Wydział Elektroniki

PRACA DOKTORSKA

Metody syntezy układów realizujących funkcje symetryczne i progowe testowalnych dla uszkodzeń typu opóźnienia

Piotr Patronik

Promotor: dr hab. inż Stanisław J. Piestrak, prof. PWr

słowa kluczowe:
cyfrowe układy scalone
niezawodność systemów komputerowych
sieci sortujące
testowanie układów cyfrowych
uszkodzenia typu opóźnienie

Wrocław 2006

Podziękowania

Chciałbym podziękować mojemu promotorowi, prof. Stanisławowi Piestrakowi za niezłomną wiarę w mój sukces, wiarę, której wielokrotnie mi brakowało. Pragnę również przekazać podziękowania dr inż Krzysztofowi Berezowskiemu za wiele cennych i pomocnych uwag.

Spis treści

Spis rysunków

Spis tablic

Wykaz akronimów

BDD	—	binarny diagram decyzyjny (ang. binary decision diagram)
BIST	—	wbudowany układ testujący (ang. built in self-test)
LFSR	—	rejestr przesuwny z liniowym sprzężeniem zwrotnym (ang. linear feedback shift register)
OE	—	układ nieparzysto-parzysty (ang. odd-even)
RG	—	własność generowania silnie testowalnych układów progowych (ang. robust generator) (def. 4.2)
SP	—	własność pojedynczej ścieżki sieci łączących (ang. single path) (def. 4.1)

Wykaz ważniejszych oznaczeń

D	—	dowolny stan logiczny
d_i	—	wektor wszystkich linii na poziomie i
$d_{i,j}$	—	linia j na poziomie i
$H(V)$	—	odpowiedź układu bez uszkodzeń po podaniu wzorca wejściowego V
$H_f(V)$	—	odpowiedź układu z uszkodzeniem f po podaniu wzorca wejściowego V
i	—	numer poziomu sieci
$i_{\max}$	—	numer najwyższego poziomu sieci (rozdz. 5.2.3)
j, k	—	numer wejścia, wyjścia lub linii sieci
$K^i_{j,k}$	—	komparator binarny z wejściami na poziomie i, łączący linie j i k
MN^n	—	sieć łącząca o n wejściach
MM^n	—	moduł łączący, element składowy sieci łączącej (rozdz. 5.2.2)
n	—	liczba wejść i wyjść układu
n_a, n_b	—	liczba wejść i wyjść elementów układów progowych, $n_a = \lfloor \frac{n}{2} \rfloor$, $n_b = \lceil \frac{n}{2} \rceil$, $n_a + n_b = n$
p	—	ścieżka, ciąg bramek i połączeń od wejścia do wyjścia układu
$p_{i \to j}$	—	ścieżka prowadząca z wejścia $T^{n_a}_i$ lub $T^{n_b}_i$ do wyjścia T^n_j
q_i	—	rozpiętość komparatora na poziomie i, $q_i = \frac{n}{2^{i+1}}$ dla sieci o $n = 2^k$
s/z	—	uszkodzenie typu sklejenie z z, $z \in \{0, 1\}$
S^n	—	funkcja symetryczna o n wejściach (def. 3.1)
S^n_m	—	elementarna funkcja symetryczna o n wejściach opisana liczbą m (def. 3.2)
S^n_A	—	złożona funkcja symetryczna o n wejściach opisana zbiorem liczb A (def. 3.3)
T^n	—	n-wejściowy, n-wyjściowy układ progowy

T^{n_a}, T^{n_b}	—	wielowyjściowe podukłady progowe o n_a lub n_b wejściach i wyjściach
T_i^n	—	i-te wyjście n-wyjściowego układu progowego (funkcja progowa n zmiennych binarnych o progu i)
T_{in}	—	wektor binarny na liniach wejściowych sieci łączącej
$w(X)$	—	waga Hamminga wektora binarnego X
U_{in}	—	wzorzec testowy dla uszkodzenia typu opóźnienie (def. 5.11)
u^{n_a}, u^{n_b}	—	wektory linii $u_j^{n_a}$ i $u_j^{n_b}$
$u_j^{n_a}$, $u_j^{n_b}$	—	linie za układem selekcji wstępnej (rozdz. 5.2.2)
V_{in}	—	wzorzec testowy dla uszkodzenia typu opóźnienie (def. 4.2)
W_{in}	—	linie za układem selekcji wstępnej, $W_{in} = \{u_1^{n_a}, \ldots, u_{n_a}^{n_a}, u_1^{n_b}, \ldots, u_{n_b}^{n_b}\}$, jak również wektor binarny na tych liniach (rozdz. 5.3.2)
z	—	stan logiczny 0 lub 1

Rozdział 1

Wprowadzenie

1.1 Wstęp

Możliwości techniczne wytwarzania układów scalonych bardzo wielkiej skali integracji (VLSI) podwajają się co dwa lata [26, 56]. Obecnie można wytworzyć układ VLSI zawierający setki milionów tranzystorów o wymiarach pojedynczych nanometrów, zaś aktualne trendy wskazują zapotrzebowanie na jeszcze większe możliwości [2]. Jednak spożytkowanie korzyści wynikających z nanotechnologii staje się niemożliwe, jeżeli nie można wystarczająco pewnie stwierdzić iż w wyprodukowanych lub użytkowanych układach nie znajdują się wady wewnętrzne. Przy dzisiejszych wymiarach układów problem ten staje się jeszcze bardziej istotny [9, 52].

Testowanie jest procesem zmierzającym do uzyskania odpowiedzi na pytanie, czy wyprodukowany egzemplarz układu jest sprawny, tj. czy działa zgodnie ze specyfikacją założoną przez projektantów. W ogólności, testowanie układu scalonego polega na podaniu na jego wejścia sekwencji pobudzeń, a następnie obserwowaniu wyjść w celu stwierdzenia czy odpowiedzi układu (wartości na wyjściach) są zgodne z oczekiwanymi. W przypadku dwuwejściowej bramki AND testowanie polega np. na podaniu kolejno wszystkich czterech kombinacji i sprawdzeniu, czy na wyjściu pojawia się jedynka tylko wtedy, gdy na wejścia podamy dwie jedynki, a w pozostałych trzech przypadkach pojawia się zero. W tym szcze-

gólnym przypadku, każda pojedyncza para bitów, czyli wektor dwubitowy, nazywana jest *wektorem testowym* albo w skrócie *testem*.

Tematyka testowania układów obejmuje m. in. następujące problemy [1, 3, 18, 74]: generowanie testów, czas testowania, szybkość testowania, dostęp do testowanego układu [17], sterowalność wejść i obserwowalność wyjść. Przewiduje się, że koszt testowania układu (w przeliczeniu na jeden tranzystor) może przewyższyć koszty jego wytworzenia [3, 15, 60]. Obok powszechnie znanych testów na występowanie uszkodzeń typu sklejenia, testy na występowanie uszkodzeń typu opóźnienia zostały zaliczone do grupy testów niezbędnych podczas wytwarzania układu [15]. Uszkodzenie typu sklejenie polega na pojawieniu się stałego sygnału logicznego 0 lub 1 niezależnie od podanego pobudzenia, jest zatem zmianą zachowania logicznego układu. Uszkodzenie typu opóźnienie polega na zmianie czasu propagacji zmiany wartości sygnału przez linie lub bramki [38, 47], jest zatem zmianą właściwości czasowych układu.

Testowanie uszkodzenia typu sklejenie z z polega na dostarczeniu przeciwnej wartości logicznej $\overline{z}$ do miejsca uszkodzenia i zapewnieniu warunków propagacji sygnału o przeciwnej niż poprawna wartości do wyjść układu. Dla układu kombinacyjnego pojedynczy test wymaga podania jednego wektora testowego na wejście układu. Przy testowaniu uszkodzeń typu opóźnienie, niezależnie od modelu uszkodzenia typu opóźnienie, testowanie uszkodzenia polega na podaniu na wejście układu pary wektorów testowych, nazywanych *wzorcem testowym* (ang. test pattern) [38]. Można zatem w skrócie powiedzieć, iż testowanie uszkodzenia typu sklejenie wymaga podania pojedynczego wektora na wejście układu, natomiast testowanie uszkodzenia typu opóźnienie wymaga podania pary wektorów binarnych, przez co testowanie uszkodzeń typu opóźnienie jest generalnie znacznie trudniejsze.

Istnieje wiele technik testowania układów logicznych [1, 18, 74], z których najbardziej istotne są przedstawione poniżej.

- Przy *testowaniu wyczerpującym* na wejście testowanego podawane są wszystkie możliwe kombinacje. Dzięki tej metodzie możliwe jest uzyskanie 100% pokrycia uszkodzeń. Istota metody jest prosta, a koszt sprzętowy niewielki: wystarczy licznik bi-

narny lub rejestr przesuwny z liniowym sprzężeniem zwrotnym (ang. linear feedback shift register (LFSR)).

- Przy *testowaniu pseudowyczerpującym* układ jest dzielony na części, z których każda jest następnie testowana wyczerpująco. Liczba kombinacji przy testowaniu części układu jest istotnie mniejsza niż przy testowaniu wyczerpującym całego układu. Aby możliwe było zastosowanie testowania pseudowyczerpującego, konieczna jest modyfikacja układu, a przez to jego powiększenie.

- Przy *testowaniu deterministycznym* konieczne jest przeprowadzenie analizy układu i sporządzenie zbioru pobudzeń z użyciem np. algorytmu PODEM (dla uszkodzeń typu sklejenie) lub RESIST czy NEST (dla uszkodzeń typu opóźnienie). Test deterministyczny pozwala wymusić wystąpienie błędu w układzie z uszkodzeniem (*sterowalność*) a następnie pozwala na propagowanie wartości błędnej wynikającej z wystąpienia uszkodzenia do obserwowalnych wyjść układu (*obserwowalność*). Użycie tej metody gwarantuje wykrycie każdego uszkodzenia z listy uszkodzeń branych pod uwagę podczas generowania testów. Główną wadą tej metody jest generalnie duża złożoność obliczeniowa analizy dla układów o dowolnej strukturze oraz konieczność przechowywania zbiorów testów, niekiedy bardzo dużych.

- Odmianą testowania wyczerpującego jest *testowanie pseudolosowe*, w którym wektory są wytwarzane są przez generatory ciągów pseudolosowych. Jako generatory ciągów pseudolosowych najczęściej wykorzystywane są LFSR lub automaty komórkowe (ang. cellular automata). LFSR i automaty komórkowe cechuje prosta struktura i możliwość wykorzystania tego samego układu do porównania wektora wyjściowego badanego układu. Ich dużą zaletą jest możliwość generowania dużych zbiorów testów bez potrzeby zasilania dodatkowymi danymi. Wadami metody testowania pseudolosowego są duże (co najmniej 10-krotnie większe) długości sekwencji koniecznej do wykrycia zbioru uszkodzeń w porównaniu z testem deterministycznym. Ciągi pseudolosowe w niewielkim stopniu nadają się też do testowania bramek logicznych o bardzo

dużej liczbie wejść.

- Przy *testowaniu mieszanym* większość uszkodzeń jest wykrywana metodą pseudolosową, natomiast zbiór trudno wykrywalnych uszkodzeń jest wykrywany relatywnie niewielkim zbiorem testów deterministycznych.

W poniższej pracy będziemy zajmowali się układami cyfrowymi realizującymi wielowyjściowe funkcje progowe i funkcje symetryczne, zdefiniowane następująco. Niech będzie dany n-elementowy zbiór zmiennych binarnych $X = \{x_1, x_2, \ldots, x_n\}$ oraz liczba całkowita m oznaczająca próg.

Funkcja *progowa* $T_m^n(X)$: jest to funkcja przełączająca n zmiennych ze zbioru X, która przyjmuje wartość 1 wtedy i tylko wtedy, gdy co najmniej m zmiennych ze zbioru X przyjmuje wartość 1. Wielowyjściowy układ progowy T^n jest to układ progowy który implementuje wszystkie n funkcji progowych n zmiennych T_m^n, $1 \leqslant m \leqslant n$.

Funkcja *symetryczna* $S_m^n(X)$ jest to funkcja przełączająca n zmiennych ze zbioru X, która przyjmuje wartość 1 wtedy i tylko wtedy, gdy dokładnie m zmiennych ze zbioru X przyjmuje wartość 1.

Istnieje wiele zastosowań wielowyjściowych układów progowych i układów implementujących funkcje symetryczne, spośród których do najważniejszych należą:

- kodery i samotestowalne układy kontrolne dla klas kodów wykrywających błędy jednokierunkowe [29, 63, 66, 67, 90],
- generatory resztowe i wielooperandowe sumatory resztowe mod n [65],
- elementy układów mnożących, takie jak kompresory (4:2) i liczniki jedynek [10],
- konwertery redukujące moc przełączania pobieraną podczas transmisji [83, 86, 87, 96],
- sygnalizatory zakończenia operacji w układach asynchronicznych niewrażliwych na opóźnienia (ang. delay-insensitive) [57, 68],
- różne filtry cyfrowe, w tym: filtry medianowe [8, 11, 12, 42, 95], filtry bazujące na statystykach porządkowych [13, 46, 50] oraz filtry boolowskie [43],

- różne elementy realizujące operacje arytmetyczne i logiczne w nanoukładach [97] oraz
- sieci neuronowe [6].

Funkcje symetryczne i wielowyjściowe układy progowe można implementować jako układy logiczne o różnych parametrach (złożoność mierzona liczbą bramek bądź wejść bramek, opóźnienie mierzone liczbą poziomów bramek). Szczegółowy przegląd implementacji wielowyjściowych układów progowych można znaleźć w [64, 66] oraz w [6]. Istnieje także obszerna klasa implementacji funkcji progowych bazująca na układach analogowych [37], która jednakże pozostaje poza sferą naszych zainteresowań. Spośród implementacji wielowyjściowych układów progowych wykorzystujących układy logiczne [21, 35, 64, 73] możemy wyróżnić następujące klasy:

- układy wielopoziomowe, zrealizowane rekursywnie jako układy o rozłącznych podukładach [21, 73];
- sieci sortujące [35, 64].

Problematyka implementacji funkcji symetrycznych jako układów logicznych była podejmowana w [14, 19, 31, 34, 66, 73, 80]. Ogólnie, implementacje funkcji symetrycznych możemy sklasyfikować następująco:

- układy o dwóch, trzech lub czterech poziomach bramek, niezależnie od liczby wejść [14, 19, 31, 34, 80] oraz
- układy zaimplementowane z użyciem dowolnego wielowyjściowego układu progowego i układu NOT-AND-[OR] [66, 73].

1.2 Cel i teza rozprawy

Celem niniejszej pracy jest zaproponowanie takich implementacji funkcji symetrycznych i wielowyjściowych układów progowych, w których testowanie uszkodzeń typu opóźnienie ścieżki jest łatwe.

Teza:

- Istnieje możliwość syntezy układów implementujących funkcje symetryczne i progowe mające własność silnej testowalności uszkodzeń typu opóźnienie o złożoności mniejszej niż najlepsze układy znane z literatury.

W rozdziale 2 przedstawiono modele uszkodzeń typu opóźnienie oraz omówiono najczęściej występujące defekty powodujące uszkodzenia typu opóźnienie. Przedstawiono również ogólne zasady testowania uszkodzeń typu opóźnienie.

W rozdziale 3 przedstawiono własności i implementacje funkcji symetrycznych i wielowyjściowych układów progowych oraz omówiono związki pomiędzy funkcjami symetrycznymi a funkcjami progowymi.

W rozdziale 4 przedstawiono własności testowalności uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych zrealizowanych jako całkowicie rozgałęzione oraz omówiono własności układów łączących przesądzające o możliwości testowania uszkodzeń typu opóźnienie ścieżki.

W rozdziale 5 przedstawiono nową metodę syntezy wielowyjściowego układu progowego mającego własność silnej testowalności uszkodzeń typu opóźnienie ścieżki, opartą na modyfikacji sieci sortującej nieparzysto-parzystej Batchera. Celem przedstawienia algorytmu modyfikacji przeprowadzono analizy struktury sieci nieparzysto-parzystej i analizy własności silnej testowalności uszkodzeń typu opóźnienie ścieżki. Na koniec podano algorytm modyfikacji sieci celem uzyskania własności silnej testowalności uszkodzeń typu opóźnienie ścieżki.

W rozdziale 6 zawarto analizę złożoności uzyskanych układów i ich porównanie z najlepszymi znanymi rozwiązaniami. Dane liczbowe uzyskano z użyciem specjalizowanego pakietu oprogramowania umożliwiającego automatyczną syntezę różnych klas układów, ocenę ich złożoności i weryfikację silnej testowalności uszkodzeń typu opóźnienie ścieżki. W zestawieniach wykazano, że zaproponowana implementacja funkcji symetrycznych mająca własność silnej testowalności uszkodzeń typu opóźnienie ścieżki ma złożoność mniejszą od złożoności którejkolwiek ze znanych implementacji.

Rozdział 7 stanowi podsumowanie problemów naukowych rozwiązanych w niniejszej rozprawie, jak również przedstawia listę otwartych problemów naukowych mogących stanowić kontynuację badań rozpoczętych w ramach niniejszej rozprawy.

Rozdział 2

Przegląd zagadnień testowania uszkodzeń typu opóźnienie

W poniższym rozdziale przedstawimy podstawowe modele uszkodzeń występujących w układach logicznych, ze szczególnym uwzględnieniem uszkodzeń typu opóźnienie, oraz podstawy testowania uszkodzeń typu opóźnienie.

2.1 Pojęcia podstawowe

2.1.1 Defekty i uszkodzenia

Defekt fizyczny (ang. physical defect, failure) występuje, gdy działanie układu cyfrowego odbiega od wynikającego ze specyfikacji. Defekty są rozpatrywane na poziomie fizycznym (elektronicznym) układu. Przykładem defektów są zwarcia linii do masy lub zasilania, zwarcia linii sygnałowych, przerwy w linii, przebicie lub zmiana rozmiaru tranzystora etc. Przyczynami wystąpienia defektów są wady materiałowe, parametry procesu fabrykacji, napięcia zasilania, warunki pracy układu i inne.

Uszkodzenie (ang. fault) jest modelem defektu i jest rozpatrywane na logicznym poziomie abstrakcji. Najczęściej stosuje się następujące modele uszkodzeń.

- Uszkodzenie typu *sklejenie*

Sklejenie (ang. stuck-at fault) występuje wtedy, gdy w połączeniu, linii wejściowej lub wyjściowej bramki logicznej pojawia się stały sygnał 1 (sklejenie z 1) lub 0 (sklejenie z 0), niezależnie od sygnałów podanych na wejście, co zapisujemy s/z, $z \in \{0, 1\}$.

- Uszkodzenie typu *zmostkowanie*

  Zmostkowanie (ang. bridging fault) występuje w przypadku galwanicznego (na poziomie fizycznym) zwarcia dwóch lub więcej linii. Dla niektórych technologii wykonania układu przyjmuje się, że na liniach zwartych realizowana jest jedna z funkcji elementarnych AND lub OR, analogicznie z sumą lub iloczynem galwanicznym (ang. wired-AND lub wired-OR).

- Uszkodzenie typu *opóźnienie*

  Uszkodzenie typu opóźnienie (ang. delay fault), w przeciwieństwie do opisanych powyżej, nie powoduje zmiany funkcji logicznej realizowanej przez układ. W przypadku jego wystąpienia, zmienia się jedynie czas generowania wartości funkcji. Uszkodzenia typu opóźnienie są podstawowym modelem uszkodzeń rozpatrywanych w niniejszej pracy, a ich dokładny i wyczerpujący opis zostanie podany w dalszej części rozdziału.

Analiza układów cyfrowych na poziomie uszkodzeń a nie defektów fizycznych ma szereg zalet [69].

1. Jeżeli uszkodzenie jest adekwatnym modelem defektów, to problem jest analizowany na poziomie logicznym a nie fizycznym.

2. Jest możliwe konstruowanie logicznych modeli uszkodzeń nadających się dla wielu różnych technologii. Tym samym, analiza wielu klas uszkodzeń staje się niezależna od technologii.

3. Stosowanie logicznych modeli uszkodzeń pozwala generować testy nawet dla tych uszkodzeń, których fizyczna przyczyna jest nieznana lub której wpływ na zachowanie układu nie jest dokładnie zrozumiany.

Uszkodzenia pojedyncze i wielokrotne

Uszkodzenie *pojedyncze* (ang. single fault) występuje wtedy, gdy jest jedynym uszkodzeniem w układzie. Uszkodzenie *wielokrotne* (ang. multiple fault) jest wtedy, gdy w układzie występuje więcej niż jedno uszkodzenie.

Błędy

Uszkodzenia na poziomie logicznym mają swoje następstwa w poprawności działania układu. Wyniki nieprawidłowego działania układu nazywa się *błędami* (ang. errors). Przy stosowanych modelach uszkodzeń błąd polega na innym ($\overline{z}$) niż oczekiwano (z) stanie logicznym, obserwowanym na wyjściu układu lub rejestru. Błędy obserwowane mogą pojawiać się jako następstwa odległych topologicznie, a w układach sekwencyjnych również czasowo, uszkodzeń.

2.1.2 Testy i testowanie uszkodzeń

Testowanie układu jest to proces zmierzający do ustalenia, czy układ jest sprawny, tj. czy nie występują w nim uszkodzenia. Testowanie polega na pobudzaniu wejść układu a następnie sprawdzaniu, czy generowane wartości są poprawne, tj. czy nie zawierają błędów.

Definicja 2.1 *Niech X będzie wektorem wejściowym układu, $H(X)$ będzie odpowiedzią układu bez uszkodzenia, a $H_f(X)$ będzie odpowiedzią układu z uszkodzeniem f. Wektor X nazywamy* testem *dla uszkodzenia f jeżeli $H(X) \neq H_f(X)$.*

Innymi słowy, test jest takim wektorem wejściowym, który sprawi, że wyjście układu z uszkodzeniem będzie różne od wyjścia układu bez uszkodzenia. Zbiór testów dla danego uszkodzenia może zawierać testy lub być zbiorem pustym. Jeżeli zbiór testów jest pusty, uszkodzenie jest *nietestowalne*.

Pokrycie zbioru testów

Zbiór testów może zawierać testy dla wszystkich uszkodzeń, które mogą wystąpić w układzie. Może również wystąpić taka sytuacja, że zbiór testów będzie zbyt duży, żeby przete-

stować wszystkie uszkodzenia, i wtedy zbiór testów będzie testował tylko część uszkodzeń. Określenie, jaka część zbioru uszkodzeń jest testowana przez dany zbiór testów, nazywa się *pokryciem zbioru testów* (ang. test set coverage).

Sterowalność i obserwowalność uszkodzeń

Niech będzie dane pewne uszkodzenie w układzie. Dane uszkodzenie jest *sterowalne*, jeżeli możliwe jest znalezienie testu, który je pobudzi. To samo uszkodzenie jest *obserwowalne*, jeżeli równocześnie istnieją warunki propagacji błędu spowodowanego tym uszkodzeniem do punktu obserwacji układu (np. wyjścia pierwotnego lub specjalnego wyjścia testowego).

2.2 Przyczyny i mechanizmy powstawania uszkodzeń typu opóźnienie

2.2.1 Defekty powodujące występowanie uszkodzeń typu opóźnienie

Wprowadzenie modelu uszkodzenia typu opóźnienie w latach 80. zostało spowodowane przez ujawnienie się nowych, niespotykanych wcześniej defektów. Skalowanie wymiaru charakterystycznego λ, zmniejszenie napięć zasilania i prądów sterujących sprawiły, że wpływ defektów wcześniej maskowanych stał się tak duży, że zaczął być obserwowany na poziomie logicznym. Do najważniejszych znanych obecnie defektów, powodujących występowanie uszkodzeń typu opóźnienie należą:

- zwarcia rezystancyjne pomiędzy liniami oraz zwarcia rezystancyjne linii do masy lub zasilania [22];
- przerwy linii o nieskończonej lub skończonej rezystancji [45], a w szczególności rozwarcia o niewielkiej rezystancji [55];
- zmiany parametrów RLC połączeń [98];

- fluktuacje, szum i zakłócenia napięcia zasilania [53, 54];
- defekty tranzystorów [94].

2.2.2 Dyskusja mechanizmów powstawania uszkodzeń typu opóźnienie

Uszkodzenia typu opóźnienie stanowią rozwinięcie modelu uszkodzenia typu sklejenie. Przy rozpatrywaniu uszkodzeń typu sklejenie, wystąpienie uszkodzenia oznacza, iż wartość logiczna odczytana z linii jest stała i nie zależy od wartości podanej na wejście. W uszkodzeniu typu opóźnienie ma znaczenie parametr określany jako czas propagacji. Pojęcie czasu propagacji wiąże się ze zmianą wartości logicznej. Przy uszkodzeniu typu opóźnienie wystąpienie uszkodzenia oznacza, iż czas propagacji zmiany wartości logicznej jest inny od założonego. Aby zrozumieć naturę uszkodzeń typu opóźnienia, konieczne jest przyjęcie, iż zjawiska w układach o rozmiarach nanometrowych mają naturę falową. Dla przykładu, rozważmy układ cyfrowy z pewnym połączeniem o opóźnieniu określonym na 1,5 ns i załóżmy, że odczyt danych z linii następuje po 1,6 ns od podania wartości na wejście. Na początku potencjał wzdłuż całej linii jest zerowy, przez co stan logiczny na wyjściu przyjmuje wartość 0. Aby wymusić na wyjściu wartość logiczną 1, konieczne jest ustalenie potencjału linii o wartości odpowiadającej logicznemu 1. Zmiana wartości na wejściu z 0 na 1 jest niejako wpuszczeniem na linię fali prostokątnej. W chwili gdy fala ta dotrze do wyjścia uzyskamy stan 1. Czas przepływu fali przez linię wynosi właśnie 1,5 ns. Jeżeli czas propagacji ulegnie np. zwiększeniu do 1,7 ns, w momencie odczytu wartość linii nie będzie wynosiła 1, lecz 0, co spowoduje wystąpienie błędu, takiego samego jak błędy występujące w uszkodzeniach typu sklejenie. Uszkodzenie typu opóźnienie nie musi oczywiście dotyczyć jedynie linii: elementami wprowadzającymi opóźnienie są także elementy przełączające, czyli tranzystory. Zatem uszkodzenia typu opóźnienia można także rozpatrywać zarówno w odniesieniu do bramek, jak i całych sieci logicznych. Test uszkodzenia typu opóźnienia polega na wymuszeniu zmiany stanu na wejściu układu, a następnie obserwacji wyjść. Aby umożliwić zmianę stanu konieczne jest uprzednie ustabilizowanie układu.

2.3 Modele uszkodzeń typu opóźnienie

2.3.1 Pojęcia podstawowe

Ścieżki i segmenty

Ścieżka w układzie jest to ciąg następujących po sobie bramek i połączeń. Początkiem ścieżki jest wejście pierwotne układu, a jej końcem jest wyjście pierwotne układu. *Segment* jest to fragment ścieżki. Początek segmentu może znajdować się na linii, bramce lub wejściu pierwotnym, a koniec na linii, bramce lub wyjściu pierwotnym.

Wejście główne i wejścia boczne

W bramkach będących elementem ścieżki, wejście będące częścią ścieżki jest nazwane *wejściem głównym* bramki (ang. on-path input). Wejścia bramki nie należące do ścieżki są nazywane wejściami *bocznymi* (ang. off-path inputs).

Zdarzenia, wartości sterujące i nie-sterujące

Zdarzenie (ang. event) jest zmianą wartości sygnału: przejściem $1 \rightarrow 0$ lub $0 \rightarrow 1$. Dowolne zdarzenie będzie oznaczane gwiazdką (*), a dowolny stan logiczny będzie oznaczany jako D. Rozważmy bramkę AND lub NAND i dowolne zdarzenie na jednym z wejść tej bramki. Niech wybrane wejście będzie wejściem głównym. Przeanalizujemy warunki, w których zdarzenie na wejściu głównym może pojawić się na wyjściu bramki. W przypadku bramki AND, aby zdarzenie na wejściu głównym mogło zostać propagowane do wyjścia bramki, konieczne jest aby stany na pozostałych wejściach (bocznych) miały wartość 1. Gdyby jedno z wejść bocznych miało wartość 0, wtedy, niezależnie od zdarzenia, stan na wyjściu bramki AND miałby wartość 0 i nie mogłaby nastąpić propagacja zdarzenia. Wartość 1 nazywa się *wartością nie-sterującą* (ang. non-controlling value) bramki AND, natomiast wartość 0 jest *wartością sterującą* (ang. controlling value) [47]. Przeciwnie niż w bramce AND lub NAND, wartością sterującą dla bramki OR lub NOR jest 1, natomiast wartością nie-sterującą jest 0.

Testy uszkodzeń typu opóźnienie

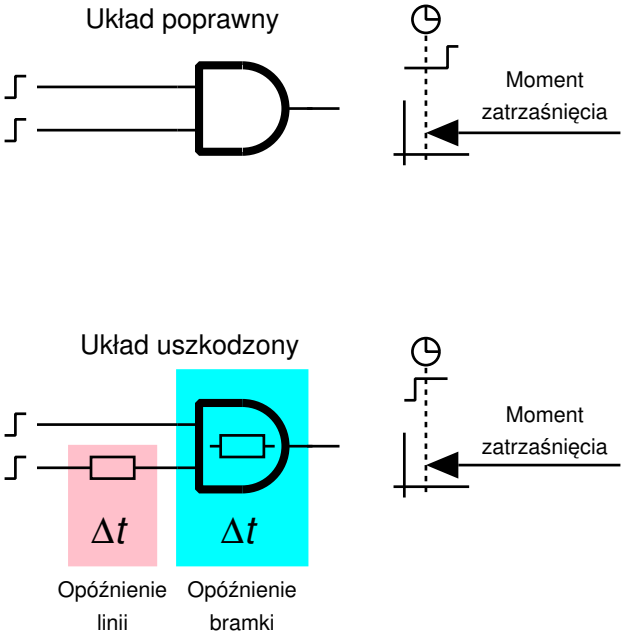
Testowanie uszkodzenia typu opóźnienie wymaga podania zdarzenia na wejście układu. Zdarzenie jest określone przez dwa wektory binarne, zapisane jako $\{v_1, v_2\}$, które różnią się między sobą na jednym lub większej liczbie bitów. Para wektorów $\{v_1, v_2\}$ jest nazywana *wzorcem testowym* (ang. test pattern) lub w skrócie *wzorcem* i jest oznaczana przez V. Zdefiniujemy teraz ponownie pojęcie testu.

Definicja 2.2 (Test uszkodzenia typu opóźnienie) *Niech V będzie wzorcem wejściowym układu, $H(V)$ będzie odpowiedzią układu bez uszkodzenia, $H_f(V)$ będzie odpowiedzią układu z uszkodzeniem f. Wzorzec V nazywamy* testem *dla uszkodzenia f jeżeli $H(V) \neq H_f(V)$.*

Wzorzec testowy podawany na wejście układu będzie nazywany *wejściowym wzorcem testowym* lub krótko *wzorcem wejściowym*. Odpowiedzią układu na wzorzec wejściowy jest para wektorów, która będzie nazywana *wyjściowym wzorcem testowym* lub *wzorcem wyjściowym*.

2.3.2 Uszkodzenia przejścia

Modelem nie będącym w zasadzie modelem uszkodzenia typu opóźnienie, ale stanowiącym bazę dla rozważania uszkodzeń typu opóźnienie jest model znany pod nazwą *uszkodzenia przejścia* (ang. transition fault) [24, 77, 78]. Istotą modelu uszkodzenia przejścia jest brak (lub duże opóźnienie) propagacji zmiany sygnału przez miejsce uszkodzenia. Testem dla modelu uszkodzenia przejścia jest para wektorów definiujących zdarzenie. Model ten można traktować jako model pośredni pomiędzy uszkodzeniem typu sklejenie, a uszkodzeniem typu opóźnienie. Tak samo jak w uszkodzeniach typu opóźnienie, testowanie polega na doprowadzeniu zdarzenia do miejsca uszkodzenia. Uszkodzenie jest skupione w punkcie układu, jak w uszkodzeniach typu sklejenie. Model uszkodzenia przejścia jest modelem pochodnym od modelu *uszkodzenia typu rozwarcie* (ang. stuck-open fault), rozwiniętego następnie do modelu *przerwy rezystancyjnej* (ang. resistive stuck-open fault) [45, 55].



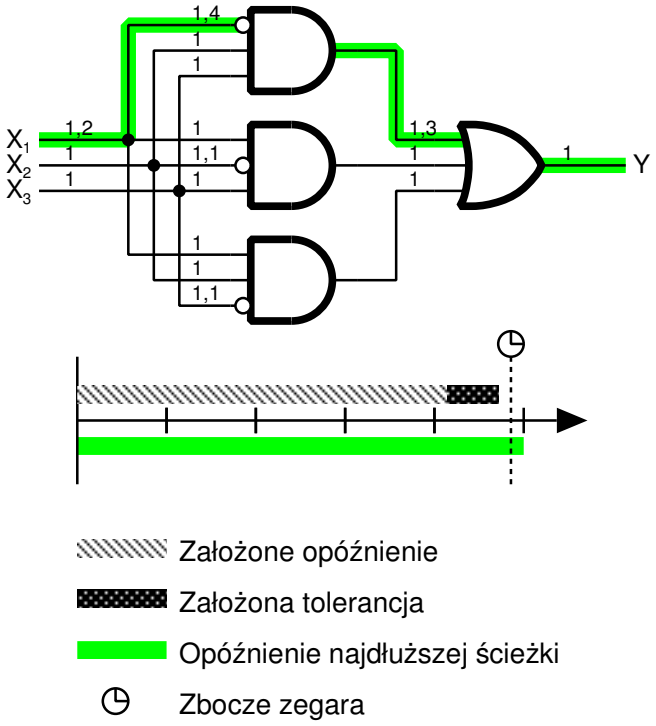
Rysunek 2.1: Ilustracja uszkodzeń typu opóźnienie linii i opóźnienie bramki

Uszkodzenie przejścia jest też związane z modelem uszkodzenia spowodowanego zwiększoną rezystancją przelotek, nazwanego *uszkodzeniem typu szeregowa rezystancja* (ang. inline resistance fault) [7].

Model uszkodzenia przejścia został przyjęty przez przemysł półprzewodnikowy [7, 24, 89, 93] i stąd jest szeroko rozpowszechniony, m. in. istnieje generator testów dla uszkodzeń przejścia TetraMAX firmy Synopsys [85]. Ostatnio wyróżniono nową klasę uszkodzenia przejścia nazwaną *uszkodzeniem typu opóźnienie propagacji* (ang. propagation delay fault) [48], który zostanie omówiony w ramach uszkodzenia typu opóźnienie segmentu w podrozdziale 2.3.5.

2.3.3 Uszkodzenia typu opóźnienie linii/bramki

W modelu *uszkodzenia typu opóźnienie linii/bramki* (ang. line/gate delay fault) przyjmuje się, że defekt powoduje zwiększenie opóźnienia sygnału przy przejściu pojedynczego elementu układu: linii lub bramki [47, 70] (rys. 2.1). W okresie gdy model ten został zaproponowany, powszechnie był już stosowany model uszkodzenia przejścia, opisany powyżej.



Rysunek 2.2: Ilustracja uszkodzeń typu opóźnienie ścieżki

Dlatego też model uszkodzenia typu opóźnienie linii/bramki nie pojawia się szerzej w znanej literaturze przedmiotu.

2.3.4 Uszkodzenia typu opóźnienie ścieżki

W rzeczywistych warunkach produkcyjnych na opóźnienie wyjścia układu względem wejścia wpływają parametry środowiska produkcyjnego takie, jak: temperatura procesu fabrykacji, skład chemiczny materiału lub atmosfery pomieszczeń produkcyjnych (ang. clean room). W wyniku zmian tych czynników, zmiany wymiarów połączeń lub tranzystorów, czy też parametrów RLC występują równomiernie na całej powierzchni układu. Zmiana opóźnienia propagacji zdarzenia przez pojedynczy element jest najczęściej tak niewielka, że zastosowanie testów dla uszkodzeń typu opóźnienie linii/bramki nie ujawni wystąpienia uszkodzenia. Zmiany opóźnień mogą zostać ujawnione dopiero wtedy, gdy wystąpi ich kumulacja, jeśli ścieżka propagacji sygnału obejmuje kilka bramek i połączeń (rys. 2.2).

Na tej podstawie został opracowany model *uszkodzenia typu opóźnienie ścieżki* (ang. path delay fault) [81].

Uszkodzenie typu opóźnienie ścieżki nie jest tak popularne w zastosowaniach przemysłowych jak uszkodzenia przejścia, mimo iż klasa uszkodzeń typu opóźnienie ścieżki jest szersza od klasy uszkodzeń przejścia. Brak popularności jest spowodowany trudnością testowania. Problem polega na tym, że w ogólnym przypadku układu liczba ścieżek rośnie wykładniczo z liczbą bramek, przez co generacja testów, jak i samo testowanie wszystkich uszkodzeń (100% pokrycia zbioru testów) jest zbyt kosztowne, a w wielu przypadkach niemożliwe. Dodatkowo, wiele spośród uszkodzeń typu opóźnienie które wykrywają testy dla uszkodzeń typu opóźnienie ścieżki, jest wykrywane testami dla innych modeli uszkodzeń typu opóźnienie. Innymi słowy, występowanie uszkodzeń typu opóźnienie ścieżki nie było dotychczas dostatecznie powszechne, aby ich testowanie stało się uzasadnione z ekonomicznego punktu widzenia. Zaproponowano częściowe rozwiązanie problemu nadmiernej liczebności zbioru ścieżek do testowania poprzez testowanie tylko stosunkowo niewielkiego podzbioru ścieżek krytycznych [23, 44, 49, 61, 71, 72, 79].

Uszkodzenia silnie testowalne

W modelu uszkodzenia typu opóźnienie ścieżki istotne jest, aby opóźnienie propagacji zdarzenia wzdłuż testowanej ścieżki było niezależne od innych opóźnień w układzie, bez względu na ich wielkość i charakter. Uszkodzenia typu opóźnienie ścieżki, które mogą być testowane niezależnie od innych opóźnień w układzie nazwano *silnie testowalnymi uszkodzeniami typu opóźnienie ścieżki* (ang. robust testable path delay faults) [81]. Uszkodzenia takie będziemy dalej nazywać krótko *uszkodzeniami silnie testowalnymi*. Formalna definicja silnej testowalności jest następująca.

Definicja 2.3 *Niech p oznacza ścieżkę w układzie. Ścieżka p jest silnie testowalna, jeżeli istnieje taki wzorzec testowy, że opóźnienie propagacji zdarzenia generowanego przez wzorzec wzdłuż ścieżki p jest niezależne od innych opóźnień w układzie.*

Tablica 2.1: Warunki silnej testowalności ścieżki [47]

Zdarzenie na wejściu głównym	Dopuszczalne wartości na wejściach bocznych	
	AND NAND	OR NOR
Narost ⎍	⎍/1	0
Opadanie ⎎	1	⎎/0

Lin i Reddy [47] wykazali, że warunki silnej testowalności ścieżki podane w [81] są spełnione, jeżeli wartości lub zdarzenia na wejściach bocznych bramek leżących na ścieżce są takie, jak podano w tablicy 2.1. O ścieżce, która jest silnie testowalna mówimy także, że ma *własność silnej testowalności*. Analogicznie, mówimy że układ, w którym każda ścieżka jest silnie testowalna, *jest silnie testowalny* lub też, że *ma własność silnej testowalności*. Wzorzec testowy, który testuje uszkodzenie typu opóźnienie ścieżki wzdłuż ścieżki p jest nazywany *silnym testem* ścieżki p.

Uszkodzenia inne niż silnie testowalne

Jakkolwiek przedstawiona problematyka pozostaje poza zakresem naszych badań i nie będzie szerzej rozwijana, dla kompletności przeglądu przedstawiamy skrócony przegląd dodatkowych zagadnień związanych z uszkodzeniami typu opóźnienia ścieżek. Uporządkowanie i klasyfikacja zagadnień została dokonana głównie na podstawie [38].

Uszkodzenia nie-silnie testowalne

W układach logicznych stosowanych w przemyśle większość ścieżek nie spełnia warunków silnej testowalności. Objawia się to tym, że dla każdego wzorca testowego zdarzenia na wyjściach bocznych są inne niż podano w tab. 2.1. Uszkodzenie typu opóźnienie ścieżki występujące na takiej ścieżce nazwano *nie-silnie testowalnym uszkodzeniem typu opóźnienie*

ścieżki (ang. non-robustly testable path delay fault) [38]. W niektórych przypadkach możliwe jest testowanie układu w którym występują takie ścieżki, aczkolwiek w przypadku wystąpienia błędu przy testowaniu może być niemożliwe wskazanie ścieżki, na której wystąpiło uszkodzenie.

Uszkodzenia zależne

Analizy przedstawione w [40] wskazują, że w dowolnym układzie logicznym, pośród wszystkich uszkodzeń typu opóźnienie ścieżki można wyróżnić dwa zbiory. Niech zbiory te będą oznaczone R i R_D. Do zbioru R zaliczono te uszkodzenia, których wystąpienie może skutkować zwiększeniem opóźnienia układu. Uszkodzenia z tego zbioru muszą być testowane. Do zbioru R należą również uszkodzenia silnie testowalne. Uszkodzenia ze zbioru R_D nie mają wpływu na opóźnienie układu, jeżeli równocześnie z nimi nie wystąpią uszkodzenia ze zbioru R. Stąd każde uszkodzenie ze zbioru R_D nazywa się *zależnym uszkodzeniem typu opóźnienie ścieżki* (ang. robust dependent path delay fault) [40, 38, 82].

Uszkodzenia proste i weryfikowalność opóźnienia

W [32] wprowadzono pojęcia *prostego uszkodzenia typu opóźnienie ścieżki* (ang. primitive path delay fault) i *weryfikowalności opóźnienia* (ang. delay verifiability). Model tego uszkodzenia jest niemal dokładnie taki sam jak model zależnego uszkodzenia typu opóźnienie ścieżki. Opóźnienie układu jest *weryfikowalne*, jeżeli prawidłowe działanie układu przy pewnej wartości częstotliwości zegara lub wartości niższej może być zagwarantowane poprzez przeprowadzenie testów z pewnego zbioru. Uszkodzenia typu opóźnienie ścieżek, które są testowane przez testy z w/w zbioru, są nazywane *uszkodzeniami prostymi*. Wykazano także, że układ silnie testowalny jest układem o weryfikowalnym opóźnieniu.

2.3.5 Uszkodzenia typu opóźnienie segmentu i opóźnienie propagacji

Uszkodzenia typu opóźnienie segmentu

Uszkodzenie typu opóźnienie segmentu (ang. segment delay fault) [25, 38] jest odmianą uszkodzenia typu opóźnienie ścieżki. Niestety, model ten wprowadza nadmierną liczbę ścieżek wymagających testowania, co uniemożliwiało testowanie uszkodzeń występujących we wszystkich ścieżkach. Z drugiej strony, model uszkodzeń typu opóźnienie linii/bramki (uszkodzenie przejścia), mimo niewątpliwej zalety w postaci liniowej zależności liczby testów od liczby bramek, nie pozwalał na wykrycie wielu uszkodzeń mających wpływ na działanie układu, a wykrywanych przez testy dla uszkodzeń typu opóźnienie ścieżki. Stąd uszkodzenie typu opóźnienie segmentu stanowi próbę znalezienia zadowalającego kompromisu pomiędzy ograniczeniami modelu uszkodzenia typu opóźnienie linii/bramki a liczbą uszkodzeń typu opóźnienie ścieżki.

Model ten opiera się na następującym założeniu [25].

Założenie 2.1 *Uszkodzenie typu opóźnienie segmentu powoduje wystąpienie uszkodzenia typu opóźnienie ścieżki na wszystkich ścieżkach przechodzących przez segment.*

Długość segmentów podlegających analizie dobiera się na podstawie statystycznej analizy defektów produkcyjnych występujących w technologii, w której będzie wykonany układ. Ze wzrostem długości segmentu wzrasta liczba segmentów podlegających analizie. Jeżeli długość segmentu jest niewielka, możliwe jest, że opóźnienia rozproszone na dłuższych segmentach i całych ścieżkach nie zostaną wykryte.

Uszkodzenia typu opóźnienie propagacji

Model *uszkodzenia typu opóźnienie propagacji* (ang. propagation delay fault) [48] jest podobny do modelu uszkodzenia typu opóźnienie segmentu. W modelu tym segment rozpoczyna się w założonym punkcie uszkodzenia (linii lub bramce), a kończy na wyjściu pierwotnym. Przyjmuje się, że opóźnienie skupione w punkcie uszkodzenia oraz opóźnienie roz-

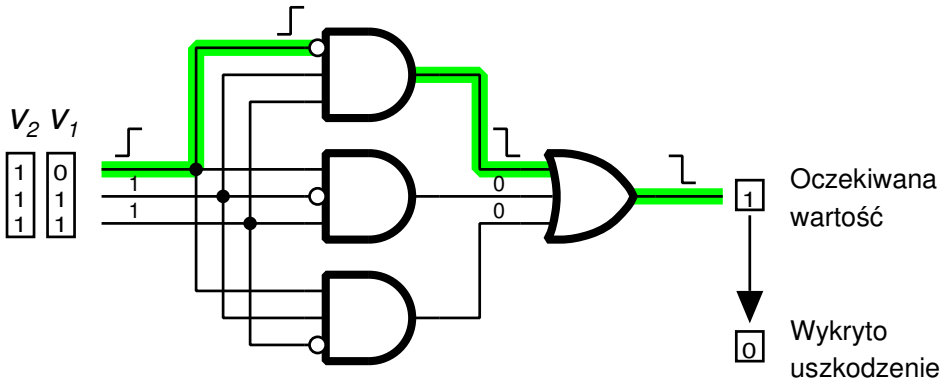
proszone wzdłuż ścieżki propagacji są dostatecznie duże, aby uszkodzenie zostało wykryte. Model ten nakłada nieco ostrzejsze wymagania na warunki propagacji zdarzeń wzdłuż ścieżek propagacji do wyjść układu niż model uszkodzenia typu opóźnienie linii/bramki.

2.3.6 Uwagi końcowe

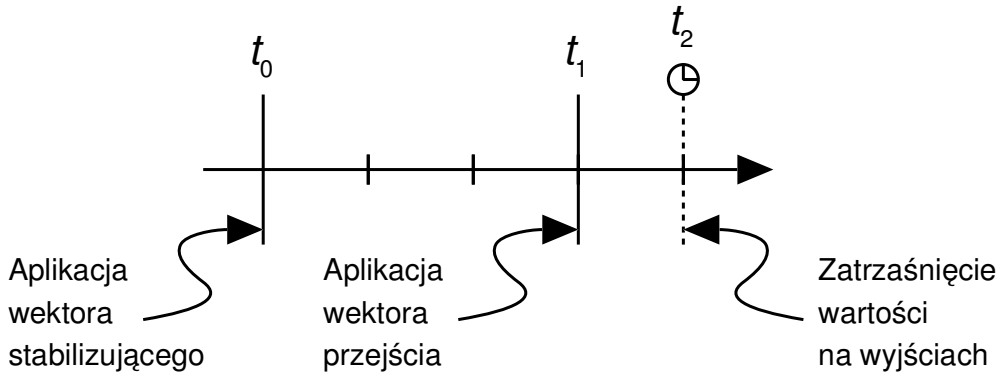
Rozwój metod i algorytmów analizy układów, a także większa wydajność komputerów i urządzeń testujących, pozwalają na testowanie układów z użyciem modelu uszkodzenia typu opóźnienie ścieżki. Przy okazji badań prowadzonych w ramach niniejszej rozprawy zaobserwowano również, że istnieją klasy układów charakteryzujące się stosunkowo małą liczbą ścieżek. Z drugiej strony, uszkodzenia typu opóźnienie ścieżki są coraz poważniejszym problemem producentów układów. Testowanie uszkodzeń typu opóźnienie ścieżki z użyciem metod np. dla uszkodzeń przejścia prowadzi do maskowania coraz większej liczby defektów, co z kolei prowadzi do nieakceptowalnych ilości układów działających nieprawidłowo u klientów. Silna testowalność natomiast daje możliwość takiego przetestowania układu, że poziom zaufania co do sprawnego działania wyprodukowanego układu będzie istotnie wyższy. Dlatego też silnie testowalne uszkodzenia typu opóźnienie ścieżki są podstawowym modelem uszkodzeń rozważanych w niniejszej pracy.

2.4 Podstawy testowania uszkodzeń typu opóźnienie

Niezależnie od przyjętego modelu uszkodzeń typu opóźnienie, testowanie każdego z tych uszkodzeń sprowadza się do pobudzenia układu sekwencją pobudzeń. Schemat testowania uszkodzeń typu opóźnienia przedstawiono na rys. 2.3 i 2.4. W chwili t_0 układ jest pobudzany wektorem v_1, z racji swojej funkcji zwanym *wektorem stabilizującym* [51, 88]. W chwili t_1, po upływie relatywnie długiego czasu $t = t_1 - t_0$, podawany jest drugi wektor binarny v_2, zwany *wektorem przejścia* [51, 88]. Następnie, po upływie czasu T równego długości cyklu zegara (w chwili $t_2 = t_1 + T$) następuje obserwacja (zatrzaśnięcie) wartości wyjść układu. Na podstawie porównania wartości zatrzaśniętej z wartością wyznaczoną na podstawie analizy

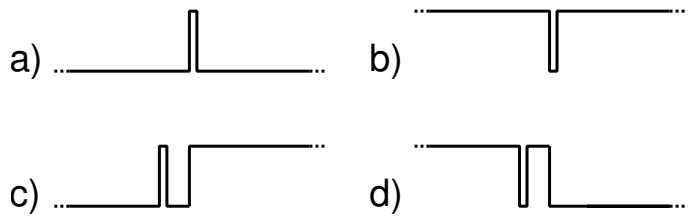


Rysunek 2.3: Testowanie uszkodzeń typu opóźnienie

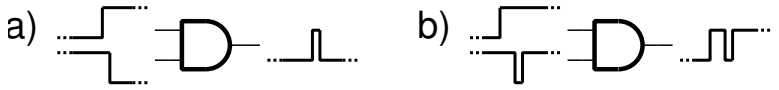


Rysunek 2.4: Przebieg testowania uszkodzeń typu opóźnienie na osi czasu

funkcji logicznych realizowanych przez układ, możemy ocenić czy propagacja zdarzenia do wyjścia układu nastąpiła czy też nie. Jeżeli wartość zatrzaśnięta na wyjściu jest przeciwna do oczekiwanej, oznacza to że czas propagacji zdarzenia był dłuższy od okresu przez nas założonego i wystąpiło uszkodzenie typu opóźnienie.



Rysunek 2.5: Zjawisko hazardu: a), b) hazard statyczny, c), d) hazard dynamiczny



Rysunek 2.6: Przykład powstawania hazardu: a) statycznego, b) dynamicznego

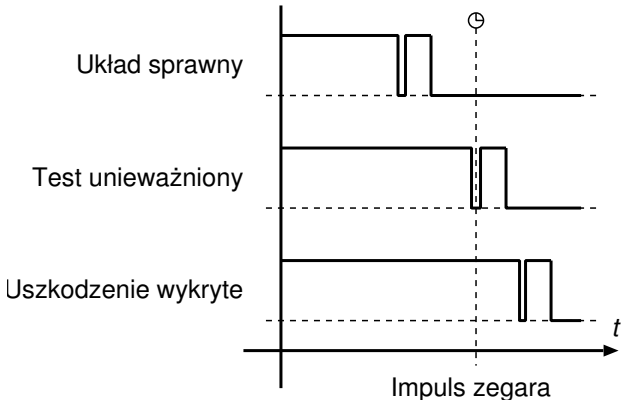
2.5 Uszkodzenia typu opóźnienie a zjawisko hazardu

2.5.1 Zjawisko hazardu

Zjawisko hazardu jest niewłaściwym zachowaniem układu wtedy, gdy wartości wyjść układu zależą od względnych opóźnień zdarzeń na różnych wejściach bramek. Eliminacja i/lub unikanie hazardu stanowi często spotykany poważny problem w syntezie układów logicznych. Wyróżnia się dwa rodzaje hazardów [39]:

- *hazard statyczny*, kiedy na wyjściu bramki pojawia się jeden lub więcej impulsów, przy czym wartość przed i po impulsach nie ulega zmianie (rys. 2.5 a,b);
- *hazard dynamiczny*, kiedy na wyjściu bramki pojawia się jeden lub więcej impulsów, przy czym wartość przed i po impulsach ulega zmianie (rys. 2.5 c,d).

Hazard statyczny może wystąpić wtedy, gdy przykładowo na jednym z wejść bramki AND poza testowaną ścieżką występuje przejście $1 \rightarrow 0$, zaś na drugim $0 \rightarrow 1$. W zależności od czasów w których przychodzą zdarzenia, na wyjściu bramki może pojawić się krótki zbędny impuls albo stała wartość 0. Hazard dynamiczny jest skutkiem nałożenia hazardu statycznego na pojedyncze zdarzenie. Przykład powstawania hazardu statycznego pokazano na rys. 2.6a), zaś dynamicznego na rys. 2.6b).



Rysunek 2.7: Wpływ zjawiska hazardu na testowanie układu

2.5.2 Zjawisko hazardu a silna testowalność uszkodzeń typu opóźnienia ścieżek

Przy testowaniu uszkodzeń typu opóźnienie ścieżki wystąpienie hazardu może prowadzić do unieważnienia silnych testów [39]. Warunki silnej testowalności podane w [47] nie chronią w żaden sposób przed wystąpieniem hazardu statycznego poza testowaną ścieżką. Określają one bowiem warunki silnej testowalności jedynie dla bramek leżących na badanej ścieżce, pomijając inne bramki w układzie. Jeżeli wystąpi hazard statyczny, to może on dotrzeć do ścieżki testowanej i doprowadzić do powstania hazardu dynamicznego. Skutkiem tego może być pozytywny wynik testu układu z uszkodzeniem, co pokazano na rys. 2.7.

2.5.3 Ochrona przed zjawiskiem hazardu

Istnieją różne propozycje ochrony przed wystąpieniem zjawiska hazardu [39]. Przykładami klas układów spełniających dostateczne warunki, aby nie powstał hazard statyczny, są: układ pozbawiony rozgałęzionych a następnie zbiegających się ścieżek (ang. reconvergent fanout) oraz układ bezinwerterowy (ang. inverter-free). Dla celów naszych analiz przyjmiemy także dodatkowe ograniczenie, że zdarzenie w wejściowym wzorcu testowym wystąpi na tylko jednej pozycji.

W literaturze pojawiają się także inne propozycje eliminacji zjawiska hazardu podczas

testowania uszkodzeń typu opóźnienie ścieżki, takie jak np. propozycja obostrzenia warunków silnej testowalności [39]. W niniejszej rozprawie są one jednak poza zakresem naszych zainteresowań.

Rozdział 3

Funkcje symetryczne i progowe oraz ich implementacje

3.1 Wstęp

Jak opisano w rozdziale 2, testowanie uszkodzeń typu opóźnienie stanowi poważny problem producentów układów scalonych VLSI, co uzasadnia duże zainteresowanie prezentowaną w niniejszej pracy tematyką. Generowanie testów z wykorzystaniem metod ogólnych jest na ogół kosztowne i nieefektywne, a nierzadko po prostu niemożliwe. Jedną z alternatyw pozwalających na znaczącą redukcję złożoności generowania testów jest synteza układów łatwo testowalnych, tj. uwzględniających możliwość wykrywania ich uszkodzeń wewnętrznych. Istnieje szereg szczególnych klas układów cyfrowych z jednej strony mających wiele zastosowań praktycznych a z drugiej strony podatnych na modyfikacje ułatwiające testowanie. Do takich układów należą niektóre układy realizujące funkcje symetryczne i progowe, których liczne zastosowania podaliśmy w rozdz. 1.

W poniższym rozdziale zostaną zdefiniowane funkcje symetryczne i progowe oraz zostanie dokonany przegląd implementacji układów realizujących funkcje symetryczne i progowe mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. Następnie zostaną sformułowane otwarte problemy naukowe dotyczące testowania uszkodzeń typu

opóźnienie w tych układach, których rozwiązania zostaną podane w rozdziałach 4, 5 i 6.

3.2 Własności funkcji symetrycznych i progowych

3.2.1 Funkcje symetryczne

Definicja 3.1 *Funkcja przełączająca $S^n(X)$ określona na n-bitowym wektorze binarnym $X = (x_1, x_2, \ldots, x_n)$ jest nazywana* funkcją symetryczną *(ang. symmetric function), jeżeli dowolna permutacja zmiennych wejściowych nie powoduje zmiany wartości funkcji.*

W szczególnych przypadkach funkcja symetryczna jest też zwana funkcją *całkowicie symetryczną* (ang. totally symmetric), jeżeli zbiór zmiennych których permutacja nie zmienia wartości funkcji zawiera wszystkie argumenty funkcji, lub też *częściowo symetryczną* (ang. partially symmetric), jeżeli zawiera tylko pewien ich podzbiór. Zmienne, których permutacja nie zmienia wartości funkcji nazywane są *zmiennymi symetrii* (ang. variables of symmetry).

Przykład 3.1 *Funkcja $f(x_1, x_2, x_3) = x_1x_2 + x_3$ jest funkcją częściowo symetryczną, zaś jej zmienne symetrii to x_1 i x_2. Natomiast funkcja $g(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1x_3$ jest funkcją całkowicie symetryczną.*

W niniejszej pracy będziemy zajmować się tylko funkcjami całkowicie symetrycznymi, zwanymi dalej po prostu funkcjami symetrycznymi. Z analizy def. 3.1 wynika, że wartość funkcji całkowicie symetrycznej zależy tylko od liczby jedynek i zer w wektorze wejściowym. Zatem funkcję symetryczną można zapisać następująco.

Niech $w(X)$ będzie *wagą Hamminga* n-bitowego wektora X (liczbą jedynek w wektorze X).

Definicja 3.2 *Niech będzie dany n-bitowy wektor binarny X i pewna liczba całkowita m, $1 \leqslant m \leqslant n$.* Elementarna funkcja symetryczna S_m^n *jest opisana liczbą m, jeżeli*

$$S_m^n(X) = 1 \Longleftrightarrow w(X) = m. \tag{3.1}$$

Przykład 3.2 *Funkcja XOR* $f(x_1, x_2) = x_1 \oplus x_2$ *jest funkcją symetryczną* S_1^2.

Definicja 3.3 *Niech będzie dany n-bitowy wektor binarny X i pewien zbiór A różnych liczb całkowitych nieujemnych takich, że* $a_i \in A$ *i* $0 \leqslant a_i \leqslant n$. Złożona funkcja symetryczna $S_A^n(X)$ *jest opisana zbiorem A, jeżeli*

$$S_A^n(X) = 1 \Longleftrightarrow w(X) \in A. \tag{3.2}$$

Przykład 3.3 *Funkcja OR* $f(x_1, x_2) = x_1 + x_2$ *jest funkcją symetryczną* $S_{\{1,2\}}^2$.

Definicja 3.4 *Jeżeli zbiór A zawiera wyłącznie kolejne liczby całkowite, wtedy funkcja symetryczna jest nazywana* kolejnościową funkcją symetryczną *(ang.* consecutive symmetric function*) [36]; zapisujemy ją z użyciem notacji* $S_{a_l \div a_r}^n$, *gdzie* $0 \leqslant a_l \leqslant a_r \leqslant n$ *[31].*

Dla $a_r = n$ występuje szczególny przypadek kolejnościowej funkcji symetrycznej, zwany *funkcją progową*, którą poniżej rozpatrujemy oddzielnie.

3.2.2 Funkcje progowe

Definicja 3.5 *Niech X będzie n-bitowym wektorem binarnym, a m pewną liczbą całkowitą,* $0 \leqslant m \leqslant n$. *Funkcja przełączająca* $T_m^n(X)$ *jest nazywana* funkcją progową, *jeżeli*

$$T_m^n(X) = 1 \Longleftrightarrow w(X) \geqslant m. \tag{3.3}$$

Definicja 3.6 Wielowyjściowy układ progowy *jest to układ, który implementuje wszystkie n funkcji progowych dla każdego* $1 \leqslant m \leqslant n$.

W dalszej części pracy, wielowyjściowy układ progowy będzie krótko nazywany *układem* T^n.

3.2.3 Związki pomiędzy funkcjami symetrycznymi a funkcjami progowymi

1. Funkcja progowa jest sumą elementarnych funkcji symetrycznych:

$$T_m^n = S_{m \div n}^n = S_m^n + S_{m+1}^n + \ldots + S_n^n. \tag{3.4}$$

2. Elementarna funkcja symetryczna może być zapisana jako funkcja funkcji progowych:

$$S_m^n = T_m^n \cdot \overline{T_{m+1}^n}. \tag{3.5}$$

3. Funkcja całkowicie symetryczna może być zapisana jako funkcja funkcji progowych:

$$S_A^n = \sum_{i \in A} T_i^n \cdot \overline{T_{i+1}^n}. \tag{3.6}$$

4. Kolejnościowa funkcja symetryczna również może być zapisana jako funkcja funkcji progowych:

$$S_{a_l \div a_r}^n = T_{a_l}^n \cdot \overline{T_{a_r+1}^n}. \tag{3.7}$$

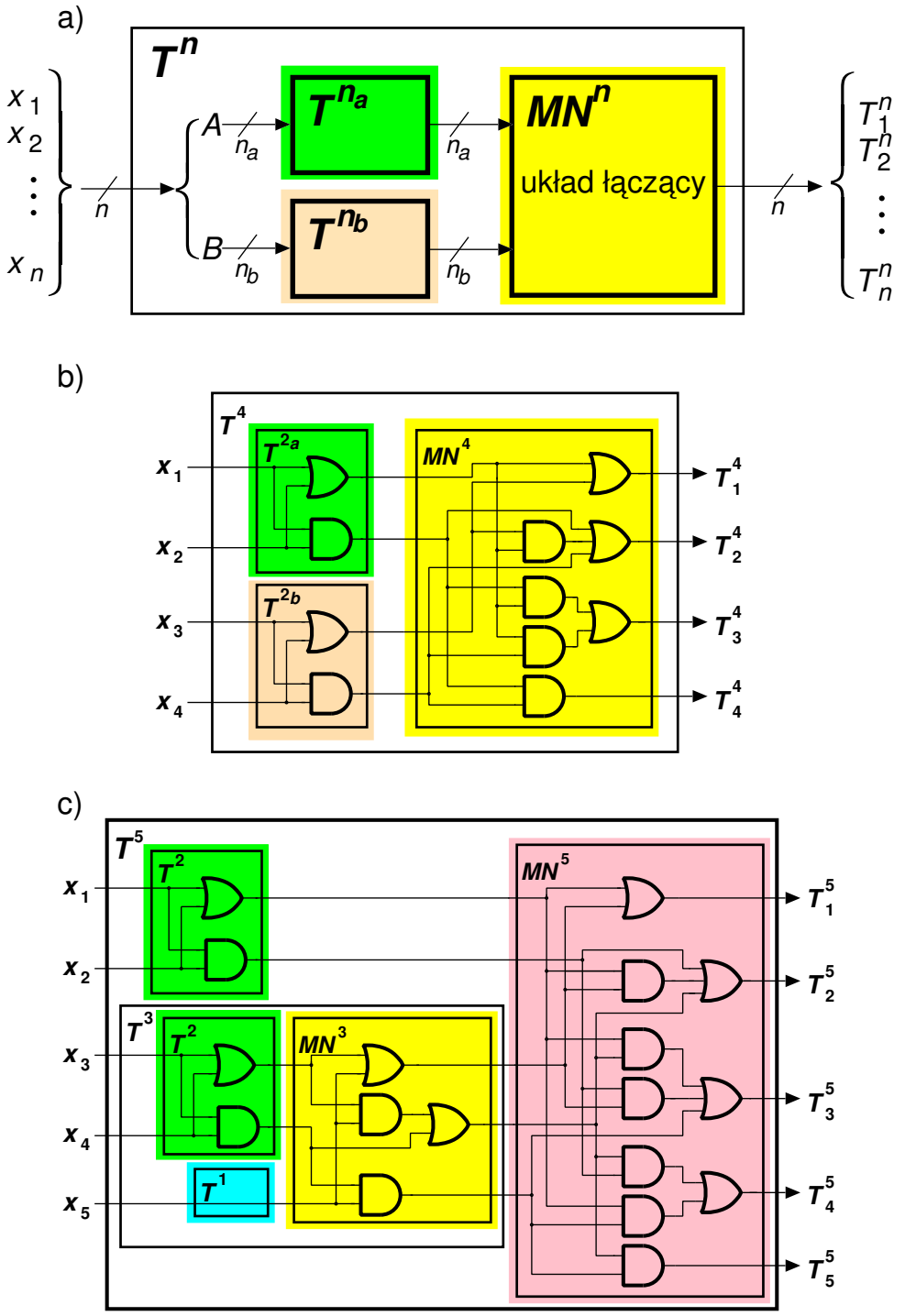
Układ implementujący elementarną funkcję symetryczną postaci S_m^n (3.5) lub kolejnościową funkcję symetryczną (3.7) można otrzymać z wielowyjściowego układu progowego stosując prosty układ NOT-AND. Implementację funkcji symetrycznej postaci S_A^n (3.6) można uzyskać dodając bramkę OR, uzyskując w ten sposób układ NOT-AND-OR [66, 73].

3.3 Układy o rozłącznych podukładach

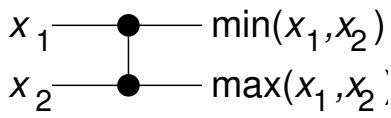
Wielowyjściowy układ progowy T^n można zrealizować jako dwu- lub wielopoziomowy. Układ T^n można także zrealizować jako układ o rozłącznych podukładach, w którym podukłady nie mają wspólnych bramek.

Definicja 3.7 *Wielowyjściowy układ progowy jest zrealizowany jako* układ o rozłącznych podukładach*, jeżeli jest złożony z dwóch układów podrzędnych T^{n_a}, T^{n_b} nie mających wspólnych bramek i takich, że $n_a + n_b = n$ oraz układu łączącego MN^n. Funkcje wyjść układu MN^n przekształcają wyjścia układów T^{n_a} oraz T^{n_b} w wynik końcowy.*

Rysunek 3.1a przedstawia schemat ogólny układu T^n złożonego z dwóch rozłącznych podukładów. Z kolei rysunek 3.1b przedstawia przykład układu czterowejściowego T^4 złożonego z dwóch rozłącznych układów dwuwejściowych T^{2_a} oraz T^{2_b} i dwupoziomowego układu łączącego MN^4. Idea układu o rozłącznych podukładach może zostać zastosowana



Rysunek 3.1: Układ T^n zrealizowany jako złożenie rozłącznych podukładów: a) schemat ogólny; b) przykład układu T^4; c) przykład układu całkowicie rozgałęzionego T^5



Rysunek 3.2: Ogólny schemat komparatora dwuwejściowego

rekurencyjnie dla każdego z podukładów. Takie postępowanie prowadzi do otrzymania układu T^n całkowicie rozgałęzionego [21], którego przykład dla $n = 5$ przedstawia rys. 3.1c.

Definicja 3.8 *Wielowyjściowy układ progowy o rozłącznych podukładach jest nazywany* całkowicie rozgałęzionym*, jeżeli spełnia następujące warunki:*

1. *ma strukturę rekurencyjną;*
2. *każda para podukładów T^{n_a} i T^{n_b} ma odpowiednio $n_a = [\frac{n}{2}]$ i $n_b = n - n_a$ linii.*

Układ T^2 może zostać przedstawiony jako złożenie dwóch układów T^1 (pojedynczych linii) oraz układu łączącego MN^2. Rysunek 3.1c zawiera przykład układu T^3 złożonego z układu T^2, układu T^1 (pojedynczej linii) oraz układu łączącego MN^3.

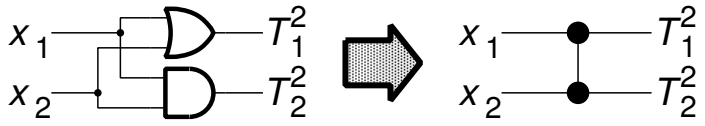
3.4 Sieci sortujące jako wielowyjściowe układy progowe

3.4.1 Istota działania sieci sortujących

Sieci sortujące są sprzętową implementacją niektórych programowych algorytmów sortowania, zbudowanych z powtarzalnych elementów – komparatorów (rys. 3.2). W ogólnym przypadku, sieć sortująca sortuje dowolne elementy porównywalne i jest zbudowana wyłącznie z komparatorów, elementów o złożoności dostosowanej do złożoności sortowanych elementów. Obszerny przegląd algorytmów sortowania oraz sieci sortujących został zamieszczony w książkach [35] oraz [16], a także w [64, 66, 84]. Jednakże tutaj interesuje

Tablica 3.1: Funkcje logiczne realizowane przez komparator binarny

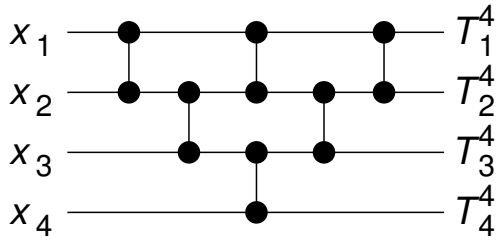
x_1	x_2	T_1^2	T_2^2
0	0	0	0
1	0	1	0
0	1	1	0
1	1	1	1



Rysunek 3.3: Komparator binarny jako układ T^2

nas wyłącznie najbardziej elementarny szczególny przypadek sieci sortującej, w którym elementami sortowanymi są bity. Rysunek 3.3 i tab. 3.1 przedstawiają schemat logiczny i tabelę prawdy dwuwejściowego komparatora takiej sieci, który de facto realizuje dwie funkcje progowe T_1^2 i T_2^2.

Przykładem jednej z najprostszych sieci sortujących jest sieć bąbelkowa [35] (rys. 3.4). W naszym przypadku istotą sieci sortującej jest własność „sortowania" zer i jedynek w wektorze binarnym. Wskutek działania sieci sortującej, jedynki zostają przesunięte na jedną stronę wektora, a zera – na drugą stronę, przy czym waga Hamminga wektora nie ulega zmianie. Na wyjściu sieci sortującej otrzymuje się ciąg posortowany. Na przykład, wektor czterobitowy postaci 0100 zostanie przekształcony w wektor 1000, zaś wektor postaci 1011 zostanie przekształcony w wektor 1110. Widać tutaj, że na wyjściu sieci sortującej jedynka na pozycji i pojawia się wtedy, gdy liczba jedynek wektora wejściowego (waga Hamminga) jest większa lub równa i. Zatem można powiedzieć, że n-wejściowa sieć sortująca jest równoważna układowi T^n, co wykazano w [41].



Rysunek 3.4: Sieć sortująca bąbelkowa jako układ T^4

3.4.2 Sieci sortujące monolityczne i całkowicie rozgałęzione

Spośród sieci sortujących można wydzielić klasę sieci o strukturze całkowicie rozgałęzionej, w których można wyodrębnić podukłady i sieci łączące. W dalszej części pracy sieci takie będą nazywane *sieciami całkowicie rozgałęzionymi*. Pozostałe sieci można zaliczyć do albo klasy *sieci monolitycznych* (jak np. wspomnianą powyżej sieć bąbelkową), albo do klasy *sieci mieszanych*.

3.4.3 Sieci sortujące Batchera

Sieci sortujące zaproponowane przez Batchera w [5] należą do klasy sieci o strukturze całkowicie rozgałęzionej. W pracy [5] Batcher zaproponował dwie klasy sieci sortujących:

1. sieci bitoniczne (ang. bitonic sorting networks) oraz
2. sieci nieparzysto-parzyste (ang. odd-even sorting networks).

Asymptotyczna złożoność obydwu sieci jest taka sama i wynosi $O(n \log^2 n)$, przy czym sieć nieparzysto-parzysta zawiera nieco mniej komparatorów. Dla niektórych wersji sieci przedstawiono dowody optymalności [4, 28, 58]. Złożoność sieci Batchera jest najmniejsza ze wszystkich znanych klas sieci sortujących o regularnej strukturze. Naturalnie, sieć Batchera należy również do najmniej złożonych spośród znanych wielowyjściowych układów progowych. Sieci sortujące o minimalnie mniejszych rozmiarach niż nieparzysto-parzysta sieć Batchera uzyskano jedynie stosując metody niedeterministyczne [20, 27, 92]. Niestety,

ich istotną wadą jest bardzo nieregularna struktura. Testowalność uszkodzeń typu opóźnienie ścieżki w sieciach nieparzysto-parzystych Batchera stanowi jeden z celów badań niniejszej pracy. Dokładny opis i analiza tej sieci zostaną podane w rozdziale 5. Inne sieci sortujące są poza obszarem zainteresowań niniejszej pracy i ich tematyka nie będzie szerzej rozwijana.

3.5 Przegląd implementacji funkcji symetrycznych

W literaturze przedstawiono wiele różnych metod implementacji funkcji symetrycznych [14, 19, 31, 73, 80]. Do najprostszych metod należy implementacja dwupoziomowa ([NOT]-AND-OR), która, niestety, staje się nadmiernie złożona i praktycznie nierealizowalna jako dwupoziomowa w technologii CMOS już dla $n \geqslant 5$. Ostatnio [73] zaproponowano implementację wielopoziomową bazującą na podanej w rozdziale 3.2.2 równoważności funkcji progowych i funkcji symetrycznych. Wiele uwagi poświęcono realizacjom mającym własność silnej testowalności uszkodzeń typu opóźnienie ścieżki [14, 31, 62, 73, 80]. W niniejszym rozdziale podsumujemy obecny stan badań w tym zakresie.

3.5.1 Implementacja bezpośrednia k-poziomowa, $2 \leqslant k \leqslant 4$

Funkcje symetryczne można implementować z użyciem układów dwupoziomowych [19] oraz trzy- i czteropoziomowych [14, 31]. W pracy [19] wskazano algorytmy generowania minimalnych pokryć dla implementacji dwupoziomowych kolejnościowych funkcji symetrycznych. Na podstawie analizy siatek Karnaugh'a sum iloczynów zdefiniowanych w [19] Ke i Menon w [31] pokazali, iż jedynie niektóre pokrycia dwupoziomowe generujące funkcje symetryczne mają własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. W szczególności, silnie testowalna realizacja dwupoziomowa istnieje tylko dla kolejnościowych funkcji symetrycznych postaci

$$S^n_{a_l \div a_r}, \text{ gdzie } a_l = 0 \text{ lub } a_r = n \text{ lub } a_l + a_r = n. \tag{3.8}$$

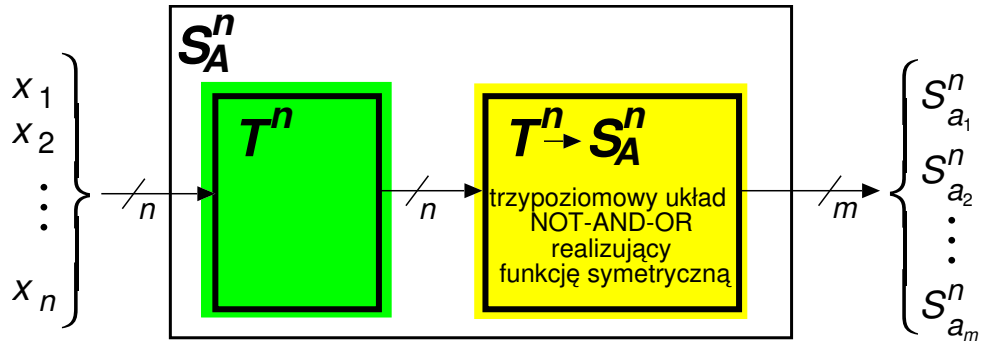
Następnie w [31] wykazano, że dla funkcji symetrycznych nie spełniających powyższych warunków istnieją trzy- i czteropoziomowe implementacje mające własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. W pracy [14] podano algorytmy generowania implementacji trzy- i czteropoziomowych, których istota sprowadza się do spostrzeżenia, iż dowolna kolejnościowa funkcja symetryczna postaci $S^n_{a_l \div a_r}$ może zostać zrealizowana jako złożenie dwóch silnie testowalnych dwupoziomowych funkcji symetrycznych o postaci podanej w (3.8).

Dla przykładu, rozważmy układ realizujący funkcję $S^n_{a_l \div a_r} = S^n_{a_l \div n} \cdot \overline{S^n_{a_r+1 \div n}}$. Dla silnie testowalnych podukładów realizujących funkcje $S^n_{a_l \div n}$ i $S^n_{a_r \div n}$, układ realizujący funkcję $S^n_{a_l \div a_r}$ również jest silnie testowalny [14]. W tym przypadku podukłady realizujące funkcje $S^n_{a_l \div n}$, $S^n_{a_r \div n}$ są dwupoziomowe, stąd układ realizujący funkcję $S^n_{a_l \div a_r}$ jest trzypoziomowy. Z kolei wykazano, że ponieważ funkcja symetryczna dowolnej postaci może zostać zrealizowana jako złożenie funkcji kolejnościowych, np. $S^n_A = \sum_{i \in A} S^n_i$, to dla silnie testowalnych układów realizujących funkcje S^n_i układ realizujący funkcję S^n_A również ma własność silnej testowalności [14]. Ponieważ podukłady realizujące funkcje S^n_i są układami, które realizują dowolne funkcje kolejnościowe, są w ogólnym przypadku trzypoziomowe, stąd układ realizujący funkcję S^n_A jest czteropoziomowy.

3.5.2 Implementacja wielopoziomowa

Motywacja dla poszukiwania implementacji wielopoziomowej

Układy o ustalonej liczbie poziomów mają w każdym przypadku powierzchnię i liczbę ścieżek o złożoności wykładniczej. O ile nie sprawia to problemu przy niewielkich układach (do kilkukilkunastu linii), to przy powszechnych dzisiaj układach o kilkudziesięciu czy nawet kilkuset liniach implementacja takiego rozwiązania staje się niemożliwa. Dodatkowo, nawet przy kilkunastu liniach występuje konieczność syntezy pojedynczych bramek o wielu wejściach. Tymczasem technologia CMOS ogranicza liczbę wejść pojedynczej bramki do czterech. Zatem, każda wielowejściowa bramka musi być realizowana jako bramka wielopoziomowa, skutkiem czego rzeczywista liczba poziomów znacznie przekracza nominalną liczbę od dwóch do czterech poziomów. Dlatego też konieczne jest znalezienie rozwiązań wielopoziomowych, które nie wymagają wprowadzania dodatkowych poziomów w faktycznej realizacji.



Rysunek 3.5: Implementacja funkcji symetrycznej S_A^n z użyciem układu T^n

Czy istnieje implementacja wielopoziomowa?

Propozycję implementacji funkcji symetrycznej w postaci układu wielopoziomowego o strukturze pokazanej na rys. 3.5 przedstawiono w [66, 73]. Warto zauważyć, iż układ T^n może być zrealizowany dowolnie: albo jako standardowy wielowyjściowy układ progowy [66], albo jako sieć sortująca [64]. Końcowy układ realizujący funkcję symetryczną jest trzypoziomowym układem NOT-AND-OR. Warto zwrócić uwagę, iż układy od dwu- do czteropoziomowych mogą realizować tylko pojedyncze funkcje symetryczne. W układzie wielopoziomowym zbudowanym z użyciem jednego wielowyjściowego układu progowego można zrealizować dowolny zbiór funkcji symetrycznych, przy czym złożoność układu wykonującego funkcję symetryczną rośnie proporcjonalnie do liczby rozłącznych przedziałów w zbiorze A (3.2), (3.6).

Czy implementacja wielopoziomowa ma własność silnej testowalności?

Układ implementujący funkcję symetryczną zrealizowany według algorytmu minimalnych pokryć dwupoziomowych [19] — jak również układ realizujący funkcje symetryczne z użyciem wielowyjściowego układu progowego zaimplementowanego jako sieć sortująca np. Batchera — nie ma własności silnej testowalności uszkodzeń typu opóźnienie ścieżki. W pracy Rahamana i innych [73] przedstawiono pewien układ T^n, będący rozwinięciem układu z [21], który razem z układem końcowym realizującym funkcję symetryczną $T^n \rightarrow S_A^n$ ma własność silnej testowalności uszkodzeń typu opóźnienie ścieżki [62, 73]. Również układ implementujący funkcję symetryczną z użyciem układu całkowicie rozgałęzionego przedstawionego przez Edwardsa [21] i układu końcowego realizującego funkcję symetryczną $T^n \rightarrow S_A^n$ ma własność silnej testowalności uszkodzeń

typu opóźnienie ścieżki, co wykazaliśmy w [62].

3.6 Problemy badawcze silnie testowalnych implementacji funkcji symetrycznych

3.6.1 Kryteria oceny wyniku

Przewidujemy, że istnieją implementacje funkcji symetrycznych mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki, które są mniej złożone i mają mniejszą liczbę ścieżek od opisanych powyżej implementacji znanych z literatury.

Złożoność układu

Złożoność implementacji dwu-, trzy- i czteropoziomowej jest taka sama jak ogólnego przypadku funkcji dwu-, trzy- i czteropoziomowej, czyli $O(2^n)$. Złożoność implementacji wielopoziomowej w postaci z rys. 3.5 jest równa złożoności układu złożonego z wielowyjściowego układu progowego i układu końcowego realizującego funkcję symetryczną $T^n \rightarrow S^n_A$. Złożoność znanych wielowyjściowych układów progowych pozwalających na realizację funkcji symetrycznych mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki jest $O(n^2)$ [21, 62, 73], złożoność układu końcowego realizującego funkcję symetryczną $T^n \rightarrow S^n_A$ jest liniowa. Stąd złożoność całego układu zrealizowanego z użyciem wielowyjściowego układu progowego pozostaje nadal $O(n^2)$. Celem naszym będzie znalezienie układu o mniejszej złożoności, a oczywistym kandydatem jest układ zbudowany na bazie sieci sortującej Batchera.

Liczba ścieżek

Liczba ścieżek w każdym przypadku układu realizującego funkcję symetryczną ma złożoność wykładniczą [14, 31, 73, 80]. Jednakże liczba ścieżek jest istotnie większa w układach dwu-, trzy- i czteropoziomowych [14, 31] niż w układach wielopoziomowych [73].

3.6.2 Sformułowanie problemu badawczego

W nawiązaniu do przedstawionych powyżej rozważań, możemy przedstawić następujące problemy badawcze:

- Od czego zależą własności silnej testowalności uszkodzeń typu opóźnienie ścieżki układu T^n zrealizowanego jako całkowicie rozgałęziony?
- Czy istnieje implementacja układu T^n mająca własność silnej testowalności uszkodzeń typu opóźnienie ścieżki, o złożoności mniejszej niż $O(n^2)$?

W rozdziałach 4, 5 i 6 przedstawimy próbę udzielenia odpowiedzi na postawione powyżej pytania.

Rozdział 4

Testowalność uszkodzeń typu opóźnienie w całkowicie rozgałęzionych wielowyjściowych układach progowych

4.1 Wstęp

W tym rozdziale zostaną zaprezentowane ogólne własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w układach T^n zrealizowanych jako całkowicie rozgałęzione. Układ całkowicie rozgałęziony jest rekurencyjnym rozwinięciem układu o rozłącznych podukładach. Poniżej wykazano, że własność silnej testowalności uszkodzeń typu opóźnienie ścieżki układu całkowicie rozgałęzionego zależy jedynie od własności sieci łączących [62].

4.2 Testowalność sieci łączącej

Wielowyjściowy układ progowy T^n może zostać zrealizowany jako układ o rozłącznych podukładach i wtedy składa się on z dwóch wielowyjściowych układów progowych T^{n_a} i T^{n_b} (każdy o liczbie linii mniejszej niż liczba linii układu bazowego i takich że $n = n_a + n_b$) oraz sieci łączą-

cej MN^n (rys. 3.1a). Wartości na wyjściach każdego podukładu T^{n_a}, T^{n_b} mogą być ustawiane niezależnie od siebie, gdyż podukłady te nie mają wspólnych wejść ani też nie współdzielą żadnych wewnętrznych linii. Dodatkowo można zauważyć, że zdarzenie podane na wejście jednego z podukładów pojawi się wyłącznie na wyjściach tego podukładu. Aby zdarzenie pojawiło się na wyjściu układu T^n, musi zostać propagowane jeszcze przez sieć łączącą MN^n, czyli przez jedną ze ścieżek w sieci MN^n. Aby test zawierający dane zdarzenie był silnym testem, muszą być spełnione warunki silnej testowalności w podukładzie przez który jest propagowane zdarzenie oraz w sieci łączącej MN^n. Ponieważ jedynymi wejściami sieci MN^n są wyjścia podukładów T^{n_a} i T^{n_b}, powyższe rozumowanie wskazuje, że warunki silnej testowalności występujące w sieci łączącej MN^n przy zdarzeniu propagowanym przez jeden z podukładów mogą być wymuszane przez drugi z podukładów.

Analiza istniejących w literaturze układów T^n realizowanych jako całkowicie rozgałęzione [5, 66, 73, 76] pozwala sądzić, iż sieci łączące mają własność, którą opisują poniższe definicje.

Definicja 4.1 *Sieć łącząca MN^n wielowyjściowego układu progowego jest nazywana* siecią o pojedynczej ścieżce*, jeżeli istnieją wyłącznie pojedyncze ścieżki:*

- *z dowolnego wejścia pierwotnego $T_i^{n_a}$ do każdego wyjścia pierwotnego T_m^n, $i \leqslant m \leqslant i + n_b$;*
- *z dowolnego wejścia pierwotnego $T_i^{n_b}$ do każdego wyjścia pierwotnego T_m^n, $i \leqslant m \leqslant i + n_a$.*

Mówimy, że sieć łącząca ma własność SP *(ang.* single path*), jeżeli jest siecią o pojedynczej ścieżce.*

W dalszej części pracy, sieć łączącą mającą własność SP będziemy nazywali krótko *układem SP*. Własność SP ściśle określa zbiór ścieżek w sieci łączącej. Dla przykładu, oznacza ona, że w czterowejściowej sieci łączącej z wejścia $T_1^{2_a}$ istnieje po jednej ścieżce do wyjść T_1^4, T_2^4 oraz T_3^4. Dla celów dalszych analiz zostanie zdefiniowane rozszerzenie własności SP.

Własność 4.1 *W układzie SP z dowolnego wyjścia do dowolnego wejścia można znaleźć co najwyżej jeden ciąg bramek i połączeń.*

Dowód

Załóżmy, że z wyjścia T^n istnieją dwa różne ciągi bramek i połączeń do wejścia T^{n_a} (T^{n_b}). Wtedy, od strony wejścia T^{n_a} (T^{n_b}) ciągi te są widoczne jako dwie różne ścieżki, co jest sprzeczne z definicją układu o pojedynczej ścieżce. ■

Zauważmy teraz, że wektor wyjściowy każdego z podukładów T^{n_a} i T^{n_b} jest posortowany. Załóżmy, że jeżeli przez podukład T^{n_a} lub T^{n_b} jest propagowane pojedyncze zdarzenie, wtedy wzorzec na wyjściu podukładu ma postać dwóch ciągów: ciągu jedynek i ciągu zer, rozdzielonych zdarzeniem. Zatem, pełny wzorzec na wejściu sieci łączącej ma postać dwóch ciągów posortowanych, przy czym elementem rozdzielającym zera i jedynki w jednym z ciągów jest zdarzenie. Załóżmy, że wtedy wzorzec na wyjściu sieci łączącej ma postać jednego ciągu posortowanego, ze zdarzeniem rozdzielającym zera i jedynki.

Definicja 4.2 *Mówimy, że sieć łącząca MN^n* generuje wielowyjściowe układy progowe mające własność silnej testowalności uszkodzeń typu opóźnienie ścieżki*, jeżeli spełnione są następujące warunki:*

1. *sieć MN^n jest układem SP;*

2. *dla wzorca wejściowego postaci*

$$V_{in} = \underbrace{\underbrace{1\ldots1}_{j-1} * \underbrace{0\ldots0}_{n_{a(b)}-j}}_{T^{n_{a(b)}}} \underbrace{\underbrace{1\ldots1}_{k} \underbrace{0\ldots0}_{n_{b(a)}-k}}_{T^{n_{b(a)}}} \tag{4.1}$$

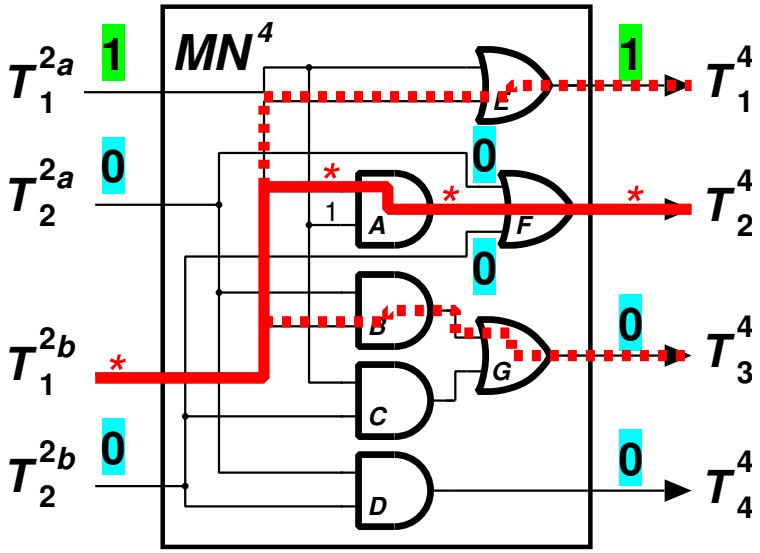
wzorzec wyjściowy ma postać

$$V_{out} = \underbrace{1\ldots1}_{j+k-1} * \underbrace{0\ldots0}_{n-(j+k)} . \tag{4.2}$$

Definicja 4.3 *Mówimy, że sieć łącząca ma* własność RG *(ang. robust generator), jeżeli generuje wielowyjściowe układy progowe mające własność silnej testowalności uszkodzeń typu opóźnienie ścieżki.*

W dalszej części pracy, sieć łączącą mającą własność RG będziemy nazywali krótko *układem RG*. Własność RG jest podobna do własności sieci łączącej polegającej na łączeniu wektorów uporządkowanych, jednak jej zdefiniowanie jest konieczne dla rozważań zamieszczonych w dalszej części pracy. Dalej wykażemy, że układ T^n zbudowany w oparciu o sieci łączące mające własność RG jest silnie testowalny.

Przykład 4.1 *Rozważmy 2-poziomową 4-wejściową sieć łączącą z rys. 4.1. Można zauważyć, iż jest on układem SP, ponieważ istnieje co najwyżej jedna ścieżka z każdego z wejść $T_1^{2_a}$, $T_2^{2_a}$, $T_1^{2_b}$*



Rysunek 4.1: Propagacja zdarzenia w układzie łączącym MN^4

oraz $T_2^{2_b}$ *do dowolnego z wyjść* $T_1^4, T_2^4, T_3^4, T_4^4$. *Równocześnie, z wejścia* $T_1^{2_a}$ *ścieżki prowadzą do wyjść* T_1^4, T_2^4, T_3^4 *(analogicznie można prześledzić ścieżki z wejść* $T_2^{2_a}$, $T_1^{2_b}$, $T_2^{2_b}$*). Niech wzorzec wejściowy ma postać* $10*0$. *Zatem jeżeli pierwszy binarny wektor wejściowy ma postać* 1000, *to drugi wektor binarny ma postać* 1010. *Wtedy pierwszy wektor wyjściowy ma postać* 1000, *zaś drugi wektor ma postać* 1100. *Zatem wzorzec wyjściowy ma postać* $1*00$. *Zdarzenie jest propagowane wzdłuż ścieżki* $T_1^{2_b} - A - F - T_2^4$. *Wejścia boczne bramek A i F mają wartości nie-sterujące. Teraz można zauważyć, iż ścieżka* $T_1^{2_b} - A - F - T_2^4$ *jest jedyną ścieżką prowadzącą z wejścia* $T_1^{2_b}$ *do wyjścia* T_2^4.

Lemat 4.1 *Układ RG jest silnie testowalny.*

Dowód

Niech $p_{i\to j}$ oznacza ścieżkę prowadzącą od wejścia $T_i^{n_a}$ ($T_i^{n_b}$) do wyjścia T_j^n, $i \leqslant j \leqslant i + n_b$. Ponieważ układ RG jest układem SP, ścieżka $p_{i\to j}$ jest jedyną ścieżką prowadzącą z wejścia $T_i^{n_a}$ ($T_i^{n_b}$) do wyjścia T_j^n, zaś warunki testowania zostały podane w definicji układu RG. Aby umożliwić propagację zdarzenia wzdłuż ścieżki $p_{i\to j}$, wejścia boczne wszystkich bramek leżących na ścieżce muszą mieć ustawione wartości nie-sterujące. Stąd ścieżka $p_{i\to j}$ jest silnie testowalna. Ponieważ powyższą analizę można przeprowadzić dla każdej pary (i, j) określonej w definicji układu SP, stąd wszystkie ścieżki w sieci są silnie testowalne. ■

Lemat 4.2 *Niech T^n będzie wielowyjściowym układem progowym złożonym z podukładów T^{n_a}, T^{n_b} oraz sieci łączącej MN^n. Jeżeli są spełnione następujące warunki:*

1. *układ MN^n jest n-wejściową siecią łączącą mającą własność RG;*
2. *$1 \leqslant n_a < n$ i $n_b = n - n_a$;*
3. *T^{n_a}, T^{n_b} są wielowyjściowymi układami progowymi spełniającymi następujące warunki:*
   a) *każda ścieżka w układzie jest silnie testowalna;*
   b) *dla każdej ścieżki prowadzącej:*
      i. *z wejścia pierwotnego x_j^a do wyjścia $T_i^{n_a}$, istnieje taki wejściowy wzorzec testowy, że wyjściowy wzorzec ma postać $\underbrace{1\ldots1}_{i-1} * \underbrace{0\ldots0}_{n_a-i}$;*
      ii. *z wejścia pierwotnego x_j^b do wyjścia $T_i^{n_b}$, istnieje taki wejściowy wzorzec testowy, że wyjściowy wzorzec ma postać $\underbrace{1\ldots1}_{i-1} * \underbrace{0\ldots0}_{n_b-i}$,*

wtedy każda ścieżka układu T^n również spełnia warunki 3a oraz 3b.

Dowód

Niech T_i^{n*} oznacza dowolne wejście sieci łączącej MN^n, a T_j^n jego dowolne wyjście. Ponieważ sieć łącząca MN^n jest układem SP, zatem z dowolnego wejścia T_i^{n*} układu łączącego do dowolnego wyjścia T_j^n prowadzi co najwyżej jedna ścieżka, którą oznaczymy przez $p_{i\to j}$. Wprowadźmy następujące oznaczenia:

A – zbiór ścieżek dochodzących do wejścia T_i^{n*};

a_k^i – elementy zbioru A, czyli $A = \{a_1^i, a_2^i, \ldots\}$;

B – zbiór ścieżek z wejścia T_i^{n*} do wyjścia T_j^n(jest to zbiór jednoelementowy, ponieważ układ MN^n na mocy założenia jest układem SP);

$C = A \times B$;

C_1 – zbiór ścieżek zawierający ścieżki powstałe z połączenia ścieżek tworzących pary w zbiorze C.

Ponieważ sieć MN^n jest układem SP, zbiór C zawiera pary postaci $(a_k^i, p_{i\to j})$. Zbiór C_1 jest zbiorem ścieżek w układzie T^n przechodzących przez wejście T_i^{n*} i dochodzących do wyjścia T_j^n, który może zostać zdefiniowany dla wszystkich par wejście T_i^{n*} – wyjście T_j^n. układu łączącego MN^n, dla których ścieżki istnieją.

Na mocy założenia, każda ścieżka ze zbioru A jest silnie testowalna, a każdy wyjściowy wzorzec testowy dla ścieżki ze zbioru A ma postać $\underbrace{1\ldots1}_{i-1} * \underbrace{0\ldots0}_{n_a-i}$ lub $\underbrace{1\ldots1}_{i-1} * \underbrace{0\ldots0}_{n_b-i}$, w zależności od tego dla którego układu (T^{n_a} lub T^{n_b}) zbiór A został zdefiniowany.

Ponieważ sieć MN^n jest układem RG, wzorzec wyjściowy testu dla każdej ścieżki ze zbioru A może być równocześnie wzorcem dla każdej ścieżki ze zbioru C_1, w zależności od drugiego członu wzorca testowego, z definicji układu RG, który jest określany przez układ odpowiednio T^{n_b} lub T^{n_a}. Ponieważ układy T^{n_b} i T^{n_a} są rozłączne i nie mają wspólnych wejść, wartości wyjść układów T^{n_b} i T^{n_a} są niezależne. Stąd każda ścieżka w zbiorze C_1 jest silnie testowalna. Ponieważ sieć łącząca MN^n jest układem RG, wyjściowy wzorzec testowy dla każdej ścieżki ze zbioru C_1 ma postać $\underbrace{1\ldots1}_{j-1} * \underbrace{0\ldots0}_{n-j}$.

Przedstawione powyżej rozumowanie można przeprowadzić dla dowolnej pary (wejście,wyjście) (T_i^{n*}, T_j^n) sieci łączącej MN^n dla której jest zdefiniowana ścieżka, a zatem układ spełnia warunki 3a oraz 3b. ■

Wniosek 4.1 *Wielowyjściowy układ progowy o rozłącznych podukładach, złożony z podukładów mających własność silnej testowalności oraz sieci łączącej mającej własności SP i RG, również ma własność silnej testowalności.*

Dowód

Wynika bezpośrednio z punktu 3a lematu 4.2. ■

Twierdzenie 4.1 *Wielowyjściowy układ progowy zrealizowany jako całkowicie rozgałęziony, w którym sieci łączące mają własności SP i RG, ma własność silnej testowalności.*

Dowód

Wynika bezpośrednio z rekurencyjnego rozwinięcia lematu 4.2 i wniosku 4.1. ■

4.3 Analiza własności sieci łączących

Powyżej opisano ogólne własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych zrealizowanych jako całkowicie rozgałęzione. Ustalono, że istnieją własności nazwane SP i RG, których spełnienie implikuje własność silnej testowalności. Teraz podejmiemy próbę wykazania, że niektóre wielowyjściowe układy progowe znane w literaturze mają własności SP i RG.

4.3.1 Dwupoziomowa sieć łącząca

Sieć łącząca dwupoziomowa realizuje następujące funkcje [66]:

$$T_m^n = \sum_i T_i^{n_a} T_{m-i}^{n_b}, \quad 0 \leqslant i \leqslant n_a, \quad 0 \leqslant m-i \leqslant n_b \ . \tag{4.3}$$

Wniosek 4.2 *Sieć łącząca dwupoziomowa jest układem SP.*

Dowód

We wzorze (4.3) opisującym sieć łączącą dwupoziomową, każdy literał $T_i^{n_a}$ lub $T_i^{n_b}$ pojawia się co najwyżej jeden raz dla każdego wyjścia T_m^n, odpowiednio $i \leqslant m \leqslant i+n_b$ lub $i \leqslant m \leqslant i+n_a$. ■

Wniosek 4.3 *Sieć łącząca dwupoziomowa jest układem RG.*

Dowód

1. Jak udowodniono we wniosku 4.2, dwupoziomowa sieć łącząca jest układem SP.

2. Niech

$$V_{in} = \underbrace{\underbrace{1\ldots1}_{j-1} * \underbrace{0\ldots0}_{n_{a(b)}-j}}_{T^{n_{a(b)}}} \underbrace{\underbrace{1\ldots1}_{k} \underbrace{0\ldots0}_{n_{b(a)}-k}}_{T^{n_{b(a)}}}, \tag{4.4}$$

   wtedy $m = i + j$, $n = n_a + n_b$, a wzorzec wyjściowy (obliczony z równania 4.3) ma postać

$$V_{out} = \underbrace{1\ldots1}_{m-1} * \underbrace{0\ldots0}_{n-m} \tag{4.5}$$

■

4.3.2 Trzypoziomowa sieć łącząca

Wyjścia sieci łączącej podanej w [73], dla podziału $n_a = n_b$ lub $n_a = n_b + 1$ mogą zostać zapisane z użyciem następujących wzorów:

$$T_m^n = \begin{cases} \sum_{i=0}^{\lceil \frac{m}{2} \rceil - 1} u_k^{n_b} u_{m-i}^{n_a} + B, & \text{jeżeli } 1 \leqslant m \leqslant n_b \\ \sum_{i=\lceil \frac{m}{2} \rceil}^{n_b} u_i^{n_a} u_{m-i}^{n_b} + B + C, & \text{jeżeli } n_b < m \leqslant n, \end{cases} \tag{4.6}$$

gdzie:

$$B = \begin{cases} 0, & \text{jeżeli } m \text{ jest nieparzyste} \\ u_{\frac{m}{2}}^{n_b}, & \text{jeżeli } m \text{ jest parzyste,} \end{cases} \tag{4.7}$$

$$C = \begin{cases} 0, & \text{jeżeli } n_a = n_b \\ T_{n_a}^{n_a} T_{m-n_a}^{n_b}, & \text{jeżeli } n_a = n_b + 1, \end{cases} \tag{4.8}$$

oraz

$$\begin{aligned} u_m^{n_a} &= T_m^{n_a} + T_m^{n_b} \\ u_m^{n_b} &= T_m^{n_a} T_m^{n_b} \\ T_0^x &= 1 \end{aligned} \tag{4.9}$$

Wniosek 4.4 *Sieć łącząca trzypoziomowa jest układem SP.*

Dowód

Rozwinięcie wzorów (4.6)–(4.9) opisujących sieć łączącą trzypoziomową prowadzi do wzoru (4.3). Sieć jest bezinwerterowa, przeto nigdzie nie występuje redukcja literałów. Zatem dowód jest równoznaczny z dowodem wniosku 4.2. Stąd sieć ma własność SP. ■

Wniosek 4.5 *Sieć łącząca trzypoziomowa jest układem RG.*

Dowód

Jak wykazano we wniosku 4.4, równania sieci trzypoziomowej są równoważne równaniom sieci dwupoziomowej. Stąd dowód sprowadza się do dowodu wniosku 4.3. Zatem sieć trzypoziomowa jest układem RG. ■

4.4 Podsumowanie

W rozdziale wykazaliśmy, że własność silnej testowalności uszkodzeń typu opóźnienia w wielowyjściowym układzie progowym może zostać uzyskana jedynie poprzez dobór sieci łączących mających łatwe do zweryfikowania własności. Wskazaliśmy, że zdefiniowane własności mają dwie klasy sieci łączących: sieć dwupoziomowa przedstawiona przez Edwardsa [21] oraz sieć trzypoziomowa przedstawiona przez Rahamana i innych [73]. Analizowane klasy sieci charakteryzują się złożonością $O(n^2 \log n)$ [64, 73], co może być trudne do zaakceptowania dla większych n. W następnych rozdziałach pokażemy propozycję wielowyjściowego układu progowego złożonego z sieci łączących mających własności SP i RG, bazującego na sieci sortującej Batchera [5], którego złożoność jest $O(n \log^2 n)$.

Rozdział 5

Testowalność uszkodzeń typu opóźnienie w wielowyjściowych układach progowych implementowanych jako sieć sortująca Batchera

5.1 Wstęp

W poniższym rozdziale przedstawimy analizę struktury sieci sortującej Batchera nieparzysto-parzystej [5]. W podrozdziale 5.2 opiszemy strukturę sieci łączącej o liczbie linii będącej potęgą liczby 2 i podamy schemat ogólny takiej sieci, następnie przedstawimy struktury sieci o parzystej i nieparzystej liczbie wejść. Omówimy również implementację sieci łączącej o nieparzystej liczbie wejść poprzez usunięcie linii z sieci łączącej Batchera o parzystej liczbie wejść i porównamy ją z siecią uzyskaną przez rekurencyjne zastosowanie algorytmu Batchera. W podrozdziale 5.3 przedstawimy analizę własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w sieci łączącej Batchera. Następnie, w podrozdziale 5.4 wykażemy istnienie modyfikacji sieci łączącej prowadzącej do nabycia przez sieć własności silnej testowalności uszkodzeń typu opóźnienie ścieżki

i pokażemy algorytmy takiej modyfikacji.

W dalszej części pracy będziemy stosować następującą uproszczoną terminologię:

– *sieć Batchera* – sieć sortująca Batchera nieparzysto-parzysta;

– *sieć łącząca Batchera OE* – sieć łącząca sieci sortującej nieparzysto-parzystej Batchera;

– *algorytm Batchera OE* – algorytm generowania sieci łączącej nieparzysto-parzystej, przedstawiony w [5].

5.2 Struktura sieci łączącej nieparzysto-parzystej Batchera

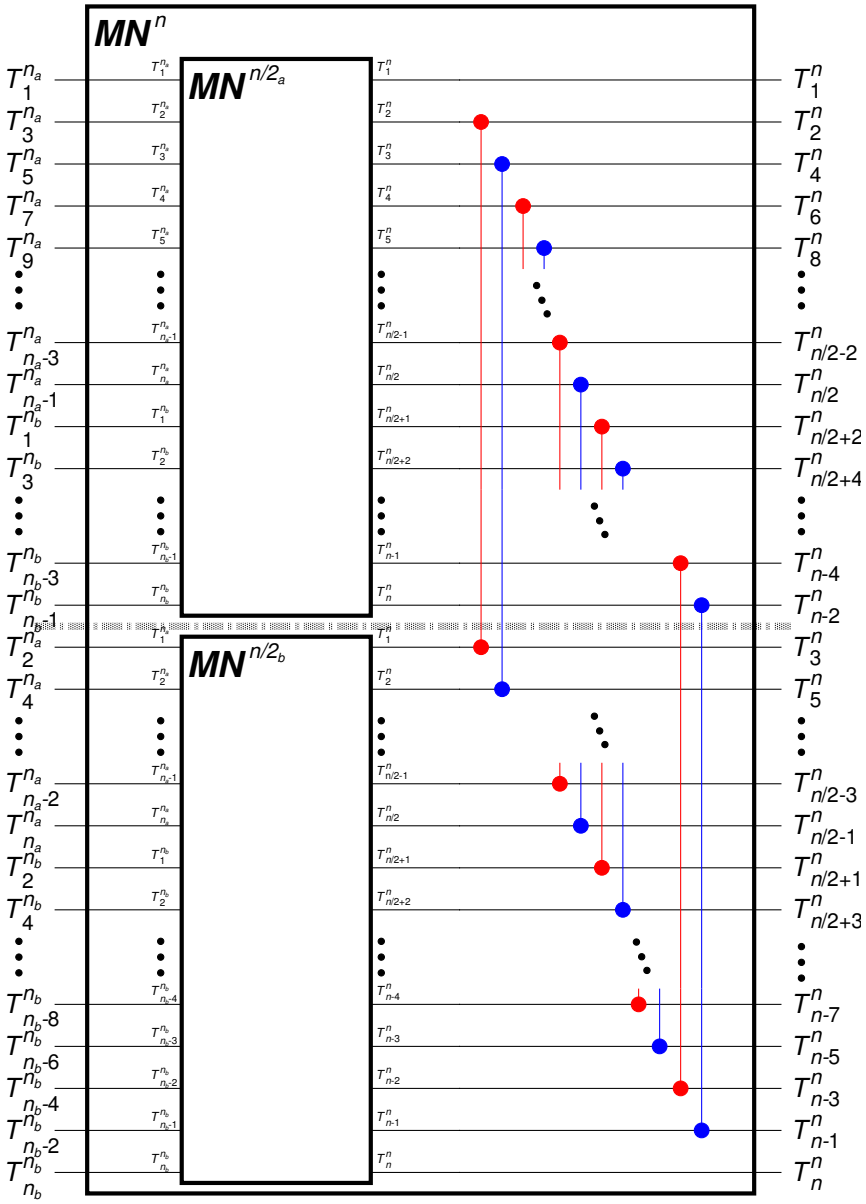
5.2.1 Uwagi ogólne

Sieć łącząca Batchera OE o n liniach jest rekurencyjnie złożona z dwóch sieci łączących o liczbie linii równej odpowiednio $n_a = \lfloor \frac{n}{2} \rfloor$ i $n_b = \lceil \frac{n}{2} \rceil$ oraz poziomu komparatorów generujących wynik końcowy [5]. Schemat ogólny sieci łączącej Batchera OE dla parzystej liczby linii n przedstawiono na rys. 5.1. Natomiast przykład sieci łączącej o 16 wejściach, w której wyjścia uporządkowano wg rosnących wag, przedstawiono na rys. 5.2.

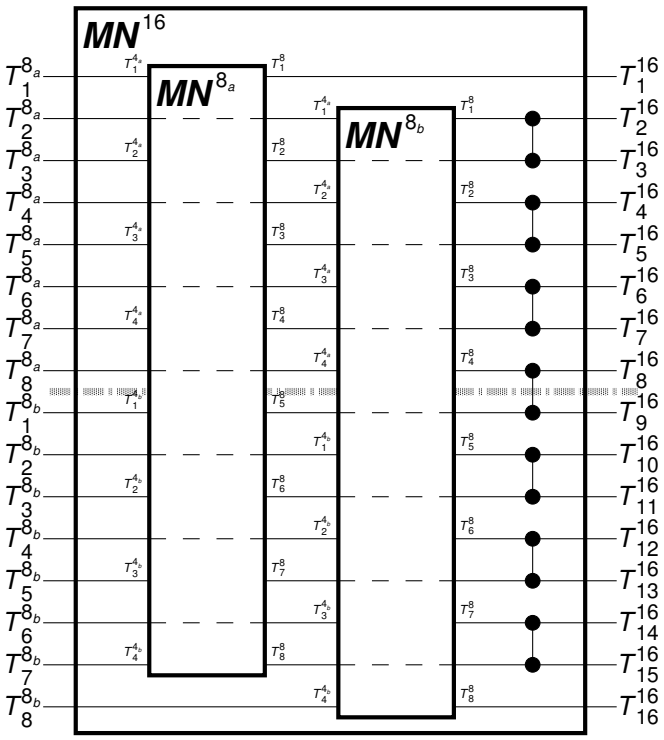
Definicja 5.1 **[84]** *Komparator znajduje się na poziomie i, jeśli jedno z jego wejść podłączone jest bezpośrednio do wyjścia poziomu $i-1$ a pozostałe – do wyjścia poziomu $i-1$ lub niższego. Każde wyjście komparatora znajdującego się na poziomie i stanowi jednocześnie jedno z wyjść poziomu i. Wejście całej sieci jest równoważne poziomowi* 0.

5.2.2 Własności sieci łączącej o liczbie wejść $n = 2^k$

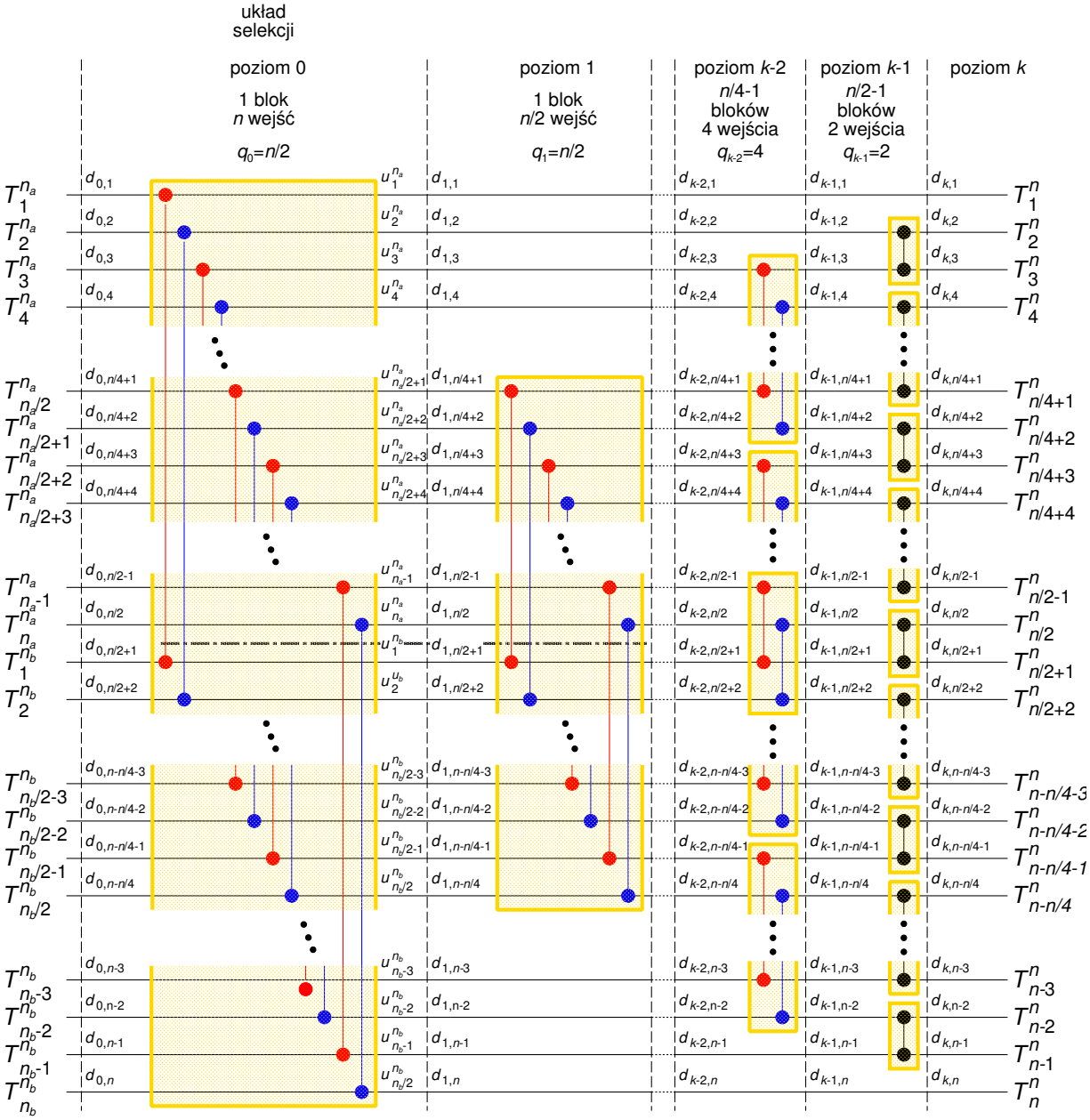
W sieci łączącej można wyróżnić linie prowadzące z wejścia do wyjścia oraz regularne grupy komparatorów (rys. 5.3), które dzielą sieć na poziomy. Na danym poziomie i, komparatory łączą linie j oraz $j \pm q_i$, gdzie $q_i = \frac{n}{2^{i+1}}$. Linie mogą być określone przez numer linii i numer poziomu $d_{i,j}$, gdzie i jest numerem poziomu, zaś j jest numerem linii. Oznaczenie d_i oznacza wektor wszystkich linii na danym poziomie, tj. $d_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,n-1}, d_{i,n}\}$. Ogólny schemat numeracji



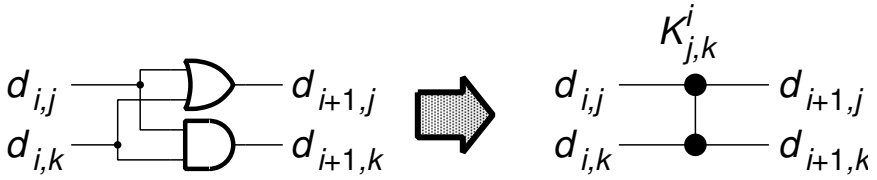
Rysunek 5.1: Ogólny schemat sieci łączącej Batchera OE dla parzystej liczby wejść



Rysunek 5.2: Przykład sieci łączącej Batchera OE o 16 wejściach



Rysunek 5.3: Sieć łącząca Batchera OE o $n = 2^k$ wejściach



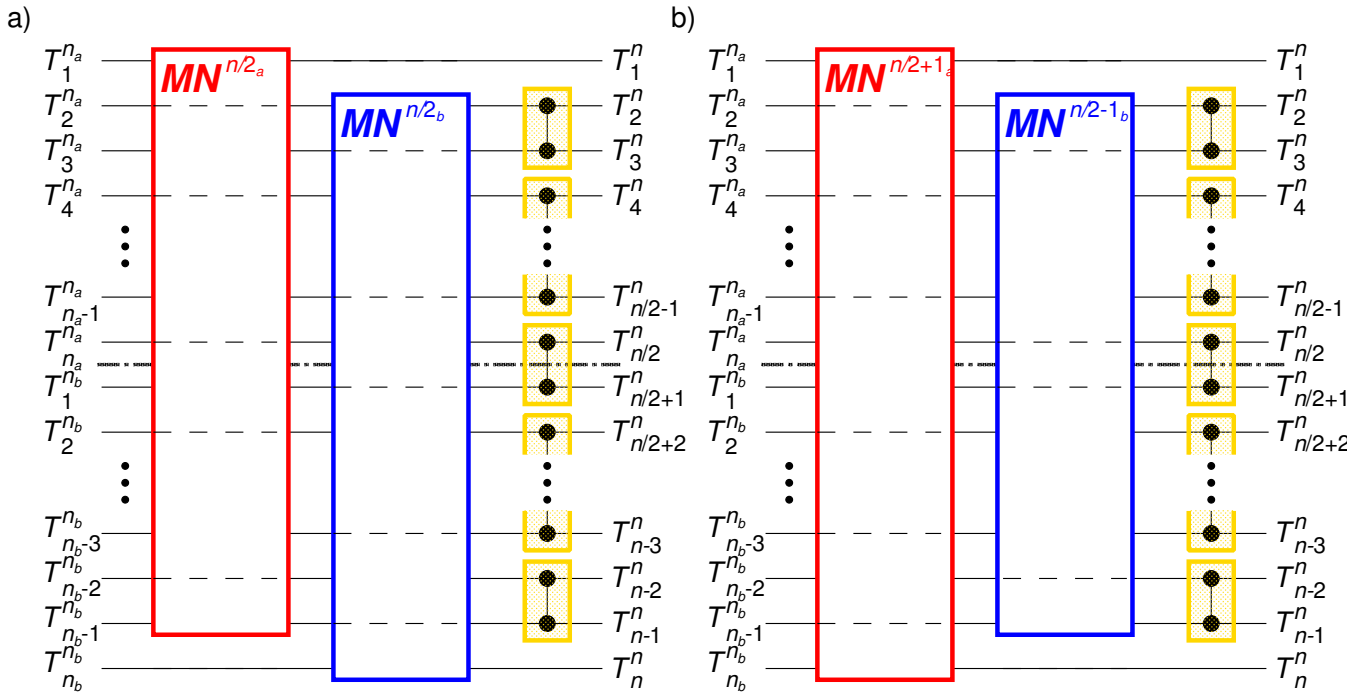
Rysunek 5.4: Oznaczenia komparatora i linii w sieci łączącej

linii i poziomów podano na rys. 5.3. Zależność logiczna pomiędzy liniami, realizowana przez komparator binarny $K^i_{j,j+q_i}$ (rys. 5.4) łączący te linie, jest zapisana jako: (i) $d_{i,j} = d_{i-1,j} + d_{i-1,j+q_i}$ lub (ii) $d_{i,j} = d_{i-1,j} \cdot d_{i-1,j-q_i}$ lub też (iii) $d_{i,j} = d_{i-1,j}$ – jeżeli na danej linii j pomiędzy poziomami $i-1$ a i komparator nie występuje.

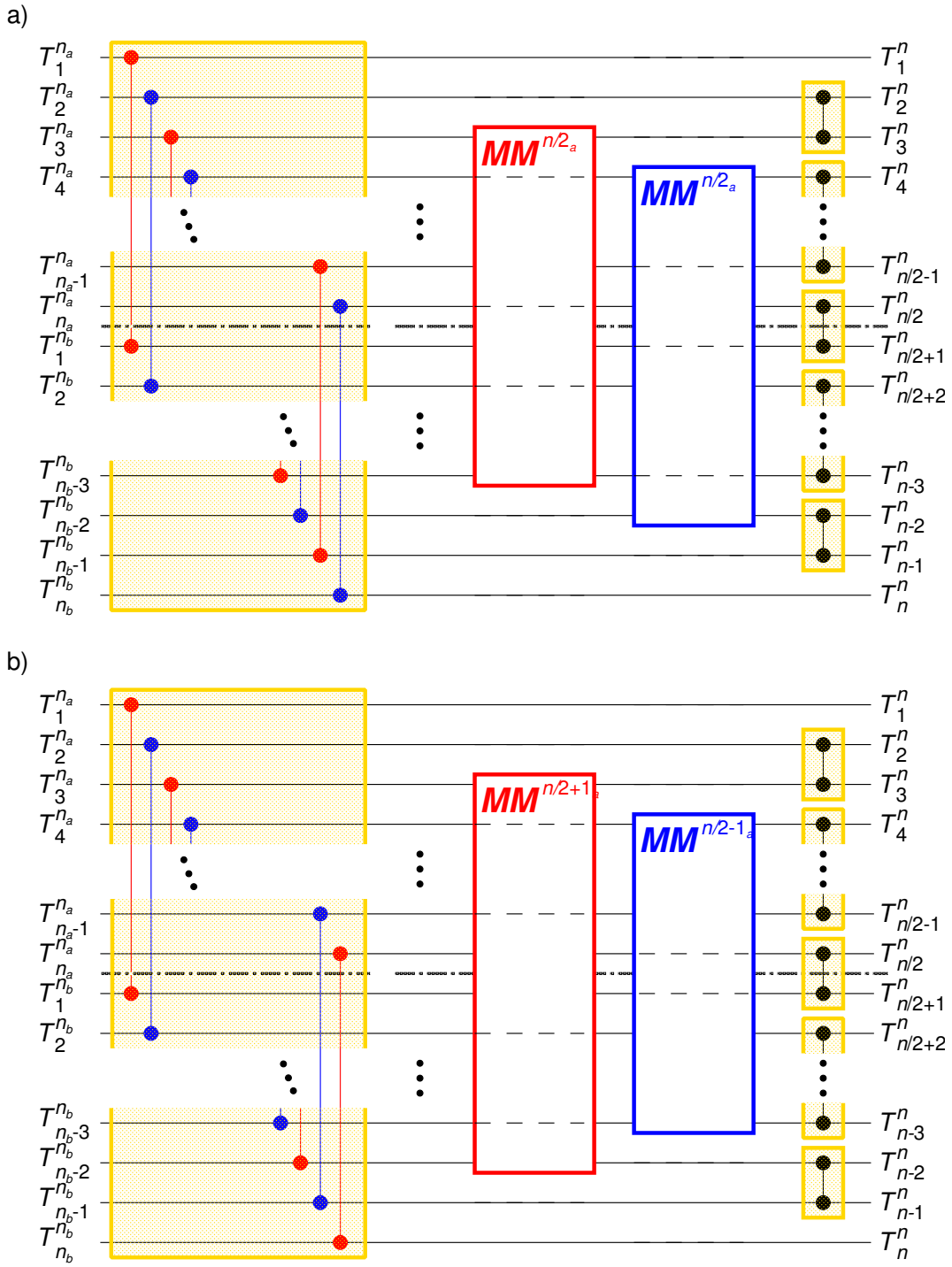
5.2.3 Własności sieci łączącej o parzystej liczbie wejść $n = 2k$

Na rys. 5.5 przedstawiono schematy dwóch sieci Batchera OE o parzystej liczbie wejść. Szczegółowa analiza struktury sieci i precyzyjne ustalenie położenia poszczególnych komparatorów dla dowolnej liczby wejść n jest tematem złożonym i wykracza poza ramy niniejszej pracy. Tutaj zajmiemy się jedynie podziałem na podsieci i ustalimy dokładne położenie niektórych komparatorów na potrzeby analiz przedstawionych w dalszej części rozprawy.

Wymóg łączenia linii parzystych z parzystymi oraz nieparzystych z nieparzystymi narzuca dobór podsieci łączących. Jeżeli $k = n_a = n_b$ jest parzyste, wtedy liczba par linii wejściowych o indeksach parzystych jest taka sama jak liczba par linii wejściowych o indeksach nieparzystych. Przykładowo, aby połączyć dwa ciągi dwuelementowe $T_1^{2_a}, T_2^{2_a}$ oraz $T_1^{2_b}, T_2^{2_b}$, trzeba połączyć następujące pary linii: (i) nieparzyste $T_1^{2_a} - T_1^{2_b}$ i (ii) parzyste $T_2^{2_a} - T_2^{2_b}$. Zatem każda z podsieci łączących jest siecią MN^2. Jeżeli $k = n_a = n_b$ jest nieparzyste, wtedy liczba par linii wejściowych o indeksach parzystych jest o jeden mniejsza niż liczba par linii wejściowych o indeksach nieparzystych. Przykładowo, przy łączeniu ciągów trzyelementowych $T_1^{3_a}, T_2^{3_a}, T_3^{3_a}$ oraz $T_1^{3_b}, T_2^{3_b}, T_3^{3_b}$, trzeba połączyć następujące pary linii: (i) nieparzyste $T_1^{3_a} - T_1^{3_b}$, $T_3^{3_a} - T_3^{3_b}$, oraz (ii) parzyste $T_2^{3_a} - T_2^{3_b}$. Tutaj konieczne jest zastosowanie różnych podsieci: MN^4 do łączenia par linii wejściowych o indeksach nieparzystych oraz MN^2 do łączenia par linii wejściowych o indeksach parzystych. W ogólnym przypadku, jeżeli n i k są parzyste (czyli n jest podzielne przez 4), to sieci łączące linie o indeksach nieparzystych i parzystych są identyczne. Natomiast, jeżeli n jest



Rysunek 5.5: Sieć łącząca Batchera OE o parzystej liczbie wejść n dla : a) n podzielnego przez 4, b) n niepodzielnego przez 4



Rysunek 5.6: Sieć łącząca Batchera OE o parzystej liczbie wejść n po wydzieleniu warstwy wejściowej dla: a) n podzielnego przez 4, b) n niepodzielnego przez 4

parzyste a k jest nieparzyste (n parzyste niepodzielne przez 4), to sieć łącząca linie o indeksach nieparzystych ma o dwie linie wejściowe więcej niż sieć łącząca linie o indeksach parzystych. Schematy ogólne sieci łączących dla k parzystego ustalone na podstawie powyższych rozważań przedstawiono na rys. 5.5a i 5.5b.

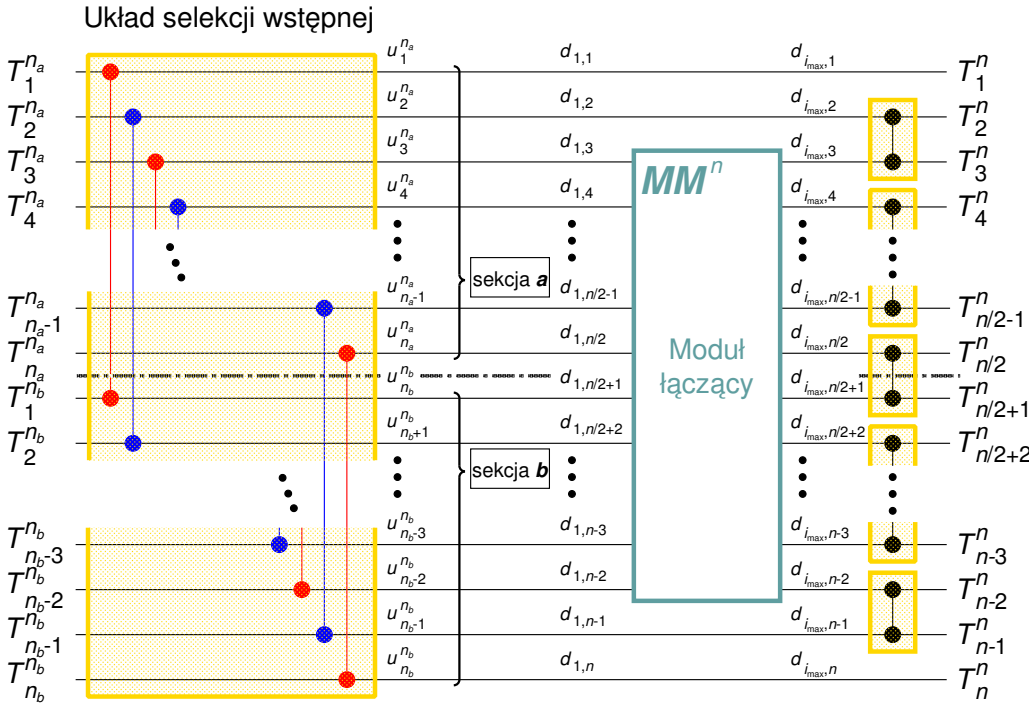
Dalsza analiza wykazała następną cechę struktury sieci o parzystej liczbie wejść n. Można w nim rozróżnić dwa podukłady: układ selekcji wstępnej, złożony z komparatorów łączących pary linii z wejść T^{n_a} i T^{n_b}, oraz moduł łączący, na który składają się dwa moduły łączące MM^{n_a} i MM^{n_b}. Układy MM^{n_a} i MM^{n_b} są identyczne dla n podzielnego przez 4 lub różnią się o dwie linie dla parzystego n niepodzielnego przez 4. Schematy tych sieci przedstawiono na rys. 5.6.

Z modułów łączących MM^{n_a} i MM^{n_b} można utworzyć pojedynczy moduł łączący MM^n o strukturze przedstawionej na rys. 5.7. Linie znajdujące się za układem selekcji wstępnej mają podwójne oznaczenia, np. u^{n_a} i $d_{1,1}$ etc. (rys. 5.7). Wyjścia pierwotne układu selekcji wstępnej są oznaczane przez $u_i^{n_a}$ i $u_i^{n_b}$, przy czym i jest numerem wejścia pierwotnego odpowiadającego danemu wyjściu: $T_i^{n_x} \rightarrow u_i^{n_x}$ $(x = \{a, b\})$. Bloki wyjść $u_1^{n_a} \ldots u_{n_a}^{n_a}$ i $u_1^{n_a} \ldots u_{n_a}^{n_a}$ określono odpowiednio jako linie $d_{1,1} \ldots d_{1,n}$. Dalsze poziomy sieci oznaczono numerami od 1 do $i_{\max}$, przy czym poziom $i_{\max}$ jest ostatnim poziomem przed komparatorami wyjściowymi. Wprowadzony tutaj przez nas system oznaczeń jest niezbędny do analizy własności silnej testowalności uszkodzeń typu opóźnienie ścieżki, przeprowadzonej w dalszej części tego rozdziału.

5.2.4 Własności sieci łączącej o nieparzystej liczbie wejść $n = 2k \pm 1$

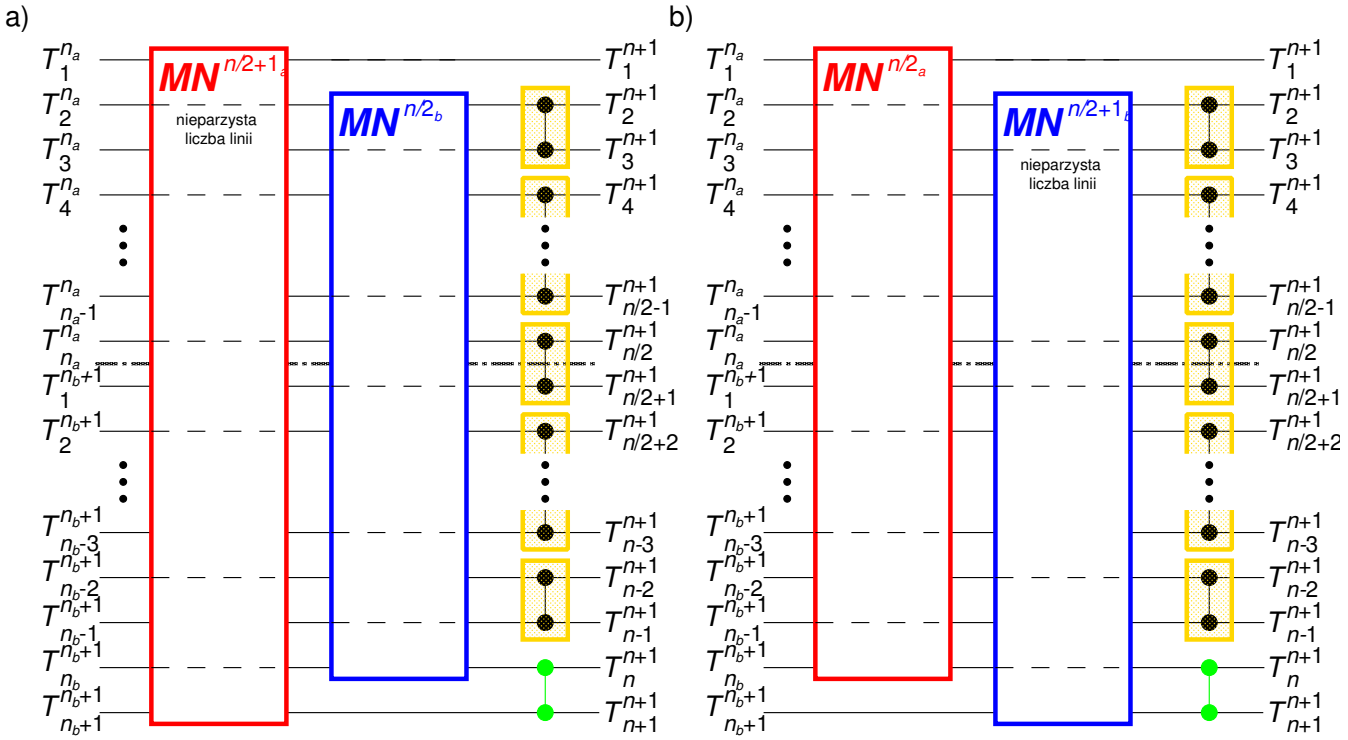
Sieć łącząca o nieparzystej liczbie wejść $n = 2k + 1$ generowana z użyciem algorytmu Batchera OE

We wcześniejszych rozważaniach wykazaliśmy, że sieć o parzystej liczbie wejść ma regularną strukturę, oraz, że sieć o parzystej liczbie linii składa się rekurencyjnie z sieci o parzystej liczbie linii. Teraz spróbujemy znaleźć takie regularności w sieci o nieparzystej liczbie wejść. Rysunek 5.8 przedstawia schemat sieci łączącej Batchera OE dla nieparzystej liczby wejść $n = 2k + 1$, a rys. 5.9 i 5.10 przykłady tej sieci dla $n = 5$, $n = 7$ i $n = 9$. Zauważmy, że przykładowe sieci o niepa-

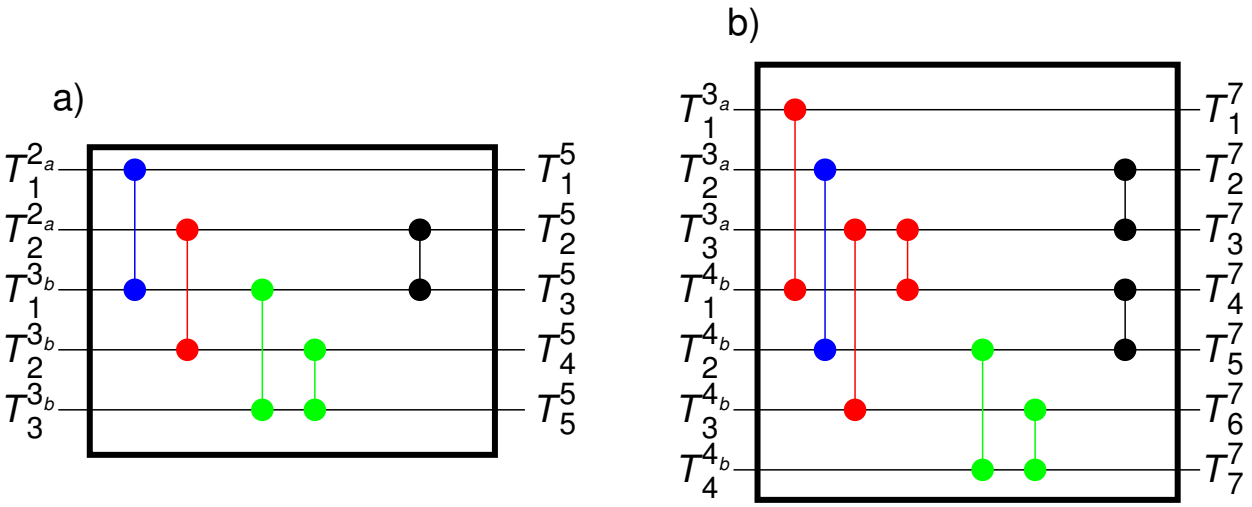


Rysunek 5.7: Oznaczenia sieci łączącej Batchera OE o parzystej liczbie wejść n

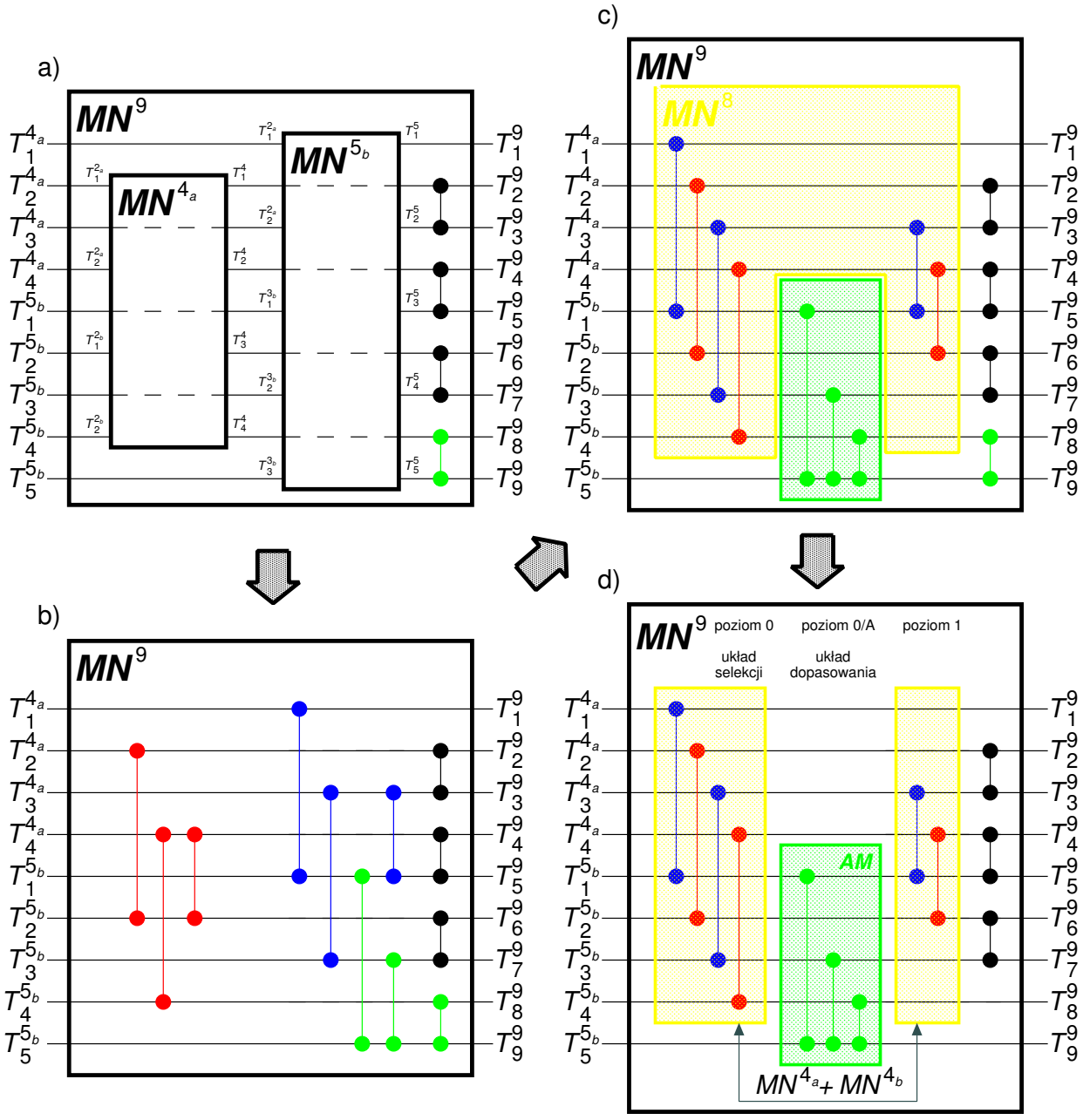
rzystej liczbie linii przypominają sieci o parzystej liczbie linii z dokładnością do komparatorów podłączonych do linii $d_{i,n+1}$. Spróbujmy teraz uogólnić ten schemat. Na rys. 5.10a przedstawiono złożenie sieci łączącej Batchera OE o 9 wejściach. Sieć tą możemy następnie przedstawić w postaci sieci pojedynczych komparatorów (rys. 5.10b). Sieć łącząca 5-wejściowa MN^{5_b}, występująca jako składowa, została przedstawiona na rys. 5.9a. Część komparatorów należących do sieci MN^{5_b}, a także dodatkowy komparator wyjściowy zostały wyróżnione na rys. 5.10c. Należy zauważyć, iż realizacja sieci łączącej MN^9 z rys. 5.10b i 5.10c pozwala na wyróżnienie dwóch grup komparatorów, jako dwóch sieci łączących 4-wejściowych MN^{4_a} i MN^{4_b} oraz pewnego układu dopasowującego, wyróżnionego na rys. 5.10c i 5.10d. Stąd, sieć MN^9 można postrzegać jako modyfikację sieci MN^8 polegającą na dostawieniu jednej, najstarszej linii oraz komparatorów dopasowujących (rys. 5.10d). Dalej, z sieci 8-wejściowej, jako sieci o parzystej liczbie linii można wydzielić warstwę komparatorów tworzących układ selekcji wstępnej, oznaczany jako poziom 0 (rozdział 5.2.2 oraz 5.2.3). W ogólnym przypadku sieć łącząca MN^n o nieparzystej liczbie wejść $n = 2k+1$ składa się z dwóch sieci łączących o parzystej liczbie wejść MN^{n_a} oraz MN^{n_b} oraz układu do-



Rysunek 5.8: Sieć łącząca Batchera o nieparzystej liczbie wejść $n = 2k+1$ dla: a) k parzystego, b) k nieparzystego



Rysunek 5.9: Przykłady sieci łączących Batchera OE dla: a) $n = 5$, b) $n = 7$



Rysunek 5.10: Analiza sieci Batchera OE dla $n = 9$

pasowania AM. Komparatory w układzie dopasowania AM łączą linię $d_{i,n}$ z liniami $d_{1+i,n-2^i}$. Układ dopasowania AM zawiera $\lceil \log_2 n \rceil - 1$ komparatorów. Ponieważ sieć o nieparzystej liczbie wejść $n = 2k + 1$ powstaje ze złożenia sieci o parzystej liczbie wejść $n = 2k$ i układu dopasowania AM, można jej strukturę przedstawić analogicznie do struktury sieci o parzystej liczbie wejść $n = 2k$ (rozdz. 5.2.3). Na rys. 5.11 podano schemat ogólny sieci o nieparzystej liczbie wejść $n = 2k + 1$ po rozdzieleniu na poszczególne podukłady.

Zauważmy teraz, że wystąpienie układu AM powoduje, że istotnie zwiększa się liczba ścieżek prowadzących do wyjścia T_{n+1}^{n+1}. Z własności SP wiemy, że do wyjścia T_{n+1}^{n+1} mogą jedynie prowadzić ścieżki z wejść $T_{n_a}^{n_a}$ oraz $T_{n_b+1}^{n_b+1}$. Przypuszczamy, że ścieżki prowadzące z wejść innych niż $T_{n_a}^{n_a}$ oraz $T_{n_b+1}^{n_b+1}$ nie są silnie testowalne.

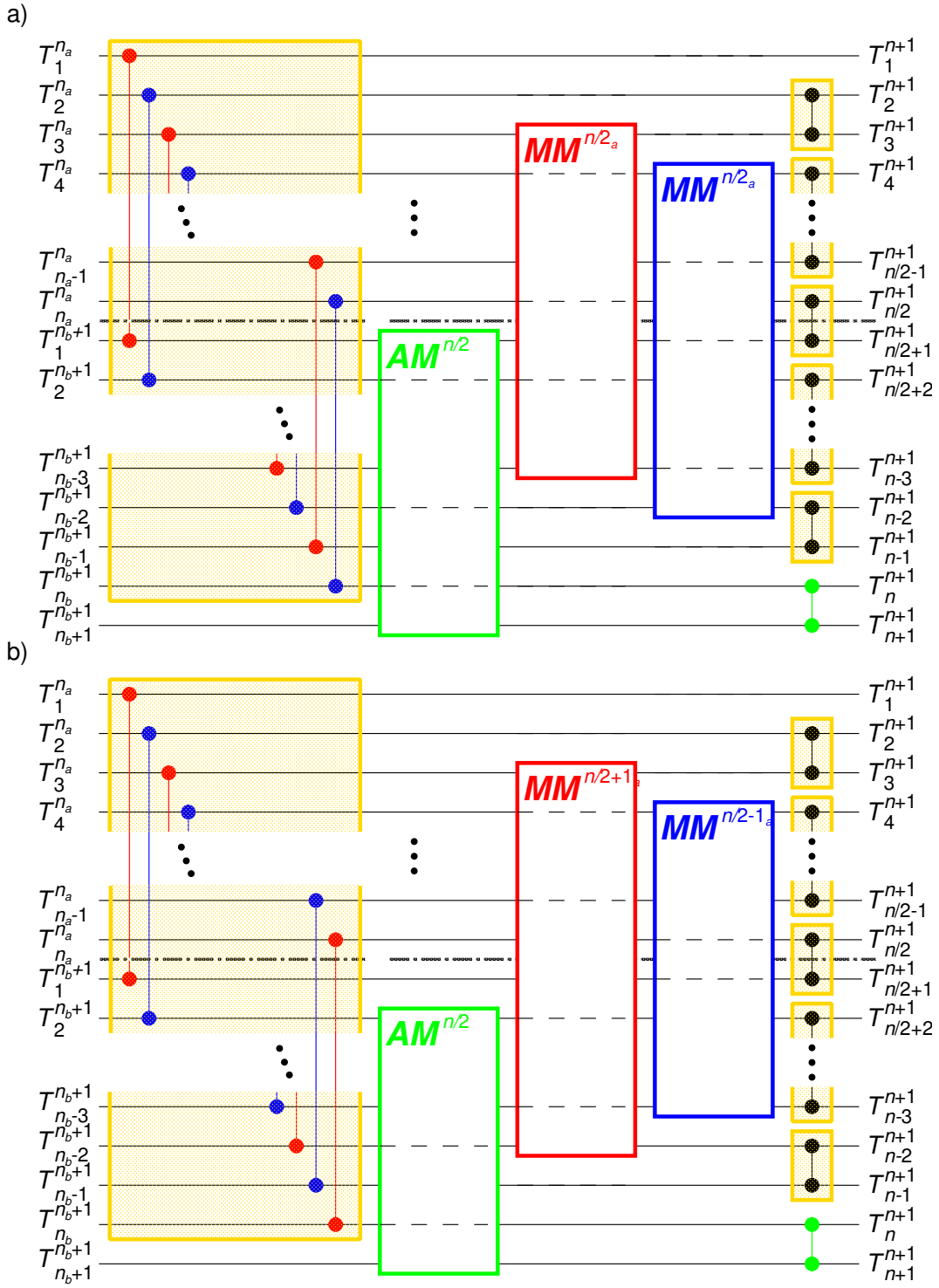
Sieć łącząca o nieparzystej liczbie linii $n = 2k - 1$ uzyskana przez usuwanie linii

Analizy, przeprowadzone przez nas w ramach badań do niniejszej pracy, których szczegóły tutaj pomijamy, wykazały, iż własności sieci łączącej Batchera OE o nieparzystej liczbie wejść $n = 2k+1$ poważnie ograniczają możliwość wprowadzania modyfikacji prowadzących do uzyskania własności silnej testowalności uszkodzeń typu opóźnienie ścieżki. Jak zaznaczyliśmy powyżej, występuje wiele ścieżek przechodzących przez linię $d_{i,n+1}$. Modyfikacja, która w naszym przypadku polega na eliminacji ścieżek poprzez podział, a następnie scalanie linii jest bardzo nieczytelna, zaś jej przejrzysty zapis analityczny lub algorytmiczny był praktycznie niemożliwy. Dlatego rozważymy tutaj inną klasę sieci łączących o nieparzystej liczbie wejść $n = 2k - 1$, realizowaną z sieci łączących Batchera OE o parzystej liczbie wejść $n = 2k$ poprzez usunięcie jednej linii.

Własność 5.1 **[35]** *Niech $C(n_a, n_b)$ oznacza liczbę komparatorów w sieci łączącej Batchera OE MN^n, gdzie $n = n_a + n_b$ oraz $n_a = n_b$. Wtedy $C(n_a, n_b - 1) = C(n_a, n_b) - 1$*

Własność 5.2 **[58]** *Niech $M(n_a, n_b)$ oznacza minimalną liczbę komparatorów w sieci łączącej MN^n, gdzie $n = n_a + n_b$ oraz $n_a = n_b$. Wtedy $M(n_a, n_b) = M(n_a, n_b - 1) + 1$*

Na rys. 5.12 przedstawiono modyfikację ogólnego przypadku sieci łączącej Batchera OE o parzystej liczbie wejść $n = 2k$ poprzez usunięcie jednej linii. Łatwo zauważyć, że sieć łącząca o parzystej liczbie wejść w wyniku usunięcia jednej linii traci dokładnie jeden komparator, natomiast z własności 5.1 wynika, że sieć łącząca powstała w wyniku takiej modyfikacji ma taką samą liczbę



Rysunek 5.11: Sieć Batchera OE o nieparzystej liczbie wejść $n = 2k + 1$ po wydzieleniu warstwy wejściowej dla: a) k parzystego, b) k nieparzystego

komparatorów jak sieć łącząca generowana zgodnie z algorytmem Batchera OE. Dodatkowo wykazano (własność 5.2), że własność taką ma również minimalny układ łączący. Na tej podstawie zakładamy, że modyfikacja sieci łączącej uzyskanej przez usunięcie linii daje układ o takiej samej złożoności jak modyfikacja sieci łączącej Batchera OE.

W trakcie dalszej analizy sieci łączącej MN^n o nieparzystej liczbie wejść $n = 2k-1$ uzyskanej przez usunięcie linii będziemy korzystać z tych samych oznaczeń co dla sieci o parzystej liczbie wejść $n = 2k$ (rys. 5.7).

5.3 Silna testowalność uszkodzeń typu opóźnienie ścieżki w sieci łączącej Batchera

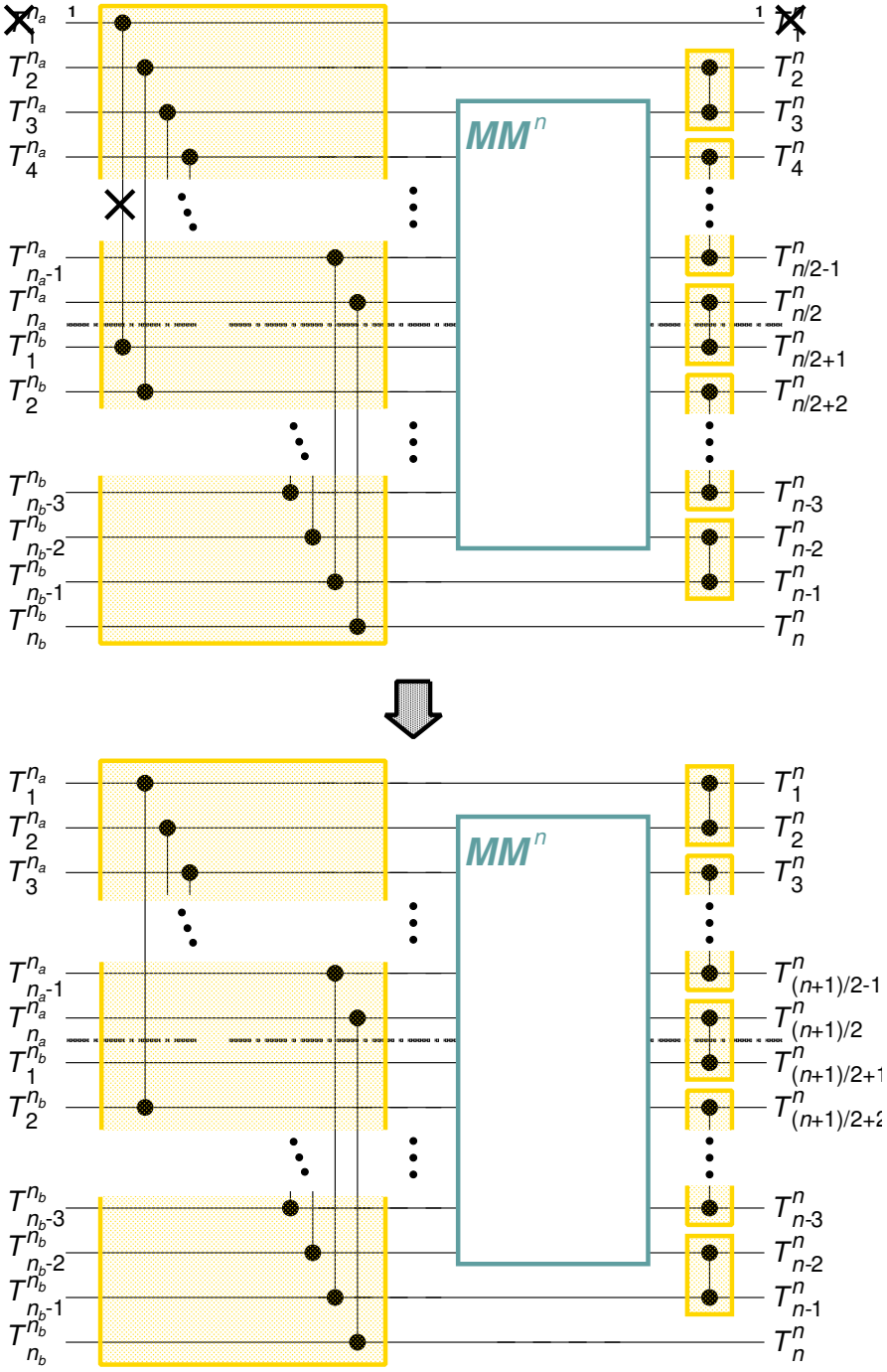
W rozdziale 4 zdefiniowaliśmy ogólne własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych. Rozdział poniższy przedstawia analizę własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w sieci łączącej Batchera OE. Analizę przeprowadzimy dla sieci łączących o parzystej liczbie wejść.

5.3.1 Analiza rozkładu ścieżek w sieci łączącej Batchera

Własność 5.3 *Niech będą dane sieci łączące MN^{n_a} oraz MN^{n_b}. Jeżeli w każdej z sieci MN^{n_a}, MN^{n_b} z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka, wtedy w sieci łączącej MN^n zrealizowanej z użyciem sieci MN^{n_a}, MN^{n_b}, według algorytmu sieci łączącej Batchera OE (rys. 5.1), z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka.*

Dowód

Rozważmy wyjście T_i^n sieci łączącej MN^n. Wyjście to jest wyjściem dwuwejściowej bramki AND lub OR komparatora. Każde z wejść tej bramki jest podłączone do oddzielnej podsieci łączącej, jedno do podsieci MN^{n_a}, drugie do podsieci MN^{n_b}. Ponieważ zbiory linii prowadzących do podsieci łączących są rozłączne, zatem dowolne wejście sieci MN^n jest podłączone do dokładnie jednej z sieci łączących MN^{n_a} lub MN^{n_b}. Na mocy założenia, w podsieciach łączących z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka. Zatem dwa rozłączne zbiory wejść generują dwa rozłączne zbiory ścieżek, które zbiegają się w bramce, której wyjście



Rysunek 5.12: Realizacja sieci sortującej o nieparzystej liczbie wejść poprzez usunięcie linii

jest wyjściem T_i^n. Rozumowanie to jest prawdziwe dla każdego wyjścia sieci łączącej MN^n poza wyjściami T_1^n oraz T_n^n, które są dołączone do tylko jednej z sieci łączących MN^{n_a} lub MN^{n_b}. Zatem na mocy założenia może do nich prowadzić co najwyżej po jednej ścieżce z dowolnego wyjścia pierwotnego. Stąd w całej sieci łączącej MN^n z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka. ■

Własność 5.4 *W sieci łączącej MN^n zrealizowanej zgodnie z algorytmem Batchera OE z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka.*

Dowód

Zgodnie z definicją sieci łączącej Batchera OE, każda z podsieci łączących jest zrealizowana rekursywnie według algorytmu łączenia nieparzysto-parzystego. Założenie o występowaniu co najwyżej jednej ścieżki można teraz rekursywnie przenieść na wszystkie podsieci. Na najniższym poziomie występuje sieć łącząca w postaci jednego komparatora lub po prostu pojedyncza linia. W obu tych przypadkach bezpośrednio widać, że z dowolnego wejścia do dowolnego wyjścia występuje co najwyżej jedna ścieżka. W sieci łączącej zrealizowanej zgodnie z algorytmem łączenia nieparzysto-parzystego występuje co najwyżej jedna ścieżka (własność 5.3). Zatem, własność tą indukcyjnie przenosimy na wszystkie podsieci, aż do sieci łączącej MN^n. Stąd w sieci łączącej MN^n z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka. ■

5.3.2 Analiza warunków pobudzenia linii w sieci łączącej Batchera

Sieć łącząca służy do łączenia dwóch uporządkowanych ciągów binarnych w jeden ciąg uporządkowany. Dodatkowo, w sieci łączącej Batchera OE o parzystej liczbie wejść, obydwa ciągi mają taką samą długość. Zatem na wejście sieci łączącej nie są podawane wszystkie kombinacje bitów, lecz jedynie pewien ściśle określony podzbiór wektorów, których niektóre pary stanowią wzorce testowe V_{in} (def. 4.2). Wynika stąd, że funkcje logiczne poszczególnych linii w sieci można rozpatrywać biorąc pod uwagę jedynie podany zbiór wektorów wejściowych. Idąc dalej tym torem rozważań, określamy zbiór wektorów, które mogą pojawić się za układem selekcji wstępnej, na liniach $W_{in} = \{u_1^{n_a}, \ldots, u_{n_a}^{n_a}, u_1^{n_b}, \ldots, u_{n_b}^{n_b}\}$. Wektor W_{in} jest równoważny wektorowi linii na

poziomie 1, czyli $W_{in} = d_1 = \{d_{1,1}, d_{1,2}, \ldots, d_{1,n}\}$. Podobnie, wektor wejść pierwotnych sieci łączącej jest równoważny wektorowi linii na poziomie 0, czyli $\{T_1^{n_a}, \ldots, T_{n_a}^{n_a}, T_1^{n_b}, \ldots, T_{n_b}^{n_b}\} = d_0 = \{d_{0,1}, d_{0,2}, \ldots, d_{0,n}\}$.

Zauważmy, że każdy wektor wejściowy T_{in} sieci łączącej ma postać

$$T_{in} = \underbrace{\underbrace{1\ldots1}_{j}\underbrace{0\ldots0}_{n_a-j}}_{T^{n_a}}\underbrace{\underbrace{1\ldots1}_{k}\underbrace{0\ldots0}_{n_b-k}}_{T^{n_b}}, \ 1 \leqslant j \leqslant n_a, \ 1 \leqslant k \leqslant n_b. \tag{5.1}$$

Własność 5.5 *Dla każdego* $1 \leqslant j \leqslant n_a$ *i dla każdego* $1 \leqslant k \leqslant n_b$ *takiego, że* $k \leqslant j$*, każdy wektor* W_{in} *w sieci łączącej* MN^n *ma postać*

$$W_{in} = \underbrace{\underbrace{1\ldots1}_{j}\underbrace{0\ldots0}_{n_a-j}}_{u^{n_a}}\underbrace{\underbrace{1\ldots1}_{k}\underbrace{0\ldots0}_{n_b-k}}_{u^{n_b}}. \tag{5.2}$$

Dowód

Przypomnijmy, że linie $u_1^{n_a} \ldots u_{n_a}^{n_a}$ są wyjściami minimalnymi komparatorów zawartych w układzie selekcji wstępnej, zaś linie $u_1^{n_b} \ldots u_{n_b}^{n_b}$ są wyjściami maksymalnymi. Rozważmy teraz dwa przypadki wektora T_{in} na wejściu sieci łączącej MN^n, w którym: (i) $j \geqslant k$ oraz (ii) $j < k$.

W przypadku (i) na wejściach k komparatorów występuje kombinacja 11, zaś na wejściach $j-k$ komparatorów występuje kombinacja 10. Ponieważ tam gdzie występuje kombinacja 10 wartość 1 musi pojawić się na wyjściach minimalnych komparatorów, zatem na wyjściach minimalnych będzie $j - k + k = j$ jedynek, a na wyjściach maksymalnych k jedynek. Ponieważ $k \leqslant j$, warunki własności w przypadku (i) są spełnione.

W przypadku (ii) na wejściach j komparatorów występuje kombinacja 11, zaś na wejściach $k-j$ komparatorów kombinacja 01. Ponieważ tam gdzie występuje kombinacja 01 wartość 1 musi pojawić się na wyjściach minimalnych komparatorów, zatem na wyjściach minimalnych będzie $k-j+j = k$ jedynek, a na wyjściach maksymalnych j jedynek. Ponieważ $j < k$, warunki własności w przypadku (ii) są spełnione. ■

Teraz określimy zależności pomiędzy liniami na poziomie 1, tj. za układem selekcji wstępnej, a liniami na dowolnym poziomie sieci.

Własność 5.6 *Na dowolnym poziomie* $i \geqslant 1$ *sieci łączącej są spełnione następujące nierówności:*

1. $(\forall\ 1 \leqslant j \leqslant \frac{n}{2})(\forall\ 1 \leqslant k \leqslant j)\ d_{i,k} \geqslant d_{1,j}$;

2. $(\forall\ \frac{n}{2}+1 \leqslant j \leqslant n)(\forall\ j \leqslant k \leqslant n)\ d_{i,k} \leqslant d_{1,j}$.

Dowód

Ponieważ każdy komparator sieci łączącej ma węzeł minimalny podłączony do linii $d_{i,a}$, zaś węzeł maksymalny podłączony do linii $d_{i,b}$, gdzie $a < b$, przepływ jedynek na liniach $d_{i,j}$ może być tylko w kierunku malejących wartości j, zaś przepływ zer na liniach $d_{i,k}$ może być tylko w kierunku rosnących wartości k.

Rozważmy teraz taki wektor binarny na poziomie d_1, w którym na liniach $\{d_{1,1} \ldots d_{1,j}\}$ $(1 \leqslant j \leqslant \frac{n}{2})$ występują jedynki, a na pozostałych liniach wartości D. Wtedy na wejściach każdego komparatora $K^i_{a,b}$ wystąpi jedna z trzech kombinacji 11, $1D$ albo DD, i taka sama kombinacja pojawi się na wyjściach tego komparatora. Stąd postać wektora na każdym poziomie będzie taka sama, czyli $d_1 = d_2 = \ldots = d_{i_{\max}}$. Podobnie, rozważmy wektor binarny na poziomie d_1, w którym na liniach $\{d_{1,j} \ldots d_{1,n}\}$ $(\frac{n}{2} \leqslant j \leqslant n)$ występują zera, a na pozostałych liniach wartości D. Wtedy na wejściach każdego komparatora $K^i_{a,b}$ wystąpi jedna z trzech kombinacji 00, $D0$ albo DD, i taka sama kombinacja pojawi się na wyjściach tego komparatora. Stąd postać wektora na każdym poziomie będzie taka sama, czyli $d_1 = d_2 = \ldots = d_{i_{\max}}$. Natomiast zgodnie z własnością 5.5: (i) jedynka na linii $d_{1,j}$ $(j \leqslant \frac{n}{2})$ implikuje jedynki na liniach młodszych od j, czyli $d_{1,k}$ $(k \leqslant j)$, zatem implikuje wektor jedynek na liniach $\{d_{1,1} \ldots d_{1,j}\}$, oraz (ii) zero na linii $d_{1,j}$ $(j \geqslant \frac{n}{2}+1)$ implikuje zera na liniach starszych od j, czyli $d_{1,k}$ $(k \geqslant j)$, zatem implikuje wektor zer na liniach $\{d_{1,j} \ldots d_{1,n}\}$. Z kolei, jak wykazaliśmy powyżej, wektory odpowiednio jedynek i zer propagują się przez wszystkie poziomy sieci. Stąd jedynka na linii $d_{i,j}$ $(1 \leqslant j \leqslant \frac{n}{2})$ implikuje jedynki na wszystkich poziomach na wszystkich liniach o indeksach od 1 do j, zaś zero na linii $d_{i,j}$ nie wyklucza jedynki na linii $d_{i,k}$ $(k < j)$. Stąd teza w pkt. 1. Zaś zero na linii $d_{i,j}$ $(\frac{n}{2} \leqslant j \leqslant n)$ implikuje zera na wszystkich poziomach na wszystkich liniach o indeksach od 1 do n, zaś jedynka na linii $d_{i,j}$, nie wyklucza zera na linii $d_{i,k}$ $(k > j)$. Stąd teza w pkt. 2. ■

Własność 5.7 *Funkcja linii $d_{i,j}$, gdzie $1 \leqslant j \leqslant n$ oraz $i \geqslant 1$, jest równoważna wyrażeniu*

$$d_{i,j} = \begin{cases} d_{1,j} + d^0_{i,j} + \sum_{k=j+1}^{\frac{n}{2}} d_{1,k} & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ d_{1,j} \cdot d^0_{i,j} \cdot \Pi_{k=\frac{n}{2}+1}^{j-1} d_{1,k} & dla\ \frac{n}{2}+1 \leqslant j \leqslant n, \end{cases} \tag{5.3}$$

gdzie:

$$d^0_{i,k} \neq f(A),\ A = \begin{cases} \{d_{1,k} : j \leqslant k \leqslant \frac{n}{2}\} & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ \{d_{1,k} : \frac{n}{2} \leqslant k \leqslant j\} & dla\ \frac{n}{2}+1 \leqslant j \leqslant n, \end{cases} \tag{5.4}$$

oraz

$$d_{0,j} = \begin{cases} 0 & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ 1 & dla\ \frac{n}{2}+1 \leqslant j \leqslant n. \end{cases} \tag{5.5}$$

Dowód

Wartość linii $d_{i,j}$ dla $i > 1$ jest funkcją wszystkich linii na poziomie 1, czyli $d_{i,j} = f(d_{1,1} \ldots d_{1,n})$. We własności 5.6 wykazano, iż dla linii $d_{1,j}$, $1 \leqslant j \leqslant \frac{n}{2}$, wartość 1 implikuje jedynki na linii $d_{i,j}$ oraz na wszystkich liniach $d_{i,1}, \ldots, d_{i,j}$, na wszystkich poziomach i większych lub równych 1. Stąd można zapisać funkcję linii $d_{i,j}$ $(j \leqslant \frac{n}{2})$ w postaci wzoru $d_{i,j} = d_{1,j} + \sum_{k=j+1}^{\frac{n}{2}} d_{1,k} + f(d_{1,1}, \ldots, d_{1,j-1}, d_{1,\frac{n}{2}+1}, \ldots, d_{1,n})$. Podobnie, wartość 0 na linii $d_{1,j}$ dla $j \geqslant \frac{n}{2}+1$ implikuje zera na linii j oraz na wszystkich liniach $d_{i,j}, \ldots, d_{i,n}$, na wszystkich poziomach większych lub równych 1. Tak samo jak powyżej, funkcję linii można zapisać w postaci $d_{i,j} = d_{1,j} \cdot \Pi_{k=\frac{n}{2}+1}^{j-1} d_{1,k} \cdot f(d_{1,1}, \ldots, d_{1,\frac{n}{2}}, d_{1,j+1}, \ldots, d_{1,n})$. ■

Własność 5.8 *Funkcja linii $d_{i,j}$, gdzie $1 \leqslant j \leqslant n$ oraz $i \geqslant 1$, jest równoważna wyrażeniu*

$$d_{i,j} = \begin{cases} d_{1,j} + d^0_{i,j} & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ d_{1,j} \cdot d^0_{i,j} & dla\ \frac{n}{2}+1 \leqslant j \leqslant n, \end{cases} \tag{5.6}$$

gdzie:

$$d^0_{i,k} \neq f(A),\ A = \begin{cases} \{d_{1,k} : j \leqslant k \leqslant \frac{n}{2}\} & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ \{d_{1,k} : \frac{n}{2} \leqslant k \leqslant j\} & dla\ \frac{n}{2}+1 \leqslant j \leqslant n. \end{cases} \tag{5.7}$$

Dowód

Ponieważ wartość 1 na linii $d_{1,j}$ występuje zawsze wtedy, gdy: (i) wyrażenie sumy w (5.3) jest równe 1, lub (ii) $d_{1,j}$ jest równe 1, zatem całą sumę można w tym wzorze pominąć, co daje $d_{i,j} = d_{1,j} + f(d_{1,1}, \ldots, d_{1,j-1}, d_{1,\frac{n}{2}+1}, \ldots, d_{1,n})$. Podobnie, wartość 0 na linii $d_{1,j}$ występuje zawsze wtedy, gdy: (i) wyrażenie iloczynu w (5.3) jest równe 0, lub (ii) $d_{1,j}$ jest równe 0, zatem cały iloczyn można pominąć, co daje $d_{i,j} = d_{1,j} \cdot f(d_{1,1}, \ldots, d_{1,\frac{n}{2}}, d_{1,j+1}, \ldots, d_{1,n})$. Przepisując (5.3) otrzymujemy zatem (5.6). ■

Innymi słowy, wartość linii $d_{1,j}$ dla $1 \leqslant j \leqslant \frac{n}{2}$ nie zależy od wartości linii $d_{1,k}$ $(j < k \leqslant \frac{n}{2})$, zaś wartość linii $d_{1,j}$ dla $\frac{n}{2}+1 \leqslant j \leqslant n$ nie zależy od wartości linii $d_{1,k}$ $(\frac{n}{2}+1 \leqslant k < j)$.

Własność 5.9 *Funkcja linii T_j^n, gdzie $1 \leqslant j \leqslant n$, jest równoważna wyrażeniu*

$$T_j^n = \begin{cases} d_{1,j} + d_{i,j}^0 & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ d_{1,j} \cdot d_{i,j}^0 & dla\ \frac{n}{2}+1 \leqslant j \leqslant n, \end{cases} \tag{5.8}$$

gdzie:

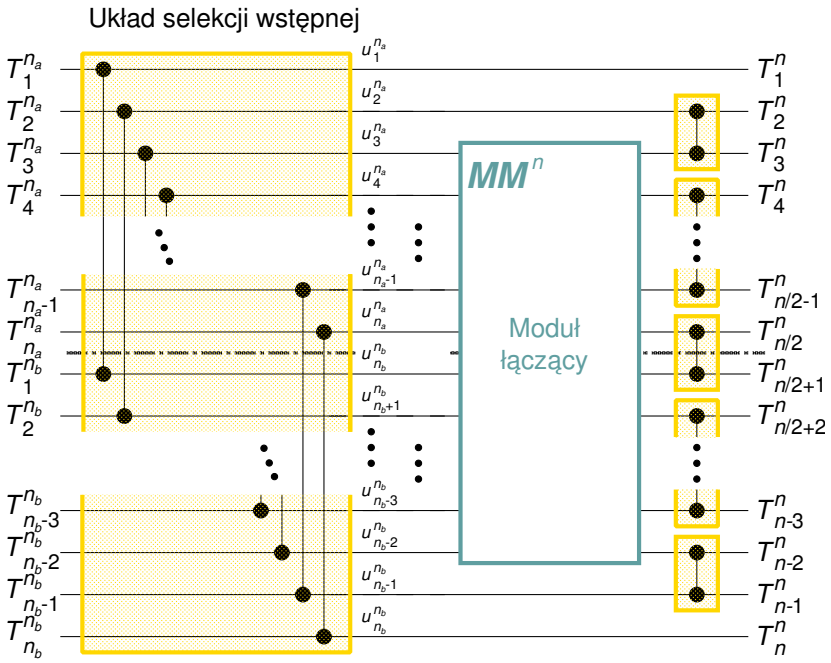
$$d_{i,k}^0 \neq f(A),\ A = \begin{cases} \{d_{1,k} : j \leqslant k \leqslant \frac{n}{2}\} & dla\ 1 \leqslant j \leqslant \frac{n}{2} \\ \{d_{1,k} : \frac{n}{2} \leqslant k \leqslant j\} & dla\ \frac{n}{2}+1 \leqslant k \leqslant n. \end{cases} \tag{5.9}$$

Dowód

Wyjście pierwotne T_j^n jest równoważne linii $d_{i\max,j}$ na najwyższym poziomie sieci łączącej MN^n, stąd dowód jest taki sam jak dowód własności 5.8. ■

5.3.3 Dyskusja własności silnej testowalności sieci łączącej

Wykazaliśmy, że w sieci łączącej Batchera OE z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka. W rozdziale 4 udowodniliśmy, iż dowolny wielowyjściowy układ progowy zrealizowany jako całkowicie rozgałęziony ma własność silnej testowalności uszkodzeń typu opóźnienie ścieżki, jeżeli każda sieć łącząca wchodząca w jego skład ma własność RG. Definicja własności RG wymaga, aby sieć łącząca miała zarówno własność RG jak i SP. Zatem, celem niniejszej analizy będzie określenie warunków, przy spełnieniu których sieć łącząca Batchera OE będzie mogła zostać uznana za spełniającą kryteria wymagane w definicji własności SP (def. 4.1) lub też znalezienie takiej modyfikacji sieci, która sprawi, że sieć wynikowa będzie spełniała kryteria własności SP i RG. Ponieważ w sieci łączącej Batchera OE z dowolnego wejścia do dowolnego wyjścia prowadzi co najwyżej jedna ścieżka, istnieje pewien zbiór ścieżek, które są silnie testowalne, oraz zbiór ścieżek, które nie są silnie testowalne. Zgodnie z kryteriami własności SP (def. 4.1), w układzie mającym własność SP mogą istnieć jedynie pojedyncze ścieżki do ściśle określonych zbiorów wyjść. Ponieważ w obecnych rozważaniach bazowym elementem jest poziom 1 sieci łączącej Batchera OE (linie $d_{1,1} \ldots d_{1,n}$ lub $u_1^{n_a} \ldots u_{n_a}^{n_a}, u_1^{n_b} \ldots u_{n_b}^{n_b}$), sformułujemy kryteria własności SP dla modułu MM^n (rys. 5.13).



Rysunek 5.13: Sieć łącząca Batchera OE MN^n zrealizowana jako złożenie układu selekcji i uzupełniającego modułu łączącego MM^n

Własność 5.10 *W sieci łączącej MN^n mającej własność SP, zrealizowanej jako złożenie układu selekcji i uzupełniającego modułu łączącego MM^n (rys. 5.13), występują jedynie następujące ścieżki:*

1. *przechodzące przez linię $u_j^{n_a}$ do wyjść $T_j^n \ldots T_{2j-1}^n$, dla $1 \leqslant j \leqslant \frac{n}{2}$ oraz*
2. *przechodzące przez linię $u_j^{n_b}$ do wyjść $T_{2j}^n \ldots T_{j+\frac{n}{2}}^n$, dla $1 \leqslant j \leqslant \frac{n}{2}$.*

Dowód

Wyjścia układu selekcji wstępnej wyznaczają granice, w których będzie mieścił się wynik łączenia. Dla przykładu, j jedynek na wyjściach układu selekcji na liniach u^{n_a} oznacza, iż na wyjściu układu selekcji na liniach u^{n_b} może być co najwyżej j jedynek. Analogicznie, k zer na liniach u^{n_b} oznacza, iż na liniach u^{n_a} może być co najwyżej j zer. Z drugiej strony, j jedynek na liniach u^{n_a} oznacza, że wynik będzie miał co najmniej j jedynek, zaś k zer na liniach u^{n_b} oznacza, że wynik będzie miał co najmniej k zer. Stąd o liniach u^{n_a} możemy powiedzieć, że z linii $u_j^{n_a}$, $1 \leqslant j \leqslant \frac{n}{2}$ prowadzą ścieżki do wyjść T_k^n $(k \geqslant j)$, zaś o liniach u^{n_b} można powiedzieć, że z linii

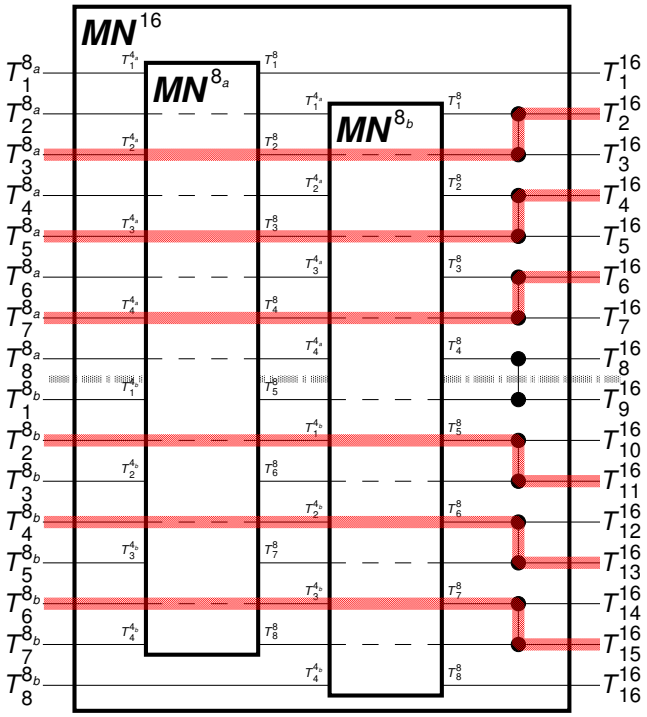
$u_j^{n_b}$, $1 \leqslant j \leqslant \frac{n}{2}$ prowadzą ścieżki do wyjść T_k^n $(k \leqslant j + \frac{n}{2})$. Tym samym określiliśmy jednostronne granice docelowych przedziałów, w których będzie mieścił się wynik łączenia.

Teraz określimy ograniczenie z drugiej strony. Ponieważ na mocy założenia układ ma własność SP, zbiory wyjść do których prowadzą ścieżki z linii $u_j^{n_a}$ oraz $u_j^{n_b}$ są rozłączne. Rozważmy linię $u_j^{n_a}$. Wartość 1 na tej linii oznacza, iż na wejściu jednej z sieci T^{n_a} lub T^{n_b} wystąpiło dokładnie j jedynek. Przyjmijmy teraz, że na wyjściu sieci odpowiednio T^{n_b} lub T^{n_a} występuje kolejno od 0 do i jedynek. Wtedy na liniach u^{n_a} liczba jedynek wynosi dokładnie j, zaś na liniach u^{n_b} występuje od 0 do $j-1$ jedynek. Na wyjściach sieci $T_1^n, \ldots, T_n^n$ występuje wtedy odpowiednio od j do $2j-1$ jedynek.

Analogicznie, jedynka na linii $u_j^{n_b}$ oznacza, iż równocześnie na obydwu liniach $T_j^{n_b}$ oraz $T_j^{n_a}$ występują jedynki. Przyjmijmy teraz, że na wejściu sieci T^{n_a} lub T^{n_b} występuje kolejno od j do $\frac{n}{2}$ jedynek, zaś na wejściu sieci odpowiednio T^{n_b} lub T^{n_a} jedynek. Wtedy na liniach u^{n_b} liczba jedynek wynosi dokładnie j, zaś na liniach u^{n_a} występuje od j do $\frac{n}{2}$ jedynek. Na wyjściach sieci $T_1^n, \ldots, T_n^n$ występuje wtedy odpowiednio od $2j$ do $j + \frac{n}{2}$ jedynek. Zatem przez linię $u_j^{n_a}$ prowadzą ścieżki do wyjść od $T_j^n, \ldots, T_{2j-1}^n$, natomiast z przez linię $u_j^{n_b}$ prowadzą ścieżki do wyjść $T_{2j}^n, \ldots, T_{j+\frac{n}{2}}^n$, co należało dowieść. ■

Jak wykazaliśmy, w sieci łączącej Batchera OE występują ścieżki określone w definicji własności SP. Gdyby były to jedyne ścieżki w tej sieci, sieć miałaby własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. Jednak poza ścieżkami określonymi w definicji własności SP, występują także ścieżki prowadzące do innych wyjść. Wykazaliśmy, że w sieci łączącej Batchera OE ścieżki te są pojedyncze. Istnienie takich ścieżek wykażemy na przykładzie. Następnie udowodnimy, że segmenty (rozdz. 2.3.1) należące do tych ścieżek także nie są silnie testowalne. Kontynuując, rozszerzymy dowody braku własności silnej testowalności na segmenty łączące wejścia pierwotne z wyjściami pierwotnymi sieci, będące w istocie ścieżkami.

Własność 5.11 *Dla wzorców wejściowych postaci V_{in} (def. 4.2), wzorce na liniach $d_{1,1} \ldots d_{1,n}$ (za układem selekcji) mają postać*



Rysunek 5.14: Niektóre nietestowalne ścieżki w sieci łączącej Batchera OE o 16 wejściach

$$U_{in} = \begin{cases} \underbrace{1\ldots1}_{j} * \underbrace{0\ldots0}_{n_a-j-1} \underbrace{1\ldots1}_{k} \underbrace{0\ldots0}_{n_b-k}, & 1 \leqslant j \leqslant n_a, 1 \leqslant k \leqslant n_b], j \geqslant k, lub \\ \underbrace{1\ldots1}_{j} \underbrace{0\ldots0}_{n_a-j} \underbrace{1\ldots1}_{k} * \underbrace{0\ldots0}_{n_b-k-1}, & 1 \leqslant j \leqslant n_a, 1 \leqslant k \leqslant n_b, j > k \end{cases} \tag{5.10}$$

Dowód

Dowód jest analogiczny do dowodu własności 5.5. ■

Przykład 5.1 *Na rys. 5.14 przedstawiono sieć T^{16} zestawioną z sieci T^{8_a} i T^{8_b} z zaznaczonymi niektórymi spośród nietestowalnych ścieżek. Można zauważyć, iż wejście $T_3^{8_a}$ jest podłączone do wejścia $T_2^{4_a}$*[(a)] *sieci MN^{8_b}. Z wejścia tego istnieje ścieżka do wyjścia T_2^{8}*[(b])*, a następnie do wyjścia T_2^{16}. Zatem istnieje ścieżka z wejścia $T_3^{8_a}$ do wyjścia T_2^{16}.*

[(a)] Zgodnie z algorytmem łączenia nieparzysto-parzystego

[(b)] Przez komparator wyjściowy

Własność 5.12 *1. Żaden segment biegnący z linii $d_{1,j}$ do linii $d_{i,k}$, gdzie $i \geqslant 1$, $1 \leqslant j \leqslant \frac{n}{2}$ oraz $1 \leqslant k < j-1$, nie jest silnie testowalny dla wzorców wejściowych postaci U_{in};*

2. żaden segment biegnący z linii $d_{1,j}$ do linii $d_{i,k}$, gdzie $i \geqslant 1$, $\frac{n}{2}+1 \leqslant j \leqslant n$ oraz $j+1 < k \leqslant n$, nie jest silnie testowalny dla wzorców wejściowych postaci U_{in}.

Dowód

1. Niech będzie dany segment przebiegający przez komparator $K^i_{k,j}$, $1 \leqslant k < j \leqslant \frac{n}{2}$. Z własności 5.6 wynika, że przy dowolnej zmianie sygnału na linii $d_{i,j}$ na linii $d_{i+1,j-k}$ występuje jedynka, czyli na wejściu bramki OR komparatora $K^i_{k,j}$ występuje stała wartość sterująca. Stąd zdarzenie na linii $d_{i,j}$ nie może propagować się do linii $d_{i+1,k}$, a zatem segment ten nie jest silnie testowalny.

2. Niech będzie dany segment przebiegający przez komparator $K^i_{j,k}$, $\frac{n}{2}+1 \leqslant j < k \leqslant n$. Z własności 5.6 wynika, że przy dowolnej zmianie sygnału na linii $d_{i,j}$ na linii $d_{i+1,k}$ występuje zero, czyli na wejściu bramki AND komparatora $K^i_{j,k}$ występuje stała wartość sterująca. Stąd zdarzenie na linii $d_{i,j}$ nie może propagować się do linii $d_{i+1,k}$, a zatem segment ten nie jest silnie testowalny. ■

Własność 5.13

1. Żaden segment biegnący z linii $d_{1,j}$ do wyjścia T^n_k, gdzie $1 < j \leqslant \frac{n}{2}$ i $1 \leqslant k < j$, nie jest silnie testowalny dla wzorców wejściowych postaci U_{in};

2. żaden segment biegnący z linii $d_{1,j}$ do linii T^n_k, gdzie $\frac{n}{2}+1 \leqslant j < n$ i $j < k \leqslant n$, nie jest silnie testowalny dla wzorców wejściowych postaci U_{in}.

Dowód

Linie najwyższego poziomu sieci łączącej $d_{i_{\max}}$ są tożsame z jej wyjściami pierwotnymi, czyli $\{d_{i_{\max},1}, \ldots, d_{i_{\max},n}\} = \{T^n_1, \ldots, T^n_n\}$. Zatem dowód jest taki sam jak dowód własności 5.12. ■

Twierdzenie 5.1 *Niech $v = \{a, b\}$, a V_{in} oznacza wzorzec testowy według def. 4.2. Sieć łącząca Batchera OE ma następujące własności:*

1. *żadna ścieżka prowadząca z wejścia $T_j^{n_v}$ do wyjścia T_k^n, gdzie $1 \leqslant k < j \leqslant\!< \frac{n}{2}$, nie jest silnie testowalna dla wzorców wejściowych postaci V_{in};*

2. *żadna ścieżka prowadząca z wejścia $T_j^{n_v}$ do wyjścia T_k^n, gdzie $\frac{n}{2} < j + \frac{n}{2} < k \leqslant n$, nie jest silnie testowalna dla wzorców wejściowych postaci V_{in}.*

Dowód

Na podstawie własności 5.11 wiemy, że wzorzec wejściowy postaci V_{in} na liniach poziomu 1 sieci łączącej przyjmuje postać U_{in}, przy czym warunki silnej testowalności są spełnione dla komparatorów układu selekcji (rozdzielających poziom 0 i poziom 1).

1. Z linii wejściowych $T_j^{n_v}$ prowadzą dwa segmenty do linii $d_{1,j}$, które leżą na początku ścieżek prowadzących do linii T_k^n ($1 \leqslant k < j \leqslant \frac{n}{2}$). Z własności 5.13 wynika, że segmenty prowadzące od linii $d_{1,j}$ do wyjść T_k^n, stanowiące dalszy ciąg ścieżek prowadzących z wejścia $T_j^{n_v}$ do wyjścia T_k^n, nie są silnie testowalne dla wzorców postaci U_{in}. Zatem ścieżki prowadzące od wejść $T_j^{n_a}$ oraz $T_j^{n_b}$ do wyjść T_k^n, nie są silnie testowalne dla wzorców postaci V_{in}.

2. Z linii wejściowych $T_j^{n_v}$ prowadzą również dwa segmenty do linii $d_{1,j+\frac{n}{2}}$, które leżą na początku ścieżek prowadzących do linii T_k^n ($\frac{n}{2} < j+\frac{n}{2} < k \leqslant n$). Z własności 5.13 wynika, że segmenty prowadzące od linii $d_{1,j}$ do wyjść T_k^n, stanowiące dalszy ciąg ścieżek z wejścia $T_j^{n_v}$ do wyjścia T_k^n, nie są silnie testowalne dla wzorców postaci U_{in}. Zatem ścieżki prowadzące od wejść $T_j^{n_a}$ oraz $T_j^{n_b}$ do wyjść T_k^n nie są silnie testowalne dla wzorców postaci V_{in}. ■

5.4 Modyfikacja sieci łączącej

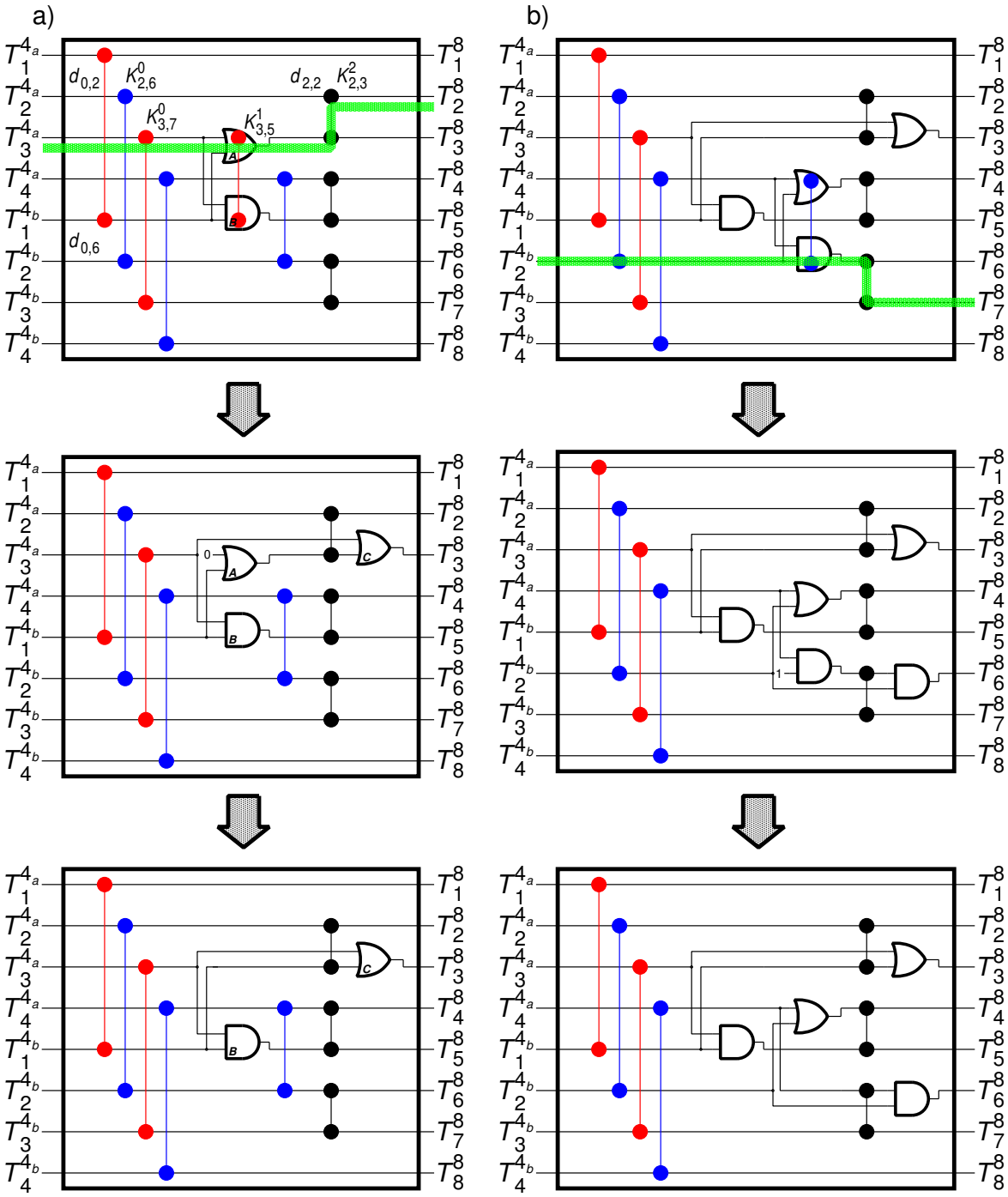
W rozdziale 5.3 przedstawiliśmy analizę własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w sieci łączącej Batchera OE, która wykazała, że sieć ta nie jest układem SP. Wykazaliśmy również, że wielowyjściowy układ progowy zrealizowany w postaci sieci sortującej Batchera OE nie jest silnie testowalny. Natomiast w rozdziale 4 wykazaliśmy, że wielowyjściowy układ progowy zrealizowany jako całkowicie rozgałęziony jest silnie testowalny, jeżeli sieci łączące mają własność SP. W poniższym rozdziale pokażemy, że istnieje modyfikacja sieci sortującej Batchera OE powodująca nabycie własności SP, a przy tym nie zwiększająca istotnie złożoności sieci.

5.4.1 Modyfikacja sieci o parzystej liczbie wejść

Prezentacja sposobu modyfikacji

We własności 5.7 wykazaliśmy, że funkcja linii $d_{i,j}$ $(1 \leqslant j \leqslant \frac{n}{2})$ może zostać przedstawiona jako suma linii $d_{1,j}$ oraz funkcji pozostałych linii z poziomu 1, z wyłączeniem linii $d_{1,k}$ $(k > j)$, czyli linii leżących powyżej linii j. Innymi słowy, linie o numerach mniejszych niż j, na każdym poziomie, nie są funkcją linii $d_{1,j}$. Bazując na tej obserwacji, można przedstawić modyfikację sieci łączącej Batchera OE nie zmieniającą jego własności. Polegać będzie ona na przecięciu linii $d_{1,j}$, a następnie dołączeniu jej poprzez bramki OR do bramek AND wszystkich komparatorów, które łączą linie $d_{i,j}$ i $d_{i,k}$. Na wejście linii $d_{1,j}$ należy podać zero. Przy wyjściu pierwotnym T_j^n odpowiadającym linii $d_{i\max,j}$, linię $d_{1,j}$ łączy się z linią $d_{i\max,j}$ przez bramkę OR. W ten sposób zachowane są funkcje linii leżących powyżej linii j, zaś linie leżące poniżej linii j nie są pobudzane przez linię $d_{1,j}$. Przykład modyfikacji pojedynczej linii w sieci ośmiowejściowej podano na rys. 5.15a i b.

Przykład 5.2 *Na rys. 5.15a i b została przedstawiona sieć łącząca MN^8 oraz nietestowalna ścieżka wiodąca z wejścia T_3^{4a} do wyjścia T_2^8. Komparator $K_{3,5}^1$ łączący linie $d_{1,3}$ i $d_{1,5}$ został przedstawiony w postaci bramek A i B. Brak własności silnej testowalności ścieżki prowadzącej z wejścia T_3^{4a} do wyjścia T_2^8 można wykazać na przykładzie wzorca testowego $V_t = \underbrace{11*0}_{T^{4a}}\underbrace{0000}_{T^{4b}}$. Widać tutaj, że wartość 1 na linii T_2^{4a} jest konieczna, aby na linii T_3^{4a} wystąpiło zdarzenie uruchamiające pożądaną zmianę na wejściu bramki OR komparatora $K_{2,3}^2$. Z drugiej zaś strony, wartość 1 na linii T_2^{4a} wymusza wartość sterującą na linii $d_{2,2}$, czyli na drugim wejściu bramki OR komparatora $K_{2,3}^2$. Wartość 1 na linii $d_{2,2}$ jest niezależna od wartości wejść T^{4b}, jako że wartość linii $d_{0,2}$ wynikająca z wejścia T_2^{4a} jest wartością sterującą dla bramki OR komparatora $K_{2,6}^0$. Stąd na wyjściu bramki OR komparatora $K_{2,6}^0$ wystąpi wartość 1 niezależnie od stanu linii $d_{0,6}$, czyli niezależnie od stanu wejścia T_2^{4b}. Stąd ścieżka od wejścia T_3^{4a} do wyjścia T_2^8 nie jest silnie testowalna. Należy tutaj zauważyć, iż nie istnieje taka kombinacja wejść, żeby zmiana stanu na wejściu T_3^{4a} spowodowała zmianę stanu na wyjściu T_2^8. Jak już powiedziano powyżej, zmiana stanu na linii T_3^{4a} wymusza stan 1 na linii T_2^{4a}, a przez to stan 1 na wejściach bramek OR komparatorów $K_{2,6}^0$ i $K_{2,3}^2$ oraz stan 1 na wyjściu T_2^8. Zatem zmiana sygnału lub wartość 1 na linii wejściowej T_3^{4a} pośrednio implikuje wartość 1 na wyjściu T_2^8, po ścieżce prowadzącej z wejścia T_2^{4a} (można rów-*



Rysunek 5.15: Przykład usuwania nietestowalnych ścieżek w sieci łączącej OE Batchera dla $n = 8$: a) krok 1, b) krok 2

nież sprawdzić, że ścieżka od wejścia $T_2^{4_a}$ do wyjścia T_2^8 jest silnie testowalna). Zatem ścieżka od wejścia $T_3^{4_a}$ do wyjścia T_2^8 nie musi wcale istnieć. Powyższe rozumowanie jest również zgodne z warunkami narzucanymi przez własność SP.

Możliwość usunięcia ścieżki od wejścia $T_3^{4_a}$ do wyjścia T_2^8 została przedstawiona na rys. 5.15a i b. Wejście bramki A z linii $d_{1,3}$ zostało zastąpione wartością 0. Linia $d_{1,3}$ pobudza bezpośrednio wyjście $d_{3,3}$, czyli wyjście T_3^8 przez bramkę C. Ponieważ na jedno z wejść bramki A podłączona jest wartość 0, zatem cała bramka A może zostać usunięta z sieci.

Teraz równanie linii $d_{2,3}$ w sieci przed modyfikacją można zastąpić równaniem

$$d_{2,3} = d_{1,5} + d_{1,3} \tag{5.11}$$

Stąd równania wyjść można zapisać następująco:

$$\begin{array}{rclcrclcl} T_2^8 & = & d_{2,3} + d_{2,2} & = & (d_{1,5} + d_{1,3}) + d_{1,2} & = & d_{1,5} + d_{1,3} + d_{1,2} \\ T_3^8 & = & d_{2,3} \cdot d_{2,2} & = & (d_{1,5} + d_{1,3}) \cdot d_{1,2} & = & d_{1,5} \cdot d_{1,2} + d_{1,3} \cdot d_{1,2} \end{array} \tag{5.12}$$

czyli jako funkcje jedynie linii poziomu 1. Ponieważ, jak udowodniono we własności 5.6, wartość 1 na linii $d_{1,3}$ implikuje 1 na linii $d_{1,2}$, zatem literał $d_{1,3}$ w wyrażeniu $d_{1,5} + d_{1,3} + d_{1,2}$ jest niepotrzebny i może zostać pominięty. Podobnie, ponieważ iloczyn $d_{1,3} \cdot d_{1,2}$ jest równoważny wartości $d_{1,3}$, zatem wyrażenia wyjść można zapisać w postaci

$$\begin{array}{rcl} T_2^8 & = & d_{1,5} + d_{1,2} \\ T_3^8 & = & d_{1,5} \cdot d_{1,2} + d_{1,3} \end{array} \tag{5.13}$$

Wyrażenia te są takie same jak implementowane przez bramki na rys. 5.15. Wykazaliśmy zatem, że sieć po modyfikacji jest równoważna pod względem logicznym sieci niezmodyfikowanej. Należy tutaj zauważyć, że wprowadzona modyfikacja wyeliminowała nie tylko segment od linii $d_{1,3}$ do wyjścia T_3^8, ale i dwie równoważne ścieżki: od wejścia $T_3^{4_a}$ do wyjścia T_2^8 oraz od wejścia $T_3^{4_b}$ do wyjścia T_2^8. Analogicznej modyfikacji dokonano w celu usunięcia segmentu od linii $d_{1,6}$ do wyjścia T_7^8. Kolejne kroki algorytmu przedstawiono na rys. 5.15. Tutaj również, poprzez usunięcie segmentu od linii $d_{1,6}$ do wyjścia T_7^8 wyeliminowano dwie równoważne ścieżki: od wejścia $T_2^{4_a}$ do wyjścia T_7^8 oraz od wejścia $T_2^{4_b}$ do wyjścia T_7^8. Łatwo teraz sprawdzić, że wynikowa sieć ma nie tylko pełną własność SP ale i własność RG, ponieważ funkcje wejść nie uległy zmianie.

Warto zauważyć, że w tym szczególnym przypadku, modyfikacja nie zmieniła liczby bramek. Zmodyfikowana sieć łącząca Batchera OE ma zatem 18 bramek i własność silnej testowalności

uszkodzeń typu opóźnienie ścieżki, podczas gdy sieci łączące ośmiowejściowe zrealizowane jako dwu- lub trzypoziomowe mają powyżej 20 bramek.

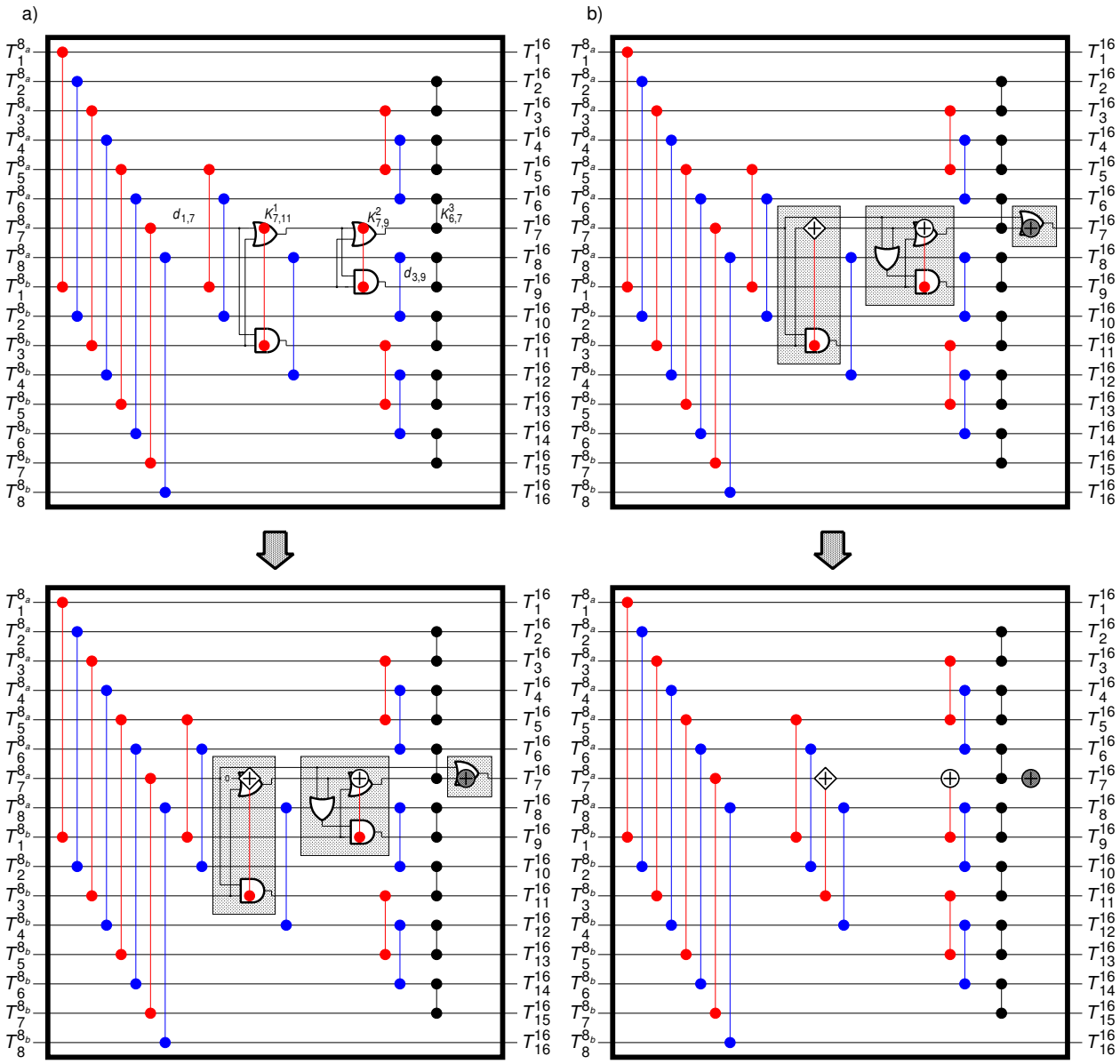
Zapis sieci po modyfikacji

Na rys. 5.16a i b przedstawiono dwa kroki modyfikacji pojedynczej linii sieci 16-wejściowej mającej na celu usunięcie segmentu prowadzącego od linii $d_{1,7}$ do wyjścia T_6^{16}. Jak wynika z własności 5.13, ścieżka te nie jest silnie testowalna. Podobnie jak w przykładzie 5.2, w komparatorze na poziomie 1 na odłączone wejście bramki OR podano stałe 0. Linię $d_{1,7}$ podłączono bezpośrednio do wyjścia T_7^{16} przez bramkę OR. Ponieważ segment prowadzący od linii $d_{1,7}$ do wyjścia T_6^{16} ma zostać wyeliminowany, komparator $K_{6,7}^3$ został pominięty. Natomiast ponieważ funkcja linii $d_{3,9}$ nie może ulec zmianie, zatem konieczna jest modyfikacja komparatora $K_{7,9}^2$, która jest widoczna w dolnej części rys. 5.16a. Modyfikacja ta polega na włączeniu bramki OR w wejście bramki AND komparatora $K_{7,9}^2$. Na podstawie analizy równań linii i wyjść sieci podobnie jak dla sieci ośmiowejściowej, można wykazać, że bramkę OR w komparatorze $K_{7,11}^1$ można pominąć i otrzymać sieć przedstawioną na rys. 5.16b. Następnie można zauważyć, że dokonane modyfikacje mogą zostać jednoznacznie przedstawione jako modyfikacje komparatorów. Komparatory po modyfikacji zostały oznaczone symbolami:

- $\oplus$ pominięcie bramki OR i początek linii równoległej
- $\odot$ pominięcie bramki AND i początek linii równoległej
- $\oplus$ dodatkowa bramka OR w komparatorze
- $\odot$ dodatkowa bramka AND w komparatorze
- $\oplus$ dodatkowa bramka OR z podłączoną linią równoległą
- $\odot$ dodatkowa bramka AND z podłączoną linią równoległą

Przykład 5.2 (c. d.) *Na rys. 5.17a i b przedstawiono modyfikacje dalszych komparatorów sieci o 16 wejściach.*

Z komparatora na poziomie 1 jest eliminowana bramka AND (rys. 5.17a), z linii $d_{1,10}$ *jest wyprowadzana linia równoległa, zaś w komparatorach na poziomach powyżej 1 dołączanych do*



Rysunek 5.16: Przykład usuwania nietestowalnych ścieżek w sieci łączącej Batchera OE dla $n = 16$: a) krok 1, b) krok 2

linii poniżej linii 10 są dołączane dodatkowe bramki AND. W tym przypadku dodatkowa bramka AND jest dołączona do komparatora pomiędzy liniami $d_{2,10}$ i $d_{2,8}$. Komparatory podłączane do linii powyżej linii 10 są pomijane, i tak tutaj pominięty został komparator pomiędzy liniami $d_{3,10}$ i $d_{3,11}$. Ostatecznie linia równoległa jest podłączona przez bramkę AND na poziomie $d_{4,10}$, czyli na poziomie wyjść sieci.

Komparatory po modyfikacji zostały zaznaczone zgodnie z użyciem w/w symboli. Następnie zostały zmodyfikowane komparatory wzdłuż linii o indeksie 7, a zapis samych komparatorów po modyfikacji przedstawiono na rys. 5.17b. Warto tutaj zwrócić uwagę, iż zmodyfikowany komparator łączący linie $d_{1,7}$ i $d_{1,11}$ nie zawiera bramek.

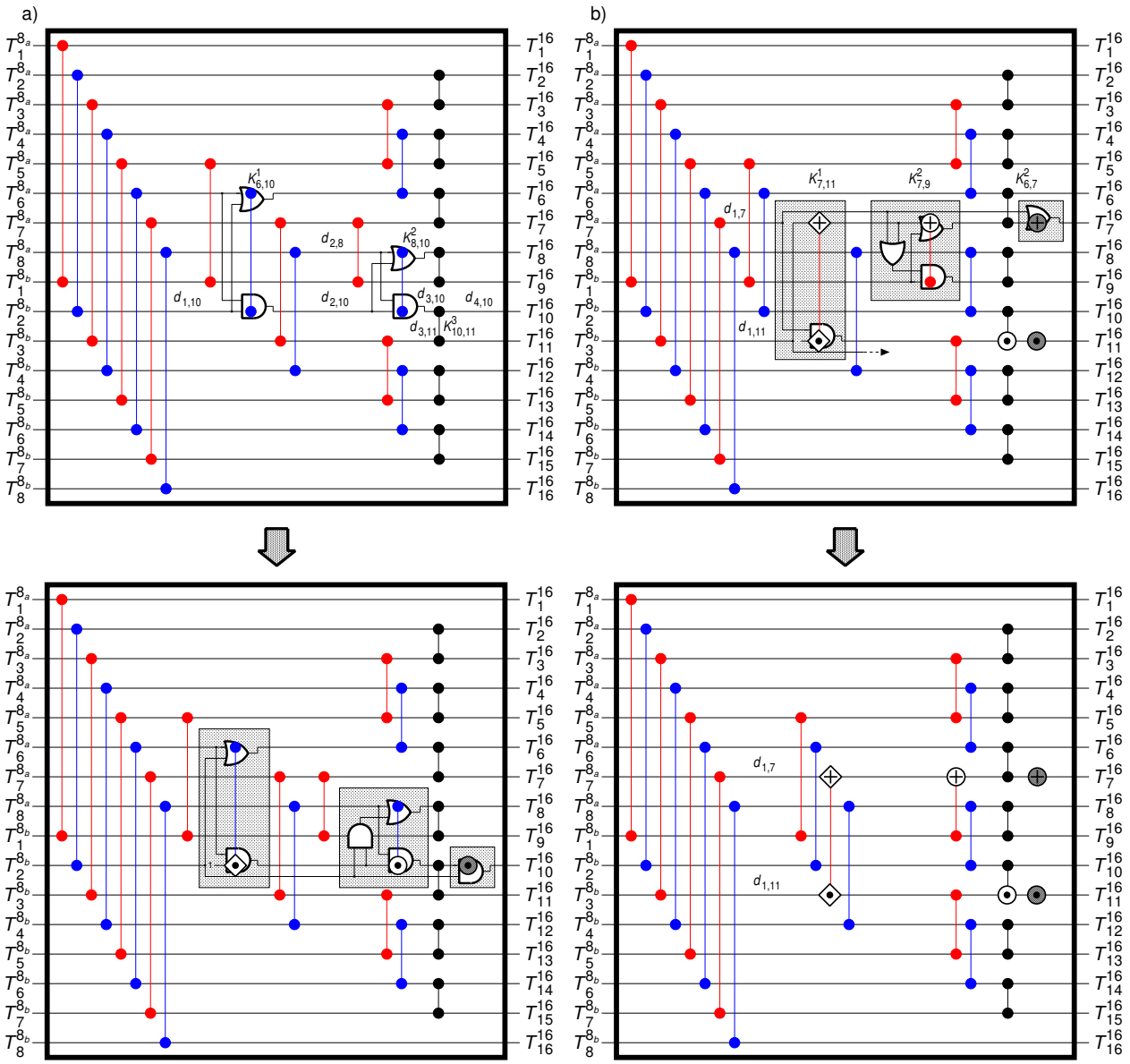
Pełna modyfikacja sieci, taka że linia równoległa jest dołączana do linii głównej na ostatnim poziomie, przy wyjściach sieci skutkuje siecią zmodyfikowaną przedstawionym w górnej części rys. 5.18. Widać tutaj, że praktycznie każdy komparator został zmodyfikowany, poza komparatorami poziomu 0. Należy jednak zauważyć, że każdy symbol w kółku na białym tle, oznacza dodatkową bramkę w sieci. Symbole w kółkach na ciemnym tle są również bramkami, jednak można je sparować z symbolami w kwadratach na komparatorach występujących tuż za poziomem 0, gdzie bramki w sieci zmodyfikowanej nie występują (występują natomiast w sieci niezmodyfikowanej). Oznacza to zatem, iż nadmiar przy takim algorytmie modyfikacji jest duży.

Redukcja liczby bramek sieci po modyfikacji

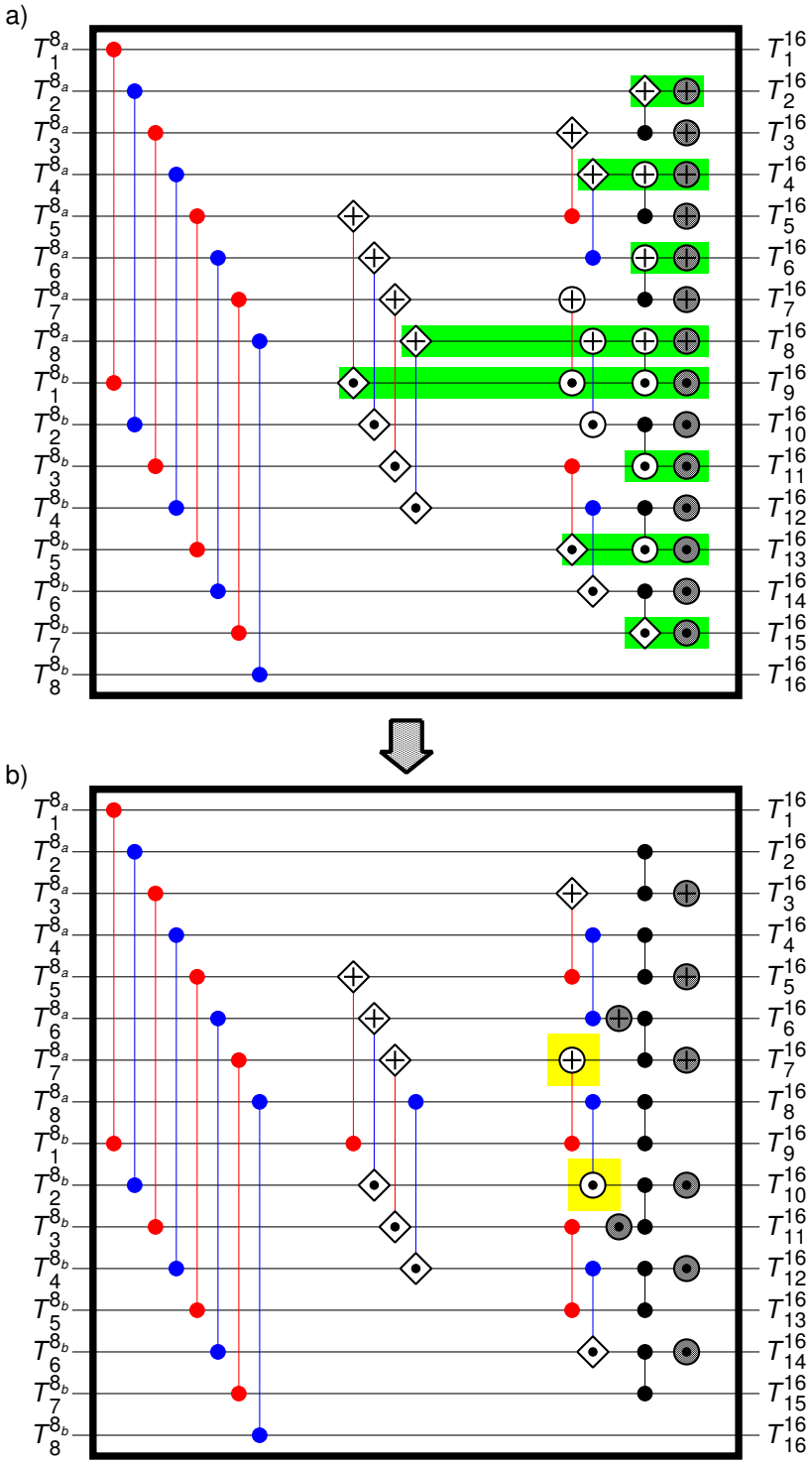
Rozważmy sieć MN^{16} po modyfikacji, przedstawioną na rys. 5.18a, na której zacieniowano elementy podlegające redukcji. Następnie na rys. 5.18b przedstawiono sieć po redukcji, w której nadmiarowe elementy, które pozostały w sieci po redukcji, zostały zacieniowane. Warto zauważyć, że liczba bramek w tej sieci (kółka na białym tle) zwiększyła się tylko o 2.

Zapis pełny i zapis skrócony sieci łączącej

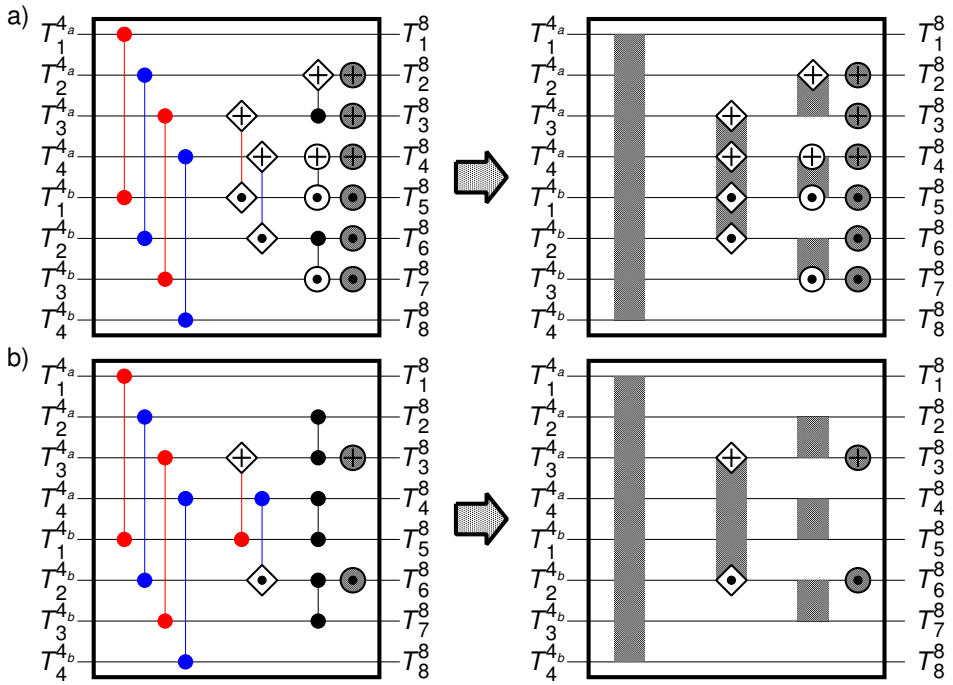
W przypadku sieci o liczbie wejść wyrażonej potęgą dwójki, komparatory układają się w bloki (rys. 5.3). Zakładamy, że komparatory można grupować w bloki także w sieciach o parzystej liczbie wejść, jednak łatwe ustalenie rozmiarów i rozkładu bloków jest możliwe tylko w sieciach, w których liczba linii $n = 2^k$. Bloki takie możemy następnie zapisać w postaci skróconej, tak jak przedstawiono to na rys. 5.19a. Widzimy także, iż taka postać zapisu może zostać użyta do zapisu



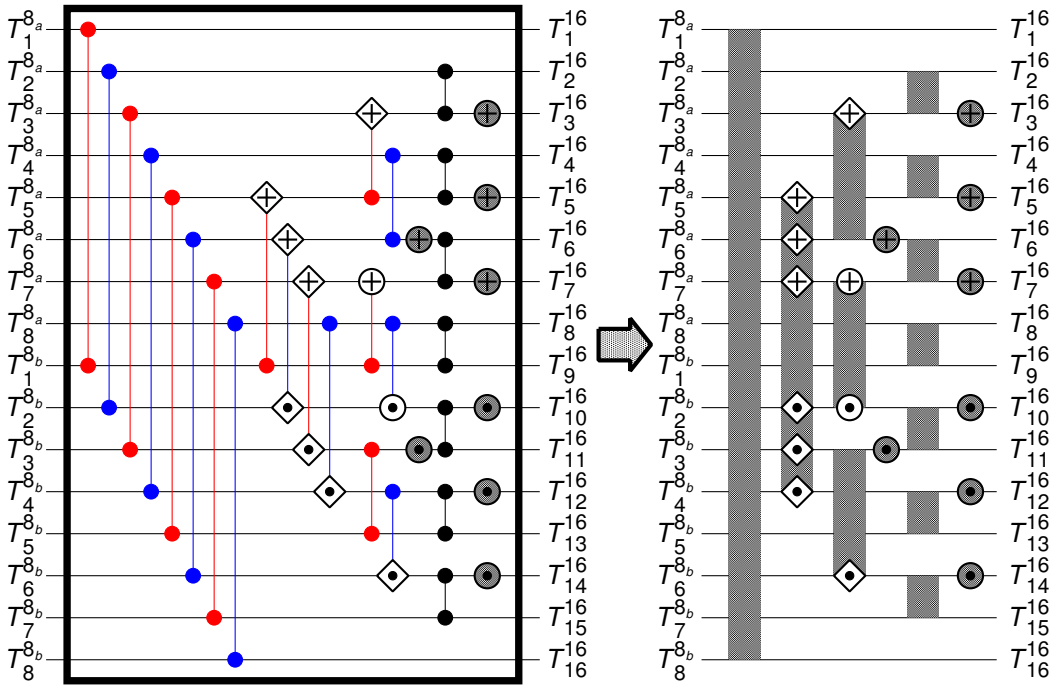
Rysunek 5.17: Przykład modyfikacji sieci łączącej Batchera OE dla $n = 16$ a) etap 1, b) etap 2



Rysunek 5.18: Redukcja bramek w zmodyfikowanej sieci MN^{16}



Rysunek 5.19: Zapis skrócony sieci zmodyfikowanej o 8 wejściach: a) bez redukcji, b) z redukcją



Rysunek 5.20: Zapis skrócony sieci zmodyfikowanej o 16 wejściach z redukcją

sieci zarówno z redukcją liczby bramek jak i bez tej redukcji. Zapis taki istotnie zwiększa przejrzystość zapisu sieci o większej liczbie linii. Na rys. 5.19b, 5.20 oraz 5.21 pokazano odpowiednio sieci o 8, 16 i 32 wejściach, po modyfikacji i redukcji.

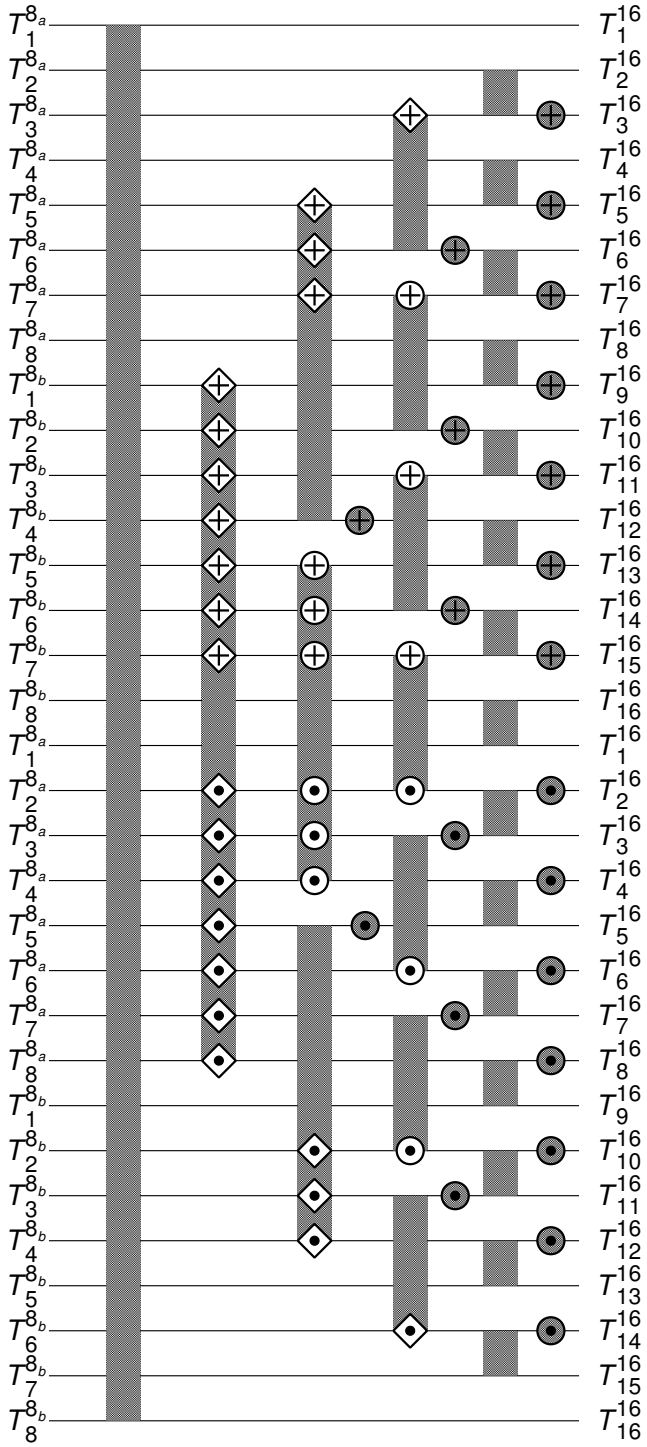
Algorytm modyfikacji

Bazując na przykładzie 5.2, podamy następujący ogólny algorytm modyfikacji sieci łączącej Batchera OE o parzystej liczbie wejść.

Algorytm 5.1

I. $1 \leqslant j \leqslant \frac{n}{2}$

1. *Usunąć bramkę OR w komparatorze podłączonym do linii* $d_{1,j}$*, a wyjście komparatora podłączyć do drugiego wejścia komparatora* $(d_{1,k})$*.*
2. *Podłączyć linię* $d_{1,j}$ *do każdego komparatora łączącego linię* j *i linię wyższą niż* j*.*
3. *Podłączyć linię* $d_{1,j}$ *i linię* $d_{i_{\max},j}$ *do bramki OR, której wyjście stanowi wyjście* T_j^n*.*



Rysunek 5.21: Sieć zmodyfikowana o 32 wejściach z redukcją (w zapisie skróconym)

II. $\frac{n}{2}+1 \leqslant j \leqslant n$

1. *Usunąć bramkę AND w komparatorze podłączonym do linii* $d_{1,j}$*, a wyjście komparatora podłączyć do drugiego wejścia komparatora* ($d_{1,k}$).
2. *Podłączyć linię* $d_{1,j}$ *do każdego komparatora łączącego linię* j *i linię niższą niż* j.
3. *Podłączyć linię* $d_{1,j}$ *i linię* $d_{i_{\max},j}$ *do bramki AND, której wyjście stanowi wyjście* T_j^n.

Algorytm należy zastosować do modyfikacji wszystkich linii sieci łączącej od 1 *do* n.

Algorytm w postaci 5.1 nie uwzględnia redukcji liczby bramek. Jeżeli założymy możliwość redukcji przedstawionej na rys. 5.18-5.21 algorytm przybiera postać następującą. Niech i_{komp} oznacza poziom w sieci, za którym występują tylko komparatory $K_{j,k}^i$, takie że: (i) $1 \leqslant j \leqslant \frac{n}{2}, k > j$ oraz (ii) $\frac{n}{2}+1 \leqslant j \leqslant n, k < j$.

Algorytm 5.2

I. $1 \leqslant j \leqslant \frac{n}{2}$

1. *Usunąć bramkę OR w komparatorze podłączonym do linii* $d_{1,j}$*, a wyjście komparatora podłączyć do drugiego wejścia komparatora* ($d_{1,k}$).
2. *Podłączyć linię* $d_{1,j}$ *do każdego komparatora łączącego linię* j *i linię wyższą niż* j.
3. *Podłączyć linię* $d_{1,j}$ *i linię* $d_{i_{\text{komp}},j}$ *do bramki OR, której wyjście stanowi nową linię* $d_{i_{\text{komp}},j}$.

II. $\frac{n}{2}+1 \leqslant j \leqslant n$

1. *Usunąć bramkę AND w komparatorze podłączonym do linii* $d_{1,j}$*, a wyjście komparatora podłączyć do drugiego wejścia komparatora* ($d_{1,k}$).
2. *Podłączyć linię* $d_{1,j}$ *do każdego komparatora łączącego linię* j *i linię niższą niż* j.
3. *Podłączyć linię* $d_{1,j}$ *i linię* $d_{i_{\text{komp}},j}$ *do bramki AND, której wyjście stanowi nową linię* $d_{i_{\text{komp}},j}$.

Algorytm należy zastosować do modyfikacji wszystkich linii sieci łączącej od 1 *do* n.

Twierdzenie 5.2 *Sieć łącząca Batchera OE zmodyfikowana zgodnie z algorytmem 5.2 jest układem SP.*

Dowód

Zauważmy, że modyfikacje przeprowadzone z użyciem algorytmu 5.2 eliminują:

1. segmenty prowadzące od linii $u_j^{n_a}$ do wyjścia T_k^n $(k < j)$ oraz
2. segmenty prowadzące od linii $u_j^{n_b}$ do wyjścia T_k^n $(k > j)$.

Ponieważ w zmodyfikowanej sieci łączącej pozostają jedynie ścieżki podane w definicji układu SP, zatem sieć po modyfikacji ma własność SP. ■

Twierdzenie 5.3 *Sieć łącząca Batchera OE zmodyfikowana zgodnie z algorytmem 5.2 jest układem RG.*

Dowód

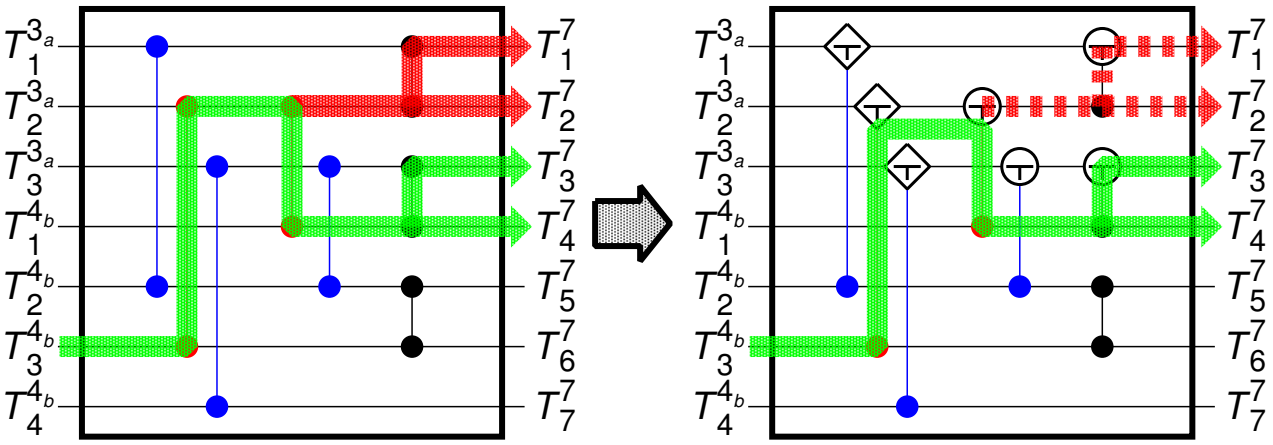
Modyfikacje sieci Batchera OE wprowadzone z użyciem algorytmu 5.2 nie zmieniają funkcji realizowanych przez poszczególne linie sieci. Zatem układ pozostaje układem łączącym, czyli dla wektorów których pary stanowią wzorce testowe V_{in} generuje wektory uporządkowane o takiej samej wadze. Ponieważ sieć zmodyfikowana jest bezinwerterowa, liczba zarówno jedynek jak i zer we wzorcu wyjściowym sieci pozostaje taka sama jak we wzorcu wejściowym. Stąd układ zmodyfikowany jest układem RG. ■

5.4.2 Modyfikacja sieci o nieparzystej liczbie wejść

Przewidujemy, że w sieciach o nieparzystej liczbie wejść modyfikacja będzie analogiczna jak w sieciach o parzystej liczbie wejść, z dodatkowymi elementami wynikającymi z nieparzystej liczby linii.

Modyfikacja sieci łączącej wygenerowanej wg algorytmu Batchera OE

Przeprowadzone przez nas analizy wykazały, iż możliwa jest modyfikacja sieci łączących generowanych zgodnie z algorytmem Batchera OE. Przykładowe sieci wynikowe miały własności SP i RG. Jednak w przypadku sieci o liczbie linii większej niż 7 modyfikacja ta jest trudna.



Rysunek 5.22: Przykład usuwania nietestowalnych ścieżek w sieci łączącej MN^7

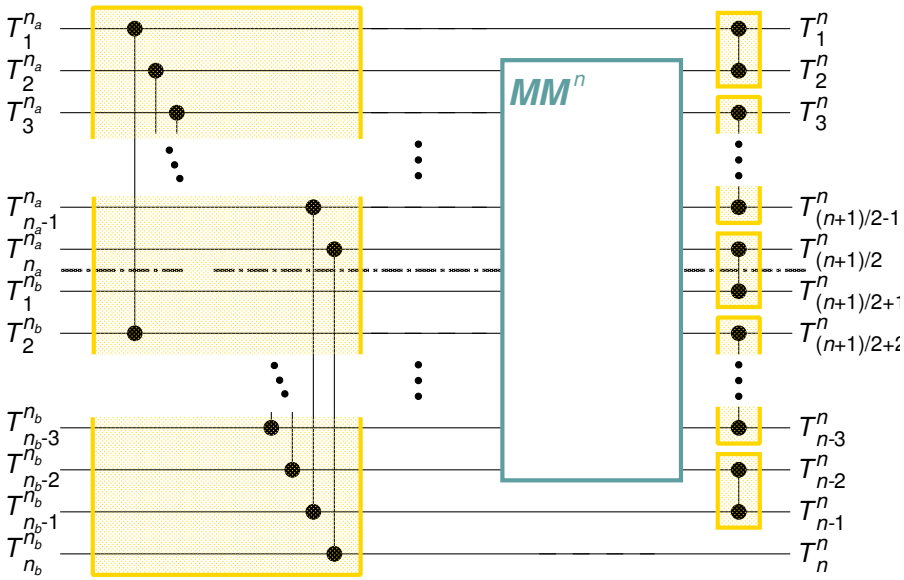
Podczas prowadzonych badań nie udało się tego problemu satysfakcjonująco rozwiązać. Udało się jednak znaleźć zadowalające rozwiązanie alternatywne. Mianowicie przeprowadzono porównanie złożoność różnych realizacji sieci o nieparzystej liczbie linii, które zostało zaprezentowane w rozdziale 5.2.4. Wynika z niego, że modyfikacja sieci uzyskanej przez usunięcie linii jest równoważna modyfikacji sieci zrealizowanej rekurencyjnie z użyciem algorytmu Batchera OE.

Modyfikacja sieci łączącej uzyskanej przez usunięcie linii

W sieciach o nieparzystej liczbie wejść, zrealizowanych poprzez usunięcie linii z sieci o parzystej liczbie linii, konieczna jest modyfikacja taka jak w sieci o parzystej liczbie linii oraz modyfikacja dodatkowa. Konieczność istnienia modyfikacji dodatkowej ilustruje poniższy przykład i rys. 5.22.

Przykład 5.3 *Na rys. 5.22 podano przykład 7-wejściowej sieci łączącej zrealizowanej poprzez usuwanie linii z sieci o parzystej liczbie linii. Na rysunku zostały zaznaczone ścieżki prowadzące z wejścia $T_3^{4_b}$. Można zauważyć, że występują tam m. in. ścieżki prowadzące do wyjść T_1^7 oraz T_2^7.*

Zweryfikujemy teraz, czy sieć ta ma własność SP. Zgodnie z definicją własności SP, w sieci mogą istnieć jedynie ścieżki prowadzące z wejścia $T_3^{4_b}$ do wyjść $T_3^7 \ldots T_6^7$. Zatem ścieżki prowadzące z wejścia $T_3^{4_b}$ do wyjść $T_3^7 \ldots T_6^7$ nie spełniają warunków definicji własności SP. Można następnie zauważyć, iż ścieżki naruszające własność SP prowadzą z wejść $T_j^{4_b}$ do wyjść typu T_k^7 $(k < j)$. Ścieżki te leżą poza przedziałem podanym w definicji układu SP.



Rysunek 5.23: Schemat ogólny sieci o nieparzystej liczbie wejść zrealizowanej przez usunięcie linii

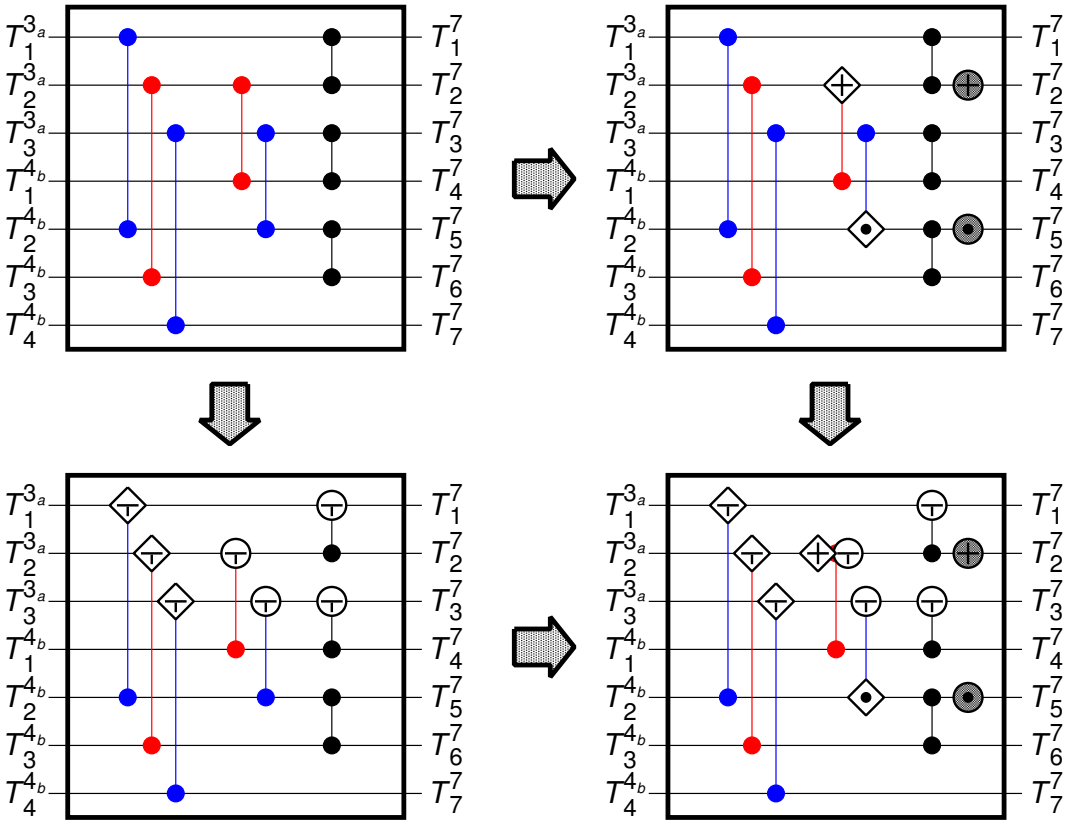
Testowalność ścieżek w sieci o nieparzystej liczbie wejść uzyskanej przez usunięcie linii

Rozważmy schemat ogólny sieci przedstawiony na rys. 5.23. Widzimy na nim, że po usunięciu linii 1, linia 2 staje się linią 1. Podobnie, po usunięciu linii pojawiają się ścieżki prowadzące z wejść $T_j^{n_a}$ do wyjść T_{j-1}^n. Poniżej wykażemy, że ścieżki te nie są silnie testowalne.

Własność 5.14 *Żadna ścieżka prowadząca z wejścia $T_j^{n_b}$ do wyjścia T_k^n $(1 \leqslant j \leqslant n_b,\ k < j)$ nie jest silnie testowalna.*

Dowód

Z wejścia $T_j^{n_b}$ istnieje segment do linii $d_{1,j-1}$ (rys. 5.23), z której istnieje bezpośrednia ścieżka do wyjścia T_{j-1}^n. Wiemy, że jest to jedyna ścieżka z wejścia $T_j^{n_b}$ do wyjścia T_{j-1}^n. Rozważmy teraz wzorzec wejściowy V_{in}, który miałby być testem dla ścieżki z wejścia $T_j^{n_b}$ do wyjścia T_k^n. We wzorcu tym zdarzenie musi wystąpić na wejściu $T_j^{n_b}$, a na wejściach T^{n_a} musi wystąpić 0 jedynek. Z definicji sieci łączącej wiemy, że na wyjściu sieci występuje $j-1$ stałych jedynek, a zdarzenie występuje na pozycji j (sieć jest bezinwerterowa). Zatem na przynajmniej jednym wejściu bocznym bramki leżącej na ścieżce z wejścia $T_j^{n_b}$ do wyjścia T_k^n musi wystąpić stała



Rysunek 5.24: Przykład modyfikacji sieci łączącej MN^7

wartość sterująca, i tym samym ta ścieżka nie jest silnie testowalna. Powyższe rozumowanie możemy przeprowadzić także dla innych ścieżek prowadzących do wyjść T_k^n $(k < j)$. ■

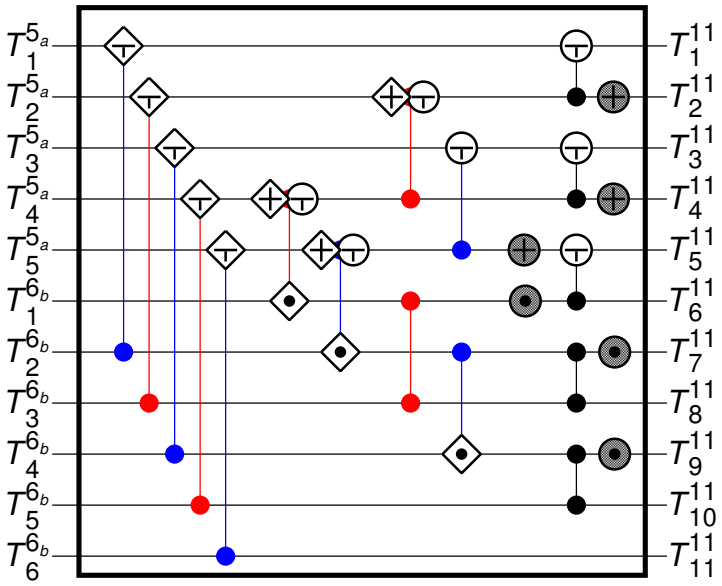
Zapis sieci po modyfikacji

- ◈ pominięcie bramki OR
- ⊕ dodatkowa bramka OR w komparatorze

Schematy logiczne modyfikacji kolejnych linii podano na rys. 5.26a-c.

Algorytm modyfikacji

Na podstawie powyższych rozważań podajemy ogólny algorytm modyfikacji sieci łączącej Batchera OE o nieparzystej liczbie wejść.



Rysunek 5.25: Przykład zmodyfikowanej sieci łączącej MN^{11}

Algorytm 5.3

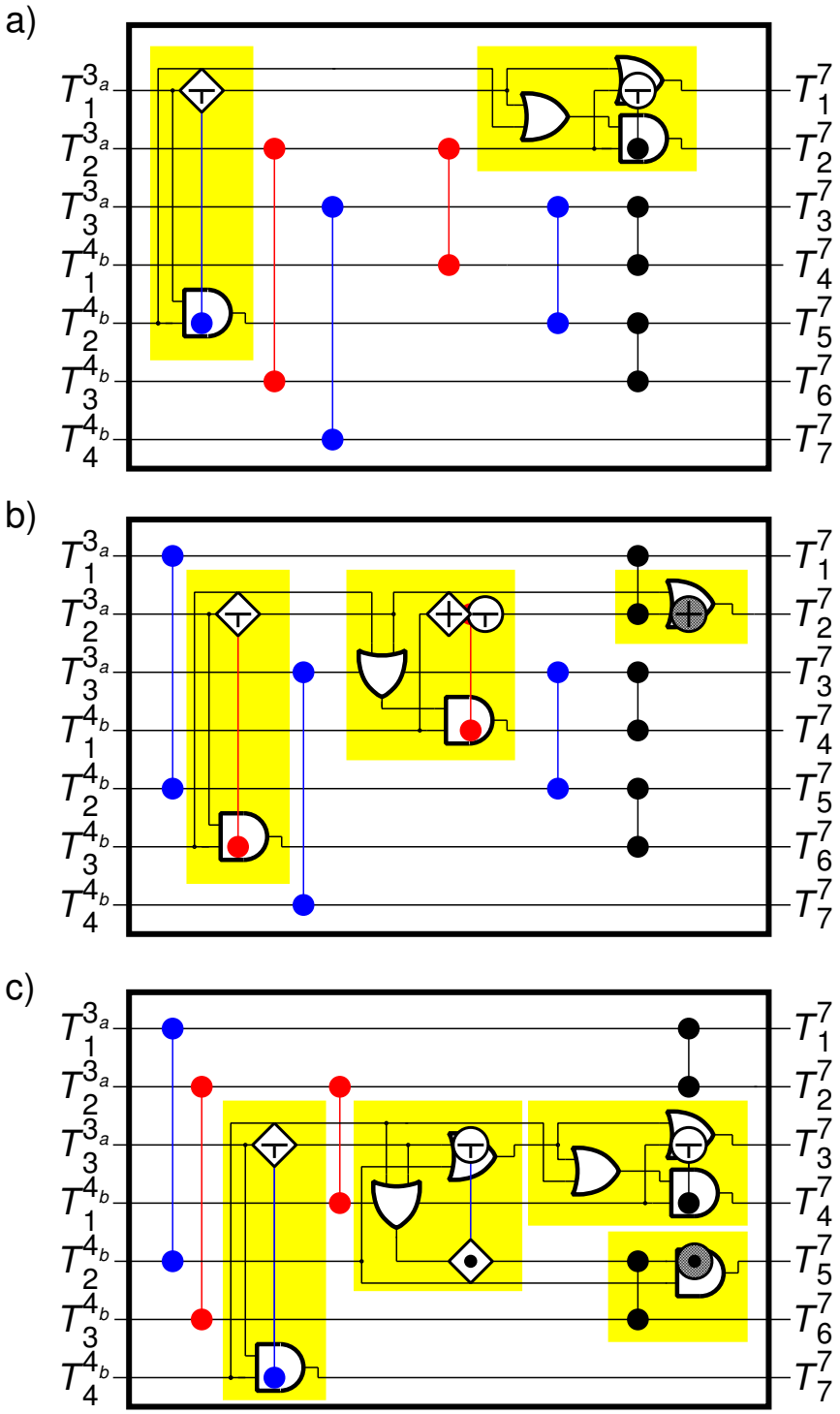
1. *Zmodyfikować sieć zgodnie z algorytmem 5.2.*
2. *Usunąć bramkę OR w komparatorze podłączonym do linii $d_{0,j}$, a wyjście komparatora podłączyć do wejścia komparatora ($d_{0,j}$).*
3. *Podłączyć linię $d_{0,k}$ do jednego z wejść bramki AND każdego komparatora łączącego linię j i linię wyższą niż j przez bramkę OR.*

Twierdzenie 5.4 *Sieć łącząca Batchera OE zmodyfikowana zgodnie z algorytmem 5.3 jest układem SP.*

Dowód

Zauważmy, że modyfikacje przeprowadzone z użyciem algorytmu 5.3 eliminują ścieżki prowadzące od wejścia $T_j^{n_b}$ $(2 \leqslant j \leqslant n_b)$ do wyjścia T_k^n $(1 \leqslant k < j)$. Ponieważ w zmodyfikowanej sieci łączącej pozostają jedynie ścieżki podane w definicji układu SP, zatem sieć po modyfikacji ma własność SP. ■

Twierdzenie 5.5 *Sieć łącząca Batchera OE zmodyfikowana zgodnie z algorytmem 5.3 jest układem RG.*



Rysunek 5.26: Usuwanie nietestowalnych ścieżek na kolejnych liniach sieci łączącej MN^7, podzielona wg linii sieci: a) linia $d_{i,1}$, b) linia $d_{i,2}$, c) linia $d_{i,3}$

Dowód

Modyfikacje sieci Batchera OE wprowadzone z użyciem algorytmu 5.3 nie zmieniają funkcji realizowanych przez poszczególne linie sieci. Zatem układ pozostaje układem łączącym, czyli dla wektorów których pary stanowią wzorce testowe, V_{in} generuje wektory uporządkowane o takiej samej wadze. Ponieważ sieć zmodyfikowana jest bezinwerterowa, liczba jedynek jak i zer we wzorcu wyjściowym sieci jest pozostaje sama jak we wzorcu wejściowym. Stąd układ zmodyfikowany jest układem RG. ■

5.5 Podsumowanie

W powyższym rozdziale wykazaliśmy istnienie modyfikacji zapewniającej wystąpienie własności SP i RG w sieci łączącej Batchera OE dla dowolnej liczby wejść, dzięki czemu uzyskaliśmy własność silnej testowalności uszkodzeń typu opóźnienie ścieżki w całej sieci sortującej Batchera OE. Wiemy, że sieć sortująca ma istotnie mniejszą złożoność od innych wielowyjściowych układów progowych. W następnym rozdziale dokonamy syntezy wszystkich układów dla różnych liczb wejść i porównamy ich złożoności i liczby ścieżek.

Rozdział 6

Wyniki syntezy układów

6.1 Wstęp

W poprzednich rozdziałach rozważaliśmy własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych. W rozdziale 4 pokazaliśmy pewne specyficzne własności wielowyjściowych układów progowych zrealizowanych jako całkowicie rozgałęzione, dotyczące sieci łączących. Wykazaliśmy, że własności te, nazwane SP i RG, zapewniają silną testowalność uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych zrealizowanych jako całkowicie rozgałęzione. Następnie, w rozdziale 5, przedstawiliśmy algorytm modyfikacji sieci sortującej Batchera OE, celem uzyskania własności SP i RG. Wszystkie podane wcześniej wyniki miały jednak charakter tylko teoretyczny. W poniższym rozdziale opiszemy: (i) próbę zweryfikowania własności SP i RG dla szeregu przypadków zmodyfikowanej sieci Batchera OE, oraz (ii) wyniki syntezy zmodyfikowanej sieci Batchera OE, jak również porównanie wyników syntezy (w sensie złożoności i liczby ścieżek) otrzymanego przez nas układu z wynikami dostępnymi w literaturze. Przedstawimy też kilka interesujących spostrzeżeń.

Tablica 6.1: Porównanie złożoności sieci łączących dla wybranych n

n	Liczba bramek dwuwejściowych		
	Oryginalna sieć Batchera [5]	Sieć zmodyfikowana	Narzut [%]
4	6	6	0,0
5	10	10	0,0
6	14	14	0,0
7	16	17	6,2
8	18	18	0,0
9	24	25	4,2
10	26	26	0,0
11	32	34	6,2
12	34	34	0,0
13	40	45	12,5
14	42	44	4,7
15	48	55	14,6
16	50	52	4,0
17	58	65	12,1
18	60	62	3,3
19	68	76	11,8
20	70	72	2,8
21	78	89	14,1
26	100	106	6,0
30	120	132	10,0
32	130	142	9,2
64	322	368	14,2
128	770	914	18,7
256	1694	2096	23,7

6.2 Weryfikacja własności uzyskanych sieci łączących

Opis pakietu oprogramowania do weryfikacji

Celem weryfikacji własności silnej testowalności uszkodzeń typu opóźnienie ścieżki i oceny złożoności układów otrzymanych z użyciem metod podanych w rozdz. 4 i 5, opracowaliśmy specjalny pakiet oprogramowania, pozwalający na:

- automatyczną syntezę sieci łączących Batchera OE,
- automatyczną syntezę sieci łączących o nieparzystej liczbie wejść poprzez usuwanie linii z sieci Batchera OE o parzystej liczbie wejść (rozdz. 5.2.4),
- automatyczną modyfikację uzyskanej sieci łączącej zgodnie z algorytmami 5.1, 5.2 i 5.3,
- zapis listy połączeń uzyskanych sieci łączącej w formacie BLIF,
- oszacowanie liczby bramek i liczby ścieżek w uzyskanych układach,
- obliczenie liczby ścieżek prowadzących z danego wejścia.

Oprogramowanie zostało napisane w języku C++, z użyciem środowiska programistycznego Eclipse. Proces pisania oprogramowania, jak i wszystkie testy przeprowadziliśmy w systemie operacyjnym Linux.

Weryfikacja własności sieci łączących

Wykorzystując napisane przez nas oprogramowanie, uzyskaliśmy szereg układów łączących dla wszystkich $3 \leqslant n \leqslant 32$ oraz dla wybranych $32 < n \leqslant 512$ zapisanych w postaci listy połączeń. Własności łączenia wektorów uporządkowanych każdej z uzyskanych sieci łączących zweryfikowaliśmy przez wykonanie symulacji w systemie SIS dla zbioru wektorów o postaci podanej w def. 5.11. Lista połączeń każdej sieci posłużyła również do wygenerowania listy wszystkich występujących w niej ścieżek. Otrzymaną listę ścieżek wykorzystaliśmy do sprawdzenia liczby wszystkich ścieżek wychodzących z każdego wejścia sieci łączącej. Okazało się, że w sieci łączącej MN^n mającej własność SP:

- dla parzystego $n = 2k$ (gdzie $k = 1, 2, 3, \ldots$), liczba ścieżek wychodzących z dowolnego wejścia jest równa $k + 1$;

– dla nieparzystego $n = 2k + 1$ (gdzie $k = 1, 2, 3, \ldots$, $n_a = k$, $n_b = k + 1$), liczba ścieżek wychodzących z dowolnego z wejść $T_i^{n_a}$ jest równa $n_b + 1$, zaś z wejść $T_i^{n_b}$, jest równa $n_a + 1$.

Zatem, jeżeli wygenerowana przez nas sieć łącząca miała własność SP (a poprzez symulację zweryfikowaliśmy również jej własność RG), to jest: (i) silnie testowalna, (ii) wielowyjściowy układ progowy zbudowany z jej użyciem jest silnie testowalny. Wszystkie testowane układy zachowały się zgodnie z przewidywaniami, zatem uzyskaliśmy częściowe praktyczne potwierdzenie wyników teoretycznych przedstawionych przez nas w rozdziałach 4 i 5, a zarazem tezy postawionej na początku niniejszej rozprawy.

6.3 Analiza porównawcza uzyskanych wyników eksperymentalnych

W tablicy 6.1 pokazano złożoność szeregu sieci łączących zrealizowanych poprzez modyfikację sieci sortującej Batchera OE zgodnie z algorytmami 5.2 i 5.3 oraz procentowy narzut modyfikacji. Za miarę złożoności przyjęto liczbę bramek dwuwejściowych. Widać tutaj, że dla $n \leqslant 10$ modyfikacja nie powiększa w żaden sposób sieci, a dla $n > 10$ narzut modyfikacji jest zaledwie kilku- lub kilkunastoprocentowy. W dalszych porównaniach pokażemy, że pomimo niewielkiego wzrostu złożoności, w każdym przypadku wielowyjściowy układ progowy zaproponowany przez nas jest nawet wielokrotnie mniejszy od najmniejszego znanego z literatury wielowyjściowego układu progowego mającego własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. Nie wykluczamy, że istnieje metoda dalszego zmniejszenia liczby bramek, jakkolwiek wydaje się to mało prawdopodobne.

W tablicy 6.2 porównano złożoność sieci łączących mających własności SP i RG, a w tablicy 6.3 złożoność wielowyjściowych układów progowych zrealizowanych z użyciem tych sieci, mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. Widać tutaj, że układy zaprojektowane z użyciem metod zaproponowanych w niniejszej rozprawie są znacznie lepsze niż najlepsze rozwiązania znane w literaturze. O ile sieci o małej liczbie wejść podane w [73] są jeszcze porównywalne z naszymi, o tyle już przy 16 wejściach sieć trzypoziomowa jest o 50% większa od naszej, a dla $n \geqslant 64$ różnica ta jest jeszcze większa (kolumna „Redukcja” w tab. 6.2). Dodatkowo,

Tablica 6.2: Porównanie złożoności różnych wersji sieci łączących wielowyjściowych układów progowych mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki

n	Liczba bramek dwuwejściowych			Redukcja†
	AND-OR [76]	SN+AND-OR [73]	BAT-M (PP)	[%]
4	4	6	6	0,00
5	12	10	10	0,00
6	12	12	14	16,67
7	24	18	17	−5,56
8	24	20	18	−10,00
9	40	28	25	−10,71
10	40	30	26	−13,33
11	60	40	34	−15,00
12	60	42	34	−19,05
13	84	54	45	−16,67
14	84	56	44	−21,43
15	112	70	55	−21,43
16	112	72	52	−27,78
17	144	88	65	−26,14
18	144	90	62	−31,11
19	180	108	76	−29,63
20	180	110	72	−34,55
21	220	130	89	−31,54
26	338	182	106	−41,76
30	420	240	132	−45,00
32	480	272	142	−47,79
64	1984	1056	368	−65,15
128	8064	4160	914	−78,03
256	32512	16512	2096	−87,31

† w porównaniu z [73]: $\frac{\text{PP}-[73]}{[73]}$

Tablica 6.3: Porównanie złożoności różnych wersji wielowyjściowych układów progowych mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki

n	Liczba bramek dwuwejściowych		
	AND-OR [76]	SN+AND-OR [73]	BAT-M (PP)
4	12	10	10
5	30	18	18
6	38	24	24
7	54	34	33
8	56	40	38
9	92	56	53
10	110	66	62
11	140	82	66
12	148	90	82
13	190	112	102
14	206	124	110
15	238	144	126
16	240	152	128
17	310	184	156
18	346	202	168
19	402	230	191
20	420	242	196
21	492	278	227
26	718	406	310
30	926	528	384
32	992	576	398
64	4032	2208	1164
128	16265	8576	3242
256	65280	33664	8680

Tablica 6.4: Porównanie złożoności różnych implementacji funkcji symetrycznych

Funkcja	Liczba wejść bramek						Redukcja†
	[31]	[12]	[80]	[64]	[73]	BAT-M (PP)	[%]
$S^{10}_{5,6}$	2100	1034	133	222	130	126	−3,08
$S^{11}_{6,7}$	4092	1957	X	282	174	136	−21,84
S^{11}_{3-7}	1815	916	137	282	174	136	−21,84
$S^{11}_{3,4}$	2805	1365	X	282	174	136	−21,84
$S^{11}_{5,6}$	4620	2220	X	282	174	136	−21,84
$S^{11}_{4,5}$	4092	1957	X	282	174	136	−21,84
$S^{12}_{7,8}$	7524	3548	X	298	182	166	−8,79
S^{12}_{4-8}	3960	1960	X	298	182	166	−8,79
$S^{12}_{6,7}$	9504	4469	189	298	182	166	−8,79
$S^{12}_{5,6}$	9504	4469	X	298	182	166	−8,79
$S^{13}_{7,8}$	18447	8451	217	382	226	206	−8,85
S^{13}_{4-8}	9295	4494	X	382	226	206	−8,85
$S^{13}_{6,7}$	20592	9530	X	382	226	206	−8,85
$S^{13}_{5,6}$	18447	8451	X	382	226	206	−8,85
$S^{14}_{8,9}$	34034	15540	X	414	250	222	−11,20
S^{14}_{5-9}	20020	9578	X	414	250	222	−11,20
$S^{14}_{5,6}$	34034	15540	X	414	250	222	−11,20
$S^{14}_{6,7}$	42042	19081	253	414	250	222	−11,20
S^{15}_{5-9}	45045	21085	X	478	290	254	−12,41
$S^{15}_{5,6}$	60060	27354	X	478	290	254	−12,41
$S^{15}_{7,8}$	90090	40362	X	478	290	254	−12,41

† w porównaniu z [73]: $\frac{\text{PP}-[73]}{[73]}$

w sieci dwu- i trzypoziomowej także występują problemy wynikające z konieczności implementacji bramek elementarnych o wielu wejściach, tak że złożoność sieci trzypoziomowej w przypadku rzeczywistych układów może być nawet nieco większa od podanej w tabeli. Dla porównania, w naszym układzie występują tylko bramki dwuwejściowe, a przy tym nasza sieć jest bardziej regularna, dzięki czemu można zarówno zmniejszyć powierzchnię zajmowaną przez układ, jak i ulepszyć wykorzystanie powierzchni zajmowanej przez połączenia czy też sieć zasilającą. Zauważmy też, że zestawy bramek które wprowadza nasza modyfikacja, mogą być implementowane jako pojedyncze bramki złożone projektowane na poziomie tranzystorów [91].

Zauważmy też, że sieć łącząca z [73] teoretycznie wnosi mniejsze opóźnienie jednakże konieczność implementacji bramek wielowejściowych jako wielopoziomowych może spowodować, że dla większych n opóźnienie układu trzypoziomowego może być porównywalne lub większe niż opóźnienie naszego układu. Gdy jednak potrzebujemy faktycznie niewielkiego opóźnienia, możemy wybrać także układ z dwupoziomową siecią łączącą [21], którego własność silnej testowalności została przez nas wykazana w [62].

W tablicach 6.4 oraz 6.5 porównano odpowiednio złożoności i liczby ścieżek różnych wersji układów implementujących funkcje symetryczne mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki. W kolumnach tablic podano pozycje literatury, z których zostały zaczerpnięte odpowiednie wyniki. Dla szeregu układów w [80] nie podano wyników syntezy, co zaznaczono symbolem „X". Kolumny [31] i [12] zawierają liczby wejść bramek implementacji dwu-, trzy- i czteropoziomowych, a kolumna [80] zawiera liczby wejść bramek implementacji uzyskanej z użyciem BDD. W kolumnach [64] oraz [73] zawarto liczby wejść bramek w implementacjach wielopoziomowych z odpowiednio dwu- i trzypoziomowymi sieciami łączącymi. W kolumnie „BAT-M (PP)" podano liczbę wejść bramek zmodyfikowanej sieci sortującej Batchera OE zaprezentowanej przez nas w rozdziale 5, a w kolumnie „Redukcja" tab. 6.4 procentową różnicę pomiędzy złożonością naszej implementacji a złożonością implementacji podanej w [73]. W przypadku układów wielopoziomowych, liczba wejść bramek wielowyjściowego układu progowego jest równa podwójnej liczbie bramek. Podobnie jak w [73] przyjęto, że układowi końcowemu generującemu funkcję symetryczną (NOT-AND) odpowiadają dwa wejścia bramek.

Implementacje dwu-, trzy- i czteropoziomowe mają złożoność wykładniczą, co przejawia się lawinowym wzrostem złożoności dla $n \geqslant 10$ (kolumny [31], [12]). Implementacje wielopoziomowe

Tablica 6.5: Porównanie liczby ścieżek w różnych implementacjach funkcji symetrycznych

Funkcja	Liczba ścieżek					
	[31]	[12]	[80]	[64]	[73]	BAT-M (PP)
$S^{10}_{5,6}$	2354	1296	781	320	320	320
$S^{11}_{6,7}$	4556	2433	X	430	430	430
S^{11}_{3-7}	2147	1108	713	294	294	294
$S^{11}_{3,4}$	3137	1675	X	270	270	270
$S^{11}_{5,6}$	5082	2740	X	430	430	430
$S^{11}_{4,5}$	4556	2433	X	358	358	358
$S^{12}_{7,8}$	8318	4330	X	520	520	520
S^{12}_{4-8}	4455	2340	X	384	384	384
$S^{12}_{6,7}$	10430	5463	5245	494	494	494
$S^{12}_{5,6}$	10430	5463	X	422	422	422
$S^{13}_{7,8}$	20165	10261	9667	622	622	622
S^{13}_{4-8}	10584	5336	X	500	500	500
$S^{13}_{6,7}$	22308	11518	X	590	590	590
$S^{13}_{5,6}$	20165	10261	X	550	550	550
$S^{14}_{8,9}$	37039	18596	X	826	826	826
S^{14}_{5-9}	22022	11262	X	700	700	700
$S^{14}_{5,6}$	37039	18596	X	700	700	700
$S^{14}_{6,7}$	45476	22877	20161	770	770	770
S^{15}_{5-9}	50052	24671	X	750	750	750
$S^{15}_{5,6}$	65067	32312	X	750	750	750
$S^{15}_{7,8}$	96525	47950	X	855	855	855

z dwu- i trzypoziomowymi sieciami łączącymi mają złożoność $O(n^2)$, przy czym złożoność sieci łączącej dwupoziomowej jest prawie dwukrotnie większa od złożoności sieci trzypoziomowej.

Zmodyfikowana sieć sortująca Batchera OE zaproponowana przez nas w niniejszej pracy ma najmniejszą złożoność. Interesujący jest również układ wykorzystujący BDD, który w analizowanych przypadkach ma złożoność podobną do złożoności układu wielopoziomowego z trzypoziomową siecią łączącą. Jednak złożoność naszego układu, aczkolwiek mniejsza o pojedyncze bramki dla małych układów, staje się mniejsza o setki procent w przypadku większych układów.

W tablicy 6.5 podano liczby ścieżek różnych implementacji funkcji symetrycznych. W każdym z analizowanych przypadków liczba ścieżek rośnie wykładniczo. Oczywiście, największa liczba ścieżek występuje w układach o stałej liczbie poziomów (kolumny [31], [12]). Podobnie w przypadku liczby bramek, liczba ścieżek ścieżek rośnie lawinowo dla $n \geqslant 12$, przy czym jest już bardzo duża dla $n \geqslant 10$. W układzie zaproponowanym przez nas wzrost liczby ścieżek jest istotnie wolniejszy niż w układach dwu-, trzy- i czteropoziomowych. Jest to bezpośrednio związane z tym, że przy stałej liczbie poziomów liczba literałów w wyrażeniu oznaczającym funkcję jest wprost proporcjonalna do liczby wejść bramek i liczby ścieżek. Implementacja wykorzystująca BDD (kolumna [80]) ma niewielką złożoność (tablica 6.4), ale gwałtowny skok liczby ścieżek jest jeszcze bardziej charakterystyczny. Wyjątkowo interesujące jest porównanie liczby ścieżek układów wielopoziomowych. Otóż wszystkie układy wielopoziomowe zrealizowane jako całkowicie rozgałęzione mają taką samą liczbę ścieżek. Wynika to z faktu, że silna testowalność implikuje występowanie własności SP, która ściśle określa zbiór ścieżek w sieci łączącej. Każda sieć łącząca, niezależnie od wersji (dwu- lub trzypoziomowy lub zmodyfikowana sieć łącząca Batchera OE), w sensie liczby i rozkładu ścieżek jest taki sam. Liczba ścieżek w wielowyjściowym układzie progowym zrealizowanym jako całkowicie rozgałęziony jest taka sama, niezależna od implementacji sieci łączącej. Przypomnijmy jednak, że zaproponowany przez nas układ, pomimo identycznej liczby ścieżek, ma wielokrotnie mniejszą złożoność (tab. 6.2, 6.3) co daje mu bezdyskusyjną przewagę nad wszystkimi innymi układami znanymi w literaturze. Przez to liczba ścieżek w kolumnach [64], [73], oraz BAT-M (PP) jest jednakowa. Prawdopodobnie także liczba testów jest taka sama dla każdego układu wielopoziomowego. Również uważamy za prawdopodobne, że jest to minimalna liczba ścieżek w wielowyjściowym układzie progowym zrealizowanym jako całkowicie rozgałęziony, lecz rozważania te wykraczają poza zakres niniejszej pracy.

6.4 Podsumowanie

W rozdziale przedstawiliśmy wyniki syntezy różnych układów implementujących funkcje symetryczne oraz obliczenia mające na celu wykazanie prawdziwości założeń przyjętych przy modyfikacji sieci Batchera, opisanej w rozdziale 5. Celem uzyskania charakterystyk numerycznych napisaliśmy oprogramowanie:

- implementujące algorytmy syntezy sieci łączących Batchera i modyfikacji sieci łączącej powodującej nabycie własności SP i RG,
- implementujące algorytmy weryfikacji sieci zmodyfikowanej oraz
- wyznaczające złożoność i liczbę ścieżek sieci łączących dwu- i trzypoziomowych.

Porównanie wyników syntezy układów według podanych przez nas algorytmów z najlepszymi wynikami zawartymi w znanej literaturze pozwala nam sądzić, że nasze implementacje mają obecnie najmniejszą złożoność.

Rozdział 7

Uwagi końcowe

Wraz z postępującą miniaturyzacją układów VLSI wzrasta liczba występujących defektów, spośród których pewne klasy modelują uszkodzenia typu opóźnienia. Testowanie uszkodzeń typu opóźnienia zostało uznane za niezbędne w procesie wytwarzania układu [15], ale wykorzystanie do tego celu metod ogólnych jest najczęściej kosztowne i nieefektywne. Alternatywą jest synteza układów łatwo testowalnych czyli takich, w których już na etapie projektowania uwzględnia się uwarunkowania procesu testowania. Istnieje szereg szczególnych klas układów cyfrowych z jednej strony mających wiele zastosowań praktycznych a z drugiej strony mających regularną strukturę i podatnych na modyfikacje ułatwiające testowanie. Do takich układów należą niektóre układy realizujące funkcje symetryczne i progowe, o wielu zastosowaniach takich jak: układy arytmetyczne, układy redukcji mocy zasilania, elementy układów asynchronicznych, filtry cyfrowe, sieci neuronowe i inne.

Celem niniejszej rozprawy było opracowanie metod syntezy układów o złożoności mniejszej od najlepszych układów znanych z literatury implementujących funkcje symetryczne i progowe mających własność silnej testowalności uszkodzeń typu opóźnienie ścieżki (silnie testowalnych). Najmniejszą złożoność mają wielowyjściowe układy progowe o strukturze całkowicie rozgałęzionej, do których należą: wielowyjściowy układ progowy z dwupoziomową siecią łączącą Edwardsa [21], wielowyjściowy układ progowy z trzypoziomową siecią łączącą Rahamana i innych [73] oraz sieć sortująca nieparzysto-parzysta Batchera [5, 64]. Najmniejszym wielowyjściowym układem progowym jest sieć sortująca nieparzysto-parzysta Batchera [5], zaś najmniejszym ze znanych silnie testowalnym wielowyjściowym układem progowym jest wielowyjściowy układ progowy z trzypo-

ziomową siecią łączącą zaprezentowany przez Rahamana i innych [73]. Zauważmy, że złożoność układu Rahamana i innych [73] jest $O(n^2)$, zaś złożoność sieci sortującej nieparzysto-parzystej Batchera [5] $O(n \log^2 n)$, a więc jest istotnie mniejsza. Zauważyliśmy, że wielowyjściowe układy progowe o strukturze całkowicie rozgałęzionej różnią się pomiędzy sobą tylko strukturą sieci łączącej. Bazując na tej obserwacji wykazaliśmy, że własności silnej testowalności wielowyjściowego układu progowego zależą tylko od własności sieci łączącej [62], co przedstawiliśmy w rozdz. 4. Następnie w oparciu o tą własność podaliśmy modyfikację sieci łączącej nieparzysto-parzystej Batchera, dzięki której układ oparty na tej sieci łączącej jest silnie testowalny przy złożoności pozostającej $O(n \log^2 n)$. Analizy struktury sieci oraz algorytmy i przykłady modyfikacji przedstawiliśmy w rozdz. 5. Algorytmy opisanej modyfikacji zostały przez nas zaimplementowane w postaci pakietu oprogramowania do automatycznej syntezy. Dokonane następnie porównanie wyników syntezy przedstawione w rozdz. 6 wykazało, że wielowyjściowy układ progowy zrealizowany jako modyfikacja sieci sortującej nieparzysto-parzystej Batchera jest silnie testowalny, a przy tym mniejszy od wielowyjściowego układu progowego z trzypoziomową siecią łączącą Rahamana i innych [73], czyli najmniejszego silnie testowalnego wielowyjściowego układu progowego i układu realizującego funkcje symetryczne znanego w literaturze, czym wykazaliśmy tezę postawioną na początku niniejszej rozprawy.

Otwartym problemem pozostaje analityczne wyznaczenie liczby ścieżek i liczby testów w wielowyjściowych układach progowych zrealizowanych jako całkowicie rozgałęzione. Znalezienie analitycznego zapisu wzorców testowych mogłoby pozwolić na zastosowanie różnych rozwiązań automatycznego generowania testów lub samotestowania (BIST) [30, 33, 59, 75, 88], jak również kompresji uzyskanych zbiorów testów. Brak jest również analiz własności silnej testowalności uszkodzeń typu opóźnienie ścieżki w wielowyjściowych układach progowych zrealizowanych jako k-ścieżkowe (ang. k-way sorters). Tą ścieżkę badań uważamy za atrakcyjną, zwłaszcza w świetle prowadzonych ostatnio badań nad k-ścieżkowymi elementami progowymi implementowanymi przez Piestraka i Berezowskiego w postaci bramek złożonych CMOS na poziomie tranzystorów.

Warto także zwrócić uwagę na ciekawą własność usuwania nietestowalnych ścieżek kosztem niewielkiego zwiększenia złożoności układu, co pokazaliśmy przy okazji modyfikacji sieci sortującej nieparzysto-parzystej Batchera. Bardzo interesującym obiektem takich badań nad możliwością takiej modyfikacji może być znany układ testowy (ang. benchmark circuit) *c6288*, w którym liczba

ścieżek wynosi ok. 10^{20}. Zauważmy, że z jednej strony istnieją propozycje modyfikacji układów celem uzyskania testowalności uszkodzeń typu sklejenia. Z drugiej strony modyfikacje ukierunkowane na ułatwienie testowania uszkodzeń typu opóźnienie ścieżki, czy też uszkodzeń typu opóźnienia w ogólności, nie pojawiają się w znanej nam literaturze, zaś podjęcie badań w tym kierunku mogłoby w istotny sposób zwiększyć możliwość syntezy wysokiej jakości układów logicznych, w których uszkodzenia typu opóźnienia są łatwo testowalne.

Bibliografia

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital System Testing and Testable Design*. Computer Science Press, 1990.

[2] T. Agerawala and S. Chatterjee. Computer architecture: Challenges and opportunities for the next decade. *IEEE Micro*, vol. 25, no. 3, pp. 58–69, May–June 2005.

[3] L. Ali, R. Sidek, I. Aris, B. S. Suparjo, and M. A. M. Ali. Challenges and directions for testing IC. *Integration, the VLSI J.*, vol. 37, no. 1, pp. 17–28, Feb. 2004.

[4] K. Amano and A. Maruoka. On optimal merging networks. *Lect. Notes Comput. Sci.*, vol. 2747, pp. 152–161, 2003.

[5] K. E. Batcher. Sorting networks and their applications. *Proc. AFIPS Spring Joint Comput. Conf.*, pp. 307–314, 1968.

[6] V. Beiu, J. M. Quintana, and M. J. Avedillo. VLSI implementations of threshold logic— a comprehensive survey. *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1217–1243, Sept. 2003.

[7] B. Benware, C. Lu, P. Krishnamurthy, R. Madge, M. Keim, M. Kassab, and J. Rajski. Affordable and effective screening of delay defects in ASICs using the inline resistance fault model. *Proc. Int. Test Conf.*, pp. 1285–1294, 2004.

[8] L. Breveglieri and V. Piuri. Digital median filters. *J. VLSI Signal Process.*, vol. 31, no. 3, pp. 191–206, July 2002.

[9] Cadence Design Systems, Inc. *Nanometer Test: Methodology and Economics*, 2004. http://www.cadence.com/whitepapers/nanometer_test_wp.pdf.

[10] P. Celinski, S. D. Cotofana, J. F. Lopez, S. Al-Sarawi, and D. Abbott. State-of-the-art in CMOS threshold-logic VLSI gate implementations and applications. *Proc. Int. Conf. VLSI Circuits Syst.*, pp. 53–64, 2003.

[11] C. Chakrabarti. Sorting network based architectures for median filters. *IEEE Trans. Circuits Syst. II*, vol. 40, no. 11, pp. 723–727, Nov. 1993.

[12] C. Chakrabarti and L. Lucke. VLSI architectures for weighted order statistic (WOS) filters. *Signal Process.*, vol. 80, no. 8, pp. 1419–1433, Aug. 2000.

[13] C. Chakrabarti and L.-Y. Wang. Novel sorting network-based architectures for rank order filters. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 502–507, Dec. 1994.

[14] S. Chakrabarti, S. Das, D. K. Das, and B. B. Bhattacharya. Synthesis of symmetric functions for path-delay fault testability. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 9, pp. 1076–1081, Sept. 2000.

[15] K. T. Cheng, S. Dey, M. Rodgers, and K. Roy. Test challenges for deep sub-micron technologies. *Proc. Des. Aut. Conf.*, pp. 142–149, 2000.

[16] T. Cormen, C. Leiserson i R. Rivest. *Wprowadzenie do algorytmów.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1998.

[17] A. L. Crouch. *Design-for-Test for Digital IC's and Embedded Core Systems.* Prentice Hall PTR, 1998.

[18] R. David. *Random Testing of Digital Circuits: Theory and Applications.* Marcel Dekker, Inc., 1998.

[19] D. L. Dietmeyer. Generating minimal covers of symmetric functions. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 5, pp. 710–713, May 1993.

[20] R. L. S. Drysdale and F. H. Young. Improved divide/sort/merge sorting networks. *SIAM J. Comput.*, vol. 4, pp. 264–270, Sept. 1975.

[21] C. R. Edwards. Some improved designs for the digital summation threshold logic (DSTL) gate. *Comput. J.*, vol. 21, no. 1, pp. 73–78, 1978.

[22] M. Favalli and C. Metra. Online testing approach for very deep-submicron ICs. *IEEE Des. & Test Comput.*, vol. 19, no. 3, pp. 16–23, Mar.–Apr. 2002.

[23] M. Fukunaga, S. Kajihara, S. Takeoka, and S. Yosimura. On effective criterion of path selection for delay testing. *Proc. Asia and South Pacific Des. Aut. Conf.*, pp. 751–756, 2003.

[24] P. Gupta and M. S. Hsiao. ALATPF: A new transition fault model and the ATPG algorithm. *Proc. Int. Test Conf.*, pp. 1053–1060, 2004.

[25] K. Heragu, J. H. Patel, and V. D. Agrawal. Segment delay faults: A new fault model. *Proc. VLSI Test Symp.*, pp. 32–39, 1996.

[26] R. Hiremane. From Moore's law to Intel innovation—prediction to reality. *Technol.@Intel Mag.*, vol. 2, Apr. 2005.

[27] J. Hugues. Evolution of non-deterministic incremental algorithms as a new approach for search in state spaces. *Proc. Int. Conf. Genetic Alg.*, pp. 351–358, 1995.

[28] S. Iwata. Lower bounds for merging networks. *Inform. Computation*, vol. 168, no. 2, pp. 187–195, 2001.

[29] S. Jiang and E. Fujiwara. Unidirectional byte error locating codes. *IEICE Trans. Fundamentals Electron. Communications Comput. Sci.*, vol. E77-A, no. 8, pp. 1253–1259, Aug. 1994.

[30] S. Kajihara, M. Fukunaga, X. Wen, T. Maeda, S. Hamada, and Y. Sato. Path delay test compaction with process variation tolerance. *Proc. Des. Aut. Conf.*, pp. 845–850, 2005.

[31] W. Ke and P. R. Menon. Delay-testable implementations of symmetric functions. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 6, pp. 772–775, June 1995.

[32] W. Ke and P. R. Menon. Synthesis of delay-verifiable combinational circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 2, pp. 213–222, Feb. 1995.

[33] M. Keim, I. Polian, H. Hengster, and B. Becker. A scalable BIST architecture for delay faults. *Proc. Eur. Test Workshop*, pp. 98–103, 1998.

[34] B.-G. Kim and D. L. Dietmeyer. Multilevel logic synthesis of symmetric functions. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 10, no. 4, pp. 436–446, Apr. 1991.

[35] D. Knuth. *Sztuka programowania, tom 3 – Sortowanie i wyszukiwanie.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2002.

[36] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, 1977.

[37] C. Kotropoulos, M. Pappas, and I. Pitas. Sorting networks using L_p mean comparators for signal processing applications. *IEEE Trans. Signal Process.*, vol. 50, no. 11, pp. 2716–2729, Nov. 2002.

[38] A. Krstič and K.-T. Cheng. *Delay Fault Testing for VLSI Circuits.* Kluwer Academic Publishers, 1998.

[39] B. Kruseman, A. K. Majhi, G. Grouthoud, and S. Eichenberger. On hazard-free patterns for fine-delay fault testing. *Proc. Int. Test Conf.*, pp. 213–222, 2004.

[40] W. K. Lam, A. Saldanha, R. K. Brayton, and A. Sangiovanni-Vincentelli. Delay fault coverage, test set size, and performance trade-offs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 1, pp. 32–44, Jan. 1995.

[41] E. A. Lamagna. The complexity of monotone networks for certain bilinear forms, routing problems, sorting, and merging. *IEEE Trans. Comput.*, vol. C-28, pp. 773–782, Oct. 1979.

[42] C. L. Lee and C. W. Jen. Bit-sliced median filter design based on majority gate. *IEE Proc. Circuits, Devices Syst.*, vol. 139, no. 1, pp. 63–71, Feb. 1992.

[43] K. D. Lee and Y. H. Lee. Threshold boolean filters. *IEEE Trans. Signal Process.*, vol. 42, no. 8, pp. 2022–2036, Aug. 1994.

[44] L. Lee, L. C. Wang, T. M. Mak, and K.-T. Cheng. A path-based methodology for post-silicon timing validation. *Proc. Int. Conf. Comput.-Aided Des.*, pp. 713–720, 2004.

[45] J. C.-M. Li and E. J. McCluskey. Diagnosis of resistive-open and stuck-open defects on digital CMOS ICs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 11, pp. 1748–1759, Nov. 2005.

[46] C. C. Lin and C. J. Kuo. Two-dimensional rank-order filter by using max-min sorting network. *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 8, pp. 941–946, Dec. 1998.

[47] C. J. Lin and S. M. Reddy. On delay-fault testing in logic circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. CAD-6, no. 5, pp. 694–703, Sept. 1987.

[48] X. Lin and J. Rajski. Propagation delay fault: A new fault model to test delay faults. *Proc. Asia and South Pacific Des. Aut. Conf.*, pp. 178–183, 2005.

[49] J.-J. Liu, L.-C. Wang, A. Krstič, and K.-T. Cheng. On effective criterion of path selection for delay testing. *Proc. Asia and South Pacific Des. Aut. Conf.*, pp. 757–762, 2003.

[50] L. Lucke and K. Parhi. Parallel processing architectures for rank order and stack filters. *IEEE Trans. Signal Process.*, vol. 42, no. 5, pp. 1178–1189, May 1994.

[51] A. K. Majhi and V. D. Agrawal. Tutorial: Delay fault models and coverage. *Proc. Int. Conf. VLSI Des.*, pp. 364–369, 1997.

[52] T. M. Mak, A. Krstič, K.-T. Cheng, and L.-C. Wang. New challenges in delay testing of nanometer, multigigahertz designs. *IEEE Des. & Test Comput.*, vol. 21, no. 2, pp. 241–247, May–June 2004.

[53] C. Metra, L. Schiano, and M. Favalli. Concurrent detection of power supply noise. *IEEE Trans. Rel.*, vol. 52, no. 4, pp. 469–475, Dec. 2003.

[54] S. Mitra, N. R. Saxena, and E. J. McCluskey. Common-mode failures in redundant VLSI systems: A survey. *IEEE Trans. Rel.*, vol. 49, no. 3, pp. 285–295, Sept. 2000.

[55] R. R. Montanes, J. P. de Gyvez, and P. Wolf. Resistance characterization for weak open defects. *IEEE Des. & Test Comput.*, vol. 19, no. 5, pp. 18–26, Sept.–Oct. 2002.

[56] G. E. Moore. Cramming more components onto integrated circuits. *Electron. Mag.*, vol. 38, no. 8, pp. 56–59, Aug. 1965.

[57] S. Moore, R. Anderson, R. Mullins, G. Taylor, and J. Fournier. Balanced self-checking asynchronous logic for smart card applications. *Microprocess. Microsyst.*, vol. 27, no. 9, pp. 421–430, Oct. 2003.

[58] G. Moronashi and S. Iwata. Some minimum merging networks. *Theoret. Comput. Sci.*, vol. 329, no. 1/2, pp. 237–250, Dec. 2004.

[59] G. Mrugalski, N. Mukherjee, J. Rajski, and J. Tyszer. High performance dense ring generators. *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 83–87, Jan. 2006.

[60] W. M. Needham. Nanometer technology challenges for test and test equipment. *Computer*, vol. 32, no. 11, pp. 52–57, Nov. 1999.

[61] S. Padmanaban and S. Tragoudas. A critical path selection method for delay testing. *Proc. Int. Test Conf.*, pp. 232–241, 2004.

[62] P. Patronik. Delay testability properties of circuits implementing threshold and symmetric functions. *Proc. Euromicro Conf. Digit. Syst. Des.*, pp. 289–297, 2005.

[63] S. J. Piestrak. The minimal test set for sorting networks and the use of sorting networks in self-testing checkers for unordered codes. *Proc. Int. Fault Tolerant Computing Symp.*, pp. 467–474, June 26–28 1990.

[64] S. J. Piestrak. The minimal test set for multioutput threshold circuits implemented as sorting networks. *IEEE Trans. Comput.*, vol. 42, no. 6, pp. 700–712, June 1993.

[65] S. J. Piestrak. Design of residue generators and multioperand adders modulo-3 built of multi-output threshold circuits. *IEE Proc. Comput. Digit. Tech.*, vol. 141, no. 2, pp. 129–134, Mar. 1994.

[66] S. J. Piestrak. *Design of self-testing checkers for unidirectional error detecting codes*, rozdział 4.2, pp. 41–43. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej, No. 92, Monografie No. 24. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1995.

[67] S. J. Piestrak. Design of encoders and self-testing checkers for some systematic unidirectional error detecting codes. *Int. J. Microelectronic Syst. Integration*, vol. 5, no. 4, pp. 246–260, 1997.

[68] S. J. Piestrak. Membership test logic for delay-insensitive codes. *Proc. Int. Symp. Adv. Res. Asynchronous Circuits Syst.*, pp. 194–204, 1998.

[69] S. J. Piestrak. Metody tolerowania uszkodzeń w układach cyfrowych. Niepublikowane materiały do wykładu z przedmiotu FTC, Instytut Cybernetyki Technicznej, Politechnika Wrocławska, Wrocław, marzec 2005.

[70] A. K. Pramanick and S. M. Reddy. On the detection of delay faults. *Proc. Int. Test Conf.*, pp. 845–856, 1988.

[71] W. Qiu, X. Lu, J. Wang, Z. Li, D. M. H. Walker, and W. Shi. A statistical coverage metric for realistic path delay faults. *Proc. VLSI Test Symp.*, pp. 37–42, 2004.

[72] W. Qiu and D. M. H. Walker. An efficient algorithm for finding the K longest testable paths through each gate in a combinational circuit. *Proc. Int. Test Conf.*, pp. 592–601, 2003.

[73] H. Rahaman, D. K. Das, and B. B. Bhattacharya. Mapping symmetric functions to hierarchical modules for path-delay fault testability. *Proc. Asian Test Symp.*, pp. 284–289, 2003.

[74] J. Rajski and J. Tyszer. *Arithmetic Built-In Self-Test.* Prentice Hall PTR, 1998.

[75] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded deterministic test. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

[76] S. M. Reddy and J. R. Wilson. Easily testable cellular realizations for the (exactly p)-out-of n and (p or more)-out-of n logic functions. *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 98–100, Jan. 1987.

[77] J. Savir and S. Patil. Scan-based transition test. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 8, pp. 1232–1241, Aug. 1993.

[78] J. Savir and S. Patil. On broad side delay test. *Proc. VLSI Test Symp.*, pp. 368–372, 1994.

[79] M. Sharma and J. H. Patel. Testing of critical paths for delay faults. *Proc. Int. Test Conf.*, pp. 634–641, 2001.

[80] J. Shi, G. Fey, and R. Drechsler. BDD based synthesis of symmetric functions with full path-delay fault testability. *Proc. Asian Test Symp.*, pp. 290–293, 2003.

[81] G. L. Smith. Models for delay faults based on paths. *Proc. Int. Test Conf.*, pp. 342–349, 1985.

[82] U. Sparman, D. Luxenburger, K.-T. Cheng, and S. M. Reddy. Fast identification of robust dependent path delay faults. *Proc. Des. Aut. Conf.*, pp. 119–125, 1995.

[83] M. R. Stan and W. P. Burleson. Bus-invert coding for low-power I/O. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[84] T. Surmacz. Metody generowania minimalnego zbioru testów dla pewnych klas sieci sortujących. Praca doktorska. Rap. tech. PRE nr 38, Instytut Cybernetyki Technicznej, Politechnika Wrocławska, Wrocław, 2004.

[85] Synopsys, Inc. *TetraMAX ATPG*, 2001. http://www.synopsys.com/products/test/tetramax_dsA4.pdf.

[86] J. F. Tabor. Noise reduction using low weight and constant weight coding techniques. Technical Report 1232, MIT Artif. Intell. Lab., 1990.

[87] L. Tallini, L. Merani, and B. Bose. Balanced codes for noise reduction in VLSI systems. *Proc. Int. Fault Tolerant Computing Symp.*, pp. 212–218, 1994.

[88] S. Tragoudas and V. Nagarandal. On-chip embedding mechanisms for large sets of vectors for delay test. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 3, pp. 488–497, Mar. 2005.

[89] J. Turino. Design for test and time to market: A personal perspective. *IEEE Des. & Test Comput.*, vol. 16, no. 3, pp. 34–43, July–Sept. 1999.

[90] G. Umanesan and E. Fujiwara. A class of codes for correcting single spotty byte errors. *IEICE Trans. Fundamentals Electron. Communications Comput. Sci.*, vol. E86-A, pp. 704–714, Mar. 2003.

[91] J. P. Uyemura. *CMOS Logic Circuit Design*. Kluwer Academic Publishers, 1999.

[92] D. C. Van Voorhis. An economical construction for sorting networks. *Proc. AFIPS National Comput. Conf.*, pp. 307–314, 1974.

[93] M. Wondolowski, B. Bennets, and A. Ley. Boundary scan: The internet of test. *IEEE Des. & Test Comput.*, vol. 16, no. 3, pp. 23–27, July–Sept. 1999.

[94] H. Yan and A. D. Singh. A delay test to differentiate resistive interconnect faults from weak transistors defects. *Proc. Int. Conf. VLSI Des.*, pp. 47–52, 2005.

[95] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted median filters: A tutorial. *IEEE Trans. Circuits Syst. II*, vol. 43, no. 3, pp. 157–192, Mar. 1996.

[96] J. H. Youn and B. Bose. Efficient encoding and decoding schemes for balanced codes. *IEEE Trans. Comput.*, vol. 52, pp. 1229–1232, Sept. 2003.

[97] R. Zhang, P. Gupta, L. Zhong, and N. K. Jha. Threshold network synthesis and optimization and its application to nanotechnologies. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 1, pp. 107–118, Jan. 2005.

[98] Q. Zhou and K. Mohanram. Analysis od delay caused by bridging faults in RLC interconnects. *Proc. Int. Test Conf.*, pp. 1044–1052, 2004.