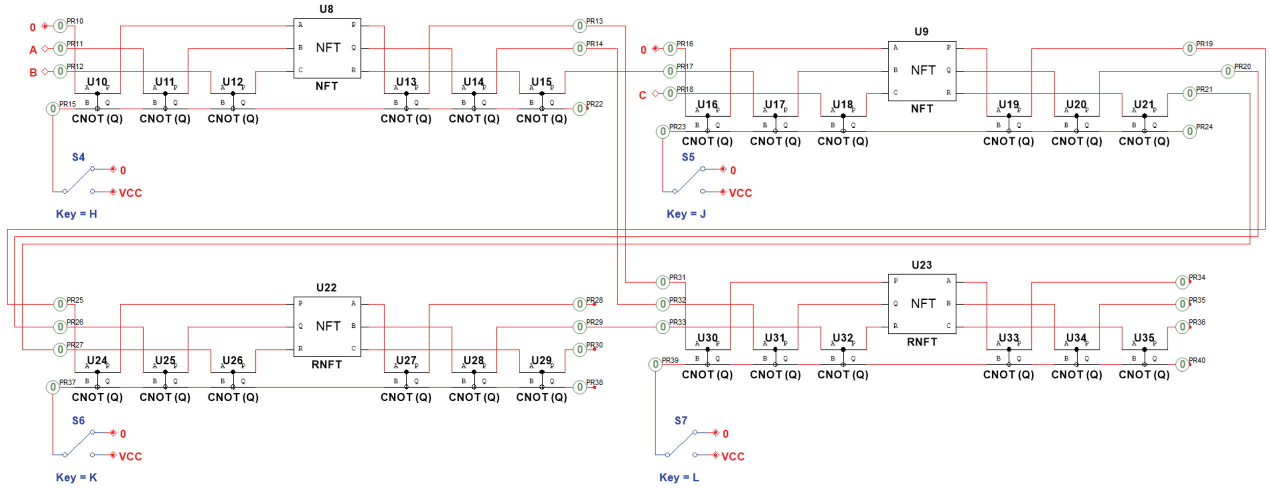
W takiej konfiguracji dwa razy zastosowano dokładnie ten sam układ, tylko w lustrzanym odbiciu. Wszystkie wyjścia stają się wejściami – i na odwrót. Korzystając z takiej metody, bezużyteczne wcześniej wyjścia stają się bardzo istotne. Obie części układu – standardowa i lustrzana – zachowują parzystość i stosują te same metody testowania w trakcie działania. Mając takie przygotowanie, można połączyć komparatorami oryginalne wejścia z lustrzanymi wyjściami – kontrolującymi tym samym poprawność przeliczania każdej ze zmiennych z osobna. W przypadku wystąpienia błędu, szybko można zlokalizować miejsce jego wystąpienia i zidentyfikować jego przyczynę. Takie podejście nie wyklucza dołożenia kolejnego kanału, aby mieć pełną redundancję na wypadek pojawienia się problemu i nie ma konieczności przerywania działania nawet w trakcie sprawnej eliminacji usterki.

5. Podsumowanie

Logika odwracalna, która swoimi korzeniami sięga fizyki kwantowej, ma wiele zastosowań w świecie algebry dwuwartościowej. Znacznie większe możliwości układów z logiką rewersyjną w zakresie testowania i wykrywania awarii (szczególnie w trakcie działania układu) są bardzo przydatne w przypadku syntezy odpornych na błędy układów sterowania cyfrowego [10]. Systemy zależnościowe, wykorzystywane w układach automatyki kolejowej, realizują najczęściej proste funkcje logiczne (w przeszłości do stero-



Rys. 8. Metoda kontroli parzystości w układach z bramkami rewersyjnymi [opracowanie własne]



Rys. 9. Jednokanałowe testowanie poprawnego funkcjonowania układu [opracowanie własne]

wania wystarczała technika przekaźnikowa). Zastosowanie w układach srk, przedstawionej w artykule logiki rewersyjnej, umożliwi realizację bezpiecznych układów jednokanałowych. Jest to szczególnie istotne ze względu na możliwość realizacji funkcji sterowania w cyfrowych strukturach programowalnych (FPGA), co umożliwi znaczne zwiększenie niezawodności systemów sterowania ruchem kolejowym [5, 8].

Literatura

1. Al Mahamud A. et.al.: *Synthesis of Fault Tolerant Reversible Logic Circuits*, Proceedings of IEEE International Conference on Testing and Diagnosis, Chengdu, China, 2009, pp. 1–4.
2. Bruce J.W. et.al.: *Efficient adder circuits based on conservative reversible logic gates*, In Proceedings of IEEE Computer Society Annual Symposium on VLSI, Pittsburg, PA, 2002, pp. 83–88.
3. Haghparast M., Navi K.: *A novel fault tolerant reversible gate for nanotechnology based systems*, American Journal of Applied Sciences, Vol. 5, No. 5, 2008, pp. 519–523.
4. Haghparast M., Navi K.: *Design of novel fault tolerant reversible full adder for nanotechnology based systems,* World Applied Sciences Journal, Vol. 3, No. 1, 2008, pps. 114–118.
5. Kawalec P., Szydłowski J., Mocki J.: *Realizacja wybranych algorytmów działania urządzeń srk w programowalnych strukturach logicznych*, International Scientific Conference Transport of the 21st Century, Warszawa, 2001.
6. Landauer R.: *Irreversibility and heat generation in the computational process*, IBM Journal of Research and Development, Vol. 5, Issue 3, 1961, pp. 183–191.
7. Parhami B.: *Fault-Tolerant Reversible Circuits*, Proceedings of 40th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2006, pp. 1722–1726.
8. Pniewski R.: *Metoda oceny bezpieczeństwa cyfrowych systemów automatyki kolejowej*, Wydawnictwo UTH, Radom 2013.
9. Pniewski R., Kornaszewski M., Chrzan M.: *Safety of electronic ATC systems in the aspect of technical and operational*, 16th International Scientific Conference Globalization and Its Socio-Economic Consequences. Proceedings, Part IV. s. 1729–1735, University of Zilina, The Faculty of Operation and Economics of Transport and Communications, Department of Economics, Rajecke Teplice, Slovak Republic, October 2016.
10. Santhi Swaroop V.G: *Implementation of Optimized Reversible Sequential and Combinational Circuits for VLSI Applications*, Int. Journal of Engineering Research and Applications Vol. 4, Issue 4 (Version 1), April 2014, pp. 382–388.
11. Siergiejczyk M.: *Wybrane zagadnienia systemów sterowania ruchem i łączności dla Kolei Dużych Prędkości w Polsce*, Logistyka, 2012, nr 3, s. 1991–2022.
12. Taha S.M.R.: *Reversible Logic Synthesis Methodologies with Application to Quantum Computing*, Springer International Publishing, Switzerland 2016.