AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI

Zbigniew Nagórny

Problem projektowania topografii systemów wielkiej skali integracji

PRACA DOKTORSKA

Promotor:
prof. dr hab. inż. Andrzej Kos

Kraków, 2007

Pragnę złożyć serdeczne podziękowania Panu Profesorowi Andrzejowi Kosowi za wszelką pomoc w realizacji niniejszej pracy, udostępnienie bogatej literatury oraz umożliwienie przeprowadzenia badań. Dziękuję za inspirację, cenne wskazówki, otwartość na współpracę, życzliwość, i za wiarę w powodzenie, która była dla mnie motywacją do dalszej pracy.

„Nawet podróż na dystans 1000 mil
musi rozpocząć się od postawienia pierwszego kroku.”
Lao Tsu

Dedykuję Mojej Rodzinie

Spis treści

Rozdział 1
Wstęp. Geneza pracy

Tradycyjny system obliczeniowy wykorzystuje procesor ogólnego przeznaczenia. Architektura takich procesorów została zaproponowana przez von Neumanna. Współczesna nauka stawia coraz większe wymagania na moc obliczeniową systemów komputerowych. Projektanci procesorów oraz systemów obliczeniowych starają się sprostać tym wymaganiom. W wyniku zastosowania wielu nowych rozwiązań, moc obliczeniowa systemów komputerowych stale wzrasta.

W ciągu minionych lat rozwoju cyfrowych systemów obliczeniowych, częstotliwość sygnału taktującego stale wzrastała. Wzrost częstotliwości sygnału taktującego jest możliwy dzięki stale postępującej miniaturyzacji układów scalonych, która umożliwia zmniejszenie czasu propagacji sygnałów w układach cyfrowych. Większa częstotliwość sygnału taktującego zapewnia wzrost mocy obliczeniowej systemu komputerowego. Dalsza miniaturyzacja układów scalonych napotyka jednak na coraz większe trudności technologiczne.

Stale są prowadzone również prace nad udoskonalaniem wewnętrznej architektury procesorów, co umożliwia zwiększenie ich wydajności. Innym rozwiązaniem, które umożliwia wzrost mocy obliczeniowej są wieloprocesorowe systemy obliczeniowe. Wzrost mocy obliczeniowej jest możliwy dzięki wprowadzeniu równoległego przetwarzania, polegającego na równoczesnej pracy wielu procesorów. Równoległość obliczeń w systemach wieloprocesorowych jest możliwa w wyniku podziału zadań między wszystkie procesory systemu. W wyniku rozwoju systemów wieloprocesorowych powstało wiele specjalizowanych procesorów, przeznaczonych do wykonywania ściśle określonych zadań. Stale są prowadzone badania nad architekturą systemów wieloprocesorowych, która zapewni optymalne komunikowanie się procesorów w systemie.

Ograniczeniem procesorów opartych na architekturze von Neumanna jest jednak konieczność ciągłego pobierania kodów rozkazów z pamięci. Z tego względu od wielu lat są stosowane specjalizowane akceleratory sprzętowe, oparte na układach logicznych. Zaletą powyższego rozwiązania jest bardzo duży wzrost mocy obliczeniowej, ze względu na możliwość równoległego przetwarzania i brak konieczności ciągłego pobierania kodów rozkazów z pamięci. Sprzętowe akceleratory obliczeń były do tej pory najczęściej wykonywane z użyciem dedykowanych układów ASIC. Postęp w dziedzinie programowalnych i rekonfigurowalnych układów logicznych umożliwia obecnie wykonywanie sprzętowych akceleratorów obliczeń z użyciem układów FPGA. Rekonfigurowalne układy FPGA są powszechnie stosowane w wielu dziedzinach elektroniki, telekomunikacji, w robotyce, astronomii, medycynie. W wyniku powstania koncepcji tworzenia wirtualnych modułów w oparciu o język opisu sprzętu VHDL (ang. VHSIC Hardware Description Language), powstał tzw. rynek własności intelektualnej IP (ang. Intellectual Property). Moduły własności intelektualnej (ang. IP Core) umożliwiają wielokrotne

wykorzystywanie raz opracowanych modułów. Możliwe jest również budowanie większych układów z wielu modułów własności intelektualnej. Powyższe rozwiązanie umożliwia znaczne skrócenie czasu projektowania układu. Możliwa jest również realizacja układu w wyniku przeniesienia jego projektu do nowej technologii wytwarzania układów scalonych. Współczesne układy scalone zawierają tak dużą liczbę tranzystorów, że możliwa jest realizacja w jednym układzie procesora z pamięcią i interfejsami oraz rekonfigurowalnej logiki. Powyższa koncepcja jest określana terminem System-on-Chip (SoC).

Część układów rekonfigurowalnych posiada zdolność do wymiany danych konfiguracyjnych w czasie pracy układu (ang. run-time reconfiguration). Układy ze zdolnością rekonfiguracji w „locie" są szczególnie użyteczne w telekomunikacji, systemach wizyjnych, robotyce, systemach obliczeniowych. W przedstawionych zastosowaniach algorytmy realizowane z użyciem układów rekonfigurowalnych „w locie" muszą jednak być wcześniej w całości przygotowane. Możliwy jest tylko wybór algorytmu spośród wcześniej przygotowanych rozwiązań. Nie jest możliwe tworzenie nowych algorytmów podczas pracy układu. Powodem tego jest zbyt długi czas potrzebny na przygotowanie takiego algorytmu. Realizacja kodu zapisanego w języku VHDL wymaga syntezy, rozmieszczenia komórek logicznych układu oraz wyznaczenia połączeń. Rozmieszczanie jest etapem, który wymaga najdłuższych obliczeń.

Przedstawione tendencje w rozwoju mikroelektroniki i systemów obliczeniowych stały się motywacją do podjęcia prac w kierunku sprzętowej realizacji obliczeń oraz zastosowania równoległego przetwarzania podczas rozmieszczania komórek logicznych w systemach VLSI. Sprzętowa realizacja algorytmu rozmieszczania komórek logicznych umożliwiłaby znaczne skrócenie czasu obliczeń, nawet o kilka rzędów wielkości, w porównaniu do czasu wykonywania programów rozmieszczania na tradycyjnych komputerach. Sprzętowe rozwiązanie problemu rozmieszczania umożliwiłoby opracowywanie i realizację nowych algorytmów przetwarzania układu rekonfigurowalnego w sposób dynamiczny, podczas pracy układu. W tym przypadku czas potrzebny na przygotowanie i realizację nowego algorytmu przetwarzania odgrywa decydującą rolę. Prace nad sprzętową realizacją algorytmu rozmieszczania i trasowania połączeń, pod kątem zastosowania w układach rekonfigurowalnych, są już prowadzone w różnych ośrodkach naukowych. Sprzętowa realizacja algorytmu rozmieszczania może znaleźć również zastosowanie w układach rekonfigurowalnych, które są poddawane częściowej rekonfiguracji (ang. partial reconfiguration). Powyższe rozwiązanie może znacznie skrócić czas częściowej rekonfiguracji układu. Sprzętowe rozwiązanie problemu rozmieszczania jest bardzo ważne dla wszystkich rodzajów układów VLSI. Narzędzia komputerowego wspomagania projektowania układów scalonych CAD nie „nadążają" za bardzo szybkim rozwojem technologii wytwarzania układów scalonych. Problem ten będzie nabrzmiewał wraz ze wzrostem liczby tranzystorów w układzie. Narzędzia projektowania muszą uwzględniać coraz większą liczbę zjawisk, które powinny być wzięte pod uwagę wraz ze wzrostem gęstości upakowania tranzystorów w układzie, w wyniku wprowadzenia nanotechnologii. Sprzętowa realizacja algorytmu rozmieszczania komórek logicznych zasługuje więc na szczególną uwagę. Powyższe rozwiązanie może być użyteczne podczas projektowania wszystkich rodzajów układów VLSI.

1.1. Cel i teza pracy

Celem pracy jest sprawdzenie możliwości zastosowania sieci Hopfielda w problemie optymalizacji rozmieszczenia komórek logicznych w cyfrowych systemach VLSI. Na wstępie dokonano przeglądu metod stosowanych podczas rozmieszczania. Szczególną uwagę zwrócono na rozwiązania, które mogą być zastosowane w sprzętowej realizacji algorytmu rozmieszczania. W chwili obecnej występuje duże zapotrzebowanie na efektywne narzędzia projektowania rozległych systemów SoC. Nowe technologie generują nowe wyzwania i stwarzają potrzebę doskonalenia istniejących metod i narzędzi. Stale postępująca miniaturyzacja wymiarów tranzystorów umożliwia ciągłe zmniejszanie opóźnień wnoszonych przez elementy aktywne układu oraz zwiększanie częstotliwości roboczej systemów VLSI. W obecnych systemach VLSI, opóźnienia wnoszone przez połączenia wywierają decydujący wpływ na właściwości funkcjonalne układu, w porównaniu do opóźnień wnoszonych przez elementy aktywne. Występuje mocno zarysowana potrzeba optymalizacji długości połączeń. Ze względu na właściwości równoległego przetwarzania, potencjalną możliwość sprzętowej realizacji oraz zachęcające rezultaty otrzymane podczas rozmieszczania modułów istotnych termicznie, wybrano metodę wykorzystującą sieć neuronową Hopfielda. Głównym celem pracy jest gruntowne zbadanie możliwości zastosowania sieci Hopfielda w problemie optymalizacji długości połączeń w systemach VLSI. W celu osiągnięcia głównego celu pracy wyznaczono następujące cele szczegółowe:

1. Przegląd metod rozmieszczania stosowanych podczas projektowania systemów VLSI
2. Opracowanie metod optymalizacji długości połączeń w systemach VLSI z wykorzystaniem zmodyfikowanej przez autora sieci Hopfielda
3. Opracowanie metody estymacji długości połączeń w układach VLSI, która może być wykorzystana podczas optymalizacji długości połączeń z użyciem sieci Hopfielda
4. Wskazanie zalet i wad zastosowania sieci Hopfielda do optymalizacji długości połączeń w systemach VLSI
5. Wskazanie możliwości aplikacyjnych sieci Hopfielda w projektowaniu topografii systemów VLSI
6. Zbadanie możliwości zastosowania opracowanych metod optymalizacji w ogólnych problemach optymalizacyjnych.

Badanie możliwości zastosowania sieci Hopfielda przeprowadzono w oparciu o symulację sieci.

Na podstawie wiedzy zawartej w literaturze oraz własnych doświadczeń autora, sformułowano następującą tezę pracy:

Teza: Optymalizację długości połączeń w układach VLSI można przeprowadzić z wykorzystaniem zmodyfikowanej sieci Hopfielda, z równoległym przetwarzaniem, która może dostarczać rozwiązań o bardzo dobrej jakości.

1.2. Organizacja pracy

Niniejsza praca składa się z 10 rozdziałów. W rozdziale 1 przedstawiono genezę pracy, cel pracy oraz tezę pracy.

W rozdziale 2 opisano różne style topografii układów VLSI. Przedstawiono etapy projektowania topografii układu VLSI. Opisano problem rozmieszczania modułów. Omówiono sposoby estymacji długości połączeń. Opisano metody minimalizacji opóźnień w układzie.

W rozdziale 3 dokonano przeglądu metod rozmieszczania modułów w układach VLSI. Przedstawiono obecnie stosowane programy, które wykorzystują różne metody rozmieszczania. Porównano metody przedstawione w przeglądzie.

W rozdziale 4 przedstawiono metodę rozmieszczania modułów w układzie VLSI, która wykorzystuje sieć neuronową Hopfielda. Celem rozmieszczenia modułów jest zapewnienie minimalnej sumarycznej długości połączeń w układzie. Określono postać funkcji kosztu, która jest minimalizowana przez sieć. Przedstawiono analog elektryczny sieci Hopfielda oraz sposób symulacji sieci.

W rozdziale 5 przedstawiono oryginalne modyfikacje klasycznego modelu sieci Hopfielda. Przedstawiono wpływ modyfikacji na rozwiązania otrzymywane przez sieć Hopfielda, podczas rozmieszczania modułów z minimalizacją długości połączeń. Wykonano rozmieszczenie modułów dla siedmiu układów, dla których zamieszczono pełne informacje określające połączenia w poszczególnych układach. Rozpatrzono różne sposoby inicjalizacji neuronów w sieci Hopfielda.

W rozdziale 6 przedstawiono oryginalne zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z ustalonym położeniem końcówek całego układu (padów). Przedstawiono oryginalną postać funkcji kosztu sieci Hopfielda dla tych układów. Opisano możliwość zastosowania sieci Hopfielda w układach, w których położenie kilku modułów jest trwale ustalone. Zbadano również równoczesne rozmieszczanie modułów oraz końcówek całego układu. Przedstawiono wyniki rozmieszczania dla badanych układów.

W rozdziale 7 przedstawiono oryginalne zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z węzłami posiadającymi wiele końcówek. Przedstawiono oryginalną postać funkcji kosztu sieci Hopfielda dla tych układów. Opisano oryginalny sposób korekcji estymacji długości połączeń dla węzłów zawierających wiele końcówek. Przedstawiono wyniki rozmieszczania otrzymane dla układów, w których węzły posiadają wiele końcówek.

W rozdziale 8 przedstawiono oryginalne rozwiązanie, polegające na zastosowaniu sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, który umożliwia minimalizację długości połączeń w układzie VLSI. Opisano rozwiązania wykorzystane podczas rozmieszczania modułów z użyciem równoległego przetwarzania. Zamieszczono wyniki rozmieszczania otrzymane z zastosowaniem równoległego przetwarzania.

W rozdziale 9 przedstawiono rezultaty zastosowania sieci Hopfielda w innych problemach optymalizacyjnych, na przykładzie problemu komiwojażera.

W rozdziale 10 opisano możliwości zastosowania sieci Hopfielda podczas projektowania topografii systemów VLSI. Zamieszczono podsumowanie rezultatów przeprowadzonych symulacji. Przedstawiono dalsze kierunki badań.

1.3. Oryginalne rozwiązania

W pracy zaproponowano i zastosowano następujące oryginalne pomysły:

1. wprowadzenie autosprzężeń (rozdział 5) w sieci Hopfielda i modyfikacja funkcji kosztu prowadzące do minimalizacji długości połączeń
2. metoda modyfikacji wartości sygnałów wejściowych neuronów sieci Hopfielda, umożliwiająca poprawę jakości rozwiązań (rozdział 5)
3. zastosowanie zmodyfikowanej sieci Hopfielda do minimalizacji długości połączeń w układzie z ustalonym położeniem padów układu (rozdział 6)
4. zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układzie z ustalonym z góry położeniem części modułów lub w przypadku, gdy pewne części podłoża nie mogą być wykorzystane do rozmieszczania modułów (rozdział 6)
5. zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układzie z równoczesnym rozmieszczaniem modułów oraz padów układu (rozdział 6)
6. zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z węzłami posiadającymi wiele końcówek (rozdział 7)
7. korekcja estymacji długości połączeń dla węzłów zawierających wiele końcówek, z użyciem aproksymacji minimalnego drzewa Steinera (rozdział 7)
8. zastosowanie sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, który umożliwia minimalizację długości połączeń w układzie (rozdział 8)

Rozdział 2
Style i etapy projektowania

Projektowanie topografii cyfrowego układu VLSI (ang. physical design process) jest bardzo trudnym problemem optymalizacyjnym. Rezultatem tego procesu jest wiedza prowadząca do fizycznej realizacji projektowanego układu. Projekt topografii (ang. layout) określa przede wszystkim obszary dyfuzji oraz metalizacji w układzie scalonym. Ogólnie można powiedzieć, że celem optymalizacji topografii układu VLSI jest minimalizacja jego powierzchni oraz zapewnienie możliwie najlepszych właściwości funkcjonalnych układu. Mniejsza powierzchnia układu umożliwia wykonanie większej liczby układów na jednostkowej powierzchni podłoża oraz powoduje zmniejszenie prawdopodobieństwa wystąpienia defektu podłoża w pojedynczym układzie. Podczas projektowania topografii należy również wziąć pod uwagę minimalizację opóźnień w krytycznych częściach układu, minimalizację traconej energii, sprzężeń między sygnałami oraz wiele innych kryteriów [1-4, 51, 70, 72, 138, 149].

Ze względu na rozmiary i złożoność obecnych układów VLSI, projektowanie topografii układów musi odbywać się z użyciem systemów projektowania wspomaganych komputerowo CAD (ang. Computer Aided Design). Systemy te umożliwiają znaczne skrócenie czasu niezbędnego do zaprojektowania układu, poprawienie jego jakości i niezawodności oraz zmniejszenie kosztów jego produkcji [3]. Współczesne systemy projektowania zawierają wiele narzędzi, których zadaniem jest usprawnienie procesu projektowania. Wśród tych narzędzi należy wymienić edytory topografii układu, narzędzia sprawdzające zgodność topografii z regułami projektowania i schematem elektrycznym układu, narzędzia generujące rzeczywisty model układu z uwzględnieniem pojemności i rezystancji pasożytniczych, pojemności, indukcyjności i rezystancji połączeń oraz inne narzędzia [1, 3, 5].

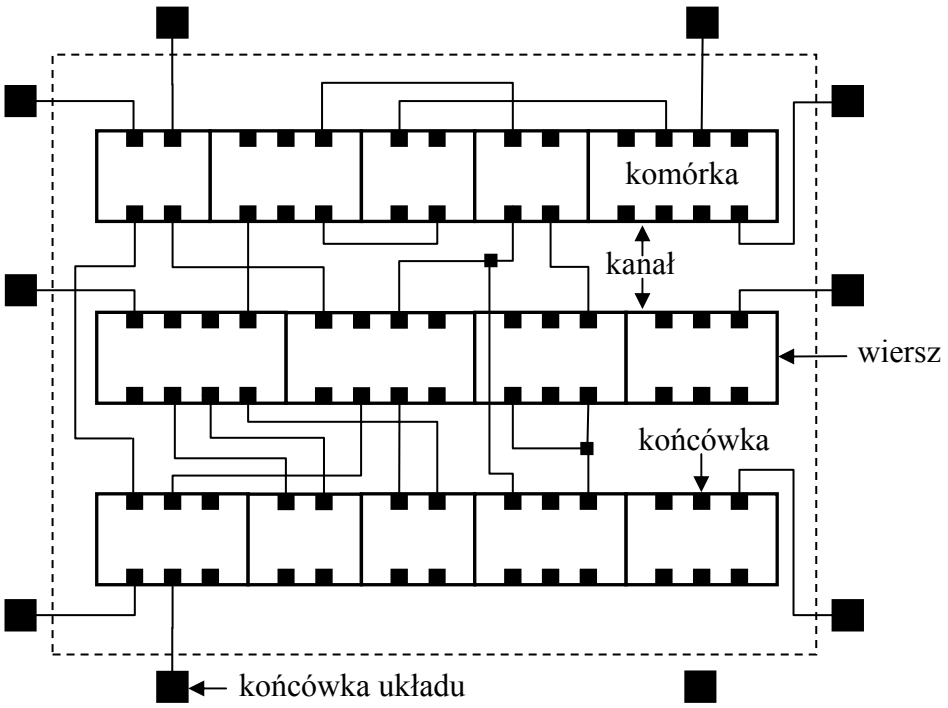
Stosowane style topografii układów przedstawiono w podrozdziale 2.1. Podrozdział 2.2 zawiera opis poszczególnych etapów projektowania topografii układu. Problem rozmieszczania modułów podczas projektowania topografii układów VLSI jest przedstawiony w podrozdziale 2.3.

2.1. Style topografii układu VLSI

Systemy projektowania topografii układów VLSI muszą uwzględniać styl (strukturę) topografii układu (ang. design style). Topografia układu VLSI może zostać zaprojektowana w jednym z trzech stylów: topografia o strukturze wierszowej - topografia wierszowa (ang. row-based), topografia o strukturze swobodnej - topografia swobodna (ang. building block) oraz topografia mieszana (ang. mixed-size, mixed block design). W przypadku topografii

wierszowej układ składa się z wierszy, w których są umieszczone elementarne komórki logiczne. Topografię wierszową posiadają: układy Standard Cell, matryce bramkowe GA (ang. Gate Array), układy Sea-of-Gates SOG, układy FPGA (ang. Field Programmable Gate Array).

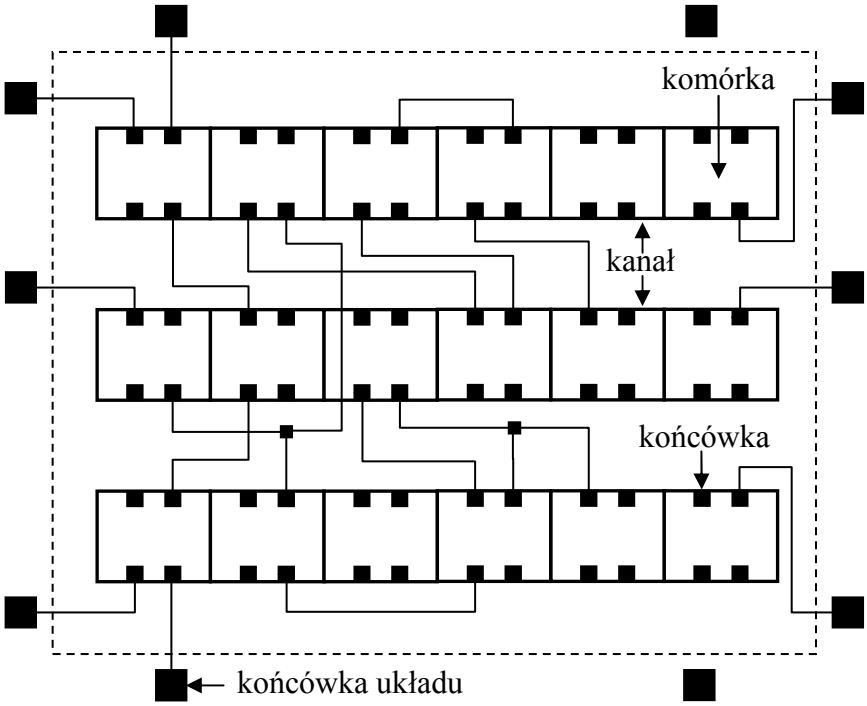
Topografia układów Standard Cell jest przedstawiona na rysunku 1. Układ składa się z wierszy elementarnych komórek logicznych - komórek standardowych, które zostały wcześniej zaprojektowane przez producenta układów. Projektant danego układu realizuje jego funkcję logiczną, używając tylko komórek standardowych zebranych w bibliotece. Komórki standardowe mają przeważnie taką samą wysokość, natomiast szerokość komórek może być różna. Połączenia między komórkami są prowadzone w kanałach, które powinny posiadać wysokość niezbędną do poprowadzenia wszystkich połączeń. Jeżeli połączenie musi przeciąć wiersz komórek, wówczas może być ono poprowadzone pionowo w specjalnej części jednej z komórek lub z wykorzystaniem specjalnej komórki, przeznaczonej wyłącznie do prowadzenia takich połączeń (ang. feedthrough). Układy Standard Cell wymagają wykonania wszystkich fotomasek przez producenta układu. Dzięki temu jest możliwe dostosowanie rozmiarów tranzystorów do potrzeb konkretnego układu, czynność ta odbywa się na poziomie biblioteki komórek. Wadą układów Standard Cell jest wysoki koszt oraz długi okres produkcji układu, w porównaniu do matryc bramkowych GA, układów Sea-of-Gates oraz układów programowalnych [1, 3, 6-9, 45].



Rys. 1. Topografia układu Standard Cell

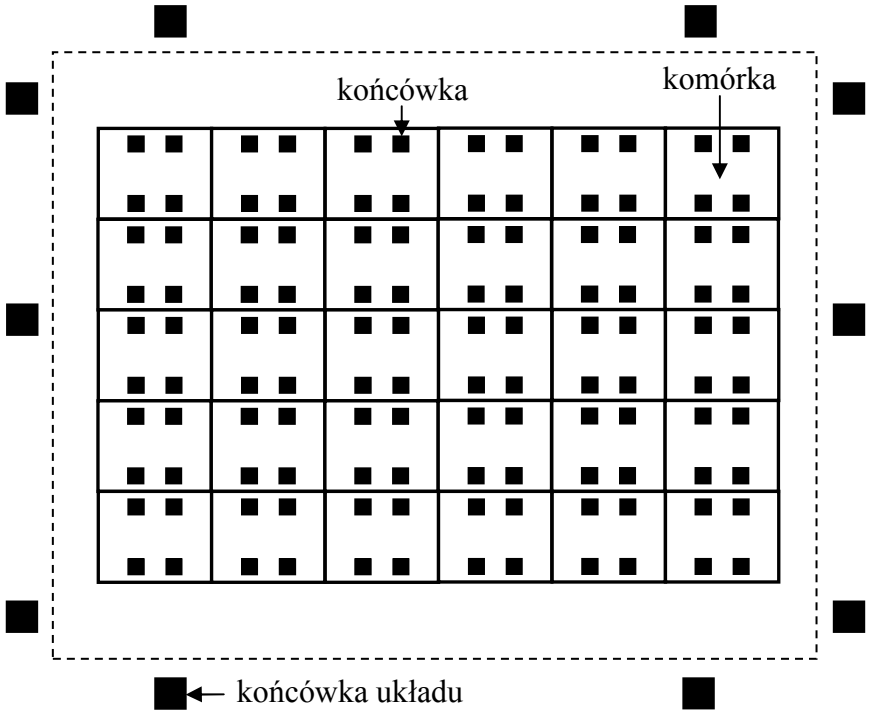
Topografię matryc bramkowych GA przedstawia rysunek 2. Układy te również składają się z wierszy elementarnych komórek logicznych, które zostały wcześniej zaprojektowane przez producenta układów. Wszystkie komórki w

układzie posiadają taką samą topografię. Połączenia są prowadzone w kanałach o ustalonej wysokości. Z tego względu są produkowane układy o różnej wysokości kanału. Produkcja układów może być masowa, co wpływa na zmniejszenie kosztu realizowanego układu. Projektant konkretnego układu określa jedynie topografię połączeń między poszczególnymi komórkami, co umożliwia skrócenie czasu jego realizacji, ponieważ producent wykonuje specjalnie dla tego układu jedynie fotomaski metalizacji. Wadą matryc bramkowych GA jest mniejsze wykorzystanie powierzchni układu oraz gorsze właściwości funkcjonalne układu, ze względu na brak możliwości dostosowania poszczególnych tranzystorów do potrzeb danego układu [1, 3, 6-7, 9].



Rys. 2. Topografia matrycy bramkowej GA

Rysunek 3 przedstawia topografię układów Sea-of-Gates. Układy te są podobne do matryc bramkowych GA, ale nie posiadają wyznaczonych kanałów. Cała powierzchnia układu składa się z elementarnych komórek logicznych, które posiadają taką samą topografię, zaprojektowaną przez producenta układów. Podczas projektowania konkretnego układu, określona liczba wierszy komórek jest rezerwowana do poprowadzenia połączeń. Komórki te nie realizują wówczas żadnych funkcji logicznych i stanowią kanały. Układy Sea-of-Gates charakteryzują się lepszym wykorzystaniem powierzchni układu w stosunku do matryc bramkowych GA [1, 3]. Układy Standard Cell, matryce bramkowe GA oraz układy Sea-of-Gates określamy mianem układów typu semi-custom (ang.), ze względu na dwuetapowy cykl projektowania układu.

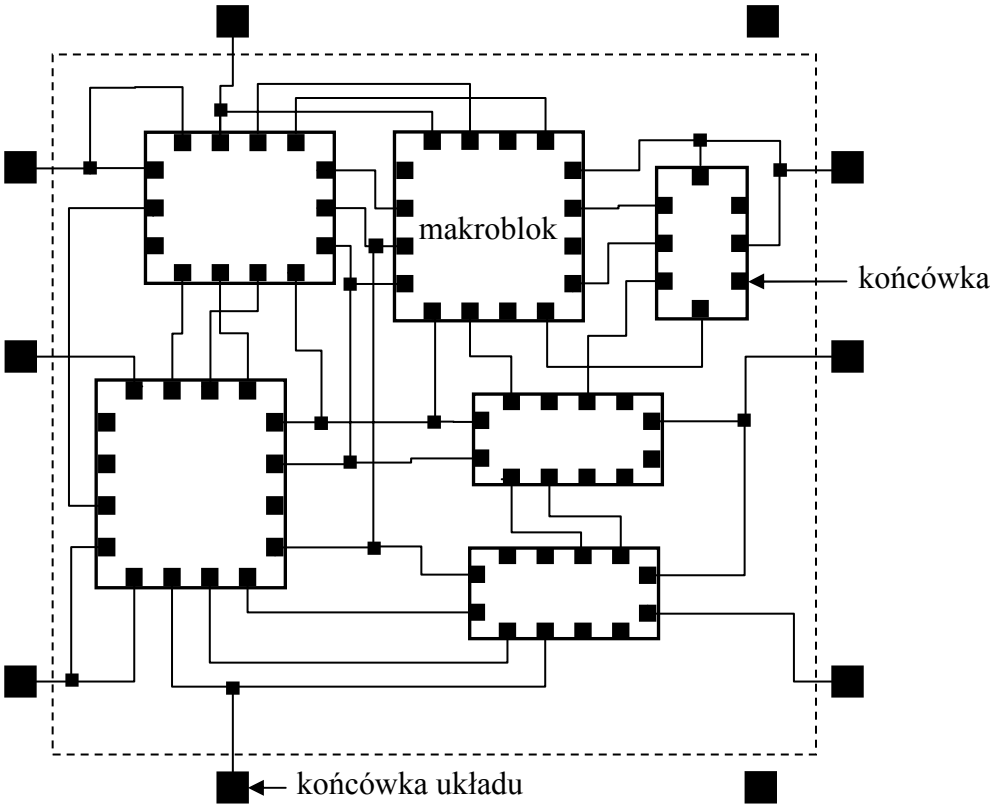


Rys. 3. Topografia układu Sea-of-Gates

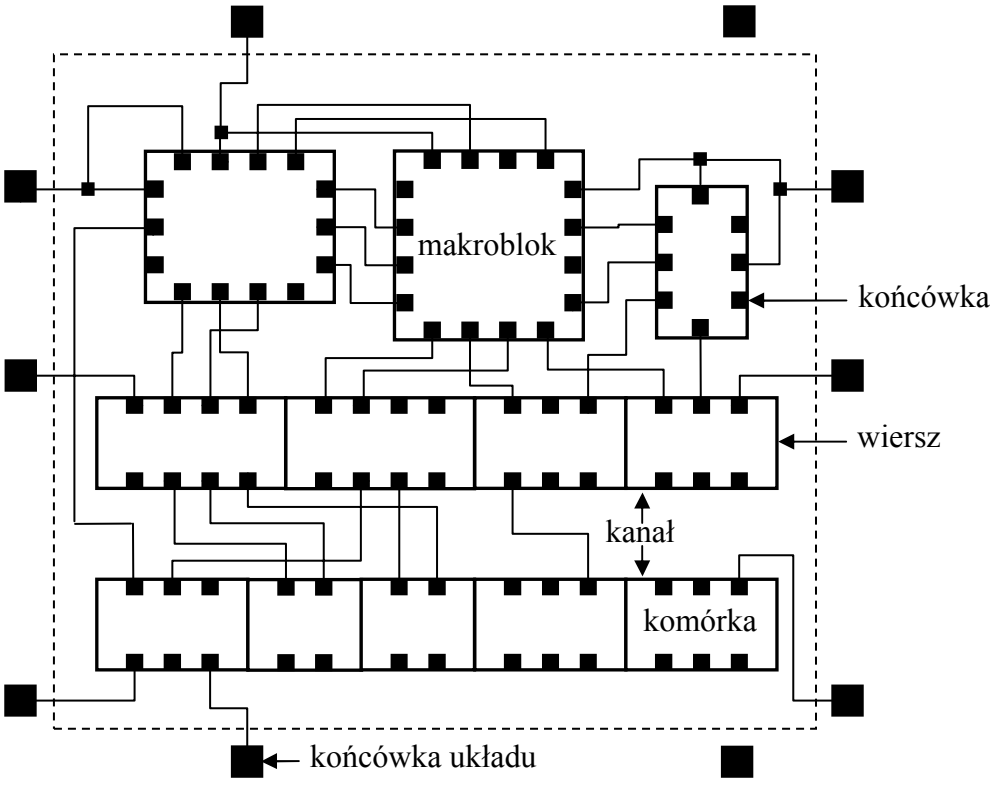
Przykład topografii swobodnej jest przedstawiony na rysunku 4. Części składowe układu - makrobloki (ang. macro block), mogą posiadać dowolne położenie i rozmiar, co umożliwia większe wykorzystanie powierzchni układu w stosunku do układów o topografii wierszowej. Przestrzenie między makroblokami służą do prowadzenia połączeń. Układy te są projektowane od podstaw, co umożliwia również osiągnięcie lepszych właściwości funkcjonalnych układu. Z tego względu określamy je mianem układów typu full-custom (ang.). Układy te wymagają wykonania wszystkich fotomasek przez producenta układu [3, 6-7, 9].

Rysunek 5 przedstawia przykład topografii mieszanej. Układ składa się z dwóch części, posiadających różną topografię: wierszową (układ Standard Cell) oraz swobodną (makrobloki). Połączenie tych topografii w jednym układzie jest możliwe, ponieważ układy Standard Cell oraz topografia swobodna wymagają wykonania wszystkich fotomasek przez producenta układu [1, 8-9, 45].

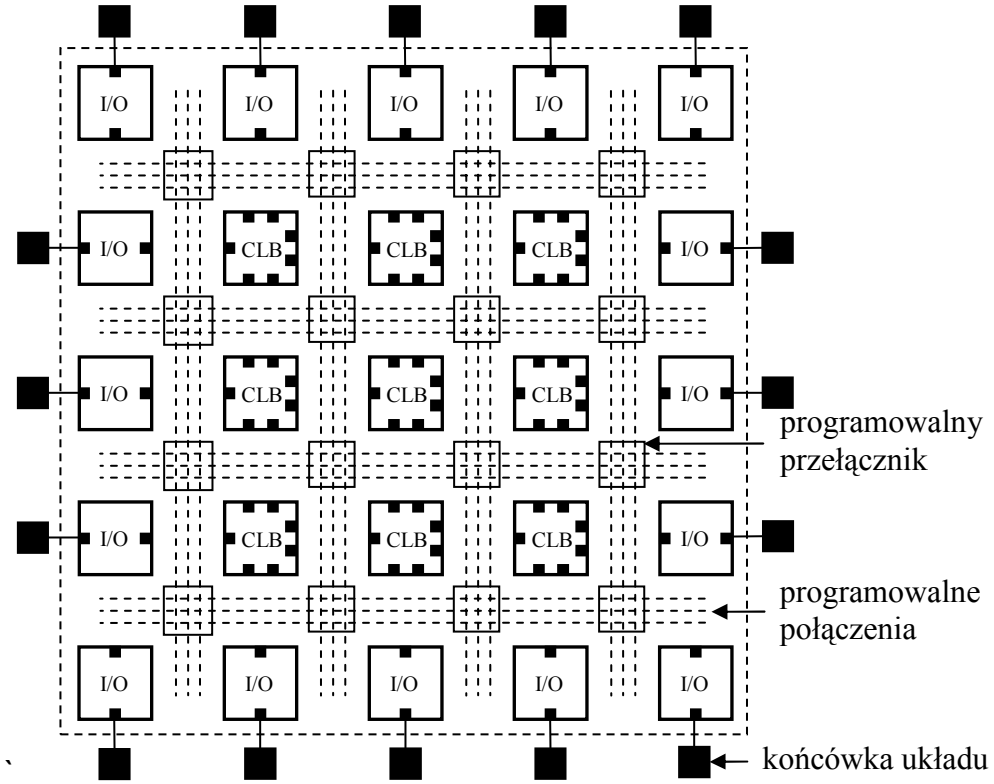
Układy FPGA są produkowane we wszystkich trzech stylach topografii. Rysunek 6 przedstawia typową topografię układu FPGA. Układ składa się z prostokątnej macierzy elementarnych komórek logicznych. Realizacja projektu konkretnego układu odbywa się programowo, bez udziału producenta. Programowanie układu odbywa się w wyniku programowania elementarnych komórek logicznych CLB (ang. Configurable Logic Block) oraz programowania połączeń między nimi. Połączenia między komórkami logicznymi są prowadzone z użyciem poziomych oraz pionowych kanałów. Programowanie połączeń między komórkami zapewniają programowalne przełączniki, znajdujące się w rogach między czterema komórkami logicznymi. Połączenia z końcówkami całego układu są realizowane za pomocą programowalnych komórek wejścia/wyjścia I/O (ang. Input-Output Block) [1, 10, 61-62].



Rys. 4. Topografia swobodna



Rys. 5. Topografia mieszana



Rys. 6. Topografia typowego układu FPGA

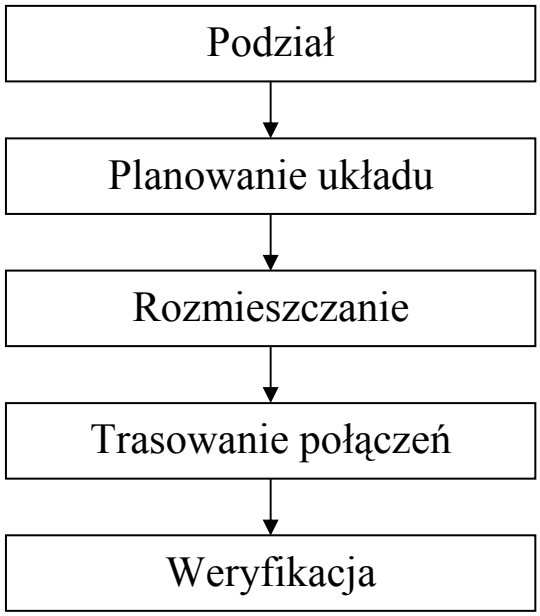
Metody rozmieszczania stosowane podczas projektowania układów VLSI muszą uwzględniać styl topografii oraz typ projektowanego układu. Niektóre metody zostały opracowane dla konkretnego typu układów. Inne mogą być stosowane dla różnych układów, po odpowiednich modyfikacjach.

2.2. Etapy projektowania topografii układu

Wejściowymi danymi dla procesu projektowania topografii układu VLSI jest lista połączeń części składowych układu (ang. netlist). Częściami składowymi układu mogą być pojedyncze tranzystory, bramki logiczne, komórki standardowe, w zależności od poziomu hierarchii, na jakim rozpatrujemy projektowany układ.

W obecnych systemach automatycznego projektowania topografii układów VLSI, proces projektowania jest podzielony na kilka etapów, ponieważ nie jest możliwe rozwiązanie tego problemu w jednym etapie. Proces projektowania topografii układu składa się zatem z następujących etapów: podziału (ang. partitioning), planowania układu (ang. floorplanning, chip planning), rozmieszczania (ang. placement), trasowania połączeń (ang. routing) oraz weryfikacji (ang. verification) [1, 3-7]. Rysunek 7 przedstawia etapy projektowania topografii układu VLSI.

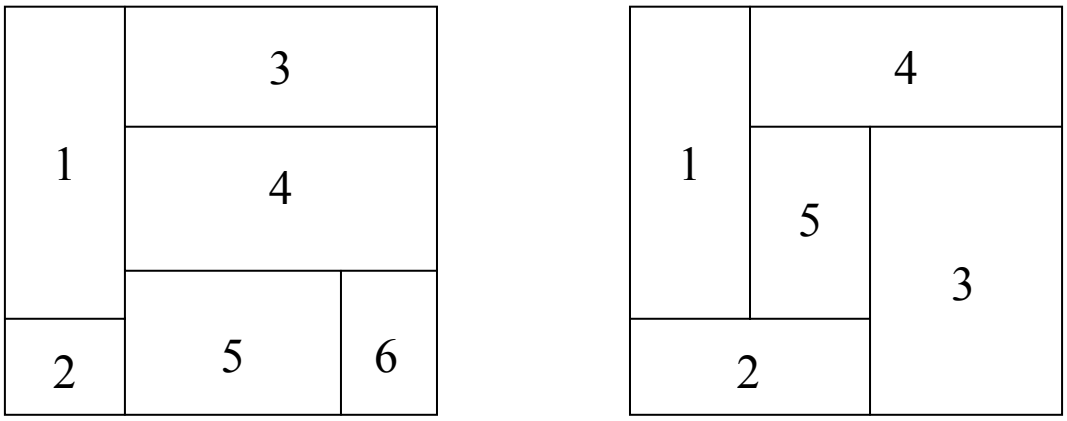
Podział może być pierwszym etapem projektowania topografii układu, gdy rozmiar układu nie pozwala na wykonanie go w jednym układzie scalonym. Podział umożliwia zastąpienie całego układu pewną liczbą układów scalonych. Podział polega na określeniu, które części składowe całego układu powinny znaleźć się w poszczególnych układach scalonych. Wykorzystywana jest tutaj metoda „dziel i rządź" (ang. „divide and conquer"), która jest bardzo często stosowana w procesie projektowania topografii. Metoda ta polega na podziale problemu na mniejsze podproblemy. Dzięki temu możliwe jest otrzymanie rozwiązania również dla bardzo dużych układów, dla których niemożliwe jest uzyskanie rozwiązania w sposób bezpośredni [1, 3-5, 30, 88-90].



Rys. 7. Etapy projektowania topografii układu VLSI

Kolejnym etapem jest planowanie układu, podczas którego następuje wstępne określenie topografii układu scalonego, w wyniku rozmieszczenia poszczególnych bloków funkcjonalnych układu. Podczas tego etapu jest określany sposób prowadzenia zasilania, sygnału taktującego oraz masy w układzie. Określane jest również położenie poszczególnych końcówek całego układu (ang. pad). Podczas tego etapu są rozpatrywane bloki, dla których rozmiary oraz położenie końcówek jest określone (ang. hard block) oraz bloki, które mają jedynie określoną powierzchnię, jaką będą zajmować w układzie (ang. soft block). Bloki typu soft nie posiadają określonego położenia końcówek, natomiast stosunek wysokości i szerokości tego bloku (ang. aspect ratio) może zmieniać się w zadanym przedziale [1, 3, 5, 7, 11, 64]. Jeżeli jest możliwy podział układu oraz jego części, liniami poziomymi i pionowymi tak, aby każdy blok stanowił w końcu jedną część, wówczas plan topografii układu jest typu slicing (ang.). W przeciwnym razie jest typu non-slicing (ang.) [1, 7-8, 86]. Rysunek 8 przedstawia plan topografii układu typu slicing oraz non-slicing. Ze względu na podobieństwa łączące planowanie układu oraz rozmieszczanie, metody stosowane podczas tego etapu zostaną dokładniej opisane.

Ze względu na dużą złożoność obliczeniową, planowanie układu odbywa się zwykle w dwóch krokach [7, 11, 64]. Podczas pierwszego kroku następuje określenie wzajemnego położenia bloków tak, aby zminimalizować całkowitą długość połączeń w układzie. Czynność ta odbywa się z wykorzystaniem teorii grafów [9, 11-13, 18]. W drugim kroku jest określane położenie poszczególnych bloków oraz rozmiary bloków typu soft tak, aby zminimalizować całkowitą powierzchnię układu, zachowując wcześniej określone wzajemne położenie bloków. W tym celu są stosowane metody wykorzystujące algorytm Ottena i Stockmeyera [14-16], algorytm podziału i ograniczenia (ang. branch and bound) [14-15], teorię grafów [13, 16-18], podział planu topografii układu [19], programowanie liniowe (ang. linear programming LP) [9, 11-12, 64].



Rys. 8. Plan topografii układu: a) typu slicing, b) typu non-slicing

Istnieją również metody, które pozwalają na minimalizację całkowitej powierzchni układu oraz długości połączeń w jednym kroku. Metody te najczęściej wykorzystują symulowane wyżarzanie, które wymaga odpowiedniego sposobu kodowania rozwiązania. Bardzo istotne jest, aby zmiany rozwiązania dokonywane podczas symulowanego wyżarzania nie powodowały powstawania rozwiązań, które są fizycznie niemożliwe do wykonania [7, 11, 20]. W przypadku planu topografii typu slicing, do kodowania rozwiązań jest stosowana Odwrotna Notacja Polska (ang. reverse Polish notation) [7]. Dla planu topografii typu non-slicing bardzo często jest stosowane kodowanie z użyciem pary nazw bloków, określających ich kolejność (ang. sequence-pair) [20-22]. Plan topografii typu non-slicing zapewnia lepsze upakowanie bloków w układzie, lecz wymaga dodatkowych nakładów podczas trasowania połączeń [8]. Obecnie badania są skupione głównie na znalezieniu efektywnego sposobu kodowania rozwiązań [23-24, 86, 106]. Odmienny sposób podejścia do problemu planowania topografii układu, w którym rozmiary planowanego układu są ściśle określone, można znaleźć w pracach [25, 120-121].

Kolejnym etapem jest rozmieszczanie. Podczas tego etapu jest określane położenie wszystkich części składowych układu w poszczególnych blokach układu. W dalszej części pracy części składowe układu będą nazywane modułami. Najczęściej stosowanymi kryteriami podczas rozmieszczania modułów jest minimalizacja całkowitej estymowanej (przewidywanej) długości

połączeń, minimalizacja opóźnień w krytycznych częściach układu oraz minimalizacja skupienia połączeń [1, 3-7, 9, 30 51-52, 84, 90, 101-104]. Rozmieszczanie zwykle składa się z dwóch kroków: rozmieszczania globalnego (ang. global placement) i rozmieszczania szczegółowego (ang. detailed placement). Podczas rozmieszczania globalnego moduły są prawie równomiernie rozmieszczane na podłożu układu, zgodnie z wybranymi kryteriami rozmieszczania. Rozmieszczanie szczegółowe ma na celu wyznaczenie ostatecznego położenia modułów, zgodnie z wybranym stylem topografii układu, jest usuwane zachodzenie modułów na siebie. Podczas rozmieszczania szczegółowego może być również wykonywana korekcja rozmieszczenia dla małych grup modułów, według wybranych kryteriów rozmieszczania. Niniejsza praca jest poświęcona metodom rozmieszczania, dlatego metody te zostaną szczegółowo opisane w następnym rozdziale. Planowanie układu i rozmieszczanie łączy wiele podobieństw, z tego względu istnieje wiele metod, które z powodzeniem mogą być stosowane podczas obydwu etapów [3, 5].

Po rozmieszczeniu modułów jest możliwe wyznaczenie połączeń między nimi. Czynność ta odbywa się podczas etapu trasowania połączeń. Etap ten składa się z dwóch kroków: trasowania globalnego (ang. global routing) i trasowania szczegółowego (ang. detailed routing). Podczas trasowania globalnego połączenia nie są wyznaczane, a jedynie planowane. Planowanie połączeń może odbywać się z wykorzystaniem grafu, którego wierzchołkami są punkty przecięcia kanałów, w których mają być prowadzone połączenia. Krawędziami grafu są kanały. Wagi krawędzi powinny odzwierciedlać długość połączeń oraz pojemność kanałów, powinny być aktualizowane po wyznaczeniu poszczególnych połączeń. W celu wyznaczenia danego połączenia, w grafie jest wyznaczana ścieżka o najmniejszym koszcie [1, 3-5, 8, 10, 73-74, 101, 135, 138]. Do trasowania globalnego może być zastosowany również algorytm symulowanego wyżarzania [6, 30, 90]. Po zaplanowaniu połączeń, następuje fizyczne wyznaczenie połączeń podczas trasowania szczegółowego. W tym kroku połączenia są prowadzone w kanałach, które zostały wcześniej przydzielone poszczególnym połączeniom podczas trasowania globalnego. Istnieje cały szereg metod trasowania szczegółowego [1, 3-5, 7-8, 10, 27-28, 117, 135, 138].

Po wyznaczeniu topografii układu następuje weryfikacja rozwiązania. Weryfikacja obejmuje sprawdzenie zgodności projektu topografii z regułami projektowania i schematem elektrycznym układu. Może polegać również na określeniu rzeczywistego modelu układu. Znajomość pojemności i rezystancji pasożytniczych, pojemności, indukcyjności i rezystancji połączeń oraz parametrów tranzystorów, umożliwia dokładną symulację układu [1, 3-5, 26]. Przed etapem weryfikacji topografii układu jest również stosowana kompresja (zagęszczanie) topografii (ang. compaction). Celem kompresji topografii jest zmniejszenie rozmiarów układu poprzez eliminację niepotrzebnych przestrzeni w układzie, przy zachowaniu reguł projektowania oraz topologii układu. Kompresja może być stosowana również w przypadku zmiany technologii wykonania układu [3, 5].

Czasową złożoność obliczeniową algorytmu mierzy się liczbą elementarnych operacji powtarzanych w algorytmie. Stosowane jest pojęcie złożoności asymptotycznej. Niech n oznacza rozmiar problemu. Algorytm posiada złożoność $O(f(n))$, jeżeli istnieją liczby dodatnie c i N takie, że liczba

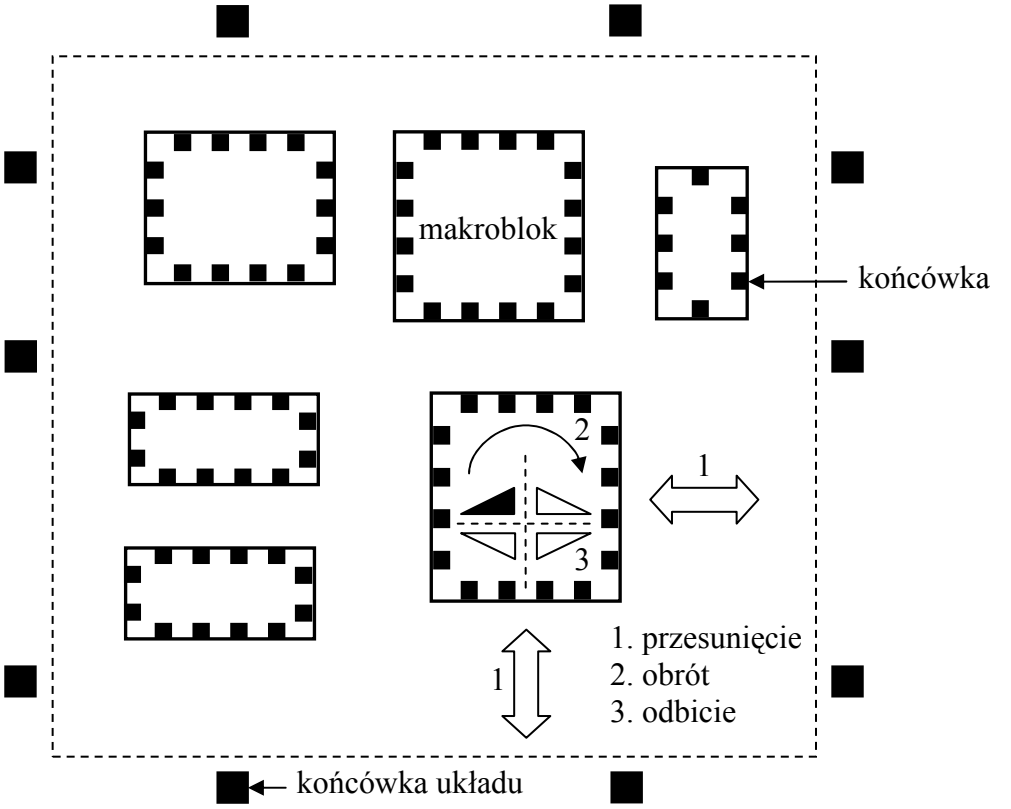
elementarnych operacji jest mniejsza lub równa niż $c\,f(n)$ dla wszystkich $n \geq N$. Wynika z tego, że dla dostatecznie dużych n liczba elementarnych operacji rośnie nie szybciej niż funkcja f [136]. Większość etapów projektowania topografii układu (z wyjątkiem weryfikacji) jest zaliczanych do tzw. problemów NP-trudnych. Dla tych problemów nie znaleziono algorytmu wyznaczenia optymalnego rozwiązania, którego czasowa złożoność obliczeniowa jest wielomianem dowolnego, skończonego stopnia względem rozmiaru problemu. Z tego powodu do rozwiązania tych problemów są stosowane metody, które aproksymują optymalne rozwiązanie [1, 3-7, 9, 14, 16-20, 24, 30].

2.3. Problem rozmieszczania modułów

W niniejszej pracy części składowe układów będą nazywane modułami. W ten sposób będą określane zarówno komórki standardowe w układach Standard Cell, jak również elementarne komórki logiczne w pozostałych układach o topografii wierszowej. Części składowe układów o topografii swobodnej również będą nazywane modułami.

2.3.1. Postawienie problemu

Rozmieszczanie modułów w układzie VLSI jest bardzo ważnym etapem projektowania jego topografii. Sposób rozmieszczenia modułów decyduje o powierzchni układu oraz jego właściwościach funkcjonalnych. Podczas następnych etapów projektowania topografii nie można naprawić błędów popełnionych podczas rozmieszczania [34, 70, 72, 82]. Danymi wejściowymi dla metod rozmieszczania są kształty, rozmiary i położenie końcówek poszczególnych modułów układu, położenie końcówek całego układu oraz lista połączeń między modułami i końcówkami całego układu. Rozwiązanie problemu rozmieszczania polega na znalezieniu współrzędnych prostokątnych, określających położenie modułów w układzie tak, aby zapewnić spełnienie celów stawianych procesowi rozmieszczania. Metody rozmieszczania muszą uwzględniać styl topografii układu, ponadto moduły nie mogą zachodzić na siebie oraz w całości muszą być położone w obszarze układu. W układach o topografii wierszowej często zakłada się, że wszystkie końcówki modułów są położone w środku geometrycznym modułów, ponieważ rozmiary modułów są niewielkie w porównaniu do rozmiarów podłoża układu. W układach o topografii swobodnej położenie końcówek modułów (makrobloków) musi być uwzględnione, ponieważ poszczególne moduły mogą znacznie różnić się od siebie rozmiarami i zajmować znaczną część podłoża układu. Uwzględniana jest również orientacja modułu (makrobloku). Każdy moduł może posiadać osiem różnych orientacji, które są rezultatem obrotu oraz odbicia względem jego osi symetrii [51, 78]. Rysunek 9 przedstawia przykład rozmieszczenia modułów.



Rys. 9. Przykład rozmieszczenia modułów

Podstawowymi celami rozmieszczania jest umożliwienie poprowadzenia wszystkich połączeń, minimalizacja powierzchni układu, minimalizacja opóźnień w krytycznych częściach układu, minimalizacja sprzężeń między sygnałami, minimalizacja wydzielanej energii w układzie, równomierny rozkład temperatury układu oraz wiele innych [1-5, 9, 30, 51-52, 75-76, 84]. Cele te trudno jest formalnie zawrzeć w algorytmach rozmieszczania, z tego względu metody rozmieszczania najczęściej stosują własne kryteria. Najczęściej stosowanymi celami metod rozmieszczania jest minimalizacja całkowitej estymowanej długości połączeń, minimalizacja skupienia połączeń, minimalizacja opóźnień w krytycznych częściach układu oraz minimalizacja powierzchni układu - w przypadku topografii swobodnej. Cele rozmieszczania są zwykle określone przez tzw. funkcję kosztu. Rozwiązanie problemu polega na znalezieniu rozmieszczenia, które posiada minimalny koszt [1, 3-10, 12, 30, 78, 84, 90, 101-104].

Minimalizacja skupienia połączeń pośrednio wpływa na minimalizację powierzchni układu, ze względu na mniejsze rozmiary kanałów do prowadzenia połączeń oraz pośrednio przyczynia się do minimalizacji długości połączeń, ponieważ silnie połączone ze sobą moduły są rozmieszczane obok siebie [1, 5, 8-9, 33, 40, 43-44, 46]. Minimalizacja całkowitej estymowanej długości połączeń jest bardzo często stosowanym celem metod rozmieszczania. Minimalizacja długości połączeń pośrednio przyczynia się do minimalizacji całkowitej powierzchni układu, ponieważ połączenia zajmują znaczną część jego powierzchni, nawet 50÷70% całkowitej powierzchni [8-9, 29-30, 33-34]. Pośrednio przyczynia się do minimalizacji skupienia połączeń i umożliwia

wykonanie wszystkich połączeń, w przypadku ograniczonej pojemności kanałów przeznaczonych do prowadzenia połączeń [4, 10, 30, 84, 101]. Powoduje zmniejszenie pojemności i rezystancji pasożytniczych, wnoszonych do układu przez połączenia oraz pojemności, indukcyjności i rezystancji połączeń. Minimalizacja długości połączeń umożliwia zatem również zmniejszenie czasu propagacji sygnałów oraz ilości ciepła wydzielanego w układzie. Minimalizacja opóźnień w krytycznych częściach układu zapewnia znaczne zmniejszenie czasu propagacji sygnałów w układzie, co umożliwia zastosowanie sygnału taktującego o większej częstotliwości [1, 3, 9, 53-54, 56-60, 77, 84, 103-104].

2.3.2. Minimalizacja długości połączeń

Połączenia w układach VLSI są zwykle prowadzone wyłącznie w kierunku poziomym oraz pionowym, dlatego do określenia długości połączeń najczęściej jest stosowana metryka Manhattan, według wzoru

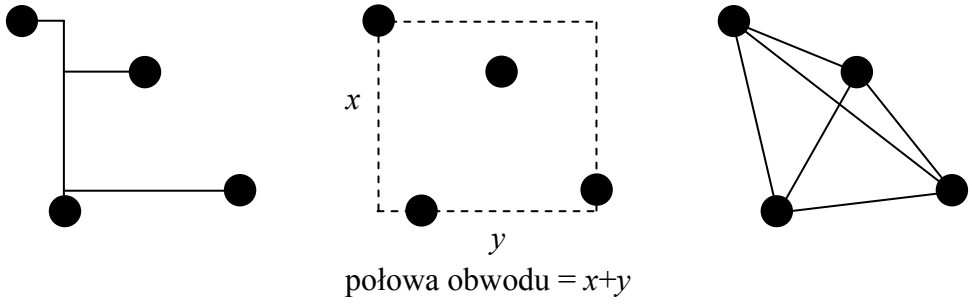
$$d_{ij} = |x_i - x_j| + |y_i - y_j| \tag{1}$$

gdzie: d_{ij} - długość połączenia między punktami o współrzędnych (x_i, y_i) oraz (x_j, y_j).

Wykorzystywana jest również metryka Euklidesa oraz kwadrat długości euklidesowej [1, 3, 9, 28, 30].

Podczas rozmieszczania modułów jest stosowana estymacja długości połączeń dla danego rozmieszczenia, ponieważ wyznaczanie ścieżek połączeń dla każdego przejściowego rozmieszczenia zajęłoby zbyt wiele czasu [4, 8]. Całkowita estymowana długość połączeń układu jest sumą estymowanych długości dla poszczególnych jego węzłów. Po ustaleniu ostatecznego rozmieszczenia modułów z wykorzystaniem estymacji długości połączeń, są wyznaczane ścieżki połączeń. Czynność ta jest jednak wykonywana podczas następnego etapu projektowania topografii - podczas trasowania połączeń. Minimalne drzewo Steinera posiada tę własność, że łączy zadane punkty używając minimalnej długości połączeń. Do estymacji długości połączeń są jednak wykorzystywane aproksymacje minimalnego drzewa Steinera, ponieważ znalezienie tego drzewa w ogólnym przypadku jest problemem NP-trudnym [29, 133-134]. Najczęściej stosowanymi sposobami estymacji długości połączeń jest metoda połowy obwodu (ang. half-perimeter) oraz metoda wykorzystująca graf pełny (ang. complete graph) [1, 3-10, 27, 30, 51, 73-74, 117]. Rysunek 10 przedstawia obydwie metody estymacji długości połączeń oraz drzewo Steinera - w przypadku grafu pełnego krawędzie grafu poprowadzono w przestrzeni Euklidesa, dla większej przejrzystości rysunku. Estymacja metodą połowy obwodu jest równa połowie obwodu prostokąta, obejmującego wszystkie końcówki danego węzła (końcówki modułów i końcówki całego układu). W metryce Manhattan ta estymacja jest równa minimalnej długości ścieżek niezbędnych do połączenia końcówek węzła, jeżeli liczba końcówek nie

jest większa niż 3. Dla większej liczby końcówek, wartość estymowana jest zwykle mniejsza od rzeczywistej minimalnej długości połączeń.



(a) (b) (c)

Rys. 10. Drzewo Steinera i metody estymacji długości połączeń: a) drzewo Steinera, b) połowa obwodu, c) graf pełny

Estymacja wykorzystująca graf pełny jest oparta na grafie pełnym, obejmującym wszystkie końcówki danego węzła, które stają się jego wierzchołkami. Graf pełny posiada krawędzie łączące wszystkie pary wierzchołków. Długość krawędzi grafu jest równa odległości między końcówkami w metryce Manhattan. Estymowana długość połączeń jest równa sumie długości wszystkich krawędzi grafu pełnego podzielonej przez $n/2$, gdzie n jest liczbą wszystkich końcówek w danym węźle. W grafie pełnym jest $n(n-1)/2$ wszystkich krawędzi, więc po podzieleniu przez $n/2$, otrzymujemy aproksymację długości $n-1$ połączeń, niezbędnych do połączenia n końcówek danego węzła. Wartość estymowana długości połączeń d jest określona wzorem

$$d = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \tag{2}$$

gdzie: d_{ij} - długość połączenia między modułami i oraz j, we wzorze występuje dzielenie przez n, ponieważ długość każdej krawędzi grafu jest liczona dwukrotnie.

Dla obydwu metod jest możliwa korekcja estymowanej długości tak, aby odpowiadała rzeczywistej długości połączeń [1, 3-4, 9-10, 27, 117].

2.3.3. Minimalizacja opóźnień w układzie

Bardzo ważnym celem rozmieszczania modułów jest minimalizacja opóźnień w krytycznych częściach układu (ang. timing-driven placement). We współczesnych układach VLSI, opóźnienia wnoszone przez połączenia mają decydujące znaczenie dla czasu propagacji sygnałów, w porównaniu do opóźnień wnoszonych przez elementy aktywne. Jest to spowodowane stale

postępującą miniaturyzacją układów oraz stałym wzrostem częstotliwości sygnału taktującego [53-54, 77].

Optymalne rozmieszczenie modułów zapewnia znaczne zmniejszenie czasu propagacji sygnałów w układzie, co umożliwia zastosowanie w układzie sygnału taktującego o większej częstotliwości. Minimalizacja opóźnień w krytycznych częściach układu z reguły jest okupiona niewielkim wzrostem całkowitej długości połączeń. Metody rozmieszczania, które minimalizują opóźnienia w układzie, dzielimy na dwie grupy: metody oparte na ścieżkach (ang. path-based) oraz metody oparte na węzłach (ang. net-based). Metody oparte na ścieżkach minimalizują opóźnienia w krytycznych ścieżkach, będących częścią różnych węzłów. Ze względu na bardzo dużą liczbę możliwych ścieżek, które mogą być rozpatrywane, metody te charakteryzują się bardzo dużą złożonością obliczeniową. Metody oparte na węzłach, minimalizują niezależnie opóźnienia w poszczególnych węzłach układu. Powszechnie stosowanym rozwiązaniem jest algorytm rozpatrujący „rezerwę" czasu na wyjściu wszystkich modułów (ang. zero-slack algorithm). W układzie musi być określony czas nadejścia sygnału na wejściach całego układu (ang. arrival time) oraz wymagany czas ustalenia się sygnału na jego wyjściach (ang. required time). W wyniku dwóch niezależnych przejść, od wejścia do wyjścia oraz od wyjścia do wejścia, są określane czasy nadejścia sygnału oraz wymagane czasy ustalenia sygnału dla wyjść wszystkich modułów. W przypadku czasu nadejścia sygnału jest brana pod uwagę najpóźniejsza jego zmiana. W przypadku wymaganego czasu ustalenia się sygnału jest brana pod uwagę najwcześniejsza konieczność jego zmiany. Uwzględniane są opóźnienia elementów aktywnych. Różnica między wymaganym czasem ustalenia się sygnału oraz czasem nadejścia sygnału określa „rezerwę" czasu dla danego modułu (ang. slack). Następnie algorytm przydziela opóźnienia do poszczególnych węzłów tak, aby „rezerwa" czasu dla wszystkich modułów była równa 0. Opóźnienia przydzielone poszczególnym węzłom są następnie zamieniane na wagi węzłów lub na ograniczenia ich maksymalnej długości. W celu określenia maksymalnej długości węzłów są wymagane odpowiednie parametry technologiczne układu, które określają opóźnienia wnoszone przez połączenia w danym układzie. Zamiana opóźnień na wagi umożliwia rozwiązanie problemu minimalizacji opóźnień bez żadnych zmian w metodach rozmieszczania oraz bez dodatkowych nakładów obliczeniowych. Jest to możliwe, jeżeli funkcja kosztu danej metody wykorzystuje wagi poszczególnych węzłów układu. Węzły posiadające większą wagę będą wówczas preferowane podczas rozmieszczania. Możliwy jest taki dobór wag węzłów, który ułatwi dalszą minimalizację opóźnień, po wykonaniu rozmieszczenia modułów (ang. in-place optimization). Dalsza minimalizacja opóźnień w krytycznych częściach układu jest możliwa np. w wyniku zmiany rozmiarów poszczególnych tranzystorów, co zwiększa ich wydajność prądową [1, 3, 56-60, 85, 103-104].

Rozdział 3
Przegląd metod rozmieszczania

Istnieje wiele metod rozmieszczania modułów w układach VLSI. Wyszukiwanie wyczerpujące (ang. exhaustive search) nie jest stosowane, ze względu na bardzo dużą liczbę możliwych rozwiązań. W przypadku układu składającego się z n części składowych o takich samych rozmiarach, liczba możliwych rozwiązań wynosi $n!$. Czasowa złożoność obliczeniowa wyszukiwania wyczerpującego jest zatem równa $O(n!)$. Metody rozmieszczania najczęściej dzieli się na: metody konstrukcyjne, iteracyjne, wykorzystujące sieci neuronowe [1, 3-5, 9, 28, 33, 44, 65, 72, 119]. Metody konstrukcyjne tworzą rozwiązanie od podstaw [1, 3-4, 9, 119]. Metody iteracyjne rozpoczynają swoje działanie od początkowego rozwiązania, które następnie poprawiają w kolejnych iteracjach [1, 3-10, 28, 30, 65, 90]. Inny sposób podziału metod rozmieszczania wyróżnia metody deterministyczne oraz stochastyczne. Metody deterministyczne rozwiązują dany problem rozmieszczania w ten sam sposób podczas kolejnych prób [9]. Poszczególne metody rozmieszczania zostaną omówione w kolejnych podrozdziałach.

3.1. Metody iteracyjne

Początkowe rozmieszczenie modułów podczas rozmieszczania modułów z użyciem metod iteracyjnych jest najczęściej wyznaczane w sposób losowy. W kolejnych iteracjach, początkowe rozmieszczenie modułów jest następnie udoskonalane. Jakość rozwiązania w metodach iteracyjnych określa funkcja kosztu. Metody iteracyjne stanowią bardzo liczną i zróżnicowaną grupę, wśród rozwiązań stosowanych podczas rozmieszczania modułów. Do metod iteracyjnych należy zaliczyć: algorytm zamiany parami oraz jego modyfikacje, symulowane wyżarzanie, algorytmy genetyczne, strategie ewolucyjne [1, 3-10, 28, 30, 65, 90].

3.1.1. Algorytm zamiany parami

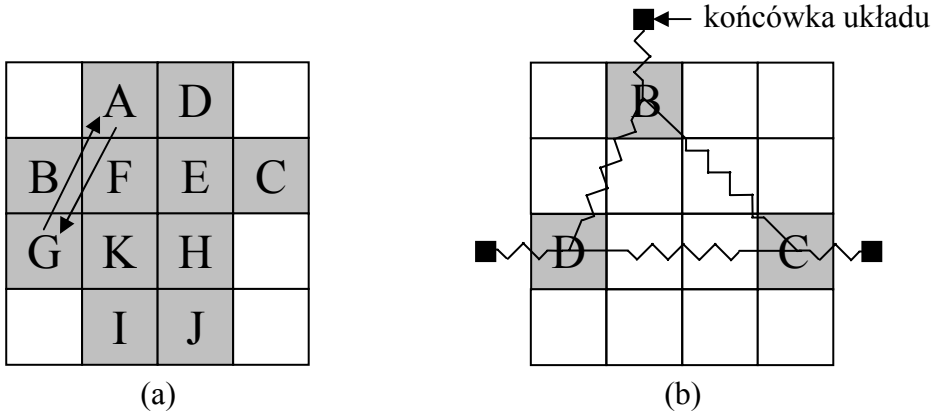
Algorytm zamiany parami (ang. pairwise interchange algorithm) polega na zamianie miejscami par modułów. Zamiana modułów jest akceptowana, jeżeli powoduje zmniejszenie kosztu połączeń (estymowanego kosztu połączeń). W przeciwnym przypadku, zamiana modułów nie jest wykonywana. Zamiany wykonuje się dla różnych par modułów. Wybór modułów do zamiany

może odbywać się losowo lub w wyniku systematycznego rozpatrywania wszystkich możliwych par modułów. W przypadku układu składającego się z n modułów, liczba wszystkich możliwych par modułów jest równa $n(n\text{-}1)/2$. Sprawdzenie wszystkich możliwych modułów składa się wtedy na jeden cykl algorytmu. Zamiany miejscami par modułów są kontynuowane do momentu, kiedy nie następuje dalsze zmniejszanie kosztu połączeń. Rysunek 11a ilustruje sposób postępowania w algorytmie zamiany parami. Dodatkowym założeniem jest jednakowy rozmiar wszystkich modułów oraz podział podłoża układu na siatkę prostokątnych części, posiadających takie same rozmiary jak moduły. Zakłada się również, że wszystkie końcówki modułów są położone w środku geometrycznym tej części podłoża układu, w której został umieszczony dany moduł [1, 3, 5-6, 8, 28, 30, 55, 65]. Istnieje wiele modyfikacji algorytmu zamiany parami. Możliwa jest równoczesna zamiana miejscami λ modułów, gdzie $\lambda > 2$, zamiana miejscami modułów znajdujących się w sąsiedztwie (ang. neighborhood interchange), zamiana miejscami modułów według algorytmu Goto (ang. generalized force-directed relaxation) [1, 3-4, 6, 9, 28, 30, 55, 65].

Istnieją również modyfikacje algorytmu zamiany parami, które wykorzystują metody relaksacyjne (ang. force-directed placement algorithms). W metodach relaksacyjnych jest rozpatrywany układ modułów, połączonych z użyciem sprężyn. Powyższą sytuację przedstawia rysunek 11b. Zgodnie z prawem Hooke'a siła oddziaływania wzajemnego **f** dla każdej pary modułów jest określona wzorem

$$\mathbf{f} = k\ \mathbf{s} \tag{3}$$

gdzie: **f** - siła oddziałująca na dany moduł, k - stała sprężyny dla danej pary modułów, **s** - wektor wyznaczony przez uporządkowaną parę współrzędnych położenia danego modułu oraz modułu oddziałującego na ten moduł.



Rys. 11. Rozmieszczanie modułów: a) algorytm zamiany parami, b) metoda relaksacyjna

Stała sprężyny k dla danej pary modułów jest równa sumie wag połączeń między tymi modułami we wszystkich węzłach układu. Uwzględniane są również połączenia między modułami a końcówkami całego układu. Całkowita

siła oddziałująca na dany moduł jest równa sumie sił wszystkich modułów lub końcówek całego układu, które oddziałują na ten moduł. W metodach relaksacyjnych jest stosowane pojęcie punktu docelowego modułu, w którym wypadkowa sił oddziałujących na moduł jest równa **0**, przy ustalonym położeniu pozostałych modułów. Celem metod relaksacyjnych jest znalezienie położenia modułów, które odpowiada stanowi równowagi układu sprężyn. W stanie równowagi wypadkowe siły oddziałujące na poszczególne moduły są równe **0**, a energia układu osiąga minimum. Powyższy stan odpowiada minimalnemu kosztowi połączeń w układzie [1, 5-6, 9, 28]. Istnieje wiele modyfikacji algorytmu zamiany parami, które wykonują zamianę modułów miejscami lub przenoszenie modułów, w oparciu o wypadkowe siły oddziałujące na moduły [1, 3, 6, 9, 28].

Algorytm zamiany parami oraz jego modyfikacje zwykle nie są stosowane samodzielnie. Istnieje wiele programów rozmieszczania, w których algorytm zamiany parami oraz jego modyfikacje są stosowane w układach Standard Cell w końcowym etapie rozmieszczania, podczas rozmieszczania szczegółowego [44, 72, 79-80, 84].

3.1.2. Symulowane wyżarzanie

Algorytm symulowanego wyżarzania (ang. simulated annealing) wykorzystuje analogię do procesu krzepnięcia ciekłego ciała. Powolne schładzanie ciekłego, rozżarzonego ciała umożliwia powstanie regularnej struktury atomów stałego ciała. Zbyt szybkie schładzanie powoduje natomiast powstanie wielu nieregularności w strukturze atomów. Nieregularność struktury atomów jest powodem wyższego stanu energetycznego ciała, w porównaniu do stanu energetycznego ciała schładzanego powoli. Zgodnie z mechaniką statystyczną stan energetyczny ciała może wzrastać w temperaturze powyżej zera absolutnego. W trakcie powolnego schładzania ciała, możliwość chwilowego wzrostu stanu energetycznego umożliwia opuszczenie minimum lokalnego funkcji energii ciała. W efekcie otrzymujemy regularną strukturę, której stan energetyczny jest równy lub bliski wartości minimum globalnego. Powyższe zjawisko stało się inspiracją do zastosowania algorytmu symulowanego wyżarzania w problemach optymalizacyjnych. Funkcja kosztu problemu optymalizacyjnego odpowiada funkcji energii ciała. Zastosowanie analogii do procesu krzepnięcia ciała umożliwia zatem znalezienie minimum globalnego funkcji kosztu problemu optymalizacyjnego [3, 5-9, 30, 69, 90-91, 93-94].

Początkowe rozwiązanie w algorytmie symulowanego wyżarzania jest tworzone najczęściej w sposób losowy. W tym celu musi być ustalony sposób reprezentacji rozwiązania problemu optymalizacyjnego. Algorytm symulowanego wyżarzania jest algorytmem iteracyjnym. Podstawową czynnością w algorytmie jest tworzenie nowego rozwiązania w wyniku zmiany obecnego rozwiązania. Nowe rozwiązanie jest akceptowane na podstawie wartości zmiany funkcji kosztu problemu optymalizacyjnego. Zmiana rozwiązania, która umożliwia zmniejszenie wartości funkcji kosztu jest zawsze akceptowana. Nowe rozwiązania, które zwiększają wartość funkcji kosztu są akceptowane, jeżeli spełniony jest następujący warunek [90]

$$e^{-\Delta E/T} > \zeta \tag{4}$$

gdzie: ΔE - zmiana wartości funkcji kosztu, równa różnicy nowej i poprzedniej wartości funkcji kosztu, ζ - liczba losowa z przedziału <0, 1>, wylosowana zgodnie z rozkładem jednostajnym, T - „temperatura" układu (parametr algorytmu), zgodnie z analogią do procesu krzepnięcia z wyżarzaniem.

Temperatura początkowa T_0 (ang. initial temperature) w algorytmie symulowanego wyżarzania jest ustalana na podstawie symulacji wstępnych. Temperatura początkowa jest ustalana tak, aby procentowa wartość akceptacji nowych rozwiązań w temperaturze początkowej przekraczała określony próg, który może mieć różną wartość, w zależności od rodzaju problemu optymalizacyjnego. Najczęściej po wykonaniu określonej liczby L generacji nowego rozwiązania, temperatura jest obniżana zgodnie z następującym harmonogramem chłodzenia (ang. cooling schedule, annealing schedule) [90]

$$T_{i+1} = \alpha\ T_i \tag{5}$$

gdzie: T_{i+1} - temperatura dla kroku i+1 zmiany temperatury, T_i - temperatura dla kroku i, α - współczynnik redukcji temperatury.

Wartość współczynnika α wynosi zwykle 0,8÷0,95. Temperatura może być obniżana również w inny sposób. Jeżeli podczas kilku kolejnych kroków obniżania temperatury nie następuje dalsza redukcja wartości funkcji kosztu problemu optymalizacyjnego lub temperatura zostanie obniżona poniżej pewnej wartości (ang. stopping criterion), to wykonywanie algorytmu jest przerywane, a otrzymane rozwiązanie jest rozwiązaniem problemu optymalizacyjnego [1, 3, 5-7, 9-10, 30, 90-92, 95, 97, 101].

Algorytm symulowanego wyżarzania jest uważany za udoskonaloną postać algorytmu zamiany parami. Algorytm zamiany parami akceptuje jedynie zmiany rozwiązania prowadzące do zmniejszenia funkcji kosztu, co jest powodem „utykania" algorytmu w minimach lokalnych funkcji kosztu. Akceptacja zmian rozwiązań prowadzących do wzrostu wartości funkcji kosztu w algorytmie symulowanego wyżarzania, umożliwia opuszczanie minimów lokalnych funkcji kosztu. Jeżeli temperatura jest wysoka to prawdopodobieństwo akceptacji gorszego rozwiązania jest bardzo duże. Obniżanie temperatury powoduje stopniowe zmniejszanie prawdopodobieństwa akceptacji gorszych rozwiązań. Algorytm symulowanego wyżarzania teoretycznie zapewnia osiągnięcie minimum globalnego funkcji kosztu, w przypadku utworzenia nieskończenie wiele nowych rozwiązań dla każdej temperatury. W praktyce stosuje się odpowiednio wolne sposoby obniżania temperatury. Algorytm symulowanego wyżarzania umożliwia otrzymanie rozwiązań o bardzo dobrej jakości. Wadą algorytmu jest jednak długi czas obliczeń i konieczność bardzo dokładnego dostrojenia parametrów algorytmu [1, 3, 5-7, 9, 30, 90, 95, 103].

Podczas rozmieszczania modułów w układach VLSI z użyciem algorytmu symulowanego wyżarzania jest wykorzystywana estymacja wartości funkcji kosztu, ponieważ wyznaczenie połączeń dla wszystkich utworzonych rozwiązań zajęłoby zbyt wiele czasu. Nowe rozwiązania są tworzone na podstawie bieżącego rozmieszczenia modułów. Nowe rozwiązania najczęściej są

tworzone w wyniku zamiany miejscami losowo wybranej pary modułów, przenoszenia losowo wybranego modułu w losowo wybrane miejsce, obrotu losowo wybranego modułu lub odbicia losowo wybranego modułu względem jednej z jego osi symetrii. Liczba nowych rozwiązań L, tworzonych podczas każdego kroku obniżania temperatury jest zwykle wielokrotnością liczby rozmieszczanych modułów [1, 3, 5-10, 30, 90, 95-99, 101-103].

Podczas rozmieszczania modułów w układach o topografii swobodnej lub podczas etapu planowania układu bardzo często jest stosowany algorytm symulowanego wyżarzania, który stosuje specjalne metody kodowania rozwiązań. Rozwiązanie jest określane wzajemnym położeniem modułów względem siebie, z użyciem specjalnego kodu. Taki sposób reprezentacji rozwiązania znacznie zmniejsza liczbę możliwych rozwiązań, w porównaniu do reprezentacji rozwiązań opartych na bezwzględnych współrzędnych modułów na podłożu układu. Powyższe kodowanie rozwiązań umożliwia zatem skrócenie czasu obliczeń. Stosowane są różne metody kodowania rozwiązania: z użyciem Odwrotnej Notacji Polskiej (ang. reverse Polish notation), z użyciem pary kolejności modułów SP (ang. sequence-pair), BSG (ang. bounded-sliceline grid), TCG (ang. transitive closure graph), O-drzewo (ang. O-tree, ordered tree), B*-drzewo (ang. B*-tree), CBL (ang. corner block list), TBS (ang. twin binary sequences). Powyższe metody kodowania wykluczają możliwość zachodzenia modułów na siebie [7, 11, 20-24, 86-87, 106, 120-121].

Algorytm symulowanego wyżarzania należy do najczęściej stosowanych rozwiązań podczas rozmieszczania modułów w układach VLSI. Programy wykorzystujące algorytm symulowanego wyżarzania umożliwiają otrzymanie rozwiązań o bardzo dobrej jakości i mogą być stosowane w układach o różnych stylach topografii. Przykładem takich programów może być: TimberWolf, iTools, MGP, MPG-MS, VPR, Parquet [6-7, 9-11, 20-24, 31, 45, 48, 54, 80, 95-103, 105-106, 120-121, 152].

3.1.3. Inne metody iteracyjne

Rozmieszczanie modułów w układach Standard Cell może być wykonane z użyciem algorytmu genetycznego lub strategii ewolucyjnej. Algorytm genetyczny operuje na wielu rozwiązaniach, natomiast strategia ewolucyjna na jednym rozwiązaniu. Obydwie metody umożliwiły otrzymanie rozwiązań, których jakość była porównywalna ze współczesną wersją programu TimberWolf. Czas potrzebny na wykonanie obliczeń metodą wykorzystującą algorytm genetyczny był porównywalny z czasem obliczeń programu TimberWolf. Metoda wykorzystująca strategię ewolucyjną zapewniała krótszy czas obliczeń [7, 9, 118-119].

3.2. Metody konstrukcyjne

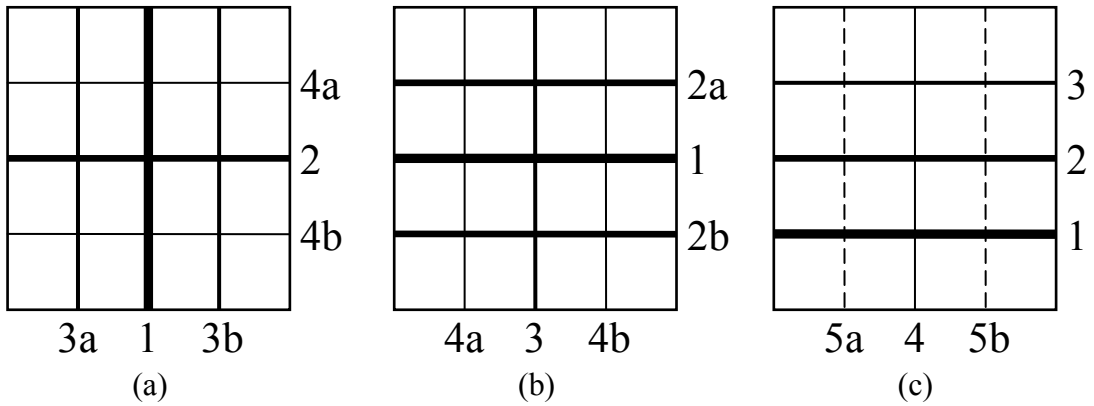
Pierwsze metody konstrukcyjne tworzyły rozwiązanie poprzez umieszczanie w każdym kroku jednego modułu, w dogodnym dla niego miejscu. Powyższe metody umożliwiają bardzo szybkie otrzymanie rozwiązania, jednak jakość rozwiązań jest niezadowalająca [3-6, 9, 28, 33, 65]. Z tego względu, obecnie metody te są czasem stosowane do tworzenia początkowych rozwiązań dla metod iteracyjnych, ze względu na ich szybkość [3, 5, 9, 65]. Do metod konstrukcyjnych zalicza się również metody wykorzystujące algorytm min-cut oraz metody analityczne. Powyższe metody należą obecnie do najpowszechniej stosowanych metod rozmieszczania [1, 3, 5, 9, 31, 45, 48, 72, 80, 119]. Algorytm min-cut wykorzystuje metodę „dziel i rządź", poprzez podział układu na części (podproblemy) [1, 3-6, 8-9, 30, 32-33]. Metody analityczne matematycznie wyznaczają położenie modułów [1, 3, 9, 30].

3.2.1. Algorytm min-cut

Algorytm min-cut opiera się na podziale układu na części tak, aby koszt połączeń między tymi częściami był minimalny. Koszt połączeń między częściami układu, które powstają w wyniku podziału jest równy sumie wag węzłów, które przecina linia podziału. Podczas podziału moduły są przenoszone między częściami układu z wykorzystaniem algorytmu Kernighana i Lina, algorytmu Fiduccia i Mattheysesa lub jego modyfikacji, w celu minimalizacji kosztu połączeń. Części powstające w wyniku podziału są dalej dzielone, dopóki każda z nich nie będzie zawierać dokładnie jednego modułu. W ten sposób każdy moduł ma określone położenie w układzie [1, 3-6, 8-9, 30, 32-33, 35].

Breuer zaproponował trzy sposoby podziału układu: podział na cztery części (ang. quadrature placement), podział na dwie połowy (ang. bisection placement) oraz podział typu slice/bisection (ang.). Podział na cztery części polega na ciągłym podziale układu na przemian liniami pionowymi i poziomymi. Ten sposób podziału jest najkorzystniejszy dla układów, które posiadają duże skupienie połączeń w środkowej części układu. Pierwszy podział układu na cztery części umożliwia zmniejszenie skupienia połączeń w środku układu. Podział na cztery części jest najczęściej stosowanym sposobem podziału. Podział na dwie połowy polega na ciągłym podziale układu na dwie równe połowy, najpierw liniami poziomymi, a następnie pionowymi. Podział liniami poziomymi umożliwia podzielenie układu na wiersze, do których są przypisane określone moduły. Następnie, podział liniami pionowymi umożliwia wyznaczenie pozycji modułów w wierszach. Ten rodzaj podziału jest korzystny dla układów Standard Cell. Podział typu slice/bisection polega na podziale układu najpierw liniami poziomymi, które mogą dzielić układ na części o różnych rozmiarach. Podział liniami poziomymi umożliwia przypisanie modułów do wierszy. Następnie układ jest dzielony liniami pionowymi na dwie równe części, co

umożliwia określenie położenia modułów w wierszach. Ten sposób podziału jest korzystny dla układów, które posiadają duże skupienie połączeń na obrzeżach układu [3-4, 9, 30]. Należy podkreślić, że po ustaleniu położenia modułów względem danej linii podziału, położenie modułów względem tej linii nie może ulec zmianie (moduły nie mogą być przeniesione na drugą stronę linii). Rysunek 12 przedstawia stosowane sposoby podziału układu, kolejność podziału określa numer oraz szerokość linii.

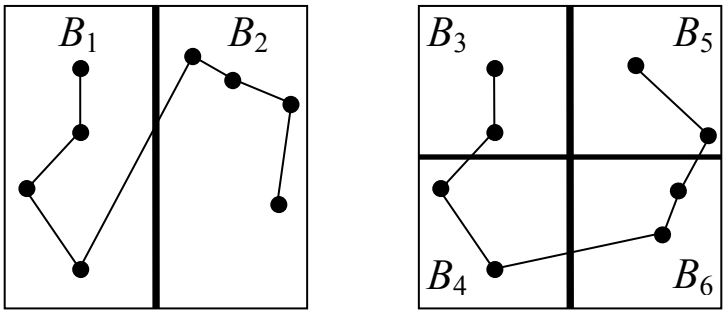


Rys. 12. Sposoby podziału układu w algorytmie min-cut: a) podział na cztery części, b) podział na dwie połowy, c) podział typu slice/bisection

Dla każdego sposobu podziału, kolejność podziału liniami poziomymi i pionowymi może być odwrócona. Podczas podziału układu, każdej części układu odpowiada określony poziom, poczynając od poziomu 1 dla całego układu. W wyniku podziału danej części układu, której odpowiada poziom *j,* powstają części z przypisanym poziomem *j*+1. Poszczególne części układu mogą być dzielone w różnej kolejności. Z reguły najpierw są dzielone wszystkie bloki posiadające przypisany poziom *j,* a następnie są dzielone części na poziomie *j*+1. Sposób ten jest podziałem wszerz (ang. breadth first). W drugim sposobie, w podziale w głąb (ang. depth first), do następnego podziału wybierana jest część, która może być dalej dzielona i posiada największy przypisany poziom. Części układu, które powstają w wyniku jego podziału, są kolejno oddzielnie przetwarzane. Umożliwia to znaczną redukcję nakładów obliczeniowych, jednak nie zapewnia osiągnięcia minimum globalnego funkcji kosztu. Korzystne może być również stosowanie odrębnych linii podziału dla poszczególnych części układu. Początkowe położenie modułów w układzie może być określone w sposób losowy. W tym celu można zastosować również prosty algorytm konstrukcyjny, umieszczający jeden moduł w każdym kroku, który może wykorzystać również ustaloną pozycję niektórych modułów w układzie [9, 30].

Rysunek 13 przedstawia podział układu na cztery części: B_3, B_4, B_5, B_6. W powyższym przykładzie, wszystkie połączenia stanowią odrębne węzły. Breuer zaproponował iteracyjny sposób podziału układu. Po wykonaniu podziału układu linią pionową na części B_1 i B_2, każda z tych części jest dalej dzielona linią poziomą. Podczas podziału części B_1 są ignorowane moduły znajdujące się w części B_2, ponieważ nie jest określone ich położenie względem linii poziomej. Podczas podziału B_2 na części B_5 i B_6, jest

uwzględniane położenie modułów znajdujących się w częściach B_3 i B_4, ponieważ ma ono wpływ na koszt podziału B_2 linią poziomą. Po podziale części B_2, jest korygowany podział części B_1. W tym celu jest wykorzystywane ustalone już położenie modułów w częściach B_5 i B_6, ponieważ ma ono wpływ na koszt podziału B_1 linią poziomą. Następnie, na tej samej zasadzie odbywa się korekcja podziału części B_2. Iteracyjna korekcja podziału części B_1 oraz B_2 jest wykonywana na przemian, aż do ustalenia wyniku. Podczas podziału układu w algorytmie zaproponowanym przez Breuera, nie jest jednak uwzględniane położenie modułów w pozostałych częściach układu.



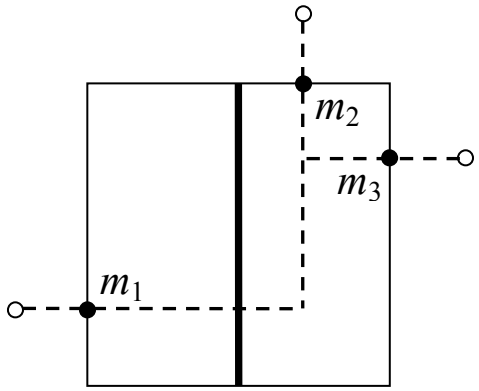
(a) (b)

Rys. 13. Iteracyjny podział układu w algorytmie min-cut: a) podział linią pionową, b) podział linią poziomą

Powyższy problem rozwiązuje modyfikacja wprowadzona przez Dunlopa i Kernighana (ang. terminal propagation). Podczas podziału układu lub jego części, są uwzględniane połączenia z modułami znajdującymi się w innych częściach oraz połączenia z końcówkami całego układu. Zakłada się, że moduły umieszczone w innych częściach układu są położone w środku geometrycznym tych części. W ogólnym przypadku, dla każdego węzła układu, moduły położone w dzielonej części, końcówki całego układu oraz moduły położone na zewnątrz dzielonej części, są łączone z użyciem minimalnego drzewa Steinera. Punkty przecięcia tego drzewa z granicami dzielonej części, są traktowane jak dodatkowe moduły, o trwale ustalonej pozycji. Dodatkowe moduły położone blisko linii podziału nie są uwzględniane. W ten sposób, moduły znajdujące się w pozostałych częściach układu mają wpływ na sposób podziału danej jego części [33]. Rysunek 14 ilustruje opisaną modyfikację. Na obrzeżach dzielonej części układu zostały umieszczone dodatkowe moduły m_1, m_2 i m_3. Powyższa modyfikacja umożliwia zmniejszenie rozmiarów układu o ok. 30%, w porównaniu do wyników uzyskanych z użyciem zwykłego algorytmu min-cut. Technika ta wymaga podziału układu wszerz [6, 9, 33]. Położenie dodatkowych modułów może być również określone bez konieczności konstruowania minimalnego drzewa Steinera. Dodatkowe moduły mogą być umieszczane w najbliższych punktach obrzeża dzielonej części układu, mogą nie być uwzględniane w przypadku, gdy znajdują się blisko linii podziału [6, 9, 33].

Przełomowym osiągnięciem dla algorytmu podziału oraz algorytmu min-cut jest jednak wprowadzenie wielopoziomowego podziału układu (ang. multilevel circuit partitioning). Metoda ta umożliwia podział układów o

bardzo dużych rozmiarach, rzędu setek tysięcy modułów. Powszechnie stosowanym algorytmem wielopoziomowego podziału układu jest algorytm hMETIS [36-39]. W ostatnich latach pojawiło się wiele programów rozmieszczania dla układów Standard Cell, które wykorzystują algorytm min-cut oraz wielopoziomowy podział układu. Przykładem takich programów może być: Capo, Dragon, Feng Shui, QUAD. W powyższych programach algorytm min-cut jest wykorzystywany do rozmieszczania globalnego, natomiast rozmieszczanie szczegółowe jest wykonywane z użyciem innych metod [31, 40-54, 56, 150].



Rys. 14. Modyfikacja Dunlopa i Kernighana

Algorytm min-cut jest oparty na minimalizacji skupienia połączeń, która z kolei pośrednio wpływa na minimalizację powierzchni układu, ze względu na mniejsze rozmiary kanałów do prowadzenia połączeń. Minimalizacja skupienia połączeń pośrednio przyczynia się do minimalizacji długości połączeń, ponieważ silnie połączone ze sobą moduły są rozmieszczane obok siebie. Chociaż algorytm min-cut powstał blisko 30 lat temu, jego modyfikacje należą do najpowszechniej stosowanych metod rozmieszczania [1, 3, 5, 8-9, 31, 33, 40, 43-46, 48, 80].

3.2.2. Metody analityczne

Metody analityczne matematycznie wyznaczają położenie modułów. W tym celu są stosowane różne metody programowania matematycznego: programowanie nieliniowe (ang. nonlinear programming NLP), programowanie kwadratowe (ang. quadratic programming QP), które jest szczególnym przypadkiem programowania nieliniowego oraz programowanie liniowe (ang. linear programming LP) [3, 63, 66-68]. Celem tych metod jest znalezienie minimum globalnego funkcji kosztu problemu rozmieszczania. W ten sposób jest wykonywane rozmieszczanie globalne modułów. Podczas rozmieszczania modułów z użyciem metod analitycznych występuje jednak problem zachodzenia modułów na siebie. Jest to spowodowane tym, że funkcja kosztu nie uwzględnia rozmiarów modułów oraz ograniczeń wynikających z wybrania

konkretnego stylu topografii układu. Moduł traktowany jest jak punkt, w którym znajdują się wszystkie końcówki modułu. W celu rozwiązania powyższego problemu powszechnie są stosowane dwa rozwiązania: jest wykonywany odpowiedni podział układu lub są wprowadzane dodatkowe warunki (siły), które wymuszają konieczność rozwiązania nowego problemu optymalizacyjnego. W wyniku zastosowania tych rozwiązań, stopniowo następuje równomierne rozmieszczenie modułów na całym podłożu układu. W przypadku równomiernego rozmieszczenia modułów jest możliwe rozmieszczenie modułów bez zachodzenia na siebie, podczas rozmieszczania szczegółowego [9, 30-31, 53, 70-72, 75-78, 81-82, 84-85].

Metody analityczne bardzo często wykorzystują kwadrat długości euklidesowej połączeń. W przypadku zastosowania kwadratu długości euklidesowej funkcja kosztu *f*, która jest ważoną sumą miar długości wszystkich połączeń w układzie, dana jest wzorem [1, 3, 9, 30-31, 70-71, 75-77, 79-82, 84]

$$f = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right) \tag{6}$$

gdzie: w_{ij} - waga połączenia między modułami *i* oraz *j*, (x_i, y_i), (x_j, y_j) - współrzędne modułów *i* oraz *j* na podłożu układu, *n* - liczba modułów w układzie.

Udowodniono, że funkcję kosztu *f* można zapisać macierzowo w następującej postaci [1, 9, 30, 81-82]

$$f = \mathbf{x}^T \mathbf{B}\ \mathbf{x} + \mathbf{y}^T \mathbf{B}\ \mathbf{y} \tag{7}$$

gdzie: **x** - wektor *n*-wymiarowy współrzędnych *x* wszystkich modułów, **y** - wektor *n*-wymiarowy współrzędnych *y* wszystkich modułów, **B** - symetryczna macierz o wymiarze *n* x *n*.

Jeżeli uwzględnimy połączenia modułów układu z końcówkami całego układu, które mają ustalone położenie w układzie, to funkcję kosztu *f* można zapisać macierzowo w następującej postaci [70, 75-77, 84]

$$f = \frac{1}{2} \mathbf{x}^T \mathbf{C}\ \mathbf{x} + \mathbf{d}_x^T \mathbf{x} + \frac{1}{2} \mathbf{y}^T \mathbf{C}\ \mathbf{y} + \mathbf{d}_y^T \mathbf{y} + \text{constans} \tag{8}$$

gdzie: **C** - symetryczna macierz o wymiarze *n* x *n*, $\mathbf{d}_x$, $\mathbf{d}_y$ - wektory *n*-wymiarowe dla osi *x* oraz osi *y*.

Wektory $\mathbf{d}_x$, $\mathbf{d}_y$ odzwierciedlają połączenia modułów układu z końcówkami całego układu. Obydwa składniki funkcji kosztu *f* we wzorze 8 są niezależne od siebie. W związku z tym minimalizacja funkcji kosztu może być wykonana oddzielnie dla osi *x* oraz osi *y* [1, 9, 30, 70, 75-77, 81-82, 84]. Dla węzłów zawierających więcej niż 2 końcówki, dla których topologia połączeń nie jest z góry narzucona, mogą być stosowane połączenia między wszystkimi końcówkami węzła, tak jak w grafie pełnym [1, 3-4, 30, 70, 73-77, 81, 84].

Zadanie programowania nieliniowego polega na znalezieniu ekstremum globalnego funkcji celu, która jest określona na zbiorze wyznaczonym przez

funkcje ograniczeń, zwanym zbiorem rozwiązań dopuszczalnych. W przypadku zadania programowania nieliniowego przynajmniej jedna z funkcji, tzn. funkcja celu lub jedna z funkcji ograniczeń jest funkcją nieliniową. Zadanie programowania kwadratowego jest szczególnym przypadkiem zadania programowania nieliniowego, w którym funkcja celu jest sumą formy liniowej i kwadratowej, natomiast ograniczenia są liniowe. Zadanie programowania kwadratowego polega na znalezieniu wektora $\mathbf{x_0}$, który minimalizuje (maksymalizuje) funkcję celu *f* [66-67]

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{D}\ \mathbf{x} + \mathbf{c}^T \mathbf{x} \tag{9}$$

pod warunkiem spełnienia następujących ograniczeń

$$\mathbf{A}\ \mathbf{x} = \mathbf{b} \tag{10}$$

$$\mathbf{x} \geq \mathbf{0} \tag{11}$$

gdzie: **x**, **c** - wektory *n*-wymiarowe, **b** - wektor *m*-wymiarowy, **A** - macierz o wymiarze *m* x *n*, **D** - macierz o wymiarze *n* x *n*, dodatkowo zakłada się, że macierz **D** jest symetryczna.

W przypadku braku ograniczeń otrzymujemy zadanie programowania kwadratowego bez ograniczeń (ang. unconstrained quadratic programming problem). Jeżeli macierz **D** jest dodatnio określona (ang. positive definite) i funkcja celu *f* jest różniczkowalna, to rozwiązanie następującego układu równań jest jedynym minimum funkcji celu *f*, które jest jednocześnie minimum globalnym

$$2\ \mathbf{D}\ \mathbf{x} + \mathbf{c} = \mathbf{0} \tag{12}$$

Rozwiązanie powyższego układu równań jest rozwiązaniem zadania programowania kwadratowego bez ograniczeń [63, 66-67].

Ze wzorów 8 i 9 wynika, że postać funkcji kosztu problemu rozmieszczania modułów dla jednej z osi układu współrzędnych jest taka sama, jak postać funkcji celu zadania programowania kwadratowego. Rozwiązanie zadania programowania kwadratowego bez ograniczeń jest równocześnie rozwiązaniem problemu rozmieszczania modułów dla jednej z osi układu współrzędnych. Rozwiązanie problemu rozmieszczania modułów sprowadza się zatem do rozwiązania układu równań. Rozwiązanie tego układu jest jedynym minimum funkcji kosztu problemu rozmieszczania, które jest jednocześnie minimum globalnym, ponieważ funkcja kosztu jest różniczkowalna, natomiast macierz **C** dla problemu rozmieszczania we wzorze 8 jest dodatnio określona. Powyższy układ musi być rozwiązany dla osi *x* oraz dla osi *y*. Istnieje wiele efektywnych metod rozwiązania powyższego układu równań. Dzięki temu, jest możliwe efektywne rozwiązanie problemu rozmieszczania globalnego modułów, nawet w przypadku bardzo dużych układów. Otrzymane rozwiązanie nie uwzględnia jednak rozmiarów modułów oraz ograniczeń wynikających z wybrania konkretnego stylu topografii układu, co powoduje m.in. występowanie zachodzenia modułów na siebie [3, 9, 30-31, 53, 63, 66-67, 70-71, 75-78, 81-82, 84-85]. W przypadku uwzględnienia powyższych ograniczeń funkcja kosztu może jednak posiadać wiele minimów lokalnych [3, 79].

Zadanie programowania liniowego polega na minimalizacji lub maksymalizacji liniowej funkcji celu w zbiorze określonym przez układ liniowych ograniczeń, tzn. równań lub nierówności liniowych. Zgodnie ze standardową (kanoniczną) postacią, zadanie programowania liniowego polega na znalezieniu wektora $\mathbf{x_0}$, który minimalizuje funkcję celu *f* [63, 66-68]

$$f(\mathbf{x}) = \mathbf{c}\ \mathbf{x} \tag{13}$$

pod warunkiem spełnienia następujących ograniczeń

$$\mathbf{A}\ \mathbf{x} = \mathbf{b} \tag{14}$$

$$\mathbf{x} \geq \mathbf{0} \tag{15}$$

gdzie: **x**, **c** - wektory *n*-wymiarowe, **c x** - iloczyn skalarny wektorów **c** i **x**, **A** - macierz o wymiarze *m* x *n*, **b** - nieujemny wektor *m*-wymiarowy, dodatkowo zakłada się, że $n \geq m$.

Możliwy jest również inny zapis zadania programowania liniowego, w którym zamiast ograniczenia wyrażonego wzorem 14, występuje ograniczenie $\mathbf{A\,x} \geq \mathbf{0}$ lub $\mathbf{A\,x} \leq \mathbf{0}$. Takie postacie zadania programowania liniowego można jednak sprowadzić do standardowej postaci zadania, w wyniku wprowadzenia nowych, nieujemnych zmiennych. Zadanie programowania liniowego można rozwiązać metodą simpleks (sympleksów) (ang. simplex method) [63, 66-68].

Problem rozmieszczania modułów może być rozwiązany z wykorzystaniem programowania liniowego. Możliwe jest rozmieszczanie modułów w układach o topografii swobodnej, w której moduły posiadają dowolne rozmiary i położenie. Funkcja kosztu *f* problemu rozmieszczania jest bardzo często ważoną sumą całkowitej estymowanej długości połączeń (metoda połowy obwodu) oraz całkowitej powierzchni układu [11, 64]

$$f = \alpha\ w\ h + \sum_{i=1}^{n}\left(w_i^{\,w} + h_i^{\,w}\right) \tag{16}$$

gdzie: *w*, *h* - szerokość oraz wysokość podłoża układu, $w_i{}^w$, $h_i{}^w$ - szerokość oraz wysokość prostokąta obejmującego wszystkie końcówki węzła *i*, α - współczynnik określający udział całkowitej powierzchni układu w funkcji kosztu, *n* - liczba węzłów w układzie.

Powyższa funkcja kosztu nie jest funkcją liniową, dlatego funkcja *f* nie może być funkcją celu w zadaniu programowania liniowego. W celu rozwiązania tego problemu, zakłada się stałą wartość jednego z wymiarów podłoża układu, np. stałą szerokość podłoża równą *W*. Wówczas składnik α *w h* jest zastępowany przez liniową funkcję α *W h*. Innym rozwiązaniem może być aproksymacja składnika α *w h* przez β (*w*+*h*), gdzie β jest współczynnikiem zapewniającym dobrą aproksymację składnika [11, 64]. Zastosowana modyfikacja umożliwia użycie funkcji kosztu *f* jako funkcji celu w zadaniu programowania liniowego. Rozwiązanie zadania programowania liniowego powinno określać optymalne rozmieszczenie modułów w układzie o topografii swobodnej. W związku z tym rozwiązanie musi spełniać również szereg dodatkowych ograniczeń: moduły nie mogą na siebie zachodzić, moduły w całości muszą być położone na podłożu układu, moduły mogą posiadać

orientację poziomą lub pionową. Powyższe ograniczenia można użyć w zadaniu programowania liniowego, ponieważ są liniowe. Ograniczenia te wymagają jednak wprowadzenia całkowitych zmiennych. Możliwości zastosowania programowania liniowego dla większych układów są jednak ograniczone. Wraz ze wzrostem rozmiaru problemu następuje bardzo szybki wzrost czasu potrzebnego na przeprowadzenie obliczeń. Programowanie liniowe może być również stosowane podczas etapu planowania topografii układu [9, 11-12, 64, 71, 85].

Metody analityczne są wykorzystywane do rozmieszczania modułów od wielu lat. Nadal są zaliczane do najczęściej stosowanych rozwiązań. W ostatnich latach pojawiło się wiele programów rozmieszczania, których podstawą są metody analityczne. Przykładem takich programów może być: GORDIAN, DOMINO, KraftWerk, FastPlace, mPL, FAR, mFAR, BloBB, APlace. Powyższe programy rozmieszczania mogą być stosowane w układach o różnych stylach topografii. Powodem ciągłego zainteresowania badaczy metodami analitycznymi jest ich duża efektywność, która umożliwia rozmieszczanie modułów w układach o bardzo dużych rozmiarach. Większość obecnych programów rozmieszczania opartych na metodach analitycznych, wykorzystuje metodę programowania kwadratowego, która umożliwia efektywne rozmieszczanie globalne modułów. Rozmieszczanie szczegółowe zwykle jest wykonywane z użyciem innych metod [31, 45, 48, 50, 52-54, 70-72, 75-87, 151].

3.3. Sieci neuronowe

Sieci neuronowe (ang. neural network) mogą być również stosowane podczas projektowania topografii układów VLSI. Istnieje wiele rodzajów sieci neuronowych [2, 69, 91-94, 107-111]. Sieci neuronowe cieszą się dużym zainteresowaniem, ze względu na ich zdolność równoległego przetwarzania danych przez wiele pracujących równocześnie elementów sieci. Zalety sieci neuronowych są najlepiej wykorzystane w przypadku sprzętowej realizacji sieci, co umożliwia uzyskanie rozwiązania w krótkim czasie. W dodatku, wzrost wymiaru problemu nie pociąga wtedy za sobą istotnego wzrostu czasu oczekiwania na rozwiązanie [108, 111]. Przykładem sieci neuronowych, które mogą być stosowane do rozmieszczania modułów w układach VLSI jest sieć samoorganizująca się [91, 112-115, 152] oraz sieć Hopfielda [2, 92, 116, 152].

3.4. Podsumowanie

Do metod rozmieszczania modułów, które są najczęściej stosowane należą: metody wykorzystujące algorytm min-cut, metody analityczne oraz metody oparte na algorytmie symulowanego wyżarzania [31, 45, 48, 80]. Obecnie, większość programów wykorzystuje minimalizację długości połączeń

podczas rozmieszczania modułów. Minimalizacja długości połączeń może być prowadzona równocześnie z minimalizacją opóźnień w krytycznych częściach układu, co zapewnia spełnienie wymagań funkcjonalnych stawianych układowi. Podczas minimalizacji długości połączeń bardzo często jest wykorzystywana estymacja długości połączeń metodą połowy obwodu. Zaletą tej metody jest niski nakład obliczeniowy. Ponadto wykazano, że rozwój technologii wytwarzania układów i systemów scalonych, który charakteryzuje się coraz większą liczbą warstw metalizacji w układzie, przyczynia się do poprawy dokładności tej metody estymacji. W układach o dużej liczbie warstw metalizacji metoda połowy obwodu staje się bardziej dokładna, ponieważ większa część węzłów jest połączona najkrótszymi połączeniami [120-121]. Minimalizacja długości połączeń pośrednio przyczynia się do minimalizacji skupienia połączeń. Ze względu na brak skutecznych i dokładnych metod estymacji skupienia połączeń, dokładne określenie skupienia połączeń ciągle wymaga wykonania trasowania globalnego. Podejmowane są próby integracji algorytmów trasowania globalnego z algorytmami wykonującymi rozmieszczanie modułów [101].

Skuteczność programów rozmieszczania zależy od cech rozmieszczanego układu, co należy uwzględnić podczas wyboru programu. Programy minimalizujące długość połączeń oparte na metodach analitycznych gorzej radzą sobie z układami, które posiadają większą liczbę połączeń obejmujących znaczną część układu. W takich układach programy oparte na metodach analitycznych muszą kłaść większy nacisk na usuwanie zachodzenia między modułami, co powoduje pogorszenie długości połączeń. Problem ten nasila się w przypadku programów, których funkcja kosztu wykorzystuje kwadrat długości połączeń, ponieważ występuje wtedy jeszcze uprzywilejowanie długich połączeń. W przypadku programów opartych na algorytmie min-cut połączenia obejmujące znaczną część układu przyczyniają się do poprawy jakości rozwiązań. W przypadku programów opartych na algorytmie symulowanego wyżarzania bardzo istotna jest dokładność, z jaką jest określane położenie modułów. Jeżeli położenie modułów jest określane przez przynależność modułów do określonych części podłoża układu, to współrzędne środków geometrycznych tych części określają położenie modułów. Zbyt duże rozmiary tych części mogą być powodem pogorszenia jakości rozwiązań. Powyższa sytuacja występuje w programie Dragon i jest szczególnie widoczna dla układów, które posiadają małą liczbę połączeń obejmujących znaczną część układu. Programy oparte na algorytmie symulowanego wyżarzania wymagają znacznie dłuższego czasu na wykonanie obliczeń, w porównaniu do programów wykorzystujących algorytm min-cut oraz metody analityczne [52-54, 84]. Programy oparte na metodach analitycznych gorzej radzą sobie z układami, w których jest mało wolnej przestrzeni na podłożu układu, ze względu na utrudnioną eliminację zachodzenia modułów na siebie. W przypadku programów wykorzystujących algorytm min-cut sytuacja przedstawia się zupełnie inaczej, powyższe programy gorzej radzą sobie z układami, w których jest zbyt dużo wolnej przestrzeni na podłożu układu [31].

Wiele istotnych informacji dotyczących całej problematyki projektowania topografii układów VLSI można znaleźć na stronie internetowej VLSI CAD Bookshelf. Strona jest redagowana przez wybitnych naukowców, zajmujących się automatyzacją projektowania topografii układów VLSI. Wiele rozwiązań jest dostępnych w postaci kodu źródłowego [125].

Rozdział 4
Minimalizacja długości połączeń z wykorzystaniem sieci Hopfielda

W niniejszym rozdziale przedstawiono podstawy teoretyczne zastosowania sieci Hopfielda do minimalizacji długości połączeń w układach VLSI. Minimalizacja długości połączeń jest bardzo często stosowana podczas projektowania topografii układów VLSI. Zmniejszenie długości połączeń powoduje zmniejszenie pojemności, indukcyjności i rezystancji wnoszonych do układu przez połączenia. Minimalizacja długości połączeń ma zatem niebagatelny wpływ na czas propagacji w układach cyfrowych oraz ilość wydzielanego ciepła. We współczesnych układach VLSI, opóźnienia wnoszone przez połączenia mają decydujące znaczenie dla czasu propagacji sygnałów, w porównaniu do opóźnień wnoszonych przez elementy aktywne. Jest to spowodowane stale postępującą miniaturyzacją układów oraz stałym wzrostem częstotliwości sygnału taktującego. Minimalizacja długości połączeń może być prowadzona równocześnie z minimalizacją opóźnień w krytycznych częściach układu (podrozdział 2.3.3), co umożliwia znaczne zmniejszenie czasu propagacji sygnałów w układzie oraz zastosowanie sygnału taktującego o większej częstotliwości [1, 3, 9, 53-54, 56-60, 77, 84, 103-104]. Jeżeli uświadomimy sobie, że połączenia zajmują nawet 50÷70% całkowitej powierzchni układu, nie bez znaczenia okaże się wpływ długości połączeń na rozmiary układu [8-9, 29-30, 33-34]. Minimalizacja długości połączeń pośrednio przyczynia się również do minimalizacji skupienia połączeń i umożliwia wykonanie wszystkich połączeń, w przypadku ograniczonej pojemności kanałów, przeznaczonych do prowadzenia połączeń [4, 10, 30, 84, 101]. Problem minimalizacji długości połączeń podczas rozmieszczania modułów w układach VLSI jest przedstawiony w podrozdziale 4.1. W podrozdziale 4.2 opisano sieć Hopfielda, która jest stosowana do rozwiązywania problemów optymalizacyjnych. Postać funkcji kosztu w problemie minimalizacji długości połączeń przedstawiono w podrozdziale 4.3. Analog elektryczny sieci Hopfielda jest omówiony w podrozdziale 4.4.

4.1. Postawienie problemu

Metody rozmieszczania modułów bardzo często wykorzystują założenie, że długość podłoża układu jest podzielona na K odcinków o rozmiarze jednostkowym, a szerokość płytki na L odcinków o rozmiarze jednostkowym. Podłoże układu jest zatem podzielone na siatkę kwadratowych części. Zakłada się, że moduły są kwadratami o rozmiarze równym jednej jednostce. Każdy moduł powinien być umieszczony w dokładnie jednej części podłoża. W każdej

części nie może być umieszczonych więcej niż jeden moduł. W związku z tym, w celu rozmieszczenia X modułów musi być spełniony warunek $X \le M = K\,L$. Ponadto zakłada się, że końcówki modułów są położone w środku geometrycznym części układu, w której znajduje się dany moduł. Całkowita ważona długość połączeń w układzie, minimalizowana w wyniku rozmieszczenia X modułów, jest określona wzorem

$$całkowita\ długość\ połączeń = \frac{1}{2}\sum_{i=1}^{X}\sum_{j=1}^{X} d_{ij}\,cm_{ij} \tag{17}$$

gdzie: d_{ij} - odległość między środkami geometrycznymi części podłoża, w których zostały umieszczone moduły i oraz j, cm_{ij} - element macierzy połączeń **CM** układu, określający wagę połączenia między modułami i oraz j.

Elementy macierzy **CM** są określone następująco

$$\begin{cases} cm_{ij} > 0 & \text{gdy jest połączenie między modułami } i \text{ oraz } j \\ cm_{ij} = 0 & \text{w przeciwnym przypadku} \end{cases} \tag{18}$$

Odległość między modułami jest wyznaczana w metryce Manhattan, zgodnie ze wzorem 1.

4.2. Opis sieci Hopfielda

Jak już wcześniej wspomniano, problemy optymalizacyjne można rozwiązywać z wykorzystaniem sieci neuronowych. Jednym ze sposobów jest zastosowanie sieci Hopfielda, wykorzystując jej własność, polegającą na minimalizacji funkcji energii.

Zastosowanie znajduje tutaj model z ciągłą charakterystyką neuronów, zamiast dwuwartościowej 0/1 lub ±1. Stosowanie dwuwartościowych neuronów nie przynosi dobrych rezultatów w rozwiązywaniu problemów optymalizacyjnych, gdyż sieć szybko osiąga lokalne minimum funkcji energii, a rozwiązanie nie jest optymalne [109].

Funkcja aktywacji neuronów w modelu ciągłym sieci Hopfielda bardzo często jest określana wzorem [93, 107-108]

$$V = f(U) = \frac{1}{2}\left(1 + \tanh\left(\frac{U}{U_0}\right)\right) \tag{19}$$

gdzie: U - sygnał wejściowy neuronu, V - sygnał wyjściowy, U_0 - parametr.

Dla małych wartości U_0 funkcja aktywacji jest bardzo stroma i zbliża się do funkcji skoku jednostkowego.

Dla tak określonej sieci wprowadza się pojęcie energii sieci. Dla małych wartości U_0 we wzorze 19, funkcja energii upraszcza się do postaci bez wyrażeń całkowych [2, 92-94, 107-108, 110]

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}V_iV_j - \sum_{i=1}^{N}I_iV_i \tag{20}$$

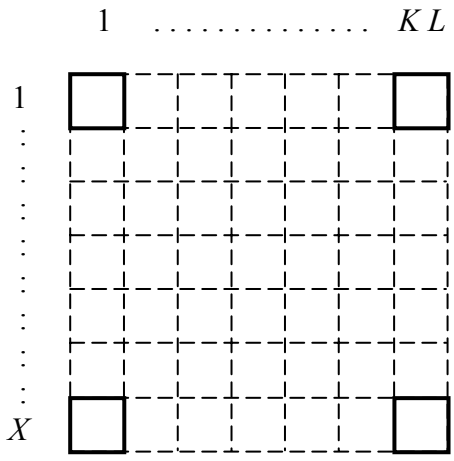
gdzie: E - energia sieci, N - liczba neuronów, ω_{ij} - waga połączenia wyjścia neuronu j z wejściem neuronu i, I_i - zewnętrzny sygnał wejściowy neuronu i.

Dla sieci z symetryczną macierzą wag połączeń funkcja energii jest funkcją nierosnącą w czasie [92-94, 107-110]. Własność ta gwarantuje, że sieć zainicjowana pewnymi wartościami początkowymi będzie zmierzać do stanu stabilnego odpowiadającemu minimum funkcji.

Jeżeli problem optymalizacyjny daje się sprowadzić do postaci odpowiadającej funkcji energii, to sieć minimalizująca swoją energię może posłużyć do rozwiązania tego problemu [2, 69, 92-94, 107-110, 126-129].

4.3. Postać funkcji kosztu w problemie minimalizacji długości połączeń

W rozpatrywanym problemie, sieć Hopfielda ma dokonać właściwego rozmieszczenia modułów w układzie, tak, aby całkowita ważona długość połączeń między modułami określona wzorem 17 była minimalna. W tym celu buduje się sieć, w której liczba neuronów $N = X\,K\,L$. Sieć składa się z X wierszy, zawierających $M = K\,L$ neuronów, zgodnie z rysunkiem 15.



Rys. 15. Podział sieci na wiersze i kolumny

Każdy neuron w sieci jest indeksowany dwoma wskaźnikami. Pierwszy wskaźnik określa numer modułu, który ma być rozmieszczany, a drugi oznacza numer części podłoża. Jeżeli neuron o współrzędnych *(x, i)* po osiągnięciu przez sieć stanu stabilnego posiada wartość $V_{x\,i} = 1$, wówczas oznacza to, że moduł *x* powinien być umieszczony w części podłoża o numerze *i*. Jeden ze sposobów numerowania części podłoża przedstawia rysunek 16.

7	8	9
4	5	6
1	2	3

Rys. 16. Jeden ze sposobów numerowania części podłoża

Należy dodać, że podział sieci na wiersze i kolumny ma charakter umowny i nie ma żadnego wpływu na topologię połączeń między neuronami.

Można teraz przystąpić do określenia funkcji kosztu, która będzie poddawana procesowi minimalizacji. Funkcja kosztu będzie zawierać składnik określony wzorem 17, ale musi zawierać także ograniczenia związane z rozmieszczaniem modułów w układzie. W rezultacie funkcja kosztu będzie składała się z następujących składników (ograniczeń):

a) Ograniczenie 1
W każdej części układu nie może znajdować się więcej niż jeden moduł. Uwzględniając podział sieci na wiersze zgodnie z rysunkiem 15, ograniczenie to wymaga, aby w stanie stabilnym żadne dwa neurony znajdujące się w jednej kolumnie nie posiadały równocześnie na wyjściu sygnału równego 1. Wówczas składnik ten będzie miał wartość 0. Ograniczenie jest więc wyrażone wzorem

$$E_1 = \frac{A}{2}\sum_{x=1}^{X}\sum_{i=1}^{M}\sum_{\substack{x'=1\\x'\neq x}}^{X} V_{x\,i}V_{x'i} = \frac{A}{2}\sum_{x=1}^{X}\sum_{i=1}^{M}\sum_{x'=1}^{X}\sum_{i'=1}^{M}\delta_{i\,i'}\left(1-\delta_{x\,x'}\right)V_{x\,i}V_{x'i'} \qquad (21)$$

gdzie: *A* jest stałą, $\delta_{i\,i'} = \begin{cases} 1 & \text{gdy } i = i' \\ 0 & \text{gdy } i \neq i' \end{cases}$, analogicznie $\delta_{x\,x'}$

b) Ograniczenie 2
Każdy moduł powinien być umieszczony w dokładnie jednej części podłoża. Ograniczenie można zapisać w następujący sposób

$$E_2 = \frac{B}{2}\sum_{x=1}^{X}\left(\sum_{i=1}^{M}V_{x\,i}-1\right)^2 = \frac{B}{2}\sum_{x=1}^{X}\left(\left(\sum_{i=1}^{M}V_{x\,i}\sum_{i=1}^{M}V_{x\,i}\right)-2\sum_{i=1}^{M}V_{x\,i}+1\right) =$$

$$= \frac{B}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{i'=1}^{M} V_{x\,i} V_{x\,i'} - B \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} + \frac{X\,B}{2} =$$

$$= \frac{B}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{x'=1}^{X} \sum_{i'=1}^{M} \delta_{x\,x'} V_{x\,i} V_{x'\,i'} - B \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} + \frac{X\,B}{2} \tag{22}$$

gdzie: B jest stałą.

Składnik E_2 nie będzie równy 0, jeżeli więcej niż jeden neuron w dowolnym wierszu sieci będzie posiadał w stanie stabilnym sygnał wyjściowy równy 1 lub jeżeli w dowolnym wierszu żaden neuron nie będzie posiadał na wyjściu sygnału równego 1. Jeżeli w każdym wierszu sieci dokładnie jeden neuron będzie posiadał na wyjściu sygnał równy 1, to składnik E_2 będzie równy 0.

c) Ograniczenie 3
Sygnały wyjściowe wszystkich neuronów w stanie stabilnym powinny wynosić 0 lub 1 lub być bliskie tym wartościom. Ograniczenie to zapisujemy następującym wzorem

$$E_3 = \frac{C}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} \left(1 - V_{x\,i} \right) = \frac{C}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} - \frac{C}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} V_{x\,i} =$$

$$= \frac{C}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} V_{x\,i} - \frac{C}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{x'=1}^{X} \sum_{i'=1}^{M} \delta_{x\,x'} \delta_{i\,i'} V_{x\,i} V_{x'\,i'} \tag{23}$$

gdzie: C jest stałą.

d) Ograniczenie 4
Składnik określający całkowitą ważoną długość połączeń jest wyrażony następującym wzorem

$$E_4 = \frac{D}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{x'=1}^{X} \sum_{i'=1}^{M} V_{x\,i} V_{x'i'} cm_{x\,x'} d_{i\,i'} \tag{24}$$

gdzie: D - stała, $d_{i\,i'}$ - odległość między częściami podłoża i oraz i ', $cm_{x\,x'}$ - element macierzy połączeń dla modułów x oraz x '.

Funkcja kosztu, która będzie poddawana minimalizacji przez sieć, będzie wyrażona więc wzorem

$$E = E_1 + E_2 + E_3 + E_4 =$$

$$= -\frac{1}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{x'=1}^{X} \sum_{i'=1}^{M} \left[-A\delta_{ii'} \left(1 - \delta_{xx'} \right) - B\delta_{xx'} + C\delta_{xx'} \delta_{ii'} - Dcm_{xx'} d_{ii'} \right] V_{xi} V_{x'i'} -$$

$$- \sum_{x=1}^{X} \sum_{i=1}^{M} \left(B - \frac{C}{2} \right) V_{x\,i} + \frac{X\,B}{2} \tag{25}$$

Przyrównując wzór 25 do wzoru 20 można otrzymać wzory określające wagi połączeń między neuronami oraz zewnętrzne sygnały wejściowe neuronów

$$\omega_{x\,i,\,x'i'} = -A\,\delta_{i\,i'}\left(1-\delta_{x\,x'}\right) - B\,\delta_{x\,x'} + C\,\delta_{x\,x'}\delta_{i\,i'} - D\,cm_{x\,x'}\,d_{i\,i'} \tag{26}$$

$$I_{x\,i} = B - \frac{C}{2} \tag{27}$$

gdzie: $\omega_{x\,i,\,x'\,i'}$ - waga połączenia między neuronami o współrzędnych *(x, i)* oraz *(x ', i ')*, $I_{x\,i}$ - zewnętrzny sygnał wejściowy neuronu *(x, i)*.

Sieć neuronowa z wagami i zewnętrznymi sygnałami wejściowymi określonymi według wzorów 26 i 27 będzie minimalizować funkcję kosztu, czyli całkowitą ważoną długość połączeń w układzie, z uwzględnieniem ograniczeń [126].

4.4. Analog elektryczny sieci Hopfielda

Jak już wcześniej wspomniano, model ciągły sieci Hopfielda (z ciągłą charakterystyką neuronów) lepiej nadaje się do rozwiązywania problemów optymalizacyjnych. Rysunek 17 przedstawia schemat układu elektronicznego, realizującego model ciągły sieci. Rolę neuronu pełni wzmacniacz operacyjny o nieliniowej charakterystyce. Rezystory $R_{i\,j}$ decydują o wartościach wag połączeń między neuronami. Rezystancja R_i oraz pojemność C_i odpowiada rezystancji oraz pojemności wejściowej membrany *i*-tego neuronu [93-94, 107, 109]. Wejściowy i wyjściowy sygnał neuronu odpowiada napięciu na wejściu oraz na wyjściu wzmacniacza operacyjnego. Zewnętrzny sygnał wejściowy *i*-tego neuronu jest określony przez wartość prądu dostarczanego przez źródło prądu I_i.

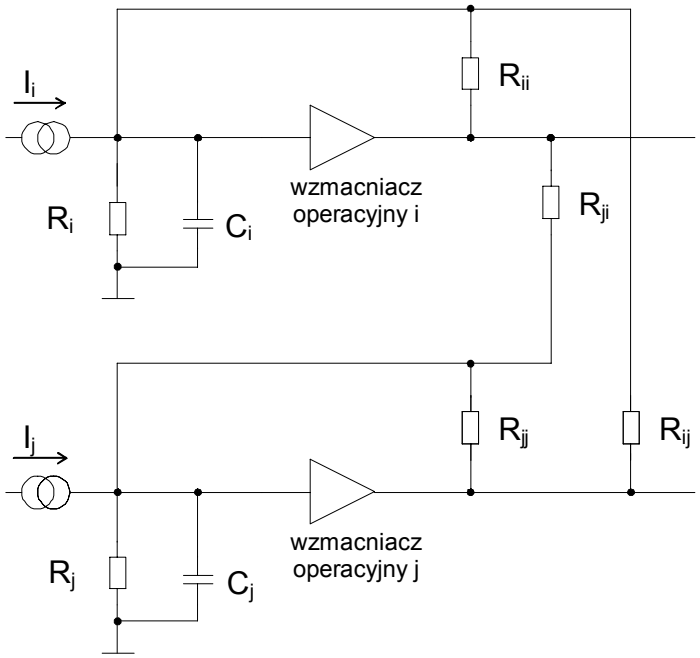
Dla układu z rysunku 17, który jest elektrycznym analogiem sieci Hopfielda, można zapisać równanie Kirchhoffa dla *i*-tego węzła

$$C_i\frac{dU_i}{dt} = \sum_{j=1}^{N}\frac{1}{R_{i\,j}}\left(V_j - U_i\right) + I_i - \frac{U_i}{R_i} =$$

$$= -U_i\left(\frac{1}{R_i} + \sum_{j=1}^{N}\frac{1}{R_{i\,j}}\right) + \sum_{j=1}^{N}\frac{1}{R_{i\,j}}V_j + I_i \tag{28}$$

Można wprowadzić rezystancję zastępczą r_i taką, że

$$\frac{1}{r_i} = \frac{1}{R_i} + \sum_{j=1}^{N}\frac{1}{R_{i\,j}} \tag{29}$$



Rys. 17. Schemat układu elektronicznego realizującego model ciągły sieci Hopfielda

Równanie sieci otrzymuje więc postać

$$C_i \frac{dU_i}{dt} = \sum_{j=1}^{N} \frac{1}{R_{ij}} V_j + I_i - \frac{U_i}{r_i} = \sum_{j=1}^{N} \omega_{ij} V_j + I_i - \frac{U_i}{r_i} \tag{30}$$

gdzie: $\omega_{ij} = 1/R_{ij}$ jest wagą połączenia wyjścia neuronu *j* z wejściem neuronu *i*.

Dobierając pojemności kondensatorów oraz rezystancje

$$C_i = C_j = C \qquad \text{dla } i, j = 1,...,N \tag{31}$$

$$r_i = r_j = R \qquad \text{dla } i, j = 1,...,N \tag{32}$$

waga ω_{ij} oraz zewnętrzny sygnał wejściowy I_i neuronu zawierają arbitralnie dobrane stałe, można więc podzielić równanie 30 przez *C* i po przedefiniowaniu $\omega_{ij} := \omega_{ij}/C$, $I_i := I_i/C$, można zapisać równanie 30 w postaci [92-94, 107]

$$\frac{dU_i}{dt} = \sum_{j=1}^{N} \omega_{ij} V_j + I_i - \frac{U_i}{RC} \tag{33}$$

gdzie: *R C* jest stałą czasową sieci, która jest oznaczana przez τ.

W rezultacie otrzymano układ równań różniczkowych, który można rozwiązać numerycznie stosując metodę Eulera. Układ równań różniczkowych jest zastępowany wówczas układem równań różnicowych [92-93]

$$U_i^{t+\Delta t} = U_i^t + \Delta t \left(\left(\sum_{j=1}^{N} \omega_{ij} V_j \right) + I_i - \frac{U_i^t}{\tau} \right) \tag{34}$$

gdzie: Δt - krok czasowy metody, U_i^t - sygnał wejściowy neuronu i w chwili t.

W przypadku, kiedy jest wymagana bardzo dokładna analiza zachowania sieci, można posłużyć się dostępnymi narzędziami do analizy układów elektronicznych np. SPICE, SMASH itp.

Rozdział 5
Minimalizacja długości połączeń z wykorzystaniem zmodyfikowanej sieci Hopfielda

W niniejszym rozdziale oraz następnych rozdziałach pracy zostaną przedstawione oryginalne modyfikacje klasycznego modelu sieci Hopfielda. W niniejszym rozdziale przedstawiono wpływ modyfikacji sieci Hopfielda na otrzymywane rozwiązania na przykładzie 7 układów. We wszystkich układach sieć Hopfielda wykonywała rozmieszczenie modułów z minimalizacją całkowitej długości połączeń. W podrozdziale 5.1 przedstawiono oryginalne modyfikacje klasycznego modelu sieci Hopfielda, które umożliwiają poprawę skuteczności sieci. Podrozdział 5.2 zawiera wyniki otrzymane przy pomocy klasycznej i zmodyfikowanej sieci Hopfielda. Podsumowanie wyników przeprowadzonych symulacji zamieszczono w podrozdziale 5.3.

5.1. Modyfikacje klasycznego modelu sieci

Podczas rozwiązywania problemów optymalizacyjnych z użyciem sieci Hopfielda zwykle pojawia się problem, polegający na osiąganiu przez sieć minimów lokalnych funkcji kosztu. Zjawisko to powoduje pogorszenie skuteczności metody. Rozwiązaniem w ramach klasycznego modelu sieci jest wykonywanie wielu prób z inicjalizacją sieci losowymi wartościami. Wielokrotne próby uwzględniające różne punkty startowe zwiększają prawdopodobieństwo znalezienia optymalnego rozwiązania. Klasyczna sieć Hopfielda umożliwia otrzymanie rozwiązań, które są znacznie gorsze w porównaniu z rozwiązaniami otrzymanymi innymi metodami. Ta obserwacja stała się inspiracją dla usprawnienia tejże sieci w możliwie prosty sposób.

Podczas rozwiązywania problemów optymalizacyjnych z wykorzystaniem sieci Hopfielda powszechnie jest stosowane rozwiązanie, w którym stała C we wzorach 26 i 27 posiada wartość porównywalną z wartościami stałych A i B lub stała C jest równa stałej B. Duże wartości stałej C stosowano, aby wartości sygnałów wyjściowych wszystkich neuronów w stanie stabilnym były równe 0 lub 1 lub były bardzo bliskie tym wartościom. Uzasadnieniem dla wyboru wartości stałej C równej wartości stałej B był warunek eliminacji sprzężeń zwrotnych z wyjścia danego neuronu na jego wejście. Model ciągły sieci Hopfielda w odróżnieniu od modelu dyskretnego, dopuszcza sprzężenia zwrotne neuronów, tzn. wartości ω_{ij} obliczone dla neuronów sieci zgodnie ze wzorem 26 mogą być różne od 0 [92]. Jednak stosowano zbyt duże wartości stałej C w porównaniu do wartości stałych A i B. Oryginalne zastosowanie

funkcji kosztu z niewielką wartością stałej *C*, w porównaniu do wartości stałych *A* i *B*, umożliwiło radykalną poprawę jakości otrzymywanych rozwiązań. W przypadku problemów o wymiarze równym 10, sieć umożliwiała z bardzo dużym prawdopodobieństwem otrzymanie optymalnego rozwiązania w sensie globalnym. Zalety powyższej oryginalnej modyfikacji funkcji kosztu były szczególnie widoczne dla problemu komiwojażera, ze względu na bardzo liczną literaturę dla tego problemu. Postacie funkcji kosztu w problemie minimalizacji długości połączeń oraz problemie komiwojażera są bardzo zbliżone. Zastosowanie oryginalnej modyfikacji funkcji kosztu dla problemu komiwojażera sprawiło, że dla problemów o wymiarze równym 10, sieć umożliwiała otrzymanie optymalnych rozwiązań w sensie globalnym dla prawie 100% prób. Takich rezultatów nie osiągnięto w żadnej z licznych prac dotyczących tego problemu. Rozwiązania otrzymywane dla problemów o większym wymiarze były również znacznie lepsze, w porównaniu do opublikowanych rezultatów.

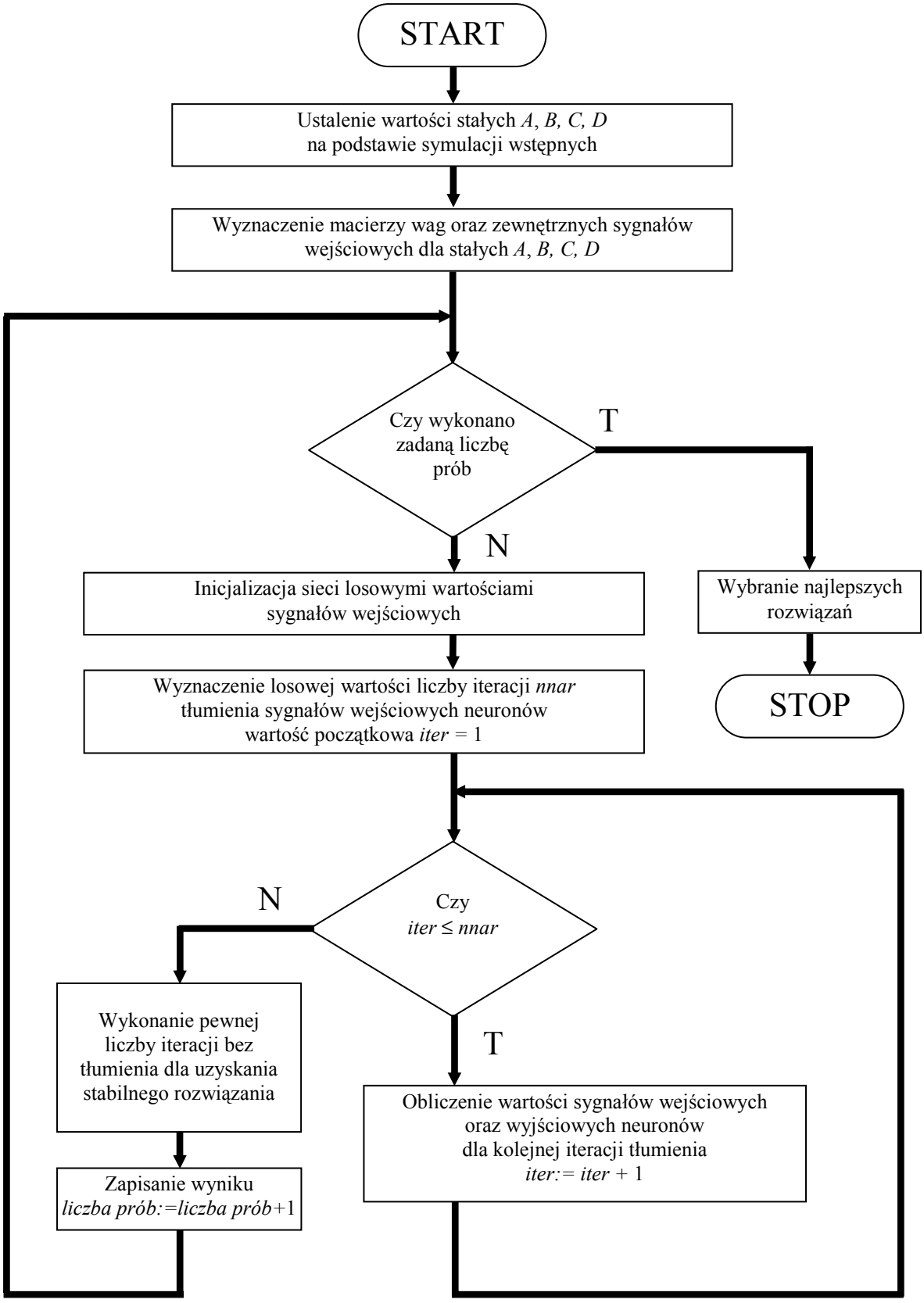
W sieci Hopfielda wprowadzono również oryginalne rozwiązanie, polegające na modyfikacji wartości sygnałów wejściowych neuronów podczas symulacji. Metoda polega na wykonaniu zadanej liczby prób, podczas których jest dokonywana modyfikacja sygnałów wejściowych. Przed rozpoczęciem symulacji ustala się wartości stałych *A*, *B*, *C*, *D* we wzorach 26 i 27 na podstawie symulacji wstępnych. Na początku każdej próby sieć jest inicjowana losowymi wartościami sygnałów wejściowych neuronów. Następnie, w sposób losowy jest wybierana liczba iteracji *nnar*, podczas wykonywania tych iteracji następuje modyfikacja sygnałów wejściowych neuronów, polegająca na tłumieniu sygnałów dochodzących do wejść poszczególnych neuronów. Liczba iteracji *nnar* jest wybierana losowo, tak, aby czas trwania tych iteracji był kilkadziesiąt razy większy od stałej czasowej sieci τ. Wartości zewnętrznych sygnałów wejściowych neuronów nie są poddawane żadnym zmianom. Należy przyjąć taki sposób modyfikacji sygnałów wejściowych, aby po wykonaniu *nnar* iteracji nie występowało już tłumienie tych sygnałów. Na końcu każdej próby jest wykonywana pewna liczba iteracji, ale już bez modyfikacji sygnałów wejściowych neuronów, w celu uzyskania stabilnego rozwiązania. Otrzymane rozwiązanie jest następnie zapisywane. Po wykonaniu zadanej liczby prób są wybierane najlepsze rozwiązania. Rysunek 18 przedstawia algorytm postępowania.

W niniejszej pracy zbadano jeden ze sposobów modyfikacji wartości sygnałów wejściowych neuronów, taki sam dla wszystkich sygnałów, polegający na liniowej zmianie tłumienia sygnałów wejściowych w kolejnych iteracjach. Wzór 34 przyjmuje dla tego przypadku postać

$$U_i^{t+\Delta t} = U_i^t + \Delta t \left(\left(\sum_{j=1}^{N} \omega_{ij} \frac{iter}{nnar} V_j \right) + I_i - \frac{U_i^t}{\tau} \right) =$$

$$= U_i^t + \Delta t \left(\frac{iter}{nnar} \left(\sum_{j=1}^{N} \omega_{ij} V_j \right) + I_i - \frac{U_i^t}{\tau} \right) \tag{35}$$

gdzie: *iter* jest numerem kolejnej iteracji, *iter* = 1,...,*nnar*.



Rys. 18. Algorytm metody z modyfikacją sygnałów wejściowych

W następnym podrozdziale zostaną przedstawione wyniki otrzymane z użyciem klasycznej i zmodyfikowanej sieci Hopfielda.

5.2. Wyniki otrzymane przy pomocy klasycznej i zmodyfikowanej sieci Hopfielda

Celem przeprowadzonych badań było zbadanie skuteczności zastosowania sieci Hopfielda do minimalizacji całkowitej długości połączeń. Badania przeprowadzono dla 7 przykładów układów. Tabele 1-5 przedstawiają macierze połączeń dla przykładów 1-5. W przykładzie 1 rozmieszczano 8 modułów, natomiast w przykładach 2-4 10 modułów. Przykład 5 polega na rozmieszczeniu 16 modułów [112]. W przykładzie 6 rozmieszczano 25 modułów, których połączenia dla optymalnego rozmieszczenia modułów tworzą regularną kwadratową siatkę (ang. grid). Rysunek 19 przedstawia połączenia między modułami w przykładzie 6 dla optymalnego rozmieszczenia modułów. Przykład 7 składa się z 40 modułów.

Nr m./ nr m.	1	2	3	4	5	6	7	8
1	0	0	1	0	0	0	0	1
2	0	0	0	1	1	0	1	0
3	1	0	0	0	1	0	0	0
4	0	1	0	0	1	1	0	0
5	0	1	1	1	0	0	0	0
6	0	0	0	1	0	0	0	0
7	0	1	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0

Tabela 1. Macierz połączeń dla przykładu 1

Nr m./ nr m.	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	1	1	1
3	1	1	0	1	0	0	0	0	0	0
4	0	0	1	0	1	1	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0
6	0	0	0	1	1	0	1	0	0	0
7	0	0	0	0	0	1	0	0	0	0
8	0	1	0	0	0	0	0	0	1	1
9	0	1	0	0	0	0	0	1	0	1
10	0	1	0	0	0	0	0	1	1	0

Tabela 2. Macierz połączeń dla przykładu 2

Nr m./ nr m.	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	1	0	1	0	0	0
2	1	0	0	1	1	0	0	1	0	0
3	1	0	0	1	0	1	1	0	1	0
4	0	1	1	0	0	1	0	1	0	1
5	1	1	0	0	0	1	0	0	0	0
6	0	0	1	1	1	0	0	0	0	0
7	1	0	1	0	0	0	0	1	1	1
8	0	1	0	1	0	0	1	0	1	1
9	0	0	1	0	0	0	1	1	0	1
10	0	0	0	1	0	0	1	1	1	0

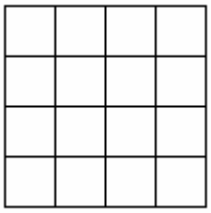
Tabela 3. Macierz połączeń dla przykładu 3

Nr m./ nr m.	1	2	3	4	5	6	7	8	9	10
1	0	0	0	1	1	1	1	1	0	1
2	0	0	1	0	0	1	0	0	1	1
3	0	1	0	1	1	1	1	0	1	0
4	1	0	1	0	1	0	1	1	0	1
5	1	0	1	1	0	1	1	1	0	0
6	1	1	1	0	1	0	0	1	1	0
7	1	0	1	1	1	0	0	1	0	1
8	1	0	0	1	1	1	1	0	0	1
9	0	1	1	0	0	1	0	0	0	1
10	1	1	0	1	0	0	1	1	1	0

Tabela 4. Macierz połączeń dla przykładu 4

Nr m./ nr m.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1
2	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0
3	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
4	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0
5	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
6	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0
9	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0
10	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0
11	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
12	1	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0
14	0	0	0	0	0	0	0	1	0	0	0	0	2	0	1	0
15	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1
16	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0

Tabela 5. Macierz połączeń dla przykładu 5



Rys. 19. Połączenia między modułami w przykładzie 6 dla optymalnego rozmieszczenia modułów

W przykładzie 5 dodatnie wartości elementów macierzy połączeń oznaczają wagę połączeń. Niektóre połączenia posiadają wagę 2, w odróżnieniu od pozostałych połączeń, posiadających wagę 1.

W przykładzie 1 moduły były rozmieszczane na podłożu o rozmiarach 3x3 części. W przykładach 2-4 moduły były rozmieszczane na podłożu o rozmiarach 4x3 części. Przykład 5 wymagał podłoża 4x4 części, przykład 6 podłoża 5x5 części, natomiast przykład 7 podłoża o rozmiarach 7x7 części. W dalszej części pracy, całkowita ważona długość połączeń dla danego rozmieszczenia modułów będzie określana jako koszt rozwiązania. Koszt optymalnego rozmieszczenia modułów (w sensie globalnym) w przykładach 1-4 został wyznaczony w wyniku wyszukiwania wyczerpującego, dla danej wielkości podłoża. Koszt optymalnego rozwiązania dla przykładów 1-4 wynosi odpowiednio: 9, 18, 29 i 48 jednostek. Koszt optymalnego rozwiązania dla przykładu 5 nie może być wyznaczony metodą wyszukiwania wyczerpującego,

ze względu na zbyt długi czas obliczeń. Znane są jednak wyniki zastosowania różnych metod rozmieszczania dla tego przykładu. Rozwiązanie otrzymane z użyciem metody wykorzystującej programowanie kwadratowe posiada koszt równy 62. W przypadku zastosowania metody wykorzystującej logikę rozmytą, koszt rozwiązania wynosi 56. Sieć samoorganizująca się umożliwiła otrzymanie rozwiązania o koszcie równym 54 [112]. Koszt optymalnego rozwiązania dla przykładu 6 jest znany ze względu na topologię połączeń między modułami i wynosi 40.

Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda z oryginalną modyfikacją funkcji kosztu oraz z użyciem sieci z oryginalną modyfikacją sygnałów wejściowych, a uzyskane wyniki zostały ze sobą porównane. Obydwa rodzaje sieci zostały zaimplementowane programowo, w języku C++. Symulacje przeprowadzono dla sieci o stałej czasowej $\tau = 1$, z zastosowaniem metody Eulera według wzoru 34 dla klasycznej sieci oraz według wzoru 35 dla sieci z modyfikacją sygnałów wejściowych. Krok czasowy w metodzie Eulera wynosił $\Delta t = 0{,}01\,\tau$, natomiast funkcja aktywacji neuronów była określona wzorem 19, dla $U_0 = 0{,}1$. Liczba iteracji *nnar* w metodzie z modyfikacją sygnałów wejściowych była losowana z przedziału 2000÷5000. Dla innych kroków czasowych Δt należy odpowiednio zmienić przedział, z którego jest losowana wartość *nnar*. W programie symulującym obydwa rodzaje sieci zastosowano synchroniczną aktualizację, tzn. w danej chwili czasowej równocześnie są obliczane sygnały wejściowe oraz wyjściowe wszystkich neuronów. Dla obydwu metod zastosowano takie same kryterium zakończenia danej próby. Otrzymane rozwiązanie jest uznawane za stabilny stan sieci, jeżeli dla wszystkich neuronów bezwzględna wartość zmiany sygnału wyjściowego neuronu jest w danej iteracji mniejsza lub równa 10^{-6}. Zastosowanie mniejszej wartości niż 10^{-6} nie ma wpływu na otrzymywane rozwiązania, natomiast powoduje wzrost liczby iteracji niezbędnych do zakończenia próby [130]. Rozwiązanie jest wyznaczane na podstawie stanu stabilnego sieci. Jeżeli sygnał wyjściowy neuronu jest mniejszy od 0,5 - zakłada się, że sygnał wyjściowy neuronu jest równy 0. Jeżeli sygnał wyjściowy neuronu jest większy lub równy 0,5 - zakłada się, że sygnał wyjściowy neuronu jest równy 1 [130-131]. Rozwiązania, które nie spełniają ograniczeń 1 i 2 z podrozdziału 4.3 są uznawane za rozwiązania nieprawidłowe i odrzucane. Dla obydwu metod zbadano również dwa sposoby inicjalizacji neuronów różnymi, losowymi wartościami początkowymi sygnałów wejściowych z przedziału

a. $-0{,}3 \le U_{xi} \le 0{,}3$ dla $x = 1,...,X$; $i = 1,...,M$ (36)

b. $-\frac{1}{10}U_0 \le U_{xi} \le \frac{1}{10}U_0$ dla $x = 1,...,X$; $i = 1,...,M$ (37)

W pierwszym sposobie inicjalizacji neuronów, wartości początkowe sygnałów wejściowych są losowane z równomiernym rozkładem prawdopodobieństwa z przedziału -0,3 do 0,3 - ponieważ dla funkcji aktywacji określonej wzorem 19 i dla $U_0 = 0{,}1$, wartości sygnałów wyjściowych dla -0,3 oraz 0,3 są bliskie 0 oraz 1. Drugi sposób polega na inicjalizacji neuronów wartościami sygnałów wejściowych bliskimi 0, ponieważ wartości sygnałów wyjściowych będą wówczas bliskie 0,5 i żaden neuron nie będzie uprzywilejowany. Taki sposób

inicjalizacji jest uważany za najbardziej korzystny [92, 132]. Losowe zaburzenie z przedziału $\pm(1/10)U_0$, przyjęto podobnie jak w pracach [92, 107, 130].

Wyniki otrzymane podczas symulacji przedstawia tabela 6. Dla każdego przykładu przeprowadzono 100 prób z użyciem klasycznej sieci Hopfielda z oryginalną modyfikacją funkcji kosztu oraz z użyciem sieci z oryginalną modyfikacją sygnałów wejściowych. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 27: $A = 5$, $B = 6$, $C = 0{,}5$. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100% (tj. 90%÷100%) i było możliwe porównanie skuteczności poszczególnych metod. W tabeli 6 są zebrane następujące wyniki: średni koszt rozwiązań, koszt najlepszego otrzymanego rozwiązania, koszt optymalnego rozwiązania, stosunek średniego kosztu do kosztu optymalnego rozwiązania, prawdopodobieństwo otrzymania optymalnego rozwiązania w jednej próbie oraz wartość stałej D.

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	11,1398	10,0556	9,5102	9,6703
	koszt najlepszego otrzymanego rozw.	9	9	9	9
	koszt optymalnego rozwiązania	9	9	9	9
	śr. koszt / koszt opt. rozw.	1,2378	1,1173	1,0567	1,0745
	prawd. opt. rozw.	9%	6%	48%	30%
	wartość stałej D	0,65	0,8	1,1	1,1
2	średni koszt	20,8778	19,3778	18,7041	20,1075
	koszt najlepszego otrzymanego rozw.	18	18	18	18
	koszt optymalnego rozwiązania	18	18	18	18
	śr. koszt / koszt opt. rozw.	1,1599	1,0765	1,0391	1,1171
	prawd. opt. rozw.	9%	27%	44%	25%
	wartość stałej D	0,35	0,45	0,45	0,35
3	średni koszt	37,9694	34,5052	34,3511	36,4845
	koszt najlepszego otrzymanego rozw.	31	31	30	30
	koszt optymalnego rozwiązania	29	29	29	29
	śr. koszt / koszt opt. rozw.	1,3093	1,1898	1,1845	1,2581
	prawd. opt. rozw.	0%	0%	0%	0%
	wartość stałej D	0,3	0,35	0,35	0,35

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
4	średni koszt	53,7895	51,8788	51,68	52,9333
	koszt najlepszego otrzymanego rozw.	48	48	48	48
	koszt optymalnego rozwiązania	48	48	48	48
	śr. koszt / koszt opt. rozw.	1,1206	1,0808	1,0767	1,1028
	prawd. opt. rozw.	2%	2%	9%	3%
	wartość stałej *D*	0,25	0,25	0,2	0,25
5	średni koszt	74,866	62,4082	58,9574	56,2391
	koszt najlepszego otrzymanego rozw.	58	54	50	48
	wartość stałej *D*	0,25	0,25	0,25	0,25
6	średni koszt	111,3913	68,9375	63,9111	51,75
	koszt najlepszego otrzymanego rozw.	93	44	40	40
	koszt optymalnego rozwiązania	40	40	40	40
	śr. koszt / koszt opt. rozw.	2,7848	1,7234	1,5978	1,2938
	prawd. opt. rozw.	0%	0%	4%	26%
	wartość stałej *D*	0,2	0,25	0,2	0,25
7	średni koszt	156,8316	140,4388	129,3838	129,5567
	koszt najlepszego otrzymanego rozw.	135	118	115	120
	wartość stałej *D*	0,15	0,15	0,15	0,15

Tabela 6. Wyniki otrzymane dla poszczególnych przykładów z użyciem klasycznej i zmodyfikowanej sieci Hopfielda

Podczas analizy wyników zawartych w tabeli 6 można zauważyć, że wartość stałej *D* zależy od rozmiarów podłoża, na którym rozmieszczano moduły. Wzrost rozmiarów podłoża powoduje konieczność zmniejszenia stałej *D*, ze względu na większy udział długości połączeń w funkcji kosztu. Dla przykładów 1-4 oraz 6 są znane optymalne rozwiązania. Podczas analizy wyników zawartych w tabeli 6 można zauważyć, że prawdopodobieństwo osiągnięcia optymalnego rozwiązania jest większe dla metody z modyfikacją sygnałów wejściowych, w porównaniu do klasycznej sieci Hopfielda. Przewaga sieci z modyfikacją sygnałów wejściowych jest bardziej widoczna w przypadku inicjalizacji neuronów wartościami z przedziału -0,3 do 0,3. W przypadku przykładów 5 i 7 koszt najlepszego rozwiązania otrzymanego z użyciem sieci z modyfikacją sygnałów wejściowych jest mniejszy, w porównaniu do klasycznej sieci. Dla przykładu 5 sieć z modyfikacją sygnałów wejściowych dostarczyła

rozwiązań, które są znacznie lepsze od rozwiązań znanych z literatury. Koszt najlepszego rozwiązania otrzymanego w innych pracach jest równy 54 [112], natomiast sieć z modyfikacją sygnałów wejściowych dostarczyła rozwiązań o koszcie równym 50 i 48, w zależności od sposobu inicjalizacji neuronów. Można również zauważyć, że średni koszt rozwiązań dla przykładów 5-7, które posiadają większy wymiar, jest wyraźnie mniejszy dla sieci z modyfikacją sygnałów wejściowych, w porównaniu do klasycznej sieci. Na podstawie otrzymanych rozwiązań można również zauważyć, że dla inicjalizacji neuronów wartościami z przedziału -0,3 do 0,3, wyniki otrzymane klasyczną siecią są znacznie gorsze w porównaniu do sieci z modyfikacją sygnałów wejściowych. Powyższa różnica staje się bardziej widoczna wraz ze wzrostem wymiaru rozpatrywanego przykładu. Klasyczna sieć Hopfielda wymaga inicjalizacji neuronów wartościami sygnałów wejściowych bliskimi 0 (sygnały wyjściowe neuronów bliskie 0,5) tak, aby żaden neuron nie był uprzywilejowany.

Dla przykładów 1-7 wykonano również symulacje z użyciem klasycznej sieci Hopfielda oraz klasycznej postaci funkcji kosztu, w której wartość stałej C jest porównywalna z wartością stałej B. Wyniki otrzymane podczas symulacji przedstawia tabela 7. Dla każdego przykładu przeprowadzono 100 prób. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 27: $A = 5$, $B = 6$, $C = 6$. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%.

Nr przykł.	Wynik	Klasyczna metoda	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	14	12,0435
	koszt najlepszego otrzymanego rozw.	11	9
	koszt optymalnego rozwiązania	9	9
	śr. koszt / koszt opt. rozw.	1,5556	1,3382
	prawd. opt. rozw.	0%	3%
	wartość stałej D	0,4	0,45
2	średni koszt	29,1111	25,2245
	koszt najlepszego otrzymanego rozw.	21	20
	koszt optymalnego rozwiązania	18	18
	śr. koszt / koszt opt. rozw.	1,6173	1,4014
	prawd. opt. rozw.	0%	0%
	wartość stałej D	0,2	0,25
3	średni koszt	43,54	40,38
	koszt najlepszego otrzymanego rozw.	35	34
	koszt optymalnego rozwiązania	29	29
	śr. koszt / koszt opt. rozw.	1,5014	1,3924
	prawd. opt. rozw.	0%	0%
	wartość stałej D	0,2	0,2

Nr przykł.	Wynik	Klasyczna metoda	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
4	średni koszt	60,8163	56,60
	koszt najlepszego otrzymanego rozw.	52	50
	koszt optymalnego rozwiązania	48	48
	śr. koszt / koszt opt. rozw.	1,267	1,1792
	prawd. opt. rozw.	0%	0%
	wartość stałej *D*	0,15	0,15
5	średni koszt	79,7778	74,16
	koszt najlepszego otrzymanego rozw.	66	62
	wartość stałej *D*	0,15	0,15
6	średni koszt	122,82	89,0612
	koszt najlepszego otrzymanego rozw.	103	68
	koszt optymalnego rozwiązania	40	40
	śr. koszt / koszt opt. rozw.	3,0705	2,2265
	prawd. opt. rozw.	0%	0%
	wartość stałej *D*	0,1	0,15
7	średni koszt	238,9149	193,58
	koszt najlepszego otrzymanego rozw.	202	167
	wartość stałej *D*	0,1	0,1

Tabela 7. Wyniki otrzymane dla poszczególnych przykładów z użyciem klasycznej sieci Hopfielda dla stałej $C = B$

Wyniki symulacji przedstawione w tabeli 7 jednoznacznie wskazują, że wartość stałej *C* porównywalna z wartością stałej *B* powoduje znaczne pogorszenie jakości otrzymywanych rozwiązań. Średni koszt otrzymanych rozwiązań jest znacznie większy w porównaniu do rozwiązań otrzymanych dla małej wartości stałej *C*. W przypadku dużej wartości stałej C nie jest możliwe również otrzymanie optymalnego rozwiązania lub prawdopodobieństwo jego otrzymania w jednej próbie jest znacznie mniejsze.

5.3. Podsumowanie

Przeprowadzone symulacje wskazują na możliwość poprawy jakości rozwiązań otrzymywanych przez sieć Hopfielda dla problemu minimalizacji długości połączeń. Poprawa jakości rozwiązań jest możliwa w wyniku zastosowania modyfikacji funkcji kosztu oraz sieci z modyfikacją sygnałów wejściowych. Model ciągły sieci Hopfielda dopuszcza sprzężenia zwrotne neuronów z wyjścia neuronu na jego wejście. Stosowanie zbyt dużych wartości stałej *C*, w porównaniu do wartości stałych *A* i *B* we wzorach 26 i 27, powoduje bardzo duże pogorszenie jakości otrzymywanych rozwiązań. Zbyt duża wartość

stałej *C* powoduje, że sieć bardzo szybko osiąga minimum lokalne funkcji kosztu, które nie może później opuścić. Zastosowanie niewielkiej wartości stałej *C*, w porównaniu do wartości stałych *A* i *B*, umożliwiło radykalną poprawę jakości otrzymywanych rozwiązań. Rozwiązania otrzymywane siecią z modyfikacją sygnałów wejściowych są lepsze, w porównaniu do klasycznej sieci Hopfielda, szczególnie dla układów o większym wymiarze. W przypadku klasycznej sieci Hopfielda najkorzystniejszym sposobem inicjalizacji sieci jest inicjalizacja neuronów wartościami sygnałów wejściowych bliskimi 0, ponieważ żaden neuron nie jest wtedy uprzywilejowany. W przypadku inicjalizacji neuronów wartościami sygnałów wejściowych z przedziału -0,3 do 0,3, niezbędne jest zastosowanie sieci z modyfikacją sygnałów wejściowych, ponieważ klasyczna sieć Hopfielda dostarcza rozwiązań o złej jakości. Lepsze rezultaty uzyskane w wyniku zastosowania metody z modyfikacją sygnałów wejściowych należy tłumaczyć tym, że podczas iteracji *nnar*, w wyniku malejącego tłumienia sygnałów wejściowych neuronów wzrasta energia sieci, określona wzorem 20. Losowe wartości liczby iteracji *nnar* określają również szybkość wzrostu energii sieci podczas tych iteracji. Pozwala to sieci na wyjście z minimów lokalnych funkcji energii dla pewnych wartości liczby *nnar*. Osiągnięcie minimum lokalnego przez klasyczną sieć Hopfielda uniemożliwia opuszczenie tego minimum. Dlatego rozwiązanie nie jest wtedy optymalne w sensie globalnym.

Rozdział 6
Minimalizacja długości połączeń w układach z ustalonym położeniem padów układu

W niniejszym rozdziale przedstawiono oryginalne zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z ustalonym położeniem końcówek całego układu. Oryginalne rozwiązanie umożliwiające zastosowanie sieci Hopfielda w układach, w których położenie końcówek całego układu jest ustalone, przedstawiono w podrozdziale 6.1. Podrozdział 6.2 zawiera rezultaty otrzymane dla układów z ustalonym położeniem końcówek całego układu. W podrozdziale 6.3 przedstawiono wyniki rozmieszczania otrzymane dla przykładów, w których położenie kilku modułów układu jest trwale ustalone. Podrozdział 6.4 zawiera rezultaty równoczesnego rozmieszczania modułów oraz końcówek całego układu. Podsumowanie wyników przeprowadzonych symulacji zamieszczono w podrozdziale 6.5.

6.1. Postać funkcji kosztu dla układów z ustalonym położeniem padów układu

Postać funkcji kosztu problemu minimalizacji długości połączeń przedstawiona w rozdziale 4, nie uwzględnia połączeń między modułami a końcówkami całego układu, które posiadają ustalone położenie w układzie. W powyższych układach składnik funkcji kosztu, określający całkowitą ważoną długość połączeń, można wyrazić następującym wzorem

$$E_4 = \frac{D}{2}\sum_{x=1}^{X}\sum_{i=1}^{M}\sum_{x'=1}^{X}\sum_{i'=1}^{M} V_{x\,i} V_{x'\,i'} cm_{x\,x'} d_{i\,i'} + D\sum_{x=1}^{X}\sum_{i=1}^{M}\sum_{p=1}^{P} V_{x\,i}\, cmp_{x\,p}\, d_{i\,p} \quad (38)$$

gdzie: P - liczba końcówek całego układu, $d_{i\,i'}$ - odległość między częściami podłoża i oraz i', $cm_{x\,x'}$ - element macierzy połączeń **CM** układu, określający wagę połączenia między modułami x oraz x', $d_{i\,p}$ - odległość między częścią podłoża i oraz końcówką całego układu p, $cmp_{x\,p}$ - element macierzy połączeń **CMP** układu, określający wagę połączenia między modułem x a końcówką całego układu p.

Elementy macierzy **CMP** są określone następująco

$$\begin{cases} cmp_{x\,p} > 0 & \text{gdy jest połączenie między modułem } x \text{ oraz końcówką } p \\ cmp_{x\,p} = 0 & \text{w przeciwnym przypadku} \end{cases} \tag{39}$$

Powyższe rozwiązanie powoduje zmianę wzoru 27, wyznaczającego wartości zewnętrznych sygnałów wejściowych neuronów, które obecnie są określone następująco

$$I_{x\,i} = B - \frac{C}{2} - D \sum_{p=1}^{P} cmp_{x\,p}\, d_{i\,p} \tag{40}$$

Sieć neuronowa z wagami i zewnętrznymi sygnałami wejściowymi neuronów, określonymi według wzorów 26 i 40, umożliwia minimalizację długości połączeń w układach z ustalonym położeniem końcówek całego układu.

6.2. Wyniki otrzymane dla układów z ustalonym położeniem padów układu

Minimalizację długości połączeń dla układów z ustalonym położeniem końcówek całego układu przeprowadzono dla wszystkich przykładów z poprzedniego rozdziału. W tym celu wprowadzono 12 końcówek całego układu w przykładzie 1, 14 końcówek w przykładach 2-4, 16 końcówek w przykładzie 5, 20 końcówek w przykładzie 6 oraz 4 końcówki w przykładzie 7. Połączenia między modułami nie uległy żadnym zmianom. Tabele 8-10 przedstawiają macierze połączeń między modułami i końcówkami całego układu dla przykładów 1-5. Rysunek 20 przedstawia połączenia między modułami i końcówkami całego układu w przykładzie 6, dla optymalnego rozmieszczenia modułów.

Nr k. / Nr m.	k. 1	k. 2	k. 3	k. 4	k. 5	k. 6	k. 7	k. 8	k. 9	k. 10	k. 11	k. 12
1	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0
6	0	0	0	1	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0	0	0	0	1
8	0	1	0	0	0	0	0	0	0	0	0	0

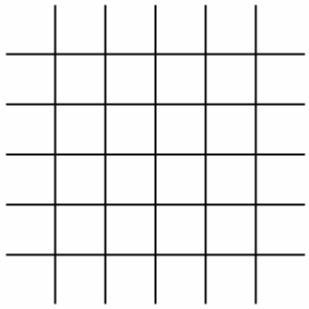
Tabela 8. Macierz połączeń między modułami i końcówkami całego układu dla przykładu 1

Nr k. / Nr m.	k. 1	k. 2	k. 3	k. 4	k. 5	k. 6	k. 7	k. 8	k. 9	k. 10	k. 11	k. 12	k. 13	k. 14
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	1	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	1	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabela 9. Macierz połączeń między modułami i końcówkami całego układu dla przykładów 2-4

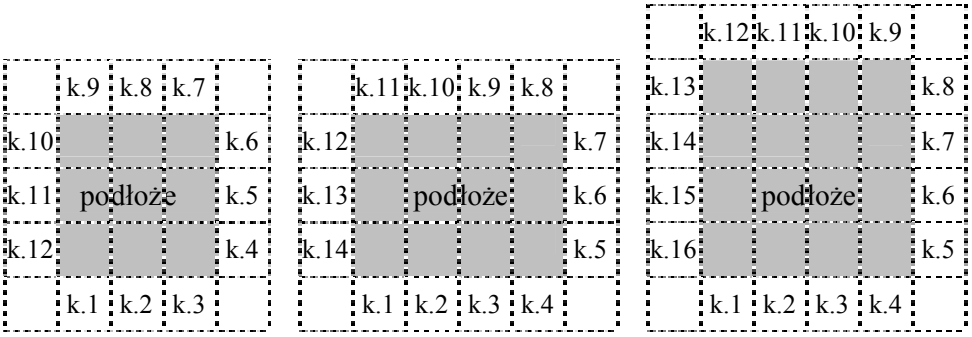
Nr k. / Nr m.	k. 1	k. 2	k. 3	k. 4	k. 5	k. 6	k. 7	k. 8	k. 9	k. 10	k. 11	k. 12	k. 13	k. 14	k. 15	k. 16
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Tabela 10. Macierz połączeń między modułami i końcówkami całego układu dla przykładu 5



Rys. 20. Połączenia między modułami i końcówkami całego układu w przykładzie 6 dla optymalnego rozmieszczenia modułów

We wszystkich przykładach moduły były rozmieszczane na podłożach o takich samych rozmiarach, jak w poprzednim rozdziale. Założono, że końcówki całego układu są położone w środkach geometrycznych jednostkowych części otaczających podłoże układu. Powyższe założenie nie musi być spełnione, położenie końcówek całego układu może być dowolne. Powyższe założenie wprowadzono jedynie po to, aby umożliwić prostą, graficzną ilustrację położenia końcówek. Rysunek 21 przedstawia położenie poszczególnych końcówek w przykładach 1-5. Koszt optymalnego rozmieszczenia modułów w przykładach 1-4 został wyznaczony w wyniku wyszukiwania wyczerpującego dla danej wielkości podłoża. Koszt optymalnego rozwiązania dla przykładów 1-4 wynosi odpowiednio: 36, 49, 67 i 81 jednostek. Koszt optymalnego rozwiązania dla przykładu 6 jest znany ze względu na topologię połączeń między modułami i wynosi 60.



Rys. 21. Położenie końcówek całego układu: a) w przykładzie 1, b) w przykładach 2-4, c) w przykładzie 5

Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda z oryginalną modyfikacją funkcji kosztu oraz z użyciem sieci z oryginalną modyfikacją sygnałów wejściowych, a uzyskane wyniki zostały ze sobą porównane. Dla obydwu metod zbadano dwa sposoby inicjalizacji neuronów. Wyniki otrzymane podczas symulacji przedstawia tabela 11. Dla każdego przykładu przeprowadzono 100 prób. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 40: $A = 5$, $B = 6$,

C = 0,5. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%.

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	41,3111	36	36,93	37
	koszt najlepszego otrzymanego rozw.	36	36	36	37
	koszt optymalnego rozwiązania	36	36	36	36
	śr. koszt / koszt opt. rozw.	1,1475	1	1,0258	1,0278
	prawd. opt. rozw.	4%	100%	7%	0%
	wartość stałej D	0,3	0,35	0,3	0,35
2	średni koszt	55,6489	50,05	49,97	50,02
	koszt najlepszego otrzymanego rozw.	49	50	49	49
	koszt optymalnego rozwiązania	49	49	49	49
	śr. koszt / koszt opt. rozw.	1,1357	1,0214	1,0198	1,0208
	prawd. opt. rozw.	3%	0%	13%	4%
	wartość stałej D	0,2	0,2	0,15	0,15
3	średni koszt	75,1444	68,52	70,53	70,70
	koszt najlepszego otrzymanego rozw.	69	67	68	70
	koszt optymalnego rozwiązania	67	67	67	67
	śr. koszt / koszt opt. rozw.	1,1216	1,0227	1,0527	1,0552
	prawd. opt. rozw.	0%	9%	0%	0%
	wartość stałej D	0,2	0,15	0,2	0,2
4	średni koszt	92,37	83,46	83,7667	83,7789
	koszt najlepszego otrzymanego rozw.	83	81	82	82
	koszt optymalnego rozwiązania	81	81	81	81
	śr. koszt / koszt opt. rozw.	1,1404	1,0304	1,0342	1,0343
	prawd. opt. rozw.	0%	1%	0%	0%
	wartość stałej D	0,15	0,15	0,2	0,2
5	średni koszt	122,62	91,40	88,5979	88,24
	koszt najlepszego otrzymanego rozw.	98	88	88	88
	wartość stałej D	0,15	0,25	0,25	0,2

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
6	średni koszt	133,0444	60	60	60
	koszt najlepszego otrzymanego rozw.	108	60	60	60
	koszt optymalnego rozwiązania	60	60	60	60
	śr. koszt / koszt opt. rozw.	2,2174	1	1	1
	prawd. opt. rozw.	0%	100%	100%	100%
	wartość stałej *D*	0,2	0,2	0,35	0,25
7	średni koszt	263,51	231,11	221,17	219,99
	koszt najlepszego otrzymanego rozw.	235	219	213	212
	wartość stałej *D*	0,15	0,2	0,2	0,2

Tabela 11. Wyniki otrzymane dla poszczególnych przykładów z ustalonym położeniem końcówek całego układu

Dla przykładów 1-4 oraz 6 jest znany koszt optymalnego rozwiązania. Podczas analizy wyników zawartych w tabeli 11 można zauważyć, że wyniki otrzymane klasyczną siecią z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 oraz otrzymane siecią z modyfikacją sygnałów wejściowych dla obydwu sposobów inicjalizacji neuronów, dla przykładów 1-4 są zbliżone. Można również zauważyć, że obecność końcówek całego układu z ustalonym położeniem wpływa na poprawę jakości otrzymywanych rozwiązań. Średni koszt otrzymanych rozwiązań jest dla przykładów 1-4 tylko o kilka procent większy od kosztu optymalnego rozwiązania. Prawdopodobieństwo otrzymania optymalnego rozwiązania w jednej próbie jest wprawdzie mniejsze, w porównaniu do przykładów bez obecności końcówek całego układu z podrozdziału 5.2, ale należy wziąć pod uwagę, że obecność końcówek całego układu spowodowała znaczny wzrost kosztu optymalnego rozwiązania. Wszystkie powyższe metody umożliwiają otrzymanie rozwiązań o koszcie większym od kosztu optymalnego rozwiązania o co najwyżej 1 jednostkę (wyjątkowo o 3 jednostki). Obecnie, należy wziąć pod uwagę, że są możliwe teraz rozwiązania, dla których stosunek kosztu rozwiązania do kosztu optymalnego rozwiązania jest bliższy jedności, w porównaniu do rozwiązań dla przykładów bez końcówek całego układu, których koszt jest większy o 1 jednostkę od kosztu optymalnego rozwiązania. Powyższe spostrzeżenie tłumaczy mniejsze prawdopodobieństwo otrzymania optymalnego rozwiązania w jednej próbie, w porównaniu do przykładów bez końcówek całego układu. Z tego względu mniejsze prawdopodobieństwo otrzymania optymalnego rozwiązania w jednej próbie nie świadczy o pogorszeniu skuteczności sieci Hofielda dla przykładów z ustalonym położeniem końcówek całego układu. Na uwagę zasługuje również bardzo dobra jakość rozwiązań otrzymanych dla przykładu 6, z regularnym sposobem połączeń między modułami. Wszystkie

powyższe metody umożliwiły otrzymanie dla tego przykładu rozwiązań, których średnia jest równa kosztowi optymalnego rozwiązania.

Dla przykładów 5 i 7, o większym wymiarze, jest widoczna przewaga metody z modyfikacją sygnałów wejściowych. Przewaga sieci z modyfikacją sygnałów wejściowych jest widoczna dla średniego kosztu rozwiązań lub dla kosztu najlepszego otrzymanego rozwiązania. Przewaga sieci z modyfikacją sygnałów wejściowych jest również widoczna dla wszystkich przykładów, w przypadku inicjalizacji neuronów wartościami sygnałów wejściowych z przedziału -0,3 do 0,3.

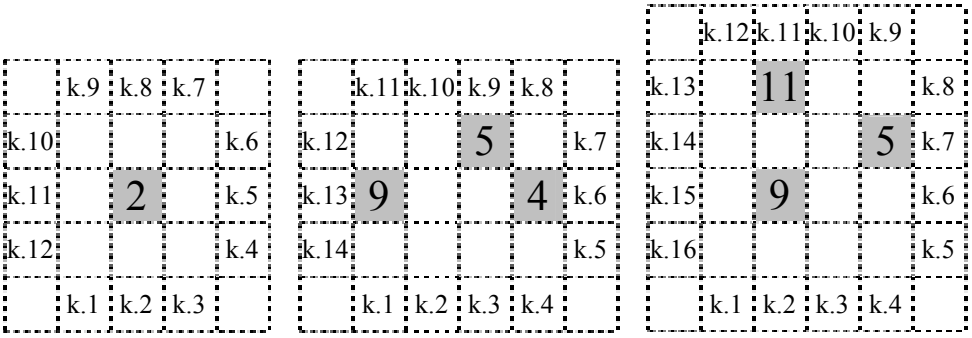
Innym rozwiązaniem, które umożliwiłoby rozmieszczanie modułów w układzie z ustalonym położeniem końcówek całego układu jest zastosowanie sieci rozmieszczającej równocześnie moduły i końcówki całego układu, w której położenie końcówek zostałoby trwale ustalone. Rozwiązanie to powoduje jednak znaczny wzrost liczby neuronów w sieci. Zaletą przedstawionego oryginalnego rozwiązania jest brak wpływu liczby końcówek całego układu na liczbę neuronów w sieci.

6.3. Wyniki otrzymane dla układów z ustalonym położeniem modułów

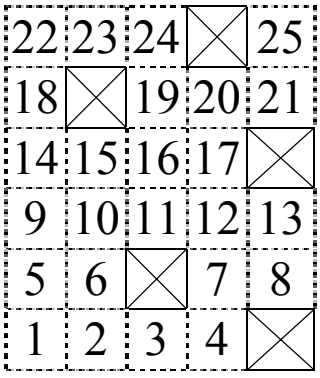
Minimalizację długości połączeń można również przeprowadzić dla układów z ustalonym położeniem kilku modułów. Sytuacja taka często występuje podczas projektowania topografii układów VLSI. W niniejszym podrozdziale przeprowadzono minimalizację długości połączeń dla przykładów 1-5 z poprzedniego podrozdziału, w których zostało trwale ustalone położenie kilku modułów. W tym celu, trwale ustalono położenie 1 modułu w przykładzie 1 oraz 3 modułów w przykładach 2-5. Moduły w przykładach 1-5 były rozmieszczane na podłożach o takich samych rozmiarach, jak w poprzednim podrozdziale. Rysunek 22 przedstawia położenie oraz numery modułów, których położenie w układzie zostało trwale ustalone (moduły oznaczono kolorem szarym). W przykładach 1-4 położenie wybranych modułów zostało trwale ustalone w oparciu o położenie modułów dla optymalnego rozmieszczenia. W związku z tym, koszt optymalnego rozwiązania dla przykładów 1-4 nie uległ zmianie. W przykładzie 5 położenie wybranych modułów zostało trwale ustalone w oparciu o położenie modułów dla rozmieszczenia otrzymanego w podrozdziale 6.2, którego koszt wynosił 88.

Obecność modułów, których położenie w układzie jest ustalone nie wymaga dużych zmian w stosowanej sieci Hopfielda. Liczba neuronów w sieci $N = (X\text{-}X_{ust})\ (K\ L\text{-}X_{ust})$, gdzie X_{ust} jest liczbą modułów, których położenie zostało trwale ustalone. Sieć składa się z $(X\text{-}X_{ust})$ wierszy, zawierających $M = (K\ L\text{-}X_{ust})$ neuronów, zgodnie z rysunkiem 15. Ustalenie położenia kilku modułów powoduje zatem zmniejszenie rozmiarów sieci. Moduły, których położenie zostało trwale ustalone są traktowane jak końcówki całego układu, posiadające ustalone położenie. Wszystkie połączenia obejmujące moduły, których położenie zostało ustalone, muszą zatem znaleźć swoje odzwierciedlenie w macierzy połączeń między modułami i końcówkami całego

układu. Połączenia między modułami, których położenie zostało ustalone, a końcówkami całego układu, nie są brane pod uwagę podczas minimalizacji długości połączeń, ponieważ ich długość nie może ulec zmianie. Części podłoża, w których zostały trwale umieszczone wybrane moduły nie są brane pod uwagę podczas rozmieszczania. Jeden ze sposobów numerowania części podłoża przedstawia rysunek 23. Każda z części podłoża musi posiadać prawidłowo określone współrzędne położenia na podłożu układu. Powyższy sposób numerowania części podłoża umożliwia również rozmieszczanie modułów w układzie, w którym pewne części podłoża nie mogą być wykorzystane do rozmieszczania modułów.



Rys. 22. Ustalone położenie wybranych modułów na podłożu układu: a) w przykładzie 1, b) w przykładach 2-4, c) w przykładzie 5



Rys. 23. Jeden ze sposobów numerowania części podłoża w przypadku obecności modułów z ustalonym położeniem

Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda z oryginalną modyfikacją funkcji kosztu oraz z użyciem sieci z oryginalną modyfikacją sygnałów wejściowych, a uzyskane wyniki zostały ze sobą porównane. Dla obydwu metod zbadano dwa sposoby inicjalizacji neuronów. Wyniki otrzymane podczas symulacji przedstawia tabela 12. Dla każdego przykładu przeprowadzono 100 prób. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 40: $A = 5$, $B = 6$, $C = 0{,}5$. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%.

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	38,02	36	36	36
	koszt najlepszego otrzymanego rozw.	36	36	36	36
	koszt optymalnego rozwiązania	36	36	36	36
	śr. koszt / koszt opt. rozw.	1,0561	1	1	1
	prawd. opt. rozw.	28%	99%	100%	100%
	wartość stałej *D*	0,35	0,4	0,4	0,4
2	średni koszt	51,7423	49,85	49,83	49,85
	koszt najlepszego otrzymanego rozw.	49	49	49	49
	koszt optymalnego rozwiązania	49	49	49	49
	śr. koszt / koszt opt. rozw.	1,056	1,0173	1,0169	1,0173
	prawd. opt. rozw.	7%	15%	17%	15%
	wartość stałej *D*	0,2	0,2	0,2	0,2
3	średni koszt	71,0825	68,23	68	68
	koszt najlepszego otrzymanego rozw.	67	68	68	68
	koszt optymalnego rozwiązania	67	67	67	67
	śr. koszt / koszt opt. rozw.	1,0609	1,0184	1,0149	1,0149
	prawd. opt. rozw.	2%	0%	0%	0%
	wartość stałej *D*	0,2	0,2	0,2	0,2
4	średni koszt	85,32	83,37	85	85
	koszt najlepszego otrzymanego rozw.	82	82	85	85
	koszt optymalnego rozwiązania	81	81	81	81
	śr. koszt / koszt opt. rozw.	1,0533	1,0293	1,0494	1,0494
	prawd. opt. rozw.	0%	0%	0%	0%
	wartość stałej *D*	0,15	0,15	0,15	0,15
5	średni koszt	98,0217	88	89,3118	89,8
	koszt najlepszego otrzymanego rozw.	88	88	88	88
	wartość stałej *D*	0,2	0,25	0,3	0,25

Tabela 12. Wyniki otrzymane dla poszczególnych przykładów z ustalonym położeniem modułów

Wyniki otrzymane klasyczną siecią Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 są zbliżone do wyników otrzymanych z użyciem sieci z modyfikacją sygnałów wejściowych dla obydwu sposobów inicjalizacji sieci. Dla przykładów 1-4 klasyczna sieć Hopfielda umożliwiła otrzymanie optymalnego rozwiązania lub koszt najlepszego rozwiązania jest większy o 1 jednostkę od kosztu optymalnego rozmieszczenia, które było podstawą do trwałego ustalenia położenia wybranych modułów. Średni koszt otrzymanych rozwiązań jest zbliżony do kosztu optymalnego rozwiązania. Dla przykładu 5 sieć umożliwiła otrzymanie rozwiązań o koszcie równym kosztowi rozmieszczenia modułów, które posłużyło do trwałego ustalenia położenia wybranych modułów, natomiast średni koszt otrzymanych rozwiązań jest zbliżony do kosztu powyższego rozmieszczenia modułów. Rozwiązania otrzymane z użyciem klasycznej sieci Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych z przedziału -0,3 do 0,3 są wyraźnie gorsze od rozwiązań otrzymanych siecią z modyfikacją sygnałów wejściowych.

6.4. Wyniki otrzymane dla równoczesnego rozmieszczania modułów oraz padów układu

Minimalizację długości połączeń w układzie można również przeprowadzić podczas równoczesnego rozmieszczania modułów na podłożu układu oraz końcówek całego układu na obrzeżach podłoża. Badania przeprowadzono dla 4 układów, składających się z 4 modułów oraz 8 końcówek całego układu. Moduły oraz końcówki całego układu były równocześnie rozmieszczane na podłożu o rozmiarach 4x4 jednostki. W przypadku równoczesnego rozmieszczania modułów i końcówek całego układu, w powyższych przykładach nie jest możliwe wyznaczenie kosztu optymalnego rozwiązania metodą wyszukiwania wyczerpującego. Dlatego zastosowano inne rozwiązanie, które umożliwia znajomość kosztu optymalnego rozwiązania. Tabele 13-16 przedstawiają macierze połączeń między samymi modułami dla przykładów 1-4.

Nr m./ nr m.	1	2	3	4
1	0	1	0	0
2	1	0	1	0
3	0	1	0	1
4	0	0	1	0

Tabela 13. Macierz połączeń dla przykładu 1

Nr m./ nr m.	1	2	3	4
1	0	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	0	1	0

Tabela 14. Macierz połączeń dla przykładu 2

Nr m./ nr m.	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Tabela 15. Macierz połączeń dla przykładu 3

Nr m./ nr m.	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

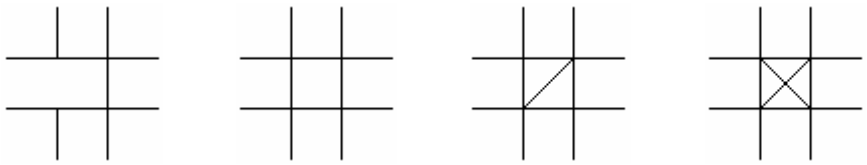
Tabela 16. Macierz połączeń dla przykładu 4

Koszt rozwiązania jest równy sumie kosztu połączeń między modułami oraz między modułami i końcówkami całego układu. Jeżeli dla danego rozmieszczenia obydwa składniki posiadają wartość minimalną z wszystkich możliwych rozwiązań, które są rozpatrywane oddzielnie dla danego składnika, to wtedy rozwiązanie jest optymalne. Powyższy warunek nie jest warunkiem koniecznym dla optymalności rozwiązania, ponieważ obydwa warunki nie muszą być równocześnie spełnione dla optymalnego rozwiązania (może to być niemożliwe). Powyższy warunek został wykorzystany jedynie do takiego określenia połączeń między modułami i końcówkami całego układu, aby ten warunek umożliwił znajomość kosztu optymalnego rozwiązania. Dla przykładów 1-4 połączenia między modułami i końcówkami całego układu, położonymi na obrzeżach układu, wyznaczono na podstawie optymalnego rozmieszczenia samych tylko modułów. Optymalne rozmieszczenie modułów oraz jego koszt zostały wyznaczone metodą wyszukiwania wyczerpującego. Moduły były rozmieszczane na podłożu o rozmiarach 2x2 jednostki, ponieważ pozostała część podłoża w rozpatrywanych przykładach jest przeznaczona dla

końcówek całego układu. Następnie, końcówki całego układu połączono jednym połączeniem z modułem, którego odległość w metryce Manhattan od danej końcówki całego układu dla optymalnego rozmieszczenia modułów wynosi dokładnie 1 jednostkę. Minimalny koszt połączeń między modułami i końcówkami całego układu jest zatem równy liczbie połączeń między końcówkami całego układu oraz modułami. W związku z powyższym, koszt optymalnego rozwiązania dla utworzonych przykładów jest znany i jest równy sumie minimalnego kosztu połączeń między samymi modułami oraz minimalnego kosztu połączeń między modułami i końcówkami całego układu, ponieważ obydwa składniki mogą posiadać równocześnie wartość minimalną. Koszt optymalnego rozmieszczenia modułów i końcówek całego układu dla przykładów 1-4 wynosi odpowiednio: 11, 12, 14 i 16 jednostek. Optymalne rozmieszczenie modułów oraz końcówek całego układu jest identyczne dla wszystkich przykładów. Rysunek 24 przedstawia położenie modułów i końcówek całego układu w przykładach 1-4, dla optymalnego rozmieszczenia modułów oraz końcówek całego układu. Rysunek 25 przedstawia połączenia między modułami i końcówkami całego układu w przykładach 1-4, dla optymalnego rozmieszczenia modułów oraz końcówek całego układu.

	k.6	k.5	
k.7	4	3	k.4
k.8	1	2	k.3
	k.1	k.2	

Rys. 24. Optymalne rozmieszczenie modułów i końcówek całego układu dla przykładów 1-4



(a) (b) (c) (d)

Rys. 25. Połączenia między modułami i końcówkami całego układu dla optymalnego rozmieszczenia modułów oraz końcówek całego układu: a) w przykładzie 1, b) w przykładzie 2, c) w przykładzie 3, d) w przykładzie 4

Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda oraz z użyciem sieci z modyfikacją sygnałów wejściowych, a uzyskane wyniki zostały ze sobą porównane. Dla obydwu metod zbadano dwa sposoby inicjalizacji neuronów. Ze względu na konieczność równoczesnego rozmieszczania końcówek całego układu oraz modułów, końcówki całego układu są traktowane tak jak moduły. Macierz połączeń jest wyznaczana

zgodnie ze wzorem 18 i obejmuje również połączenia między modułami a końcówkami całego układu. Nie jest tworzona zatem specjalna macierz dla połączeń między modułami a końcówkami całego układu. Podłoże przeznaczone do rozmieszczania modułów jest rozszerzone o części podłoża przeznaczone do rozmieszczania końcówek całego układu. We wszystkich przykładach sieć dokonywała równoczesnego rozmieszczenia modułów i końcówek całego układu na podłożu o rozmiarach 4x4 jednostki. Równoczesne rozmieszczanie modułów oraz końcówek całego układu nie wymaga dużych zmian w zastosowanej sieci Hopfielda. Liczba neuronów w sieci $N = (X+X_{kon})\ K\ L$, gdzie X_{kon} jest liczbą końcówek całego układu, które są rozmieszczane. Sieć składa się z $(X+X_{kon})$ wierszy, zawierających $M = K\ L$ neuronów, zgodnie z rysunkiem 15. Wartości wag połączeń między neuronami w sieci oraz zewnętrznych sygnałów wejściowych neuronów są wyznaczane na podstawie wzorów 26 i 27. Ze względu na ograniczenie położenia końcówek całego układu na obrzeżach obszaru zarezerwowanego do rozmieszczenia modułów, podczas symulacji sieci zgodnie ze wzorem 34 lub 35, powyższe ograniczenie musi być uwzględnione. W tym celu, w każdej iteracji symulacji pracy sieci, dla wszystkich neuronów, które odpowiadają położeniu dowolnej końcówki całego układu w dowolnej części podłoża, zarezerwowanej wyłącznie dla modułów, jest ustalana wartość sygnału wyjściowego równa 0. W związku z tym, nie jest możliwe umieszczenie przez sieć końcówki całego układu w obszarze przeznaczonym wyłącznie do rozmieszczania modułów. W ten sam sposób wyklucza się możliwość umieszczenia modułu w obszarze przeznaczonym dla końcówek całego układu. Neurony, dla których w każdej iteracji jest ustalana wartość ich sygnału wyjściowego równa 0, mogą być usunięte z sieci, ponieważ ich obecność nie ma żadnego wpływu na pozostałe neurony. Wyniki otrzymane podczas symulacji przedstawia tabela 17. Dla każdego przykładu przeprowadzono 100 prób. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 27: $A = 5$, $B = 6$, $C = 0{,}5$. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%.

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	12,9899	11,7174	11,4688	11,4725
	koszt najlepszego otrzymanego rozw.	11	11	11	11
	koszt optymalnego rozwiązania	11	11	11	11
	śr. koszt / koszt opt. rozw.	1,1809	1,0652	1,0426	1,043
	prawd. opt. rozw.	20%	43%	57%	52%
	wartość stałej D	0,6	0,65	0,75	0,8

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
2	średni koszt	14	12,8526	12,5	12,4043
	koszt najlepszego otrzymanego rozw.	12	12	12	12
	koszt optymalnego rozwiązania	12	12	12	12
	śr. koszt / koszt opt. rozw.	1,1667	1,0711	1,0417	1,0337
	prawd. opt. rozw.	23%	53%	72%	75%
	wartość stałej *D*	0,6	0,75	0,8	0,85
3	średni koszt	15,5326	14,7634	14,625	14,4842
	koszt najlepszego otrzymanego rozw.	14	14	14	14
	koszt optymalnego rozwiązania	14	14	14	14
	śr. koszt / koszt opt. rozw.	1,1095	1,0545	1,0446	1,0346
	prawd. opt. rozw.	19%	39%	42%	49%
	wartość stałej *D*	0,55	0,55	0,55	0,7
4	średni koszt	17,26	16,4375	16	16
	koszt najlepszego otrzymanego rozw.	16	16	16	16
	koszt optymalnego rozwiązania	16	16	16	16
	śr. koszt / koszt opt. rozw.	1,0788	1,0273	1	1
	prawd. opt. rozw.	56%	78%	92%	97%
	wartość stałej *D*	0,45	0,55	0,6	0,7

Tabela 17. Wyniki otrzymane dla poszczególnych przykładów dla równoczesnego rozmieszczania modułów oraz końcówek całego układu

Podczas analizy wyników zawartych w tabeli 17 można zauważyć, że wyniki otrzymane klasyczną siecią z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 są zbliżone do wyników otrzymanych siecią z modyfikacją sygnałów wejściowych, dla obydwu sposobów inicjalizacji neuronów. Widoczna jest jednak przewaga sieci z modyfikacją sygnałów wejściowych. Średni koszt rozwiązań otrzymanych powyższymi metodami jest tylko o kilka procent większy od kosztu optymalnego rozwiązania. Dla wszystkich przykładów otrzymano optymalne rozwiązanie, natomiast prawdopodobieństwo otrzymania optymalnego rozwiązania w jednej próbie jest bardzo duże. Rozwiązania otrzymane z użyciem klasycznej sieci Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych z przedziału -0,3 do

0,3 są wyraźnie gorsze od rozwiązań otrzymanych siecią z modyfikacją sygnałów wejściowych.

Należy podkreślić, że rozmieszczanie końcówek całego układu razem z modułami, powoduje wzrost liczby neuronów oraz połączeń między neuronami w sieci Hopfielda. Z tego względu, w rozważanych przykładach rozmieszczano jedynie 4 moduły.

6.5. Podsumowanie

W niniejszym rozdziale przedstawiono oryginalne rozwiązania, umożliwiające zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z ustalonym położeniem końcówek całego układu. Rozważono również minimalizację długości połączeń w układach z trwale ustalonym położeniem części modułów lub w przypadku, gdy pewne części podłoża nie mogą być wykorzystane do rozmieszczania modułów. Zaproponowano również rozwiązanie umożliwiające równoczesne rozmieszczanie modułów oraz końcówek całego układu.

Analiza otrzymanych rezultatów pozwala zauważyć, że obecność końcówek całego układu o ustalonym położeniu, umożliwia znaczną poprawę jakości otrzymywanych rozwiązań. Obecność końcówek całego układu o ustalonym położeniu sprawia, że rezultaty otrzymywane z użyciem klasycznej sieci Hopfielda, dla inicjalizacji neuronów wartościami sygnałów wejściowych bliskimi 0, są zbliżone do rezultatów otrzymywanych z użyciem sieci z modyfikacją sygnałów wejściowych dla obydwu sposobów inicjalizacji neuronów. Średni koszt otrzymywanych rozwiązań jest tylko o kilka procent większy od kosztu optymalnego rozwiązania. Trwałe ustalenie położenia części modułów na podłożu układu, nie powoduje pogorszenia jakości otrzymywanych rozwiązań.

W przypadku układów o większym wymiarze, sieć z modyfikacją sygnałów wejściowych umożliwia otrzymanie lepszych rezultatów, w porównaniu do klasycznej sieci Hopfielda. Przewaga sieci z modyfikacją sygnałów wejściowych jest również widoczna dla wszystkich przykładów, w przypadku inicjalizacji neuronów wartościami sygnałów wejściowych z przedziału -0,3 do 0,3. Rezultaty otrzymane klasyczną siecią dla powyższego sposobu inicjalizacji neuronów są znacznie gorsze. Klasyczna sieć Hopfielda wymaga inicjalizacji neuronów wartościami sygnałów wejściowych bliskimi 0 (sygnały wyjściowe neuronów bliskie 0,5) tak, aby żaden neuron nie był uprzywilejowany. Przewaga sieci z modyfikacją sygnałów wejściowych jest również widoczna w przypadku równoczesnego rozmieszczania modułów oraz końcówek całego układu, ponieważ w układzie nie występują wówczas końcówki o ustalonym położeniu. W przypadku braku końcówek całego układu o ustalonym położeniu, sieć z modyfikacją sygnałów wejściowych umożliwia otrzymanie rozwiązań o lepszej jakości. Potwierdzają to również badania przeprowadzone w poprzednim rozdziale.

Rozdział 7
Minimalizacja długości połączeń w układach z węzłami posiadającymi wiele końcówek

Dotychczasowe badania przeprowadzono przy założeniu, że wszystkie węzły układu posiadają jedynie dwie końcówki. W niniejszym rozdziale przedstawiono oryginalne zastosowanie sieci Hopfielda do minimalizacji długości połączeń w układach z węzłami posiadającymi wiele końcówek. Oryginalna postać funkcji kosztu sieci Hopfielda, która umożliwia minimalizację długości połączeń w układach z węzłami posiadających wiele końcówek jest przedstawiona w podrozdziale 7.1. W podrozdziale 7.2 opisano oryginalny sposób korekcji estymacji długości połączeń dla węzłów zawierających wiele końcówek. W podrozdziale 7.3 przedstawiono wyniki rozmieszczania otrzymane dla układów, w których węzły posiadają wiele końcówek. Podsumowanie wyników przeprowadzonych symulacji zamieszczono w podrozdziale 7.4.

7.1. Postać funkcji kosztu dla układów z węzłami posiadającymi wiele końcówek

Postać funkcji kosztu problemu minimalizacji długości połączeń przedstawiona w rozdziałach 5 i 6 zakłada, że wszystkie węzły układu posiadają jedynie dwie końcówki. Obecność węzłów w układzie, które posiadają wiele końcówek, pociąga za sobą konieczność estymacji długości (kosztu) połączeń, ponieważ końcówki węzła mogą być połączone w różny sposób i nie można określić sposobu ich połączenia przed rozmieszczeniem modułów. Sieć Hopfielda musi więc wykorzystać estymację długości połączeń podczas rozmieszczania modułów, podobnie jak inne metody rozmieszczania. Ze względu na postać funkcji energii sieci Hopfielda, która jest określona wzorem 20, do estymacji długości połączeń może być zastosowana metoda oparta na grafie pełnym, opisana w podrozdziale 2.3.2. Składnik funkcji kosztu sieci Hopfielda określający całkowitą ważoną długość połączeń dla układów z ustalonym położeniem końcówek, w których węzły mogą posiadać dowolną liczbę końcówek, jest wyrażony wzorem

$$E_4 = \frac{D}{2} \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{x'=1}^{X} \sum_{i'=1}^{M} V_{xi} V_{x'i'} cm_{xx'} d_{ii'} + D \sum_{x=1}^{X} \sum_{i=1}^{M} \sum_{p=1}^{P} V_{xi} cmp_{xp} d_{ip} \quad (41)$$

gdzie: P - liczba końcówek całego układu, $d_{i\,i'}$ - odległość między częściami podłoża i oraz i', $cm_{x\,x'}$ - element macierzy połączeń **CM** układu, określający wagę połączenia między modułami x oraz x', $d_{i\,p}$ - odległość między częścią podłoża i oraz końcówką całego układu p, $cmp_{x\,p}$ - element macierzy połączeń **CMP** układu, określający wagę połączenia między modułem x a końcówką całego układu p.

Elementy macierzy **CM** są określone następująco

$$cm_{x\,x'} = \sum_{i=1}^{S} \frac{2}{t_i} cm_{i\,x\,x'} \tag{42}$$

gdzie: S - liczba węzłów w układzie, t_i – liczba końcówek należących do węzła i, $cm_{i\,x\,x'}$ - waga połączenia między modułami x oraz x' w węźle i.

Waga $cm_{i\,x\,x'}$ jest określona następująco

$$cm_{i\,x\,x'} = \begin{cases} w_i > 0 & \text{gdy jest połączenie między modułami } x \text{ oraz } x' \text{ w węźle } i \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{43}$$

gdzie: w_i – waga połączeń w węźle i.

Waga w_i umożliwia preferencję połączeń w wybranych węzłach układu. Przy braku preferencji, w_i = 1 dla wszystkich węzłów układu.
Elementy macierzy **CMP** są określone następująco

$$cmp_{x\,p} = \sum_{i=1}^{S} \frac{2}{t_i} cmp_{i\,x\,p} \tag{44}$$

gdzie: S - liczba węzłów w układzie, t_i – liczba końcówek należących do węzła i, $cmp_{i\,x\,p}$ - waga połączenia między modułem x a końcówką całego układu p w węźle i.

Waga $cmp_{i\,x\,p}$ jest określona następująco

$$cmp_{i\,x\,p} = \begin{cases} w_i > 0 & \text{gdy jest połączenie między modułem } x \\ & \text{a końcówką całego układu } p \text{ w węźle } i\,. \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{45}$$

Sieć neuronowa z wagami i zewnętrznymi sygnałami wejściowymi neuronów określonymi według wzorów 26, 40 oraz 42-45, umożliwia minimalizację długości połączeń w układach z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek.

7.2. Korekcja estymacji długości połączeń

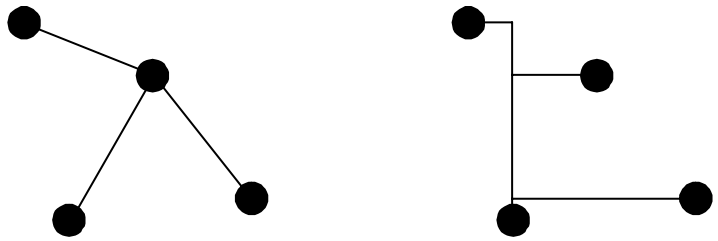
Metoda estymacji długości połączeń zastosowana w poprzednim podrozdziale jest niedokładna, szczególnie dla węzłów zawierających dużą liczbę końcówek. W niniejszym podrozdziale zostanie przedstawiony oryginalny sposób wyznaczenia współczynników korygujących wartość estymowanej długości połączeń, w zależności od liczby końcówek w danym węźle układu. Przedstawione rozwiązanie wykorzystuje aproksymację minimalnego drzewa Steinera, która umożliwia wyznaczenie długości połączeń dla węzłów zawierających określoną liczbę końcówek. Metoda wyznaczania ścieżek połączeń z wykorzystaniem aproksymacji minimalnego drzewa Steinera jest opisana w podrozdziale 7.2.1. W podrozdziale 7.2.2 jest przedstawiony sposób wyznaczenia współczynników korygujących estymowaną długość połączeń.

7.2.1. Zmodyfikowany algorytm Prima

Algorytm Prima umożliwia wyznaczenie minimalnego drzewa rozpinającego grafu (ang. minimal spanning tree), które jest zbiorem krawędzi łączących wszystkie wierzchołki grafu i posiadających minimalną sumaryczną długość (wagę) krawędzi. Minimalne drzewo rozpinające grafu jest wykorzystywane w projektowaniu topografii układów VLSI do łączenia końcówek danego węzła układu lub do estymacji długości jego połączeń [1, 9, 28, 55, 73-74, 134, 136]. Aby wyznaczyć to drzewo należy rozpatrzyć graf pełny, którego wierzchołkami są wszystkie końcówki danego węzła. Suma długości krawędzi minimalnego drzewa rozpinającego w metryce Manhattan jest dłuższa od sumy długości krawędzi minimalnego drzewa Steinera o nie więcej niż 50%, przeciętnie o 12% [29, 133]. Minimalne drzewo Steinera umożliwia połączenie wszystkich końcówek węzła (wierzchołków grafu) używając minimalnej długości połączeń. Krawędzie drzewa Steinera łączą się w tzw. punktach Steinera, które nie muszą być wierzchołkami grafu. Istnieje wiele sposobów aproksymacji minimalnego drzewa Steinera, które wymagają różnych nakładów obliczeniowych [29, 133-134]. Rysunek 26 przedstawia minimalne drzewo rozpinające oraz drzewo Steinera.

W celu wyznaczenia współczynników korygujących estymację długości połączeń, zastosowano aproksymację minimalnego drzewa Steinera, która jest stosowana do wyznaczania połączeń między modułami w układach VLSI [10, 135]. Powyższa aproksymacja minimalnego drzewa Steinera wykorzystuje zmodyfikowany algorytm Prima oraz algorytm Lee (ang. Lee algorithm, maze router). W powyższej metodzie, podobnie jak w przypadku minimalnego drzewa rozpinającego, losowo jest wybierana jedna z końcówek węzła, która tworzy zbiór końcówek odwiedzonych. Pozostałe końcówki tworzą zbiór końcówek nie odwiedzonych. Następnie jest wyznaczana ścieżka połączenia, która łączy wybraną końcówkę z najbliższą końcówką nie odwiedzoną. Końcówki nie odwiedzone, które zostały połączone z końcówkami odwiedzonymi, są za

każdym razem przesuwane do zbioru końcówek odwiedzonych. Następnie, są wyznaczane ścieżki połączeń o minimalnej długości, które łączą dowolną końcówkę nie odwiedzoną z dowolną końcówką odwiedzoną lub ścieżką, która została wcześniej wyznaczona. Ścieżki mogą łączyć się ze sobą w dowolnym miejscu. Ścieżki nie muszą łączyć się tylko w miejscu położenia końcówek odwiedzonych, jak to ma miejsce w przypadku minimalnego drzewa rozpinającego. Powyższe czynności są wykonywane w sumie (*n*-1) razy, gdzie *n* jest liczbą końcówek w węźle. Ścieżki są wyznaczane z użyciem algorytmu Lee [1, 3-5, 8, 10, 28].



(a) (b)

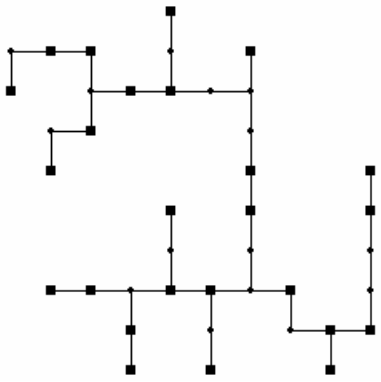
Rys. 26. Stosowane drzewa połączeń: a) minimalne drzewo rozpinające, b) drzewo Steinera

Rysunek 27 przedstawia sposób wyznaczania najkrótszej ścieżki między końcówkami *A* i *B* na podłożu, które jest siatką kwadratowych części, o rozmiarze równym jednej jednostce. Proces wyznaczania ścieżki rozpoczyna się od części podłoża zawierającej końcówkę *A*, która jest oznaczana liczbą 0. Na początku wpisuje się liczbę 1 w każdą pustą część sąsiadującą z tą częścią. Następnie, wpisuje się liczbę 2 do wszystkich pustych części sąsiadujących z częściami zawierającymi liczbę 1, itd. Proces znaczenia części kończy się, gdy zostanie osiągnięta końcówka docelowa *B*. Liczba wpisana do części podłoża, w której znajduje się końcówka *B* jest długością najkrótszej ścieżki między końcówkami *A* i *B*.

2	1	2	3	4	5	6	7
1	*A*0	1	2	3	4	5	6
2	1	2	3	4	5	6	7
3	2	3	4	5	6	7	
4	3	4	5	6	*B*7		
5	4	5	6	7			
6	5	6	7				
7	6	7					

Rys. 27. Wyznaczanie najkrótszej ścieżki połączenia z użyciem algorytmu Lee

W celu przeprowadzenia prawidłowego znaczenia części podłoża należy zastosować kolejkę [73-74, 136]. Każda część po oznaczeniu powinna zostać umieszczona w kolejce wraz z przydzieloną jej liczbą. Po oznaczeniu wszystkich pustych sąsiadów danej części, z kolejki jest pobierana pierwsza w kolejności część wraz z liczbowym oznaczeniem *d* i są znaczone wszystkie jej puste sąsiednie części z użyciem liczby *d*+1. Powyższe czynności są powtarzane, aż do osiągnięcia końcówki *B*. Rozpatrywane jest sąsiedztwo jedynie w kierunku poziomym oraz pionowym. Wyznaczenie trasy ścieżki łączącej końcówkę *B* z końcówką *A* odbywa się podczas etapu cofania. Poczynając od końcówki *B*, są wybierane sąsiednie części podłoża, oznaczone liczbą mniejszą o 1. Po osiągnięciu części podłoża zawierającej końcówkę *A*, ścieżka jest wyznaczona. Należy ustalić porządek wybierania kierunków, gdyż prowadzi to do porządkowania przebiegu ścieżek, które rzadziej wtedy zmieniają kierunek [28]. W przypadku, gdy jest już wyznaczona przynajmniej jedna ścieżka, przed rozpoczęciem wyznaczania kolejnej ścieżki, kolejka jest opróżniana i są kasowane oznaczenia wszystkich części. Następnie, wszystkie części, przez które przebiegają wyznaczone już ścieżki otrzymują oznaczenie 0 i są umieszczane w kolejce wraz z tym oznaczeniem. Po pobraniu pierwszej w kolejności części oraz jej liczbowego oznaczenia, rozpoczyna się proces wyznaczania następnej ścieżki. Proces znaczenia części kończy się po osiągnięciu kolejnej końcówki nie odwiedzonej węzła. Etap cofania kończy się po napotkaniu dowolnej części oznaczonej liczbą 0 (część podłoża z końcówką lub wcześniej wyznaczona ścieżka). Rysunek 28 przedstawia aproksymację minimalnego drzewa Steinera dla węzła zawierającego 25 końcówek, która została otrzymana w wyniku zastosowania zmodyfikowanego algorytmu Prima oraz algorytmu Lee.



Rys. 28. Przykład aproksymacji minimalnego drzewa Steinera dla węzła zawierającego 25 końcówek

Przedstawiona metoda aproksymacji minimalnego drzewa Steinera jest stosowana do wyznaczania połączeń podczas trasowania globalnego oraz trasowania szczegółowego w układach VLSI [10, 135].

7.2.2. Wyznaczenie współczynników korygujących estymację długości połączeń

Dokładność estymacji długości połączeń jest uzależniona od liczby końcówek należących do węzła. W przypadku metody połowy obwodu estymowana długość połączeń jest zwykle zbyt mała, gdy liczba końcówek należąca do węzła jest większa niż 3. Rozbieżność między wartością estymowaną a rzeczywistą długością połączeń niezbędnych do połączenia końcówek węzła, wzrasta wraz ze wzrostem liczby końcówek w węźle [1, 9-10]. W przypadku estymacji długości połączeń opartej na grafie pełnym, wartość estymowana jest zbyt duża dla węzłów zawierających więcej niż 2 końcówki, a rozbieżność wzrasta wraz z liczbą końcówek w węźle. W celu rozwiązania tego problemu można wyznaczyć współczynniki korygujące estymację długości połączeń opartą na grafie pełnym, w zależności od liczby końcówek w węźle. Współczynniki te są równe rzeczywistej, minimalnej długości połączeń, niezbędnej do połączenia wszystkich końcówek danego węzła, podzielonej przez wartość estymowaną, zgodnie z następującym wzorem [1]

$$f_i = \frac{d_{rz}}{d_{est}} \tag{46}$$

gdzie: f_i – współczynnik korygujący estymację długości połączeń dla węzła zawierającego i końcówek, d_{rz} – rzeczywista, minimalna długość połączeń, niezbędnych do połączenia wszystkich końcówek węzła, d_{est} - estymowana długość połączeń w danym węźle, wyznaczona z użyciem metody opartej na grafie pełnym, bez użycia współczynników korygujących estymację (podrozdział 2.3.2).

W przedstawionym rozwiązaniu jako rzeczywistą długość połączeń przyjęto długość połączeń otrzymaną w wyniku zastosowania zmodyfikowanego algorytmu Prima, ponieważ metoda ta jest stosowana do wyznaczania połączeń w układach VLSI [10, 135]. Ponadto metoda ta jest lepszą metodą aproksymacji minimalnego drzewa Steinera, w porównaniu do metody wykorzystującej minimalne drzewo rozpinające. Współczynniki korygujące estymację opartą na grafie pełnym wyznaczono dla węzłów zawierających od 3 do 50 końcówek. Końcówki węzłów były losowo rozmieszczane na podłożu o rozmiarze 100x100 części. Dla każdej liczebności końcówek w węźle wykonano 100 prób, dla których obliczono średnią arytmetyczną wartości współczynników, wyznaczonych w każdej próbie. Tabela 18 przedstawia otrzymane wartości współczynników dla estymacji opartej na grafie pełnym, po obliczeniu średniej. Węzły zawierające 2 końcówki nie wymagają korekcji estymacji długości połączeń, a współczynnik korygujący posiada wartość równą 1.

Analiza zawartości tabeli 18 wyraźnie pokazuje, że dla estymacji opartej na grafie pełnym, estymowana długość połączeń bez użycia współczynników korygujących estymację jest większa od rzeczywistej długości połączeń. Rozbieżność ta wzrasta wraz ze wzrostem liczby końcówek w węźle. Wyznaczone współczynniki umożliwiają dokładną estymację długości połączeń

w funkcji kosztu sieci Hopfielda dla problemu minimalizacji długości połączeń w układach, w których węzły mogą posiadać dowolną liczbę końcówek. W wyniku zastosowania współczynników korygujących estymację długości połączeń, wzory 42 i 44 przyjmują postać

$$cm_{x\,x'} = \sum_{i=1}^{S} f_i \frac{2}{t_i} cm_{i\,x\,x'} \tag{47}$$

$$cmp_{x\,p} = \sum_{i=1}^{S} f_i \frac{2}{t_i} cmp_{i\,x\,p} \tag{48}$$

gdzie: f_i – współczynnik korygujący estymację długości połączeń dla węzła zawierającego *i* końcówek.

Liczba końc.	Wartość wsp.	Liczba końc.	Wartość wsp.	Liczba końc.	Wartość wsp.	Liczba końc.	Wartość wsp.
3	0,778518	15	0,325748	27	0,235157	39	0,195314
4	0,657925	16	0,318217	28	0,231896	40	0,191714
5	0,586268	17	0,303703	29	0,228596	41	0,191387
6	0,528721	18	0,293221	30	0,222555	42	0,189338
7	0,490405	19	0,287067	31	0,22015	43	0,184204
8	0,451731	20	0,280056	32	0,217958	44	0,181387
9	0,423296	21	0,271281	33	0,214117	45	0,179508
10	0,404378	22	0,265728	34	0,207537	46	0,179513
11	0,385386	23	0,257337	35	0,205604	47	0,177683
12	0,36574	24	0,251247	36	0,204178	48	0,173229
13	0,355437	25	0,247934	37	0,201443	49	0,171331
14	0,336212	26	0,240326	38	0,198371	50	0,170131

Tabela 18. Wartości współczynników korygujących estymację długości połączeń dla metody opartej na grafie pełnym

W taki sam sposób można wyznaczyć współczynniki korygujące estymację długości połączeń dla metody połowy obwodu. Wyznaczone współczynniki mogą być zastosowane do korekcji estymacji długości połączeń przez dowolną metodę rozmieszczania. Metoda wykorzystująca zmodyfikowany algorytm Prima lepiej aproksymuje minimalne drzewo Steinera, w porównaniu do minimalnego drzewa rozpinającego. Większa dokładność aproksymacji minimalnego drzewa Steinera nie jest konieczna, ponieważ podczas wyznaczania połączeń w układach VLSI, połączenia mogą być przenoszone do innych kanałów o mniejszym skupieniu połączeń.

7.3. Wyniki otrzymane dla układów z węzłami posiadającymi wiele końcówek

W niniejszym podrozdziale zostaną przedstawione rezultaty rozmieszczania modułów z minimalizacją długości połączeń dla trzech przykładów z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek. Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda oraz z użyciem sieci z modyfikacją sygnałów wejściowych, a uzyskane wyniki zostały ze sobą porównane. Dla obydwu metod zbadano dwa sposoby inicjalizacji neuronów. Wartości wag połączeń między neuronami w sieci oraz zewnętrznych sygnałów wejściowych neuronów były wyznaczone na podstawie wzorów 26, 40, 43, 45 oraz 47-48, ze względu na minimalizację długości połączeń w układach z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek.

Wszystkie przykłady są układami FPGA, które oryginalnie są zapisane w formacie SEGA [137]. Przykłady poddano konwersji do formatu BLIF, z użyciem zalecanego programu [137]. Następnie zastosowano program VPACK, który umożliwił implementację układów z wykorzystaniem komórek CLB, składających się z tablic LUT (ang. look-up table) oraz przerzutników D [61-62, 105]. Program VPACK umożliwił utworzenie dla każdego przykładu listy połączeń między modułami (komórkami CLB), w postaci pliku NET, dla której było następnie wykonywane rozmieszczanie [105]. Przykład 1 (*term1*) składa się z 14 modułów (komórek CLB), 34 wejść całego układu, 10 wyjść całego układu oraz 66 węzłów, zawierających 2÷7 końcówek. Moduły były rozmieszczane na podłożu o rozmiarach 4x4 części. Przykład 2 (*9symml*) składa się z 19 modułów (komórek CLB), 9 wejść całego układu, 1 wyjścia całego układu oraz 70 węzłów, zawierających 2÷13 końcówek. Moduły były rozmieszczane na podłożu o rozmiarach 5x5 części. Przykład 3 (*Test*) składa się z 20 modułów (komórek CLB), 5 wejść całego układu, 16 wyjść całego układu oraz 25 węzłów, zawierających 2÷17 końcówek. Moduły były rozmieszczane na podłożu o rozmiarach 5x5 części. W przykładzie 3 komórki CLB posiadały 4 wejścia i 1 wyjście (4-wejściowa tablica LUT oraz przerzutnik D). Przykłady 1 oraz 2 powstały w wyniku zaimplementowania układów *term1* oraz *9symml* z użyciem komórek CLB, które posiadały 10 wejść i 4 wyjścia.

Wyniki otrzymane podczas symulacji przedstawia tabela 19. Dla każdego przykładu przeprowadzono 100 prób. Dla wszystkich symulacji zastosowano następujące wartości stałych ze wzorów 26 i 40: $A = 5$, $B = 6$, $C = 0{,}5$. Wartości stałej D dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%. Zastosowano współczynniki korygujące wartość estymowanej długości połączeń przedstawione w tabeli 18. We wszystkich przykładach każdy węzeł posiadał taką samą wagę $w_i = 1$, zastosowaną we wzorach 43 i 45. W tabeli 19 są zebrane następujące wyniki: średnia estymowana długość połączeń, najmniejsza estymowana długość połączeń dla otrzymanych rozwiązań oraz wartość stałej D. Długość połączeń jest estymowana z użyciem grafu pełnego i jest wyrażona z użyciem jednostek,

które są równe rozmiarowi komórki CLB. W tabeli 19 zamieszczono również estymowaną długość połączeń dla rozmieszczenia modułów otrzymanego z użyciem narzędzia rozmieszczania modułów programu VPR, wersja 4.30 (rozmieszczenie wykonano zgodnie z opisem zamieszczonym na płycie CD - Dodatek). VPR jest zaliczany do wiodących akademickich programów rozmieszczania oraz trasowania połączeń dla układów FPGA [10, 103-105]. Program VPR również wykorzystuje minimalizację całkowitej długości połączeń podczas rozmieszczania modułów, ponieważ minimalizacja długości połączeń pośrednio przyczynia się do minimalizacji skupienia połączeń i umożliwia wykonanie wszystkich połączeń, w przypadku ograniczonej pojemności kanałów przeznaczonych do prowadzenia połączeń [4, 10, 30, 84, 101]. We wszystkich przykładach wejścia i wyjścia całego układu (końcówki całego układu) posiadały ustalone położenie, takie samo jak w rozmieszczeniu wykonanym z użyciem narzędzia rozmieszczania programu VPR, ponieważ dzięki temu było możliwe porównanie jakości rozmieszczania modułów przez sieć Hopfielda i program VPR.

Nr przykł.	Wynik (estymacja dla długości połączeń)	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średnia długość połączeń	183,6629	162,6456	163,1931	162,5215
	najmniejsza dł. poł. otrzyman. rozw.	164,1764	158,3135	158,3135	162,3484
	wartość stałej *D*	0,1	0,13	0,11	0,11
	dł. poł. dla rozm. programu VPR	158,3135	158,3135	158,3135	158,3135
2	średnia długość połączeń	275,9115	253,7566	253,2274	247,7692
	najmniejsza dł. poł. otrzyman. rozw.	260,6218	245,9107	242,2974	243,2985
	wartość stałej *D*	0,09	0,13	0,11	0,11
	dł. poł. dla rozm. programu VPR	241,5157	241,5157	241,5157	241,5157
3	średnia długość połączeń	106,3597	101,6110	101,4515	101,4328
	najmniejsza dł. poł. otrzyman. rozw.	101,1062	101,4328	101,4328	101,4328
	wartość stałej *D*	0,35	0,45	0,45	0,45
	dł. poł. dla rozm. programu VPR	101,4328	101,4328	101,4328	101,4328

Tabela 19. Wyniki otrzymane dla przykładów z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek

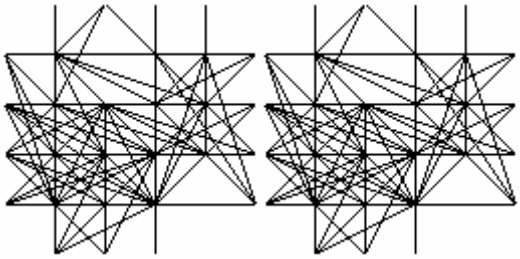
Z danych zawartych w tabeli 19 wynika, że sieć Hopfielda umożliwia otrzymanie rozwiązań, których jakość jest bardzo dobra. Estymowana długość połączeń dla rozwiązań otrzymanych z użyciem sieci Hopfielda jest porównywalna z estymowaną długością połączeń dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania programu VPR. Otrzymanie wyników porównywalnych z programem VPR jest możliwe w przypadku zastosowania klasycznej sieci Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 oraz sieci z modyfikacją sygnałów wejściowych, niezależnie od sposobu inicjalizacji neuronów. Dla przykładów 1-3, średnia estymowana długość połączeń dla rozwiązań otrzymanych z użyciem powyższych sieci jest nieznacznie większa, w porównaniu do estymowanej długości połączeń dla rozmieszczenia otrzymanego z użyciem narzędzia rozmieszczania programu VPR. Estymowana długość połączeń dla najlepszych rozwiązań, otrzymanych z użyciem sieci Hopfielda jest równa estymowanej długości połączeń dla rozmieszczenia otrzymanego z użyciem narzędzia rozmieszczania programu VPR lub jest nieznacznie większa.

Dla najlepszych rozwiązań otrzymanych klasyczną siecią Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0, wykonano trasowanie połączeń z użyciem narzędzia trasowania połączeń programu VPR (zgodnie z opisem zamieszczonym na płycie CD – Dodatek). Tabela 20 przedstawia wyniki otrzymane dla przykładu 1. W tabeli zamieszczono rezultaty dla 5 najlepszych rozwiązań otrzymanych siecią Hopfielda oraz dla rozmieszczenia otrzymanego z użyciem narzędzia rozmieszczania modułów programu VPR. Dla rozwiązań zamieszczonych w tabeli 20 wykonano również trasowanie połączeń z użyciem aproksymacji minimalnego drzewa Steinera, przedstawionej w podrozdziale 7.2.1. Długość połączeń wyznaczona z użyciem aproksymacji minimalnego drzewa Steinera była traktowana jako wzorzec podczas wyznaczania współczynników korygujących estymację długości połączeń w podrozdziale 7.2.2. Podczas wyznaczania aproksymacji minimalnego drzewa Steinera nie jest uwzględniane położenie końcówek komórek CLB oraz skupienie połączeń w poszczególnych kanałach układu, podobnie jak w przypadku minimalizacji długości połączeń z użyciem sieci Hopfielda lub narzędzia rozmieszczania programu VPR. Wyznaczenie długości aproksymacji minimalnego drzewa Steinera umożliwia jednak sprawdzenie dokładności zastosowanej estymacji długości połączeń. W tabeli 20 zamieszczono następujące wyniki: estymowaną długość połączeń dla danego rozmieszczenia, wyznaczoną z użyciem grafu pełnego, długość połączeń wyznaczonych z użyciem aproksymacji minimalnego drzewa Steinera dla danego rozmieszczenia modułów, długość połączeń po wykonaniu trasowania połączeń z użyciem narzędzia trasowania połączeń programu VPR oraz minimalną pojemność kanałów, która jest niezbędna do poprowadzenia wszystkich połączeń z użyciem narzędzia trasowania połączeń programu VPR. Minimalna pojemność kanałów jest równa maksymalnej liczbie poprowadzonych połączeń, w dowolnym kanale układu. Długość połączeń jest wyrażona z użyciem jednostek, które są równe rozmiarowi komórki CLB. Rysunek 29 przedstawia połączenia między modułami w przykładzie 1 dla najlepszego rozwiązania otrzymanego klasyczną siecią Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 oraz dla rozmieszczenia modułów wykonanego programem VPR. Połączenia między

modułami poprowadzono tak jak w grafie pełnym. Połączenia poprowadzono w przestrzeni Euklidesa dla większej przejrzystości rysunku.

Rozmieszczenie	Estymowana długość połączeń	Długość połączeń (min. drzewo Steinera)	Długość połączeń (VPR)	Minimalna pojemność kanału (VPR)
sieć Hopfielda 1	158,3135	159	302	8
sieć Hopfielda 2	159,9182	160	321	9
sieć Hopfielda 3	161,4034	159	329	9
sieć Hopfielda 4	162,3484	158	329	9
sieć Hopfielda 5	162,4339	162	328	9
VPR	158,3135	159	302	8

Tabela 20. Wyniki trasowania otrzymane dla najlepszych rozwiązań w przykładzie 1



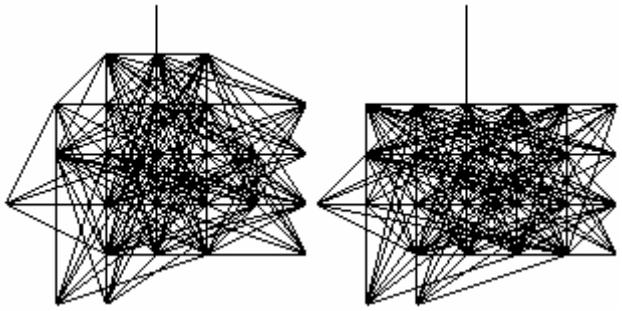
(a) (b)

Rys. 29. Połączenia między modułami w przykładzie 1: a) najlepsze rozmieszczenie otrzymane siecią Hopfielda, b) rozmieszczenie programu VPR

Tabele 21-22 oraz rysunki 30-31 przedstawiają rezultaty otrzymane dla przykładów 2-3, dla najlepszych rozwiązań uzyskanych klasyczną siecią Hopfielda z inicjalizacją neuronów wartościami sygnałów wejściowych bliskimi 0 oraz dla rozmieszczenia modułów wykonanego programem VPR.

Rozmieszczenie	Estymowana długość połączeń	Długość połączeń (min. drzewo Steinera)	Długość połączeń (VPR)	Minimalna pojemność kanału (VPR)
sieć Hopfielda 1	245,9107	248	517	10
sieć Hopfielda 2	246,5967	248	509	11
sieć Hopfielda 3	246,9424	248	504	11
sieć Hopfielda 4	247,6436	248	501	10
sieć Hopfielda 5	247,6962	249	523	10
VPR	241,5157	235	504	11

Tabela 21. Wyniki trasowania otrzymane dla najlepszych rozwiązań w przykładzie 2

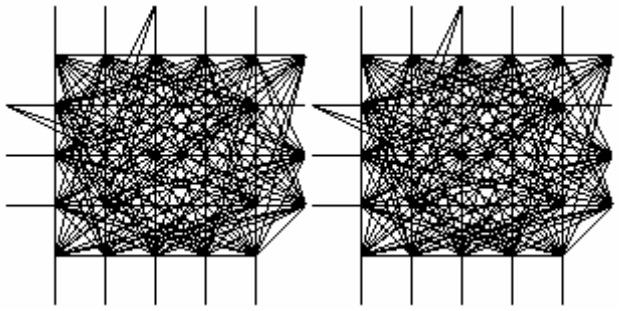


(a) (b)

Rys. 30. Połączenia między modułami w przykładzie 2: a) najlepsze rozmieszczenie otrzymane siecią Hopfielda, b) rozmieszczenie programu VPR

Rozmieszczenie	Estymowana długość połączeń	Długość połączeń (min. drzewo Steinera)	Długość połączeń (VPR)	Minimalna pojemność kanału (VPR)
sieć Hopfielda 1	101,4328	93	156	4
sieć Hopfielda 2	102,8398	94	145	3
sieć Hopfielda 3	103,3088	95	132	4
VPR	101,4328	93	154	4

Tabela 22. Wyniki trasowania otrzymane dla najlepszych rozwiązań w przykładzie 3



(a) (b)

Rys. 31. Połączenia między modułami w przykładzie 3: a) najlepsze rozmieszczenie otrzymane siecią Hopfielda, b) rozmieszczenie programu VPR

Rezultaty zawarte w tabelach 20-22 potwierdzają wysoką jakość rozwiązań, otrzymanych z użyciem sieci Hopfielda. Długości połączeń po wykonaniu trasowania połączeń z użyciem narzędzia trasowania programu VPR są porównywalne dla rozmieszczeń modułów wyznaczonych z użyciem sieci Hopfielda oraz dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR. Niektóre rozmieszczenia otrzymane siecią Hopfielda umożliwiają nawet połączenie modułów z użyciem mniejszej długości połączeń lub wymagają kanałów, których pojemność jest

mniejsza o 1. Rezultaty zamieszczone w tabelach 20-22 umożliwiają również weryfikację zastosowanej metody estymacji długości połączeń. Dla wszystkich rozpatrywanych rozmieszczeń modułów, estymowana długość połączeń oraz długość połączeń aproksymacji minimalnego drzewa Steinera jest zbliżona. Otrzymane rezultaty potwierdzają dokładność zastosowanej metody estymacji długości połączeń. Dla wszystkich rozpatrywanych rozwiązań otrzymanych z użyciem sieci Hopfielda oraz dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR, długości połączeń aproksymacji minimalnego drzewa Steinera są porównywalne. Rozbieżność między estymowaną długością połączeń, długością połączeń aproksymacji minimalnego drzewa Steinera oraz długością połączeń, wyznaczoną z użyciem narzędzia trasowania połączeń programu VPR, wynika z zastosowanego rozwiązania w programie VPR. Podczas trasowania połączeń, program VPR przenosi połączenia między kanałami, w zależności od skupienia połączeń w kanałach. Narzędzie trasowania połączeń programu VPR stara się wykonać wszystkie połączenia z użyciem kanałów o możliwie najmniejszej pojemności, co może być powodem większej długości połączeń, w porównaniu do estymowanej długości połączeń. Powyższa rozbieżność występuje również dla rozmieszczeń modułów, wykonanych z użyciem narzędzia rozmieszczania programu VPR, ponieważ podczas rozmieszczania modułów program VPR również nie uwzględnia skupienia połączeń w kanałach.

7.4. Podsumowanie

Program VPR jest zaliczany do wiodących akademickich programów rozmieszczania oraz trasowania połączeń dla układów FPGA. Program VPR bardzo często jest stosowany jako „wzorcowe” narzędzie, które umożliwia porównanie rezultatów rozmieszczania otrzymanych różnymi metodami.

Korekcja estymacji długości połączeń oparta na grafie pełnym umożliwia dostarczenie przez sieć Hopfielda rozwiązań, których jakość jest porównywalna z rozmieszczeniami modułów, otrzymanymi z użyciem programu VPR. Niektóre rozwiązania otrzymane siecią Hopfielda umożliwiają wyznaczenie wszystkich połączeń z użyciem kanałów, których pojemność jest mniejsza o 1 lub wymagają mniejszej długości połączeń. Podczas projektowania topografii układów VLSI jest wskazane rozpatrzenie kilku najlepszych rozwiązań, ze względu na częste stosowanie różnych estymacji. Jedynie wyznaczenie połączeń umożliwia wiarygodną ocenę jakości projektu układu. Rozwiązania, które posiadają większy estymowany koszt, mogą zapewnić lepsze właściwości funkcjonalne układu po wyznaczeniu wszystkich połączeń. Potwierdzają to przedstawione przykłady, w których minimalizacja estymowanej długości połączeń jest tylko w przybliżeniu równoważna minimalizacji rzeczywistej długości połączeń, po wykonaniu trasowania połączeń. Rezultaty otrzymane w niniejszym rozdziale należy uznać za bardzo dobre, ze względu na zbliżoną jakość rozwiązań otrzymanych z użyciem sieci Hopfielda, w porównaniu do programu VPR.

Rozdział 8
Minimalizacja długości połączeń z wykorzystaniem równoległego przetwarzania

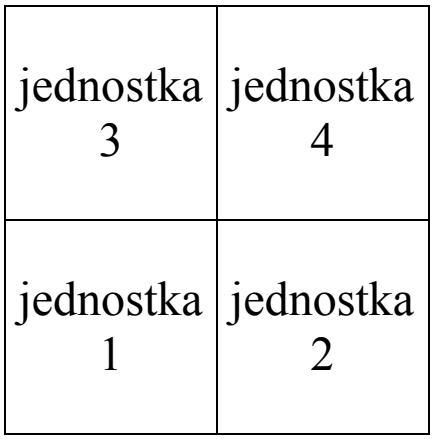
Przeprowadzone badania potwierdzają skuteczność zastosowania sieci Hopfielda podczas rozmieszczania modułów w układach VLSI. Wadą przedstawionego rozwiązania jest jednak bardzo szybki wzrost liczby neuronów oraz połączeń między neuronami, który występuje wraz ze wzrostem liczby rozmieszczanych modułów. W niniejszym rozdziale przedstawiono oryginalne rozwiązanie, polegające na zastosowaniu sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, który umożliwia minimalizację długości połączeń w układach VLSI. W podrozdziale 8.1 scharakteryzowano różne rozwiązania, które są stosowane podczas rozmieszczania modułów z użyciem równoległego przetwarzania. Zastosowanie sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, który umożliwia minimalizację długości połączeń w układzie VLSI, przedstawiono w podrozdziale 8.2. W podrozdziale 8.3 zamieszczono wyniki rozmieszczania otrzymane z zastosowaniem równoległego przetwarzania. Podsumowanie wyników przeprowadzonych symulacji zamieszczono w podrozdziale 8.4.

8.1. Rozmieszczanie modułów z wykorzystaniem równoległego przetwarzania

Rozmieszczanie modułów z użyciem równoległego przetwarzania (ang. parallel placement) jest stosowane od wielu lat podczas projektowania topografii układów VLSI. Równoległe przetwarzanie umożliwia znaczne skrócenie czasu potrzebnego na wykonanie obliczeń, ponieważ obliczenia są prowadzone przez wiele pracujących równocześnie jednostek przetwarzających. Równoległość obliczeń jest realizowana w wyniku podziału zadań lub podziału rozmieszczanych modułów między wszystkie jednostki przetwarzające systemu. Podział zadań polega na wykonywaniu przez poszczególne jednostki przetwarzające tylko ściśle określonych zadań, w których dana jednostka przetwarzająca jest wyspecjalizowana. Najczęściej podczas rozmieszczania modułów jest stosowany podział rozmieszczanych modułów między różne jednostki przetwarzające. Moduły znajdujące się w poszczególnych częściach układu są przydzielane do różnych jednostek przetwarzających. W każdej części układu jednostki przetwarzające

równocześnie wykonują rozmieszczanie modułów, wymieniając między sobą informacje dotyczące położenia poszczególnych modułów układu. Stosowane są również rozwiązania, w których każda jednostka przetwarzająca wykonuje oddzielnie rozmieszczanie modułów w całym układzie. W powyższym rozwiązaniu, w określonych odstępach czasu, spośród wszystkich rozwiązań jest wybierane jedno rozwiązanie, które jest następnie dalej oddzielnie przetwarzane przez wszystkie jednostki przetwarzające. Jednostki przetwarzające mogą stosować różne metody rozmieszczania modułów: algorytm zamiany parami, symulowane wyżarzanie, metody analityczne, strategie ewolucyjne [3, 119, 123-124, 139-147].

Podczas rozmieszczania modułów z użyciem równoległego przetwarzania z podziałem rozmieszczanych modułów, moduły znajdujące się w poszczególnych częściach układu są przydzielane do różnych jednostek przetwarzających. Rysunek 32 przedstawia przykład podziału układu między różne jednostki przetwarzające.



Rys. 32. Przykład podziału układu między różne jednostki przetwarzające

Przydział modułów do poszczególnych części układu i jednostek przetwarzających nie może być stały, ponieważ spowodowałoby to znaczne pogorszenie jakości otrzymywanych rozwiązań. Z tego względu w algorytmie przetwarzania jednostek dopuszcza się możliwość przenoszenia modułów do innych części układu. Innym rozwiązaniem jest dynamiczna zmiana przydziału modułów do różnych części układu w trakcie przetwarzania. Powyższe rozwiązania umożliwiają znaczną poprawę jakości otrzymywanych rozwiązań. Równoczesne rozmieszczanie modułów przez wszystkie jednostki przetwarzające jest jednak obarczone błędem. Jednostki przetwarzające podejmują decyzje o rozmieszczeniu modułów w swoich częściach, na podstawie rozmieszczenia modułów w innych częściach układu, które może ulec zmianie w danej iteracji programu jednostki przetwarzającej. W związku z tym, zmiany rozmieszczenia modułów wykonane przez jednostki przetwarzające w różnych częściach układu, często mogą kolidować ze sobą wzajemnie i mogą powodować wzrost całkowitego kosztu rozmieszczenia wszystkich modułów. Ponadto, jednostki przetwarzające często wykonują rozmieszczenie modułów na podstawie położenia modułów w innych częściach

układu, które nie jest już aktualne. Powodem tego są ograniczone możliwości komunikacji między jednostkami przetwarzającymi. W związku z tym, podczas rozmieszczania modułów z wykorzystaniem równoległego przetwarzania często można zaobserwować oscylacje funkcji całkowitego kosztu rozmieszczenia wszystkich modułów. Oscylacje funkcji całkowitego kosztu rozmieszczenia modułów są powodem pogorszenia jakości otrzymywanych rozwiązań [3, 119, 123, 139-147].

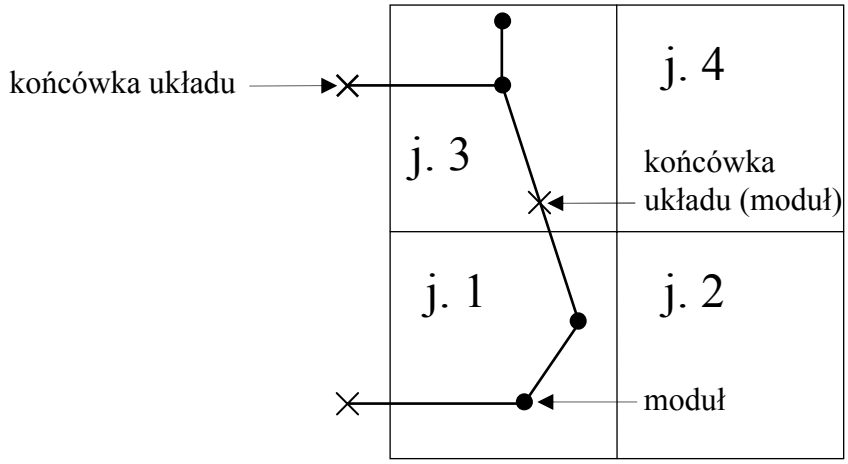
8.2. Zastosowanie sieci Hopfielda jako jednostki przetwarzającej

Wadą sieci Hopfielda w przypadku zastosowania sieci do rozwiązywania problemów optymalizacyjnych jest szybko wzrastająca liczba neuronów oraz połączeń między neuronami wraz ze wzrostem wymiaru problemu. W przypadku rozmieszczania X modułów na podłożu o rozmiarach K i L jednostek, liczba neuronów $N = X\,K\,L$, natomiast liczba połączeń między neuronami wynosi $X^2K^2L^2$. Argumentem przeciwko stosowaniu sieci Hopfielda do rozwiązywania problemów optymalizacyjnych była również niska jakość otrzymywanych rozwiązań. Badania przeprowadzone w poprzednich rozdziałach wskazują, że dla problemu minimalizacji długości połączeń, sieć Hopfielda umożliwia otrzymanie rozwiązań o dobrej jakości, szczególnie dla układów o niewielkim rozmiarze.

Powyższe obserwacje stały się motywacją do zbadania skuteczności zastosowania sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, którego celem jest minimalizacja długości połączeń w układzie. Zastosowanie sieci Hopfielda do rozmieszczania niewielkiej liczby modułów umożliwia otrzymanie bardzo dobrych rezultatów, a liczba neuronów i połączeń między neuronami nie jest duża. Podział układu na odpowiednio małe części, umożliwia zatem efektywne zastosowanie sieci Hopfielda jako jednostki przetwarzającej dla poszczególnych części układu. W dodatku podczas symulacji, sieć (jednostka przetwarzająca) może wielokrotnie wykorzystywać informację o położeniu modułów w innych częściach układu, przed ostatecznym ustaleniem rozwiązania. W różnych ośrodkach naukowych są badane rozwiązania, w których liczba jednostek przetwarzających, wykorzystujących algorytm zamiany parami lub symulowanego wyżarzania, jest nawet równa liczbie rozmieszczanych modułów w układzie. Celem powyższych badań jest sprzętowe rozwiązanie problemu rozmieszczania modułów, które może być zastosowane w układach rekonfigurowalnych [123].

Sieć Hopfielda można zastosować do rozmieszczania modułów z użyciem równoległego przetwarzania. W tym celu należy podzielić układ na wystarczająco małe części, dla których można zastosować jednostki przetwarzające w postaci sieci Hopfielda. Liczba neuronów N danej jednostki przetwarzającej jest równa $X\,K\,L$, gdzie X jest liczbą modułów położonych w części układu odpowiadającej danej jednostce przetwarzającej, natomiast K i L są rozmiarami danej części układu. Na wartości wag połączeń między neuronami danej jednostki przetwarzającej mają wpływ jedynie moduły

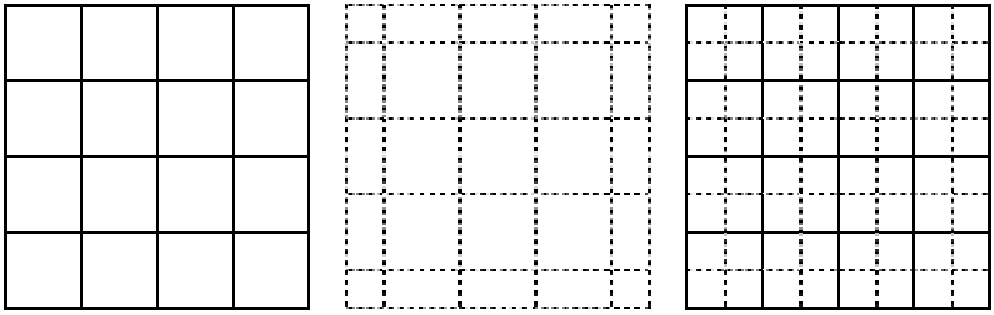
położone w części układu odpowiadającej danej jednostce przetwarzającej. Połączenia modułów danej jednostki przetwarzającej z modułami położonymi w pozostałych częściach układu są traktowane jak połączenia z końcówkami całego układu, które są położone w środku odcinka łączącego obydwa moduły. Rysunek 33 przedstawia sposób interpretacji połączeń między modułami położonymi w różnych częściach układu. W przypadku połączeń między modułami znajdującymi się w różnych częściach układu, połowa kosztu połączenia jest w równym stopniu uwzględniana w funkcji kosztu rozmieszczenia modułów obydwu jednostek przetwarzających. Połączenia modułów danej jednostki przetwarzającej z modułami położonymi w pozostałych częściach układu oraz końcówkami całego układu mają zatem jedynie wpływ na wartość zewnętrznych sygnałów wejściowych neuronów danej jednostki przetwarzającej. W trakcie symulacji sieci Hopfielda wszystkich jednostek przetwarzających, zmiana położenia modułów znajdujących się w pozostałych częściach układu wymaga aktualizacji wartości sygnałów wejściowych neuronów danej części układu. Powyższe rozwiązanie umożliwia równoczesne rozmieszczanie modułów przez wszystkie jednostki przetwarzające, które w trakcie przetwarzania uwzględniają zmiany położenia modułów w pozostałych częściach układu. Podział układu na części wymaga wyznaczenia wartości wag połączeń między neuronami oraz zewnętrznych sygnałów wejściowych neuronów sieci Hopfielda, które pełnią funkcje jednostek przetwarzających poszczególnych części układu. W tym celu są stosowane wzory 26, 40, 43, 45 oraz 47-48, ze względu na minimalizację długości połączeń w częściach układu z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek. Sieć Hopfielda każdej części układu minimalizuje własną funkcję kosztu. Całkowity koszt rozmieszczenia wszystkich modułów jest równy sumie kosztów rozmieszczenia modułów w poszczególnych częściach układu.



Rys. 33. Interpretacja połączeń między modułami położonymi w różnych częściach układu

Przydział modułów do danej części układu z reguły nie jest przydziałem optymalnym. Z tego względu należało zapewnić możliwość zmiany przydziału

modułów do poszczególnych części układu. W tym celu zastosowano dwa sposoby podziału układu. Podział układu jest zmieniany na przeciwny, po każdym ustaleniu się rezultatu rozmieszczenia modułów dla danego podziału. Części układu dla obydwu sposobów podziału układu zachodzą na siebie wzajemnie. Zmiana sposobu podziału układu zapewnia więc możliwość przemieszczania się modułów po całej powierzchni układu. Początkowe rozmieszczenie modułów może być wyznaczone losowo lub z użyciem innej metody rozmieszczania. Rysunek 34 przedstawia przykład dwóch sposobów podziału układu na części.



(a) (b) (c)

Rys. 34. Podział układu na części: a) pierwszy sposób, b) drugi sposób, c) obydwa sposoby razem

Zmiana sposobu podziału układu wymaga wyznaczenia wartości wag połączeń między neuronami oraz zewnętrznych sygnałów wejściowych neuronów sieci Hopfielda jednostek przetwarzających. Symulacja sieci Hopfielda w jednostkach przetwarzających odbywa się zgodnie ze wzorem 34. Poszczególne iteracje są wykonywane kolejno dla wszystkich jednostek przetwarzających, aż do ustalenia się rozwiązania we wszystkich jednostkach. Po każdej iteracji jest możliwa aktualizacja zewnętrznych sygnałów wejściowych neuronów, na podstawie bieżącego położenie modułów, które znajdują się w innych częściach układu. Podczas aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów, położenie modułów w innych częściach układu wyznacza neuron o największej wartości sygnału wyjściowego.

W programie symulującym pracę jednostek przetwarzających zastosowano szereg nowych rozwiązań. Wprowadzono zmianę umożliwiającą otrzymywanie tylko prawidłowych rozwiązań, tzn. rozwiązań, które uwzględniają wszystkie ograniczenia. Jeżeli rozwiązanie otrzymane przez jednostkę przetwarzającą jest nieprawidłowe (nie uwzględnia ograniczeń 1 i 2 z podrozdziału 4.3), w części układu odpowiadającej danej jednostce przetwarzającej, jest przywracane początkowe rozmieszczenie modułów, tzn. rozmieszczenie modułów występujące po ostatniej zmianie podziału układu na części.

Symulacje przeprowadzone w poprzednich rozdziałach wskazują, że optymalna wartość stałej D we wzorach 26 i 40 zależy od udziału długości połączeń w funkcji kosztu sieci Hopfielda. W rozpatrywanej metodzie rozmieszczania, jeżeli początkowe rozmieszczenie modułów jest wyznaczane w

sposób losowy, to udział długości połączeń w funkcji kosztu jest początkowo duży. Następnie, udział długości połączeń w funkcji kosztu jednostek przetwarzających stopniowo zmniejsza się, wraz z postępującą minimalizacją długości połączeń w układzie. W związku z tym, jest konieczne stopniowe zwiększanie wartości stałej D, po każdej zmianie sposobu podziału układu na części, aby umożliwić osiągnięcie rozwiązań o dobrej jakości.

W poprzednich rozdziałach stosowano rozwiązanie polegające na tym, że w stanie stabilnym sieci Hopfielda, sygnał wyjściowy neuronu był uważany za równy 1, jeżeli sygnał wyjściowy był większy lub równy 0,5. Jeżeli żaden z neuronów przydzielonych do danego modułu układu nie posiadał sygnału wyjściowego większego lub równego 0,5, to rozwiązanie było nieprawidłowe, ponieważ położenie modułu na podłożu układu nie było wtedy określone. Przeprowadzone symulacje wskazują, że jest możliwe wyznaczenie położenia modułu jedynie na podstawie znajomości neuronu o największym sygnale wyjściowym, a otrzymane rozwiązanie może być prawidłowe. Neurony odpowiadające danemu modułowi nie muszą posiadać wartości sygnału wyjściowego większego lub równego 0,5. Jeżeli żaden z neuronów odpowiadających dowolnemu modułowi układu, położonemu w dowolnej części układu, nie posiada sygnału wyjściowego większego lub równego 0,5, to powyższa sytuacja jest traktowana jedynie jako sygnał, że sieć pracuje na granicy między prawidłowymi i nieprawidłowymi rozwiązaniami. Podczas wyznaczania parametrów sieci Hopfielda dla następnego sposobu podziału układu na części, należy wtedy zmniejszyć wartość stałej D we wzorach 26 i 40. W przeciwnym przypadku, otrzymane rozwiązanie może być niepoprawne. Jeżeli powyższa sytuacja nie występuje w żadnej grupie neuronów przydzielonych do dowolnego modułu, położonego w dowolnej części układu, to wartość stałej D jest zwiększana. W przypadku, gdy dopuszcza się tylko niewielki wzrost wartości stałej D, rozwiązania otrzymywane przez jednostki przetwarzające bardzo rzadko są niepoprawnymi rozwiązaniami, które nie spełniają ograniczeń. Powyższe modyfikacje umożliwiają otrzymanie rozwiązania w wyniku mniejszej liczby zmian podziału układu na części.

W następnym podrozdziale zostaną szczegółowo opisane rezultaty rozmieszczania, otrzymane w wyniku zastosowania równoległego przetwarzania z użyciem sieci Hopfielda jako jednostki przetwarzającej.

8.3. Wyniki rozmieszczania otrzymane z użyciem równoległego przetwarzania

W niniejszym podrozdziale zostaną przedstawione rezultaty rozmieszczania modułów z minimalizacją długości połączeń dla 5 przykładów z ustalonym położeniem końcówek całego układu, w których węzły mogą posiadać dowolną liczbę końcówek. Każdy z przykładów został rozwiązany z zastosowaniem równoległego przetwarzania, z jednostkami przetwarzającymi w postaci sieci Hopfielda.

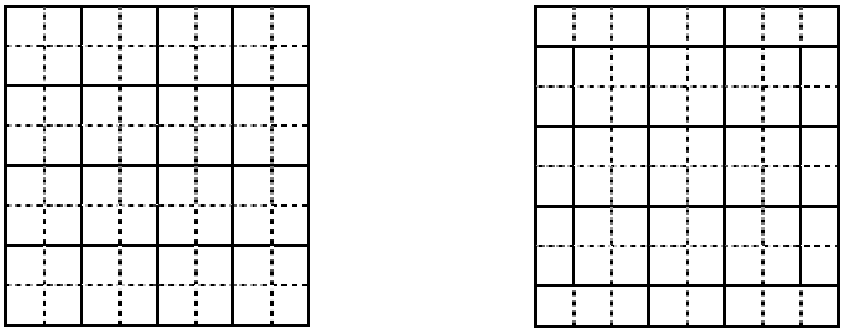
Wszystkie przykłady są układami FPGA [137]. Lista połączeń układów została utworzona w taki sam sposób jak w podrozdziale 7.3. Wszystkie układy

zostały zaimplementowane z wykorzystaniem komórek CLB, składających się z 4-wejściowej tablicy LUT oraz przerzutnika D. Tabela 23 zawiera informacje dotyczące badanych układów: nazwę układu, liczbę modułów (komórek CLB), liczbę wejść całego układu, liczbę wyjść całego układu, liczbę węzłów w układzie, liczbę końcówek w poszczególnych węzłach układu oraz rozmiary podłoża układu, na którym rozmieszczano moduły.

Nazwa układu	Liczba modułów	Liczba wejść układu	Liczba wyjść układu	Liczba węzłów	Liczba końcówek węzłów	Rozmiary podłoża
term1	54	34	10	88	2÷11	8x8
9symml	70	9	1	79	2÷23	9x9
apex7	77	49	37	126	2÷39	11x11
c1355	104	41	32	145	2÷23	11x11
c880	114	60	26	174	2÷25	11x11

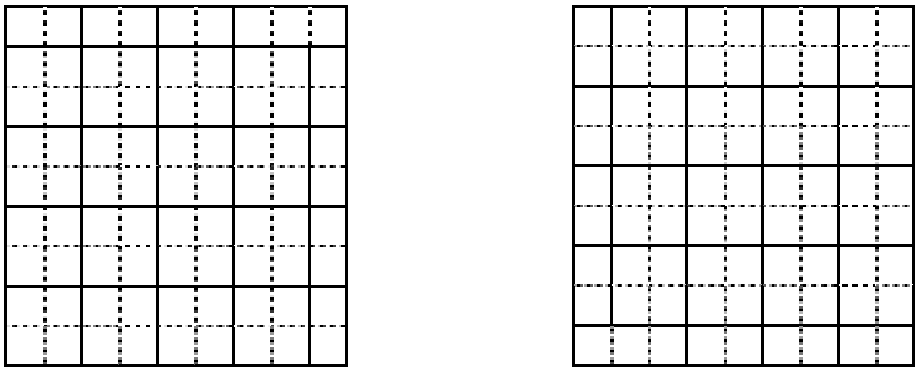
Tabela 23. Dane poszczególnych przykładów

Podczas rozmieszczania modułów zastosowano podział podłoża układu na części o maksymalnym rozmiarze 2x2 jednostki, zgodnie z rozwiązaniem przedstawionym w poprzednim podrozdziale. Rysunki 35-37 przedstawiają sposoby podziału podłoża układu w badanych przykładach. Linie ciągłe oznaczają granice poszczególnych części układu, liniami przerywanymi oznaczono jednostkową siatkę podłoża układu. Obydwa sposoby podziału układu na części utworzono w wyniku podziału układu regularną siatką o rozmiarach 2x2 jednostki. W drugim sposobie podziału, siatka podziału jest przesunięta o jedną jednostkę w kierunku poziomym i pionowym (połowa rozmiaru siatki). Powyższe rozwiązanie gwarantuje, że części układu dla obydwu sposobów podziału zachodzą na siebie wzajemnie. Części układu o rozmiarze 1x1 jednostkę są dołączane do sąsiedniej części układu.



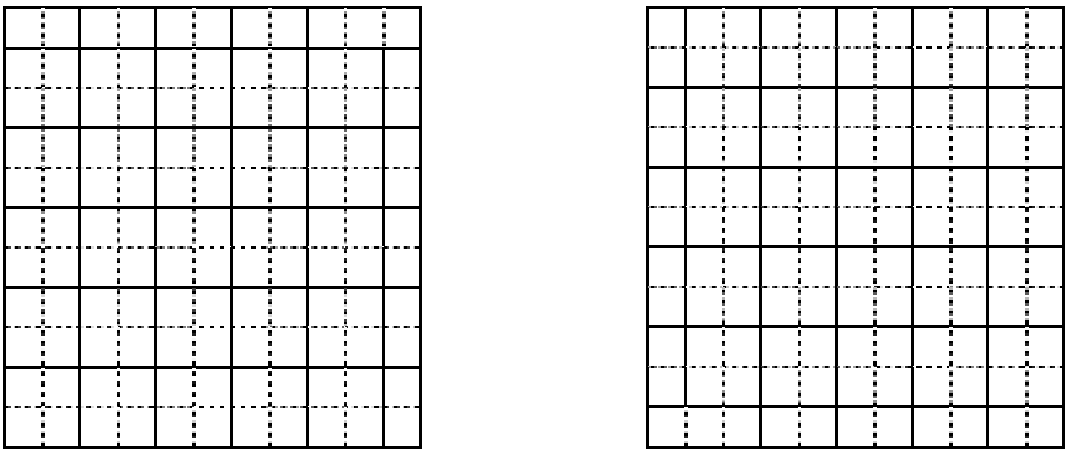
(a) (b)

Rys. 35. Podział podłoża układu o rozmiarach 8x8 jednostek na części: a) pierwszy sposób, b) drugi sposób



(a) (b)

Rys. 36. Podział podłoża układu o rozmiarach 9x9 jednostek na części: a) pierwszy sposób, b) drugi sposób

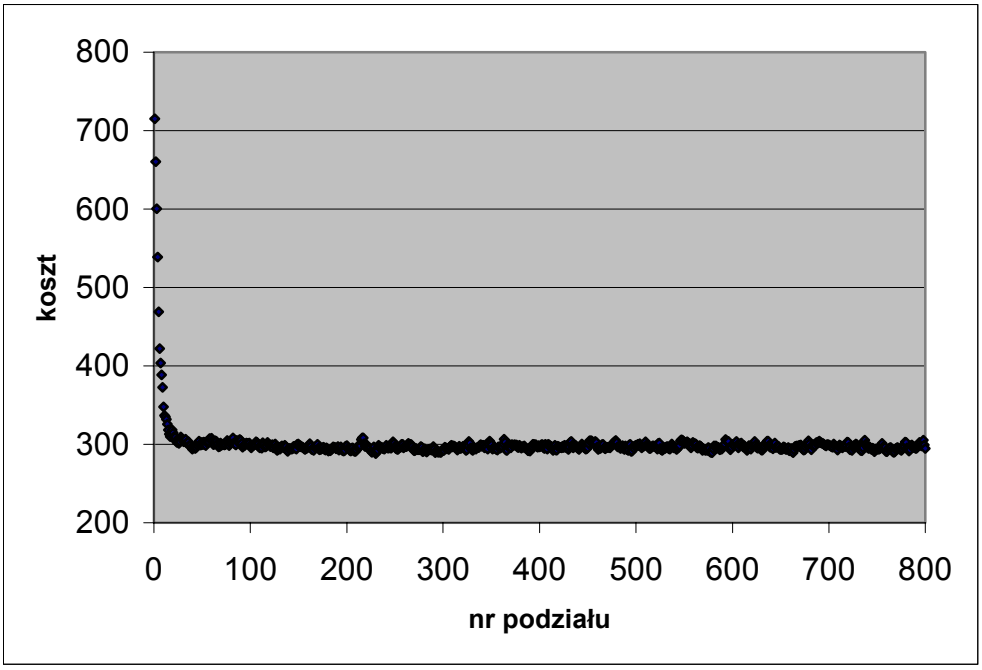


(a) (b)

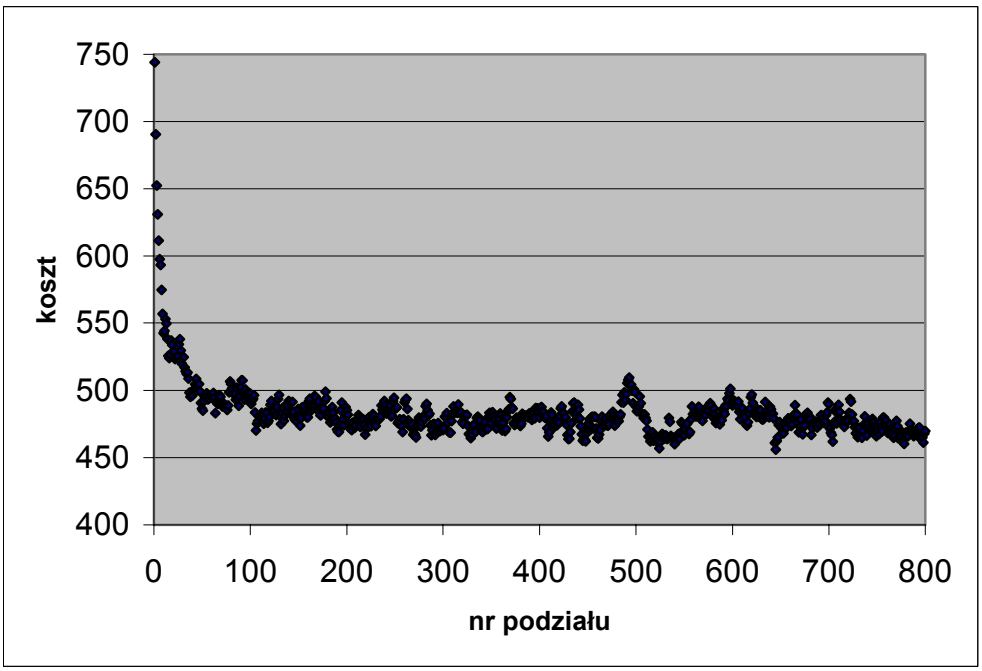
Rys. 37. Podział podłoża układu o rozmiarach 11x11 jednostek na części: a) pierwszy sposób, b) drugi sposób

Dla wszystkich przykładów wykonano minimalizację długości połączeń, z użyciem równoległego przetwarzania, z jednostkami przetwarzającymi w postaci sieci Hopfielda. Symulacje przeprowadzono z wykorzystaniem programu napisanego w języku C++. W programie zastosowano wszystkie rozwiązania opisane w poprzednim podrozdziale. Symulację sieci Hopfielda jednostek przetwarzających przeprowadzono w oparciu o wzór 34, ponieważ dla układów o niewielkim wymiarze, klasyczna sieć Hopfielda umożliwia otrzymanie rozwiązań o dobrej jakości. Jednostki przetwarzające w każdej iteracji symulacji pracy jednostek aktualizowały położenie modułów znajdujących się w pozostałych częściach układu, w celu aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów jednostek przetwarzających. Zastosowano inicjalizację neuronów wartościami sygnałów wejściowych bliskimi 0 (-0,01÷0,01), ponieważ taki sposób inicjalizacji neuronów umożliwił otrzymanie bardzo dobrych rozwiązań w badaniach przeprowadzonych w poprzednich rozdziałach. Początkowe rozmieszczenie modułów w układzie wyznaczono w sposób losowy. W symulacjach zastosowano następujące wartości stałych ze wzorów 26 i 40 dla sieci Hopfielda jednostek przetwarzających: $A = 5$, $B = 6$, $C = 0{,}5$. Początkowa wartość stałej D wynosiła

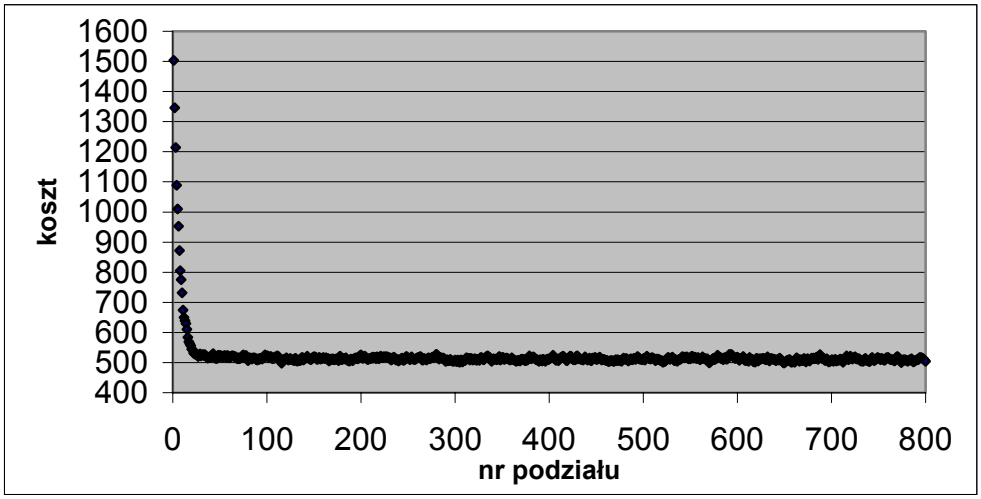
0,1, po każdej zmianie sposobu podziału układu na części, następowała zmiana wartości stałej *D* o ±0,01. We wszystkich przykładach wejścia i wyjścia całego układu (końcówki całego układu) posiadały ustalone położenie, które zostało określone w taki sam sposób, jak w podrozdziale 7.3. Dla wszystkich układów wykonano 800 zmian sposobu podziału układu na części. Dla każdego podziału układu na części, jednostki przetwarzające wykonały rozmieszczenie modułów. Rysunki 38-42 przedstawiają estymowaną długość połączeń po każdym podziale układu na części dla wszystkich badanych układów.



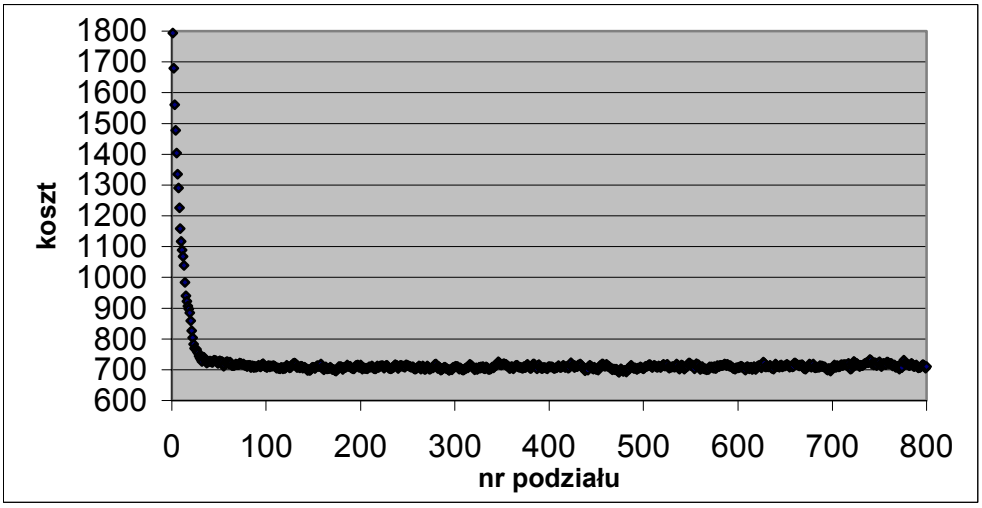
Rys. 38. Wykres zmiany kosztu rozwiązania dla układu *term1*



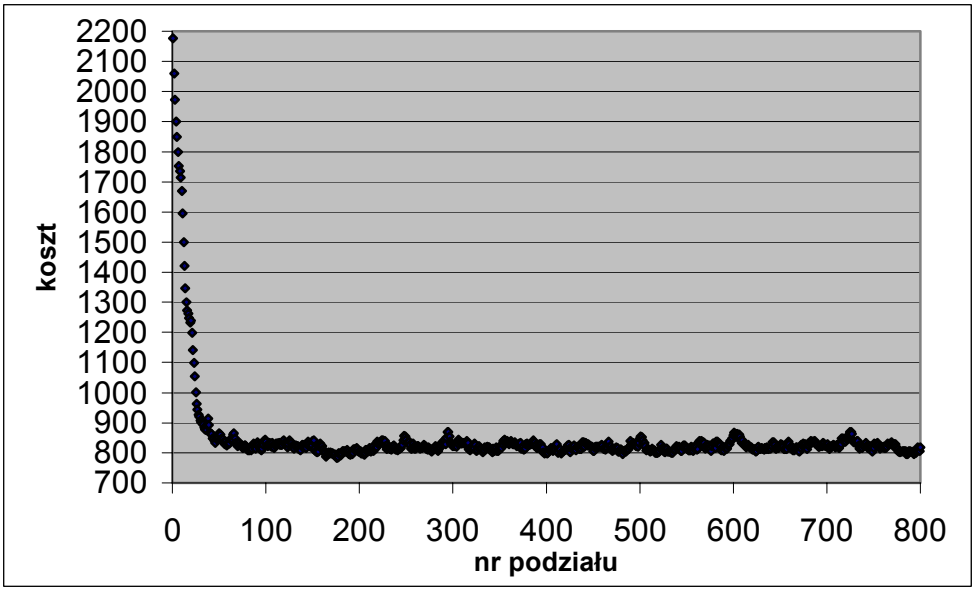
Rys. 39. Wykres zmiany kosztu rozwiązania dla układu *9symml*



Rys. 40. Wykres zmiany kosztu rozwiązania dla układu *apex7*

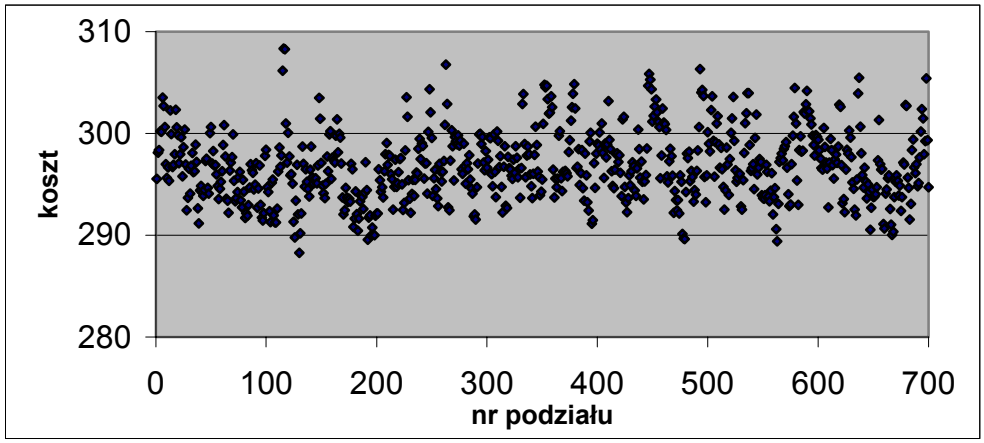


Rys. 41. Wykres zmiany kosztu rozwiązania dla układu *c1355*

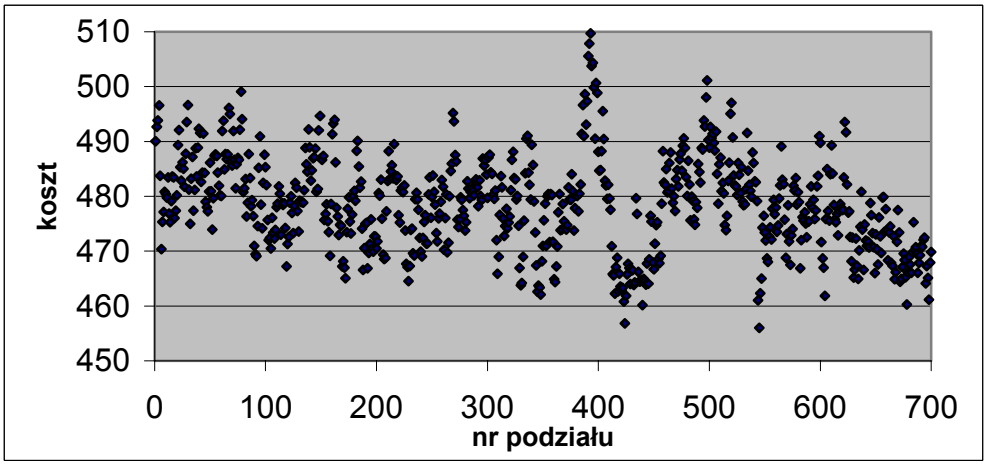


Rys. 42. Wykres zmiany kosztu rozwiązania dla układu *c880*

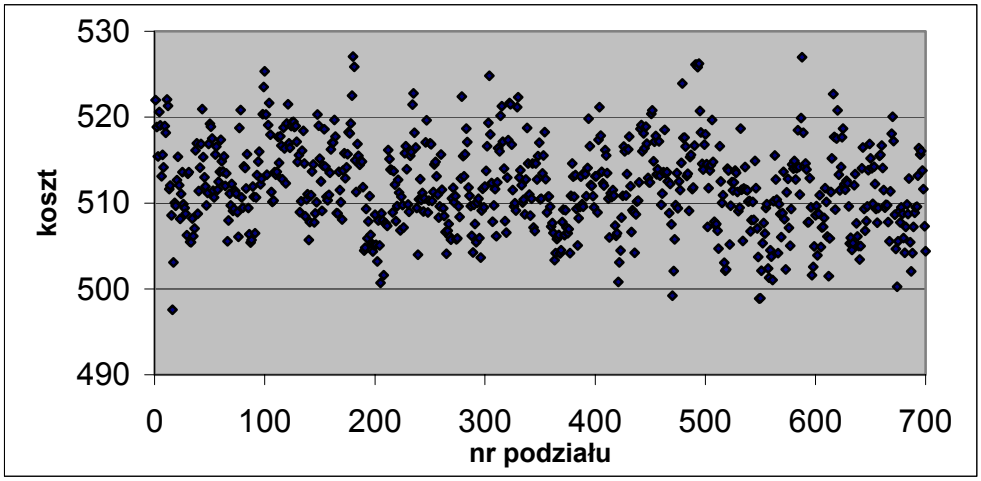
Analiza rysunków 38-42 pozwala zauważyć, że koszt początkowego losowego rozmieszczenia modułów bardzo szybko maleje podczas początkowych podziałów układu na części. Po wykonaniu określonej liczby podziałów, średnia wartość kosztu otrzymywanych rozwiązań w kolejnych podziałach ulega stabilizacji. Dla badanych układów po wykonaniu około 100 podziałów, średnia wartość otrzymywanych rozwiązań nie ulega istotnym zmianom. Można jednak zauważyć fluktuacje kosztu otrzymywanych rozwiązań wokół wartości średniej. Fluktuacje kosztu otrzymywanych rozwiązań wynikają z zastosowania równoległego przetwarzania. Jako jednostki przetwarzające zastosowano sieć Hopfielda, która umożliwia otrzymywanie różnych rozwiązań. Koszt rozmieszczenia modułów w części układu odpowiadającej danej sieci Hopfielda może wzrosnąć w porównaniu do kosztu bieżącego rozwiązania, zwłaszcza w przypadku, gdy całkowity koszt rozwiązania jest zbliżony do kosztu rozwiązania optymalnego. Z tego względu może występować niewielki, chwilowy wzrost całkowitego kosztu otrzymywanych rozwiązań. Rysunki 43-47 przedstawiają fluktuacje kosztu otrzymywanych rozwiązań dla badanych układów. Rysunki 43-47 przedstawiają estymowaną długość połączeń dla otrzymanych rozwiązań, począwszy od podziału numer 101. Amplituda fluktuacji kosztu rozwiązań wynosi maksymalnie około 10% średniej wartości kosztu otrzymywanych rozwiązań.



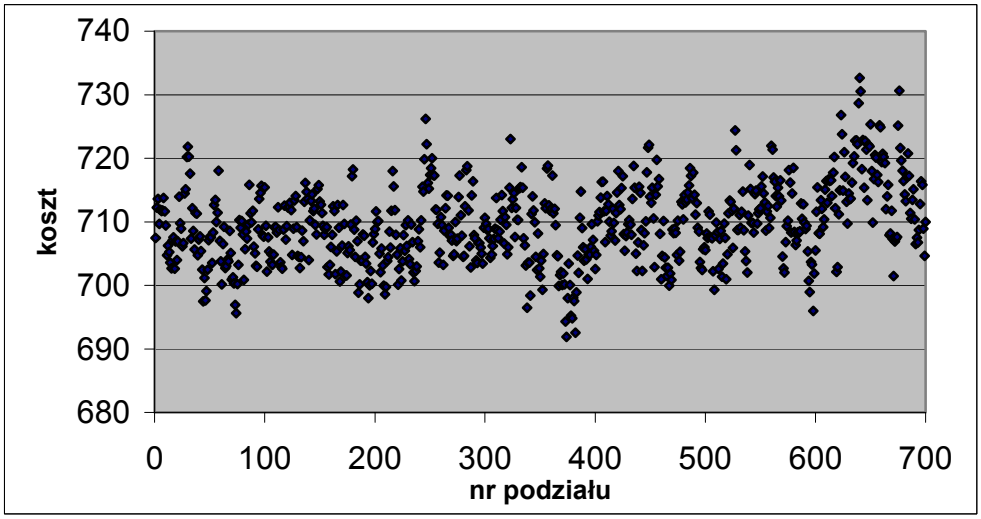
Rys. 43. Fluktuacje kosztu rozwiązania dla układu *term1*



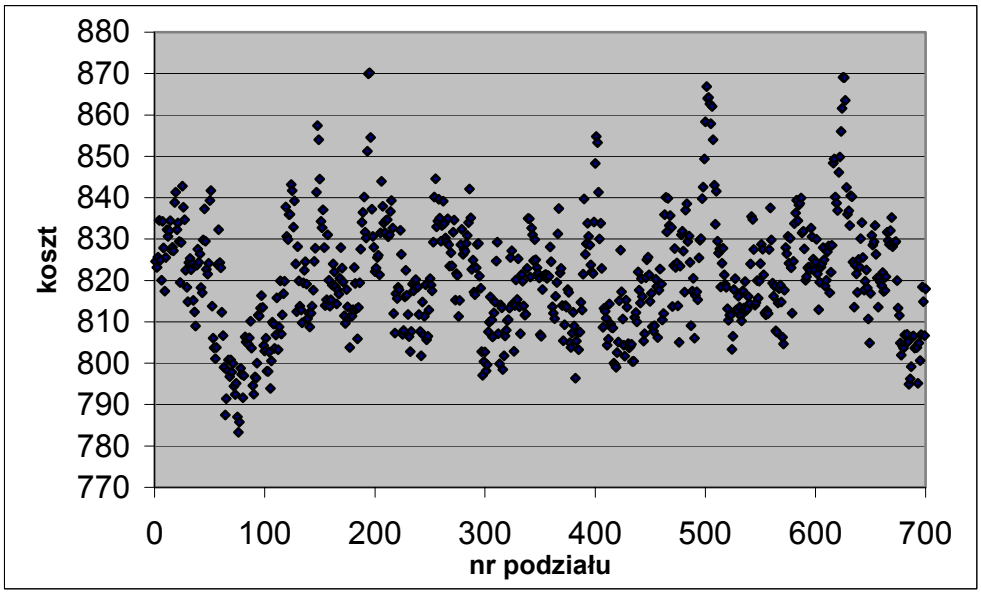
Rys. 44. Fluktuacje kosztu rozwiązania dla układu *9symml*



Rys. 45. Fluktuacje kosztu rozwiązania dla układu *apex7*

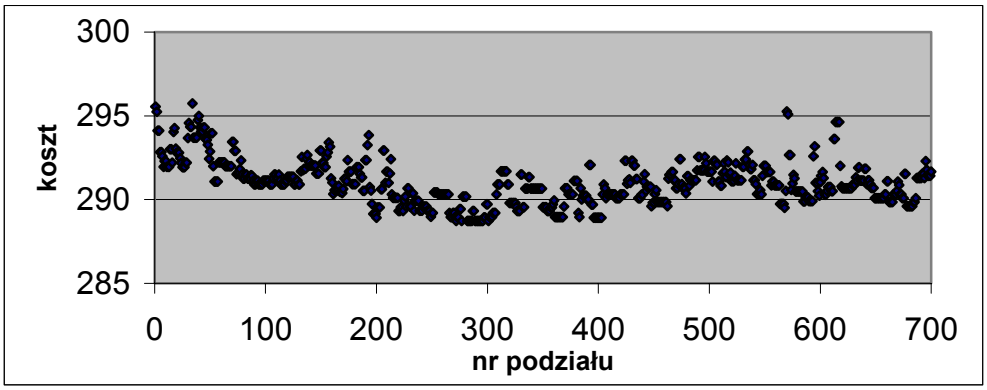


Rys. 46. Fluktuacje kosztu rozwiązania dla układu *c1355*

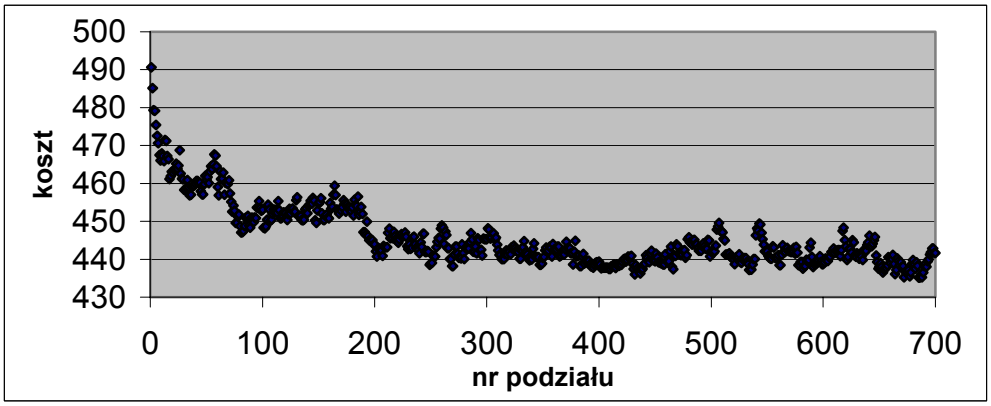


Rys. 47. Fluktuacje kosztu rozwiązania dla układu *c880*

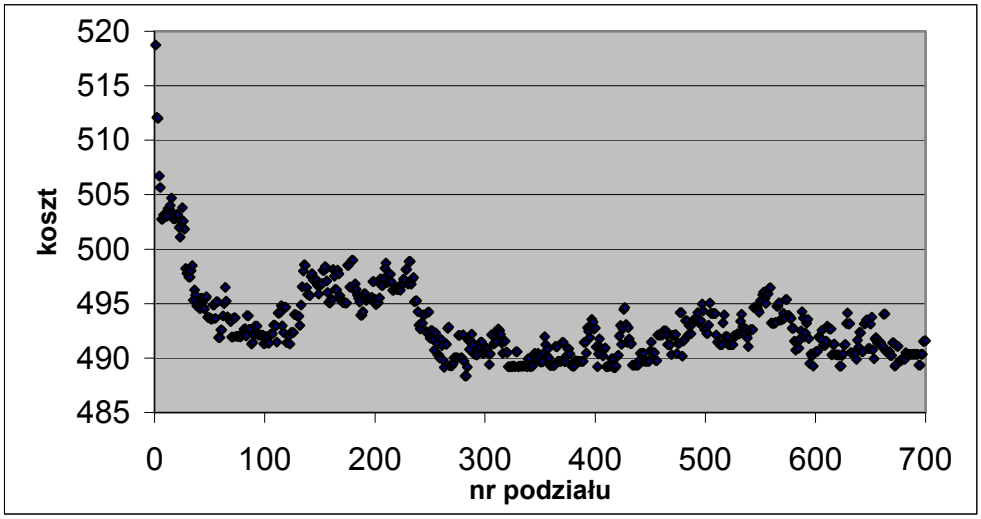
W celu zmniejszenia wpływu fluktuacji na jakość otrzymywanych rozwiązań zastosowano rozwiązanie, polegające na akceptacji rozwiązań, które umożliwiają zmniejszenie kosztu rozwiązania. Na początku jest wykonywanych 100 podziałów, w taki sam sposób jak poprzednio. W wyniku wykonania tych podziałów następuje stabilizacja średniego kosztu otrzymywanych rozwiązań. Następnie, poszczególne jednostki akceptują jedynie te rozwiązania, które lokalnie, w danej części układu powodują zmniejszenie kosztu rozwiązania. Decyzja jest podejmowana wyłącznie na podstawie kosztu rozwiązania danej części układu. Nie jest rozpatrywany sumaryczny koszt rozwiązań wszystkich jednostek przetwarzających, co wymagałoby oczekiwania przez wszystkie jednostki przetwarzające na wynik obliczeń oraz decyzję jednostki nadrzędnej systemu równoległego przetwarzania. W przypadku braku akceptacji nowego rozwiązania, w danej części układu jest przywracane poprzednie rozwiązanie. Podobne rozwiązanie, polegające na akceptacji jedynie rozwiązań, które zmniejszają koszt rozmieszczenia w danej części układu zastosowano w innych pracach [139, 148]. Istotną modyfikacją jest początkowa akceptacja wszystkich rozwiązań, co umożliwia ustabilizowanie się kosztu otrzymywanych rozwiązań, zapobiegając ewentualnemu osiągnięciu minimum lokalnego funkcji kosztu rozmieszczenia wszystkich modułów, któremu może odpowiadać znacznie większy koszt rozwiązania od kosztu rozwiązania optymalnego. Dla wszystkich układów wykonano 800 zmian sposobu podziału układu na części z zastosowaniem powyższego rozwiązania. Rysunki 48-52 przedstawiają estymowaną długość połączeń, począwszy od podziału numer 101.



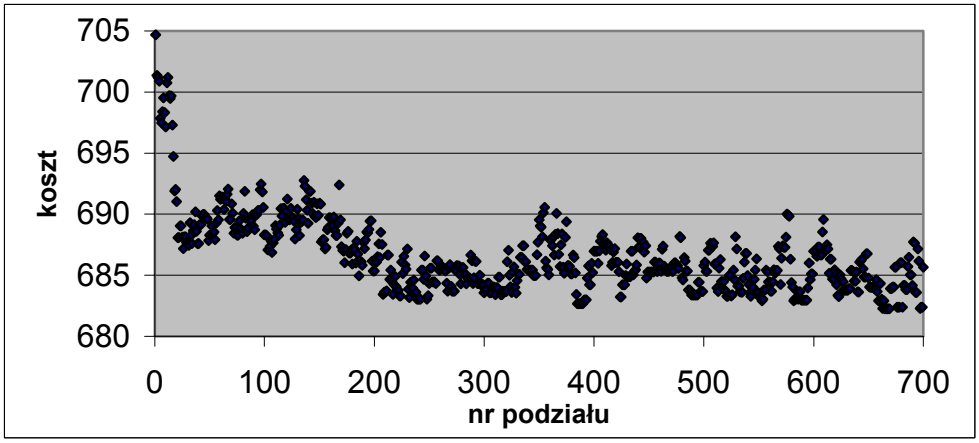
Rys. 48. Wykres zmiany kosztu rozwiązania dla układu *term1* po modyfikacji



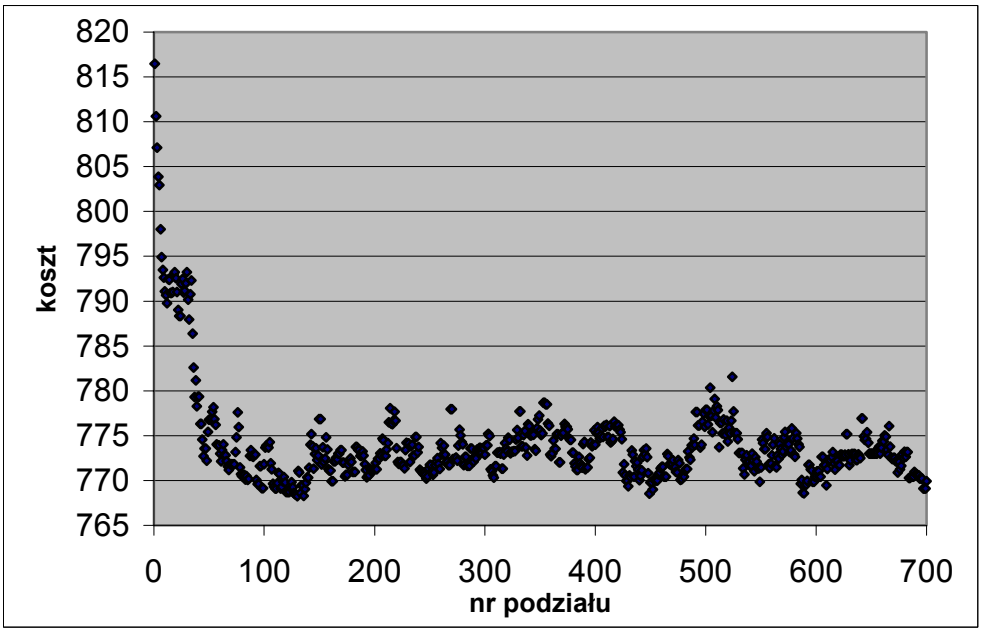
Rys. 49. Wykres zmiany kosztu rozwiązania dla układu *9symml* po modyfikacji



Rys. 50. Wykres zmiany kosztu rozwiązania dla układu *apex7* po modyfikacji



Rys. 51. Wykres zmiany kosztu rozwiązania dla układu *c1355* po modyfikacji



Rys. 52. Wykres zmiany kosztu rozwiązania dla układu *c880* po modyfikacji

W wyniku analizy rysunków 43-47 oraz 48-52 można zauważyć, że w przypadku akceptacji przez jednostki przetwarzające jedynie rozwiązań, które zmniejszają koszt rozwiązania danej jednostki przetwarzającej, fluktuacje całkowitego kosztu rozwiązań uległy znacznemu zmniejszeniu. Podczas kolejnych podziałów układu na części, średni koszt otrzymywanych rozwiązań uległ również stopniowemu zmniejszeniu, umożliwiając otrzymanie lepszych rozwiązań. Fluktuacje całkowitego kosztu rozwiązania wynikają z przyjętego rozwiązania, które wykorzystuje równoległe przetwarzanie. Jednostki przetwarzające akceptują jedynie rozwiązania, które z punktu widzenia danej jednostki przetwarzającej zmniejszają lokalny koszt rozwiązania. Pozostałe rozwiązania nie są akceptowane. W przypadku akceptacji nowego rozwiązania przez daną jednostkę przetwarzającą, kalkulacja zmiany kosztu rozwiązań pozostałych jednostek przetwarzających może być obarczona błędem. Błąd kalkulacji zmiany kosztu jednostek przetwarzających może być powodem chwilowego, niewielkiego wzrostu całkowitego kosztu rozwiązania. Usunięcie powyższych fluktuacji wymagałoby jednak wyznaczenia całkowitego kosztu rozwiązania przez jednostkę nadrzędną systemu równoległego przetwarzania, która podejmowałaby decyzję o akceptacji lub odrzuceniu rozwiązań wszystkich jednostek przetwarzających. Jednostki przetwarzające musiałyby oczekiwać na decyzję jednostki nadrzędnej, co znacznie wydłużyłoby czas obliczeń. Podobne fluktuacje całkowitego kosztu rozwiązań obserwowano w innych pracach [3, 123]. Błąd kalkulacji zmiany kosztu jednostek przetwarzających odgrywa decydującą rolę w momencie, gdy położenie modułów w układzie osiąga duży stopień uporządkowania, a średni koszt otrzymywanych rozwiązań jest zbliżony do kosztu optymalnego rozwiązania [139].

Tabela 24 przedstawia wyniki otrzymane w wyniku zastosowania opisanej modyfikacji. Dla wszystkich układów wykonano 800 zmian sposobu podziału układu na części. Podczas pierwszych 100 podziałów wszystkie rozwiązania były akceptowane. Następnie, jednostki przetwarzające akceptowały rozwiązania, które umożliwiły zmniejszenie kosztu rozwiązania danej jednostki przetwarzającej. Rozwiązania otrzymane dla podziałów 101÷800 zostały wcześniej graficznie przedstawione na rysunkach 48-52. W tabeli 24 są zebrane następujące wyniki: średnia estymowana długość połączeń dla rozmieszczeń otrzymanych we wszystkich 800 podziałach układu, najmniejsza estymowana długość połączeń dla powyższych rozwiązań oraz estymowana długość połączeń dla rozmieszczenia modułów otrzymanego z użyciem narzędzia rozmieszczania modułów programu VPR (rozmieszczenie wykonano zgodnie z opisem zamieszczonym na płycie CD - Dodatek). Z danych przedstawionych w tabeli 24 wynika, że równoległe przetwarzanie umożliwia otrzymanie rozwiązań, których jakość jest bardzo dobra. Estymowana długość połączeń dla najlepszych rozwiązań otrzymanych z zastosowaniem równoległego przetwarzania jest porównywalna lub nawet mniejsza, w porównaniu do estymowanej długości połączeń dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania programu VPR. Należy podkreślić, że średnia estymowana długość połączeń zamieszczona w tabeli 24 uwzględnia wszystkie rozwiązania, począwszy od losowego rozmieszczenia modułów. Z tego względu średnia estymowana długość połączeń nie może być porównana z estymowaną długością połączeń dla rozmieszczenia wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR. Analiza rysunków 48-52 pozwala jednak zauważyć, że po

ustabilizowaniu się kosztu otrzymywanych rozwiązań podczas początkowych podziałów układu, średnia estymowana długość połączeń dla otrzymywanych rozwiązań jest zbliżona do estymowanej długości połączeń dla rozmieszczenia wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR. Otrzymane rozwiązania należy uznać za bardzo dobre, ponieważ program VPR należy do programów zapewniających najwyższą jakość rozwiązań [104].

Nazwa układu	Wynik	Rezultat (estymacja)
term1	średnia długość połączeń	294,9542
	najmniejsza dł. poł. otrzyman. rozw.	288,7202
	dł. poł. dla rozm. programu VPR	285,1448
9symml	średnia długość połączeń	454,5210
	najmniejsza dł. poł. otrzyman. rozw.	435,1888
	dł. poł. dla rozm. programu VPR	442,1362
apex7	średnia długość połączeń	503,7622
	najmniejsza dł. poł. otrzyman. rozw.	488,3382
	dł. poł. dla rozm. programu VPR	494,2260
c1355	średnia długość połączeń	702,8630
	najmniejsza dł. poł. otrzyman. rozw.	682,2555
	dł. poł. dla rozm. programu VPR	688,7998
c880	średnia długość połączeń	803,6733
	najmniejsza dł. poł. otrzyman. rozw.	768,2418
	dł. poł. dla rozm. programu VPR	774,6796

Tabela 24. Wyniki otrzymane dla badanych układów z zastosowaniem akceptacji rozwiązań zmniejszających koszt

Podczas symulacji, których rezultaty zostały przedstawione w tabeli 24, jednostki przetwarzające aktualizowały położenie modułów znajdujących się w pozostałych częściach układu, w celu aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów jednostek przetwarzających. Aktualizacja odbywała się w każdej iteracji symulacji pracy jednostek dla danego sposobu podziału układu na części. Tabela 25 zawiera rezultaty otrzymane w przypadku braku aktualizacji położenia modułów znajdujących się w pozostałych częściach układu, w trakcie wykonywania iteracji danego podziału układu. W przypadku braku aktualizacji, wartość zewnętrznych sygnałów wejściowych neuronów jednostek jest wyznaczana na początku danego podziału układu na części, na podstawie położenia modułów, które zostało wyznaczone podczas poprzedniego podziału układu na części. Wartość zewnętrznych sygnałów wejściowych neuronów jest wtedy stała podczas wykonywania wszystkich iteracji danego podziału układu. Dla wszystkich układów wykonano 800 zmian sposobu podziału układu na części, w taki sam sposób jak poprzednio.

Nazwa układu	Wynik	Rezultat (estymacja)
term1	średnia długość połączeń	299,4241
	najmniejsza dł. poł. otrzyman. rozw.	289,5921
	dł. poł. dla rozm. programu VPR	285,1448
9symml	średnia długość połączeń	440,5858
	najmniejsza dł. poł. otrzyman. rozw.	429,0679
	dł. poł. dla rozm. programu VPR	442,1362
apex7	średnia długość połączeń	502,8861
	najmniejsza dł. poł. otrzyman. rozw.	490,5128
	dł. poł. dla rozm. programu VPR	494,2260
c1355	średnia długość połączeń	708,3567
	najmniejsza dł. poł. otrzyman. rozw.	691,6721
	dł. poł. dla rozm. programu VPR	688,7998
c880	średnia długość połączeń	804,6892
	najmniejsza dł. poł. otrzyman. rozw.	773,3495
	dł. poł. dla rozm. programu VPR	774,6796

Tabela 25. Wyniki otrzymane dla badanych układów w przypadku braku aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów w trakcie wykonywania iteracji danego podziału układu

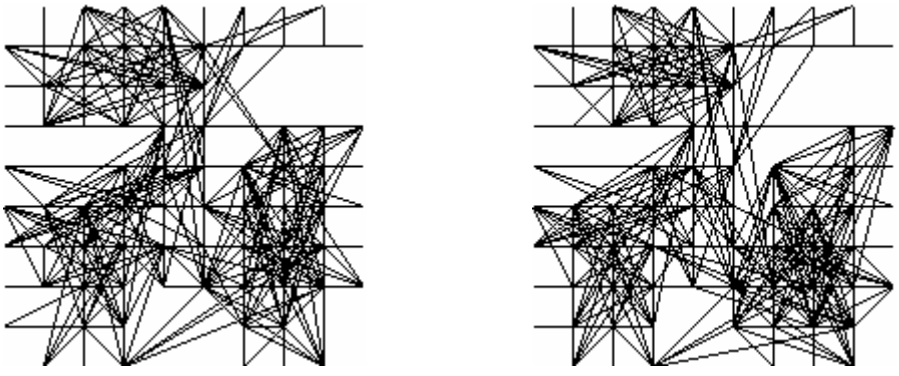
Z rezultatów zamieszczonych w tabeli 25 wynika, że brak aktualizacji położenia modułów znajdujących się w pozostałych częściach układu w trakcie wykonywania danego podziału układu (brak aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów jednostek przetwarzających), nie powoduje istotnego pogorszenia jakości otrzymywanych rozwiązań. Średnia estymowana długość połączeń dla 800 podziałów układu oraz najmniejsza estymowana długość połączeń dla otrzymanych rozwiązań jest zbliżona do wcześniejszych rezultatów, uzyskanych w wyniku pełnej aktualizacji wartości zewnętrznych sygnałów wejściowych neuronów. Powyższa właściwość jest bardzo istotna, ponieważ umożliwia otrzymywanie rozwiązań o zbliżonej jakości, przy znacznie mniejszej ilości informacji, które muszą być wymieniane między jednostkami przetwarzającymi poszczególnych części układu. Tabela 26 zawiera rezultaty uzyskane dla najlepszych rozmieszczeń otrzymanych z użyciem równoległego przetwarzania bez aktualizacji położenia modułów znajdujących się w pozostałych częściach układu, w trakcie wykonywania danego podziału układu. Dla wszystkich badanych rozmieszczeń wykonano trasowanie połączeń z użyciem narzędzia trasowania połączeń programu VPR (zgodnie z opisem zamieszczonym na płycie CD – Dodatek). W tabeli 26 zamieszczono rezultaty trasowania dla 5 najlepszych rozwiązań otrzymanych z użyciem równoległego przetwarzania, dla ostatniego rozwiązania otrzymanego w serii 800 podziałów układu na części oraz dla rozmieszczenia otrzymanego z użyciem narzędzia rozmieszczania programu VPR. Dla wszystkich rozwiązań wykonano również trasowanie połączeń z użyciem aproksymacji minimalnego drzewa Steinera, przedstawionej w podrozdziale 7.2.1. W tabeli 26 zamieszczono następujące wyniki: estymowaną długość połączeń, długość połączeń wyznaczoną w wyniku zastosowania aproksymacji minimalnego drzewa Steinera, długość połączeń po

wykonaniu trasowania połączeń z użyciem narzędzia trasowania programu VPR oraz minimalną pojemność kanałów, niezbędną do poprowadzenia wszystkich połączeń z użyciem narzędzia trasowania połączeń programu VPR.

Rozmieszczenie		Estymowana długość połączeń	Długość połączeń (min. drzewo Steinera)	Długość połączeń (VPR)	Minimalna pojemność kanału (VPR)
term1	równ. prz. 1	289,5921	282	439	5
	równ. prz. 2	289,8383	283	433	5
	równ. prz. 3	290,0981	282	444	5
	równ. prz. 4	290,3354	282	454	4
	równ. prz. 5	290,3443	283	419	4
	równ. prz. ost.	289,5921	282	439	5
	VPR	285,1448	275	408	5
9symml	równ. prz. 1	429,0679	436	658	5
	równ. prz. 2	429,3665	433	642	6
	równ. prz. 3	429,3896	435	670	6
	równ. prz. 4	429,3976	432	665	6
	równ. prz. 5	429,4116	435	662	6
	równ. prz. ost.	429,4903	437	682	5
	VPR	442,1362	438	669	6
apex7	równ. prz. 1	490,5128	459	644	5
	równ. prz. 2	490,5452	457	642	5
	równ. prz. 3	490,6018	458	678	4
	równ. prz. 4	490,6091	456	640	5
	równ. prz. 5	490,6890	458	698	5
	równ. prz. ost.	491,3798	458	668	5
	VPR	494,2260	459	689	5
c1355	równ. prz. 1	691,6721	673	969	6
	równ. prz. 2	692,0024	672	946	6
	równ. prz. 3	692,1779	674	976	6
	równ. prz. 4	692,2365	674	979	6
	równ. prz. 5	692,2605	673	972	5
	równ. prz. ost.	692,5668	673	977	5
	VPR	688,7998	663	967	6
c880	równ. prz. 1	773,3495	750	1110	6
	równ. prz. 2	773,6247	748	1044	6
	równ. prz. 3	773,9940	753	1083	6
	równ. prz. 4	774,0850	752	1068	6
	równ. prz. 5	774,1084	756	1100	6
	równ. prz. ost.	776,0803	757	1097	6
	VPR	774,6796	739	1086	6

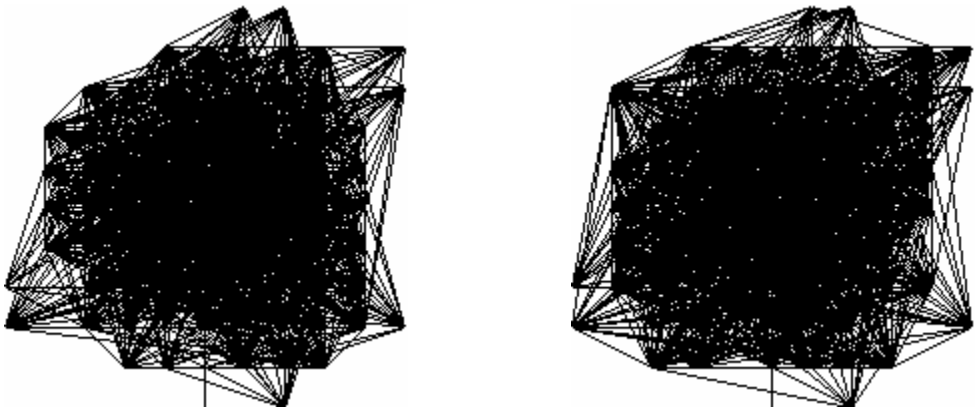
Tabela 26. Wyniki trasowania otrzymane dla najlepszych rozwiązań

Rezultaty zawarte w tabeli 26 są dowodem na to, że równoległe przetwarzanie umożliwia otrzymanie bardzo dobrych rozwiązań. Długości połączeń po wykonaniu trasowania połączeń z użyciem narzędzia trasowania programu VPR dla rozmieszczeń modułów wyznaczonych z użyciem równoległego przetwarzania oraz narzędzia rozmieszczania modułów programu VPR - są porównywalne. Niektóre rozmieszczenia otrzymane z użyciem równoległego przetwarzania umożliwiają nawet połączenie modułów z użyciem mniejszej długości połączeń. Rozmieszczenia otrzymane z użyciem równoległego przetwarzania wymagają kanałów, których pojemność jest taka sama lub mniejsza o 1. Porównywalne wyniki otrzymano nie tylko dla najlepszych rozwiązań uzyskanych z użyciem równoległego przetwarzania. W tabeli 26 zamieszczono również rezultaty dla ostatniego rozwiązania, otrzymanego w serii 800 podziałów układu na części. Powyższe rozwiązania rozpatrzono w celu zbadania wpływu fluktuacji kosztu otrzymywanych rozwiązań na jakość rozmieszczenia. Dla powyższych rozwiązań otrzymano zbliżone rezultaty podczas trasowania połączeń. Estymowana długość połączeń oraz długość połączeń aproksymacji minimalnego drzewa Steinera, dla rozmieszczeń modułów wyznaczonych z użyciem równoległego przetwarzania oraz narzędzia rozmieszczania modułów programu VPR - są porównywalne, co potwierdza dokładność zastosowanej metody estymacji długości połączeń. Rozbieżność między estymowaną długością połączeń, długością połączeń aproksymacji minimalnego drzewa Steinera oraz długością połączeń wyznaczoną z użyciem narzędzia trasowania połączeń programu VPR, wynika z rozwiązania zastosowanego podczas trasowania połączeń w programie VPR. Narzędzie trasowania połączeń programu VPR przenosi połączenia między kanałami, w zależności od skupienia połączeń w kanałach. Estymacja długości połączeń zastosowana w równoległym przetwarzaniu, a także w narzędziu rozmieszczania modułów programu VPR, nie uwzględnia skupienia połączeń w kanałach. Z tego względu powyższa rozbieżność występuje zarówno dla rozmieszczeń wykonanych z użyciem równoległego przetwarzania, jak również dla rozmieszczeń wykonanych narzędziem rozmieszczania modułów programu VPR. Rysunki 53-57 przedstawiają połączenia między modułami dla ostatniego rozmieszczenia otrzymanego z użyciem równoległego przetwarzania oraz dla rozmieszczenia wykonanego narzędziem rozmieszczania programu VPR (tabela 26). Połączenia między modułami poprowadzono tak jak w grafie pełnym, w przestrzeni Euklidesa (dla większej przejrzystości rysunków).



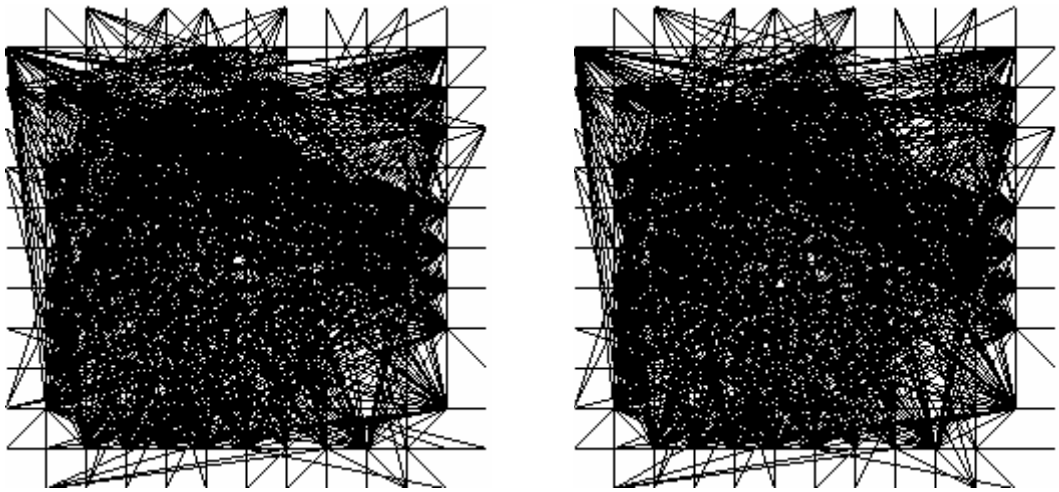
(a) (b)

Rys. 53. Połączenia między modułami w układzie *term1*: a) ostatnie rozmieszczenie równoległego przetwarzania, b) rozmieszczenie programu VPR



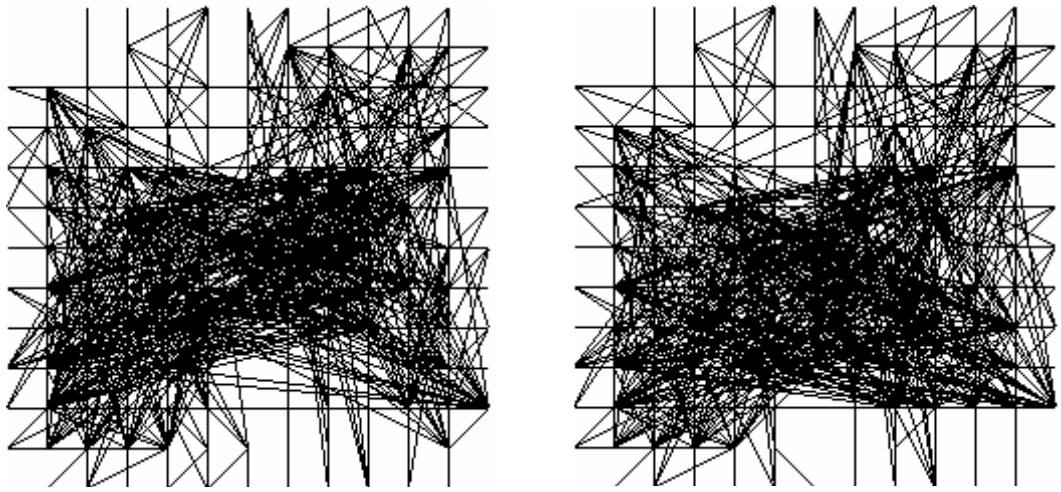
(a) (b)

Rys. 54. Połączenia między modułami w układzie *9symml*: a) ostatnie rozmieszczenie równoległego przetwarzania, b) rozmieszczenie programu VPR



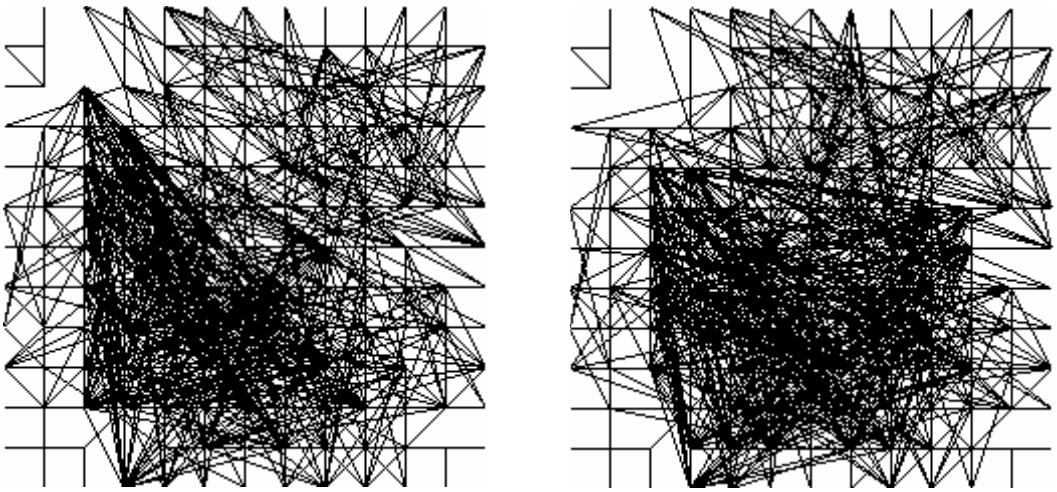
(a) (b)

Rys. 55. Połączenia między modułami w układzie *apex7*: a) ostatnie rozmieszczenie równoległego przetwarzania, b) rozmieszczenie programu VPR



(a) (b)

Rys. 56. Połączenia między modułami w układzie *c1355*: a) ostatnie rozmieszczenie równoległego przetwarzania, b) rozmieszczenie programu VPR



(a) (b)

Rys. 57. Połączenia między modułami w układzie *c880*: a) ostatnie rozmieszczenie równoległego przetwarzania, b) rozmieszczenie programu VPR

Rozwiązanie przedstawione w niniejszym rozdziale, polegające na zastosowaniu równoległego przetwarzania oraz jednostek przetwarzających w postaci sieci Hopfielda, umożliwia znaczne zmniejszenie liczby neuronów oraz liczby połączeń między neuronami, w porównaniu do rozwiązania, w którym jedna sieć Hopfielda wykonuje rozmieszczenie wszystkich modułów układu. W przypadku rozmieszczania X modułów na podłożu o rozmiarach K i L jednostek, liczba neuronów $N = X\,K\,L$, natomiast liczba połączeń między neuronami wynosi $X^2K^2L^2$. Dla większości badanych układów nie jest możliwe przeprowadzenie symulacji w przypadku, gdy jedna sieć Hopfielda wykonywałaby rozmieszczenie wszystkich modułów. W przypadku zastosowania równoległego przetwarzania z jednostkami przetwarzającymi obejmującymi maksymalnie 4 elementarne, jednostkowe części podłoża układu, liczba neuronów w jednostce przetwarzającej może wynosić maksymalnie 16 neuronów, natomiast maksymalna liczba połączeń między neuronami jest równa 256. W przypadku rozmieszczania np. 64 modułów na podłożu o rozmiarach 8x8 jednostek, liczba neuronów niezbędnych do wykonania rozmieszczenia modułów z użyciem równoległego przetwarzania jest odpowiednio 16 lub około 19 razy mniejsza, w porównaniu do rozwiązania wykorzystującego jedną sieć Hopfielda do rozmieszczenia wszystkich modułów układu, w zależności od sposobu podziału układu na części podczas równoległego przetwarzania. Liczba połączeń między neuronami jest wtedy odpowiednio 4096 lub około 6088 razy mniejsza. Podczas równoległego przetwarzania jednostki przetwarzające muszą jednak komunikować się między sobą. Nawet w przypadku użycia dedykowanych połączeń między każdą parą jednostek przetwarzających, równoległe przetwarzanie wymagałoby około 3979 lub około 5657 razy mniej połączeń, w zależności od sposobu podziału układu na części podczas równoległego przetwarzania. Wraz ze wzrostem liczby rozmieszczanych modułów oraz rozmiarów podłoża, na którym są rozmieszczane moduły, liczba neuronów oraz połączeń między neuronami w jednej sieci Hopfielda, która wykonywałaby rozmieszczenie wszystkich modułów układu, ulegają dalszemu, gwałtownemu wzrostowi, w porównaniu do rozwiązania wykorzystującego równoległe przetwarzanie.

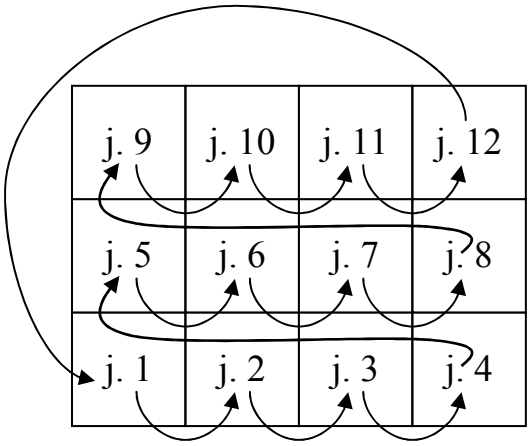
Wraz ze wzrostem liczby rozmieszczanych modułów liczba podziałów układu, które są niezbędne do ustabilizowania się kosztu otrzymywanych rozwiązań będzie wzrastać. Załóżmy, że moduły są rozmieszczane na podłożu, które jest kwadratem, liczba rozmieszczanych modułów jest równa X, każdy moduł zajmuje kwadratową część podłoża układu o rozmiarze jednej jednostki oraz moduły wypełniają całe dostępne podłoże układu. Liczba podziałów układu niezbędnych do ustabilizowania się kosztu otrzymywanych rozwiązań będzie wówczas proporcjonalna do $\sqrt{X}$, ponieważ w najgorszym przypadku dany moduł musi być przeniesiony wzdłuż przekątnej kwadratowego podłoża układu, która posiada długość $2\sqrt{X}$ w metryce Manhattan. Podobne uzasadnienie można znaleźć w pracach [123, 139].

8.3.1. Wyniki rozmieszczania otrzymane w przypadku, gdy jednostki przetwarzające przekazują położenie modułów wyłącznie do sąsiedniej jednostki

Dotychczasowe symulacje równoległego przetwarzania przeprowadzono przy założeniu, że jednostki przetwarzające przekazywały informację o położeniu swoich modułów do wszystkich pozostałych jednostek. Przekazywanie informacji o położeniu modułów odbywało się w każdej iteracji symulacji pracy jednostek lub po zakończeniu danego podziału układu na części. Powyższe założenie utrudnia jednak sprzętową realizację metody, ponieważ jednostki przetwarzające musiałyby komunikować się ze wszystkimi pozostałymi jednostkami. W niniejszym podrozdziale zostanie sprawdzone rozwiązanie, polegające na przekazywaniu informacji o położeniu modułów wyłącznie między sąsiednimi jednostkami w utworzonej pętli jednostek [123].

Rysunek 58 przedstawia sposób przekazywania informacji o położeniu modułów między jednostkami przetwarzającymi. Jednostki przetwarzające przekazują informację o położeniu modułów wyłącznie do sąsiedniej jednostki w pętli, utworzonej z jednostek. Przekazywanie informacji o położeniu modułów odbywa się w trakcie wykonywania rozmieszczania modułów przez jednostki przetwarzające, dla danego podziału układu na części. Podczas pierwszego podziału układu na części każda jednostka przekazuje informację o położeniu swoich modułów do sąsiedniej jednostki w pętli, która odbiera tę informację. Podczas następnych podziałów każda jednostka przekazuje dalej informację o położeniu modułów, którą odebrała podczas poprzedniego podziału układu na części. Po wykonaniu n-1 podziałów, gdzie n jest liczbą jednostek przetwarzających, informacja o położeniu swoich modułów, przekazana przez wszystkie jednostki podczas pierwszego podziału układu na części dotrze do wszystkich pozostałych jednostek. Podczas następnego podziału układu na części, proces przekazywania położenia modułów przez jednostki rozpoczyna się na nowo od przekazania przez każdą jednostkę aktualnej informacji o położeniu swoich modułów do sąsiedniej jednostki w pętli. W przedstawionym rozwiązaniu jednostki przetwarzające posiadają informację o przybliżonym położeniu modułów, znajdujących się w pozostałych częściach układu.

Powyższa informacja może nie być aktualna (dokładna), ponieważ w każdym podziale układu każda jednostka aktualizuje jedynie położenie modułów przekazane przez sąsiednią jednostkę. Przedstawione rozwiązanie znacznie ułatwia jednak sprzętową realizację metody, ponieważ jednostki przetwarzające komunikują się wyłącznie z sąsiednimi jednostkami w pętli jednostek, a przekazywanie informacji o położeniu modułów może odbywać się w trakcie wykonywania rozmieszczania w danym podziale układu na części.



Rys. 58. Sposób przekazywania informacji o położeniu modułów między jednostkami przetwarzającymi

Informacja o położeniu modułów znajdujących się w pozostałych częściach układu jest przechowywana w lokalnej pamięci. Dla każdego modułu jest przechowywana w lokalnej pamięci lista modułów i padów, które są połączone z danym modułem. Listy modułów i padów są przechowywane oddzielnie dla każdego węzła, do którego należy dany moduł. Liczba powyższych węzłów jest ograniczona, ze względu na ograniczoną liczbę końcówek modułu (komórki CLB). Razem z listą modułów połączonych z danym modułem są również przechowywane współrzędne tych modułów na podłożu układu, które są przybliżonymi współrzędnymi, ze względu na ich sposób aktualizacji. Jeżeli dana jednostka przetwarzająca odbierze informację o położeniu modułów, przekazaną przez sąsiednią jednostkę w pętli, to dana jednostka sprawdza, czy powyższe moduły występują na listach połączeń modułów położonych w danej jednostce przetwarzającej. Odebrane informacje przez daną jednostkę przetwarzającą umożliwiają aktualizację położenia (współrzędnych) modułów występujących w listach połączeń modułów danej jednostki przetwarzającej [123]. Powyższe informacje, przechowywane w pamięci lokalnej, umożliwiają wyznaczenie wag połączeń między neuronami oraz zewnętrznych sygnałów wejściowych neuronów sieci Hopfielda poszczególnych jednostek przetwarzających, bez konieczności odwoływania się do pamięci wspólnej systemu równoległego przetwarzania. Wartość zewnętrznych sygnałów wejściowych neuronów nie zmienia się podczas wykonywania wszystkich iteracji danego podziału układu na części. Początkowe rozmieszczenie modułów na podłożu układu jest wyznaczane w sposób losowy, w taki sam sposób jak poprzednio. Współrzędne modułów połączonych z

danym modułem, przechowywane razem z listą połączeń danego modułu, na początku są inicjowane współrzędnymi środka podłoża układu.

Dla wszystkich układów wykonano minimalizację długości połączeń, z użyciem równoległego przetwarzania, jednostki przetwarzające przekazywały informację o położeniu modułów wyłącznie do sąsiedniej jednostki w pętli jednostek. Symulacje przeprowadzono z wykorzystaniem programu napisanego w języku C++. W programie zastosowano wszystkie rozwiązania opisane w niniejszym podrozdziale. Rozwiązania opisane w poprzednich podrozdziałach zastosowano w taki sam sposób, jak poprzednio. Dla wszystkich układów wykonano 800 zmian sposobu podziału układu na części. Podczas pierwszych 300 podziałów jednostki przetwarzające akceptowały wszystkie prawidłowe rozwiązania, następnie jednostki przetwarzające akceptowały rozwiązania, które umożliwiły zmniejszenie kosztu rozwiązania danej jednostki przetwarzającej. Tabela 27 zawiera otrzymane rezultaty.

Nazwa układu	Wynik	Rezultat (estymacja)
term1	średnia długość połączeń	302,6536
	najmniejsza dł. poł. otrzyman. rozw.	288,4458
	dł. poł. dla rozm. programu VPR	285,1448
9symml	średnia długość połączeń	476,4987
	najmniejsza dł. poł. otrzyman. rozw.	451,1988
	dł. poł. dla rozm. programu VPR	442,1362
apex7	średnia długość połączeń	524,0212
	najmniejsza dł. poł. otrzyman. rozw.	496,2432
	dł. poł. dla rozm. programu VPR	494,2260
c1355	średnia długość połączeń	727,4756
	najmniejsza dł. poł. otrzyman. rozw.	690,5440
	dł. poł. dla rozm. programu VPR	688,7998
c880	średnia długość połączeń	843,4619
	najmniejsza dł. poł. otrzyman. rozw.	764,8251
	dł. poł. dla rozm. programu VPR	774,6796

Tabela 27. Wyniki otrzymane dla badanych układów dla pętli jednostek

Z danych zawartych w tabeli 27 wynika, że rezultaty rozmieszczania otrzymane w przypadku, gdy jednostki przetwarzające przekazują informację o położeniu modułów wyłącznie do sąsiedniej jednostki są porównywalne z wcześniej osiągniętymi rezultatami, gdy jednostki przetwarzające przekazywały informację o położeniu swoich modułów do wszystkich pozostałych jednostek. Średnia estymowana długość połączeń dla 800 podziałów układu oraz najmniejsza estymowana długość połączeń dla otrzymanych rozwiązań jest zbliżona do wcześniejszych rezultatów. Tabela 28 zawiera rezultaty trasowania połączeń z użyciem narzędzia trasowania połączeń programu VPR dla 5 najlepszych rozmieszczeń otrzymanych z użyciem równoległego przetwarzania, dla ostatniego rozmieszczenia otrzymanego w serii 800 podziałów układu na części oraz dla rozmieszczenia otrzymanego z

użyciem narzędzia rozmieszczania programu VPR. Dla wszystkich rozmieszczeń wykonano również trasowanie połączeń z użyciem aproksymacji minimalnego drzewa Steinera, przedstawionej w podrozdziale 7.2.1.

Rozmieszczenie		Estymowana długość połączeń	Długość połączeń (min. drzewo Steinera)	Długość połączeń (VPR)	Minimalna pojemność kanału (VPR)
term1	równ. prz. 1	288,4458	282	437	5
	równ. prz. 2	288,7267	281	434	5
	równ. prz. 3	288,9177	283	431	4
	równ. prz. 4	289,0781	282	436	4
	równ. prz. 5	289,3488	285	421	4
	równ. prz. ost.	288,7267	281	434	5
	VPR	285,1448	275	408	5
9symml	równ. prz. 1	451,1988	446	687	6
	równ. prz. 2	451,6454	449	659	6
	równ. prz. 3	453,2187	450	658	6
	równ. prz. 4	453,4592	450	697	6
	równ. prz. 5	453,5884	451	703	6
	równ. prz. ost.	451,1988	446	687	6
	VPR	442,1362	438	669	6
apex7	równ. prz. 1	496,2432	463	671	5
	równ. prz. 2	496,3794	463	674	5
	równ. prz. 3	496,6409	463	673	5
	równ. prz. 4	497,6233	464	696	5
	równ. prz. 5	500,0715	475	669	5
	równ. prz. ost.	496,3794	463	674	5
	VPR	494,2260	459	689	5
c1355	równ. prz. 1	690,5440	668	942	6
	równ. prz. 2	691,0602	672	965	5
	równ. prz. 3	691,1163	669	946	5
	równ. prz. 4	692,3249	674	977	5
	równ. prz. 5	692,4333	674	977	5
	równ. prz. ost.	690,5440	668	946	6
	VPR	688,7998	663	967	6
c880	równ. prz. 1	764,8251	754	1095	6
	równ. prz. 2	765,0918	754	1092	6
	równ. prz. 3	765,3481	754	1072	7
	równ. prz. 4	765,7471	754	1048	6
	równ. prz. 5	765,8536	755	1082	6
	równ. prz. ost.	764,8251	754	1095	6
	VPR	774,6796	739	1086	6

Tabela 28. Wyniki trasowania otrzymane dla najlepszych rozwiązań dla pętli jednostek

Dane zawarte w tabeli 28 potwierdzają wysoką jakość rozwiązań, otrzymanych w przypadku, gdy jednostki przetwarzające przekazują informację o położeniu modułów wyłącznie do sąsiedniej jednostki. Długości połączeń po wykonaniu trasowania połączeń z użyciem narzędzia trasowania połączeń programu VPR są porównywalne dla rozmieszczeń modułów wyznaczonych z użyciem równoległego przetwarzania oraz dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR. Porównywalne wyniki otrzymano nie tylko dla najlepszych rozwiązań uzyskanych z użyciem równoległego przetwarzania, ale również dla ostatniego rozmieszczenia, otrzymanego w serii 800 podziałów układu na części. Niektóre rozmieszczenia otrzymane z użyciem równoległego przetwarzania umożliwiają nawet połączenie modułów z użyciem mniejszej długości połączeń lub wymagają kanałów, których pojemność jest mniejsza o 1. Dla wszystkich rozpatrywanych rozwiązań otrzymanych z użyciem równoległego przetwarzania oraz dla rozmieszczenia modułów wykonanego z użyciem narzędzia rozmieszczania modułów programu VPR, długości połączeń aproksymacji minimalnego drzewa Steinera są porównywalne.

W niniejszym podrozdziale przedstawiono rozwiązanie polegające na równoległym przetwarzaniu przez jednostki przetwarzające, które przekazują informację o położeniu modułów wyłącznie do sąsiedniej jednostki w pętli jednostek. Przedstawione rozwiązanie umożliwia otrzymanie rozmieszczeń o zbliżonej jakości, w porównaniu do wcześniejszych rozwiązań równoległego przetwarzania, ale wymaga mniejszej ilości informacji, które muszą być wymieniane między jednostkami przetwarzającymi. Rozwiązanie przedstawione w niniejszym podrozdziale ułatwia sprzętową realizację rozmieszczania, ponieważ jednostki przetwarzające muszą komunikować się wyłącznie ze swoją sąsiednią jednostką, a nie z wszystkimi pozostałymi jednostkami.

8.4. Podsumowanie

W niniejszym rozdziale przedstawiono oryginalne rozwiązanie, polegające na zastosowaniu równoległego przetwarzania z jednostkami przetwarzającymi w postaci sieci Hopfielda. Zaproponowana metoda rozmieszczania umożliwia otrzymanie rozwiązań o bardzo dobrej jakości. Niektóre rozmieszczenia otrzymane z użyciem równoległego przetwarzania umożliwiają nawet połączenie modułów z użyciem mniejszej długości połączeń lub z użyciem kanałów o mniejszej pojemności, w porównaniu do rozmieszczeń narzędzia rozmieszczania programu VPR, które wykorzystuje algorytm symulowanego wyżarzania. Czas wykonywania programu symulującego równoległe przetwarzanie jest dłuższy od czasu wykonywania programu VPR, ale przedstawione rozwiązanie jest przeznaczone do sprzętowej realizacji. Sprzętowa realizacja równoległego przetwarzania umożliwiłaby skrócenie czasu rozmieszczania modułów nawet o kilka rzędów wielkości, w porównaniu do programów uruchamianych na klasycznych komputerach. Przedstawione rozwiązanie umożliwia znaczne zmniejszenie liczby neuronów oraz liczby połączeń między neuronami, w porównaniu do rozwiązania, w którym jedna sieć Hopfielda wykonuje rozmieszczenie wszystkich modułów układu, co jest

niezwykle istotne z punktu widzenia sprzętowej realizacji algorytmu. W niniejszym rozdziale rozważono różne rozwiązania, które można zastosować podczas równoległego przetwarzania. Na szczególną uwagę zasługuje rozwiązanie przedstawione w podrozdziale 8.3.1, które polega na równoległym przetwarzaniu przez jednostki przetwarzające, które przekazują informację o położeniu modułów wyłącznie do sąsiedniej jednostki w pętli jednostek. Powyższe rozwiązanie ułatwia sprzętową realizację algorytmu rozmieszczania, zapewniając otrzymanie rozmieszczeń o bardzo dobrej jakości.

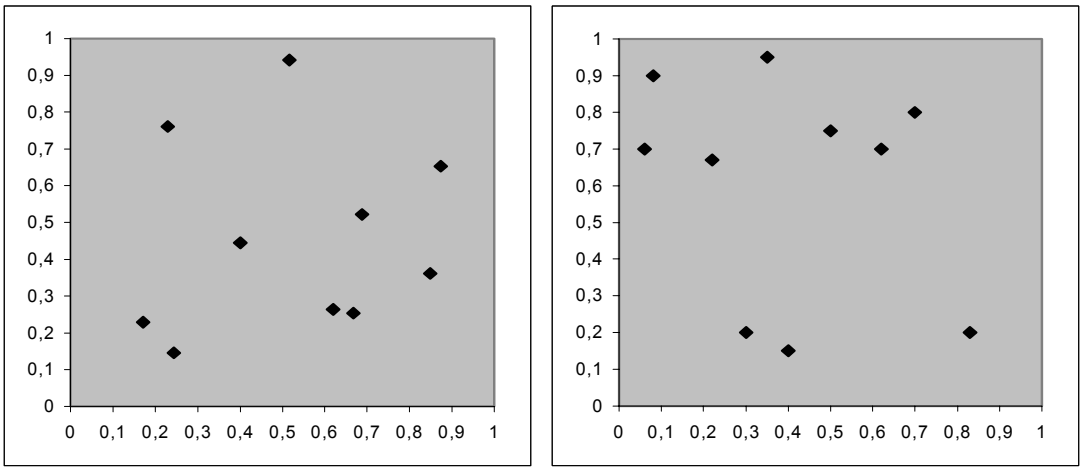
Zaproponowana metoda rozmieszczania nie „utykała" w minimum lokalnym funkcji kosztu, którego koszt jest znacznie większy od kosztu rozwiązania otrzymanego z użyciem programu VPR. Zaproponowana metoda rozmieszczania modułów zawdzięcza powyższą właściwość zastosowaniu sieci Hopfielda jako jednostki przetwarzającej podczas równoległego przetwarzania. Metody wykorzystujące sieć Hopfielda są metodami stochastycznymi, ponieważ losowa inicjalizacja neuronów w sieci Hopfielda umożliwia otrzymanie różnych rozwiązań w kolejnych próbach. Stochastyczny charakter zaproponowanego rozwiązania zapobiega „utykaniu" rozwiązań w minimach lokalnych funkcji kosztu. Po ustabilizowaniu się średniego kosztu otrzymywanych rozwiązań, zastosowano akceptację jedynie rozwiązań, które zmniejszają koszt rozwiązania poszczególnych jednostek przetwarzających. Zastosowane rozwiązanie umożliwia zmniejszenie fluktuacji kosztu otrzymywanych rozwiązań oraz otrzymanie rozwiązań o lepszej jakości. Powyższe rozwiązanie odpowiada rozwiązaniu stosowanemu przez algorytmy zachłanne lub symulowanemu wyżarzaniu z temperaturą układu równą 0. Powyższe rozwiązanie jest jednak stosowane dopiero w momencie, gdy nie następuje wyraźne zmniejszanie się średniego kosztu otrzymywanych rozwiązań, a koszt rozwiązań jest zbliżony do kosztu rozwiązań otrzymanych innymi metodami. W przeciwnym razie jest możliwe „utykanie" rozwiązań w minimach lokalnych funkcji kosztu, których koszt jest wyraźnie większy od kosztu rozwiązań otrzymanych innymi metodami.

W niniejszej pracy rozpatrzono jednostki przetwarzające, które mogą równocześnie rozmieszczać maksymalnie 4 moduły. Możliwe jest również zastosowanie jednostek przetwarzających obejmujących większą liczbę modułów. Przedstawiony schemat równoległego przetwarzania może być również zastosowany dla jednostek przetwarzających, wykorzystujących inną metodę rozmieszczania.

Rozdział 9
Zastosowanie sieci Hopfielda w innych problemach optymalizacyjnych

W niniejszym rozdziale zostaną przedstawione rezultaty zastosowania sieci Hopfielda w innych problemach optymalizacyjnych, na przykładzie problemu komiwojażera. Problem komiwojażera jest jednym z klasycznych kombinatorycznych problemów optymalizacyjnych, jest to tzw. problem NP-trudny [30, 90, 92-94, 107-110]. Problem ten w najczęściej spotykanej postaci polega na znalezieniu trasy przejazdu komiwojażera łączącej *n* miast tak, aby koszt przejazdu był najmniejszy z możliwych, każde miasto było odwiedzone dokładnie jeden raz oraz podróż komiwojażera zakończyła się w mieście, z którego wyruszył. Ponadto zakłada się, że koszt przejazdu między dwoma dowolnymi miastami spośród wszystkich *n* miast nie zależy od kierunku przejazdu między tymi miastami. Dla tak postawionego problemu, dla $n > 2$ istnieje $n!/2n$ różnych tras przejazdu, ponieważ każda trasa spośród $n!$ możliwych tras może rozpoczynać się w jednym spośród *n* miast oraz przebiegać w jednym z dwóch możliwych kierunków. Problem komiwojażera można rozwiązać z użyciem sieci Hopfielda. Funkcje kosztu dla problemu komiwojażera oraz problemu rozmieszczania modułów w układzie VLSI posiadają zbliżoną postać [128].

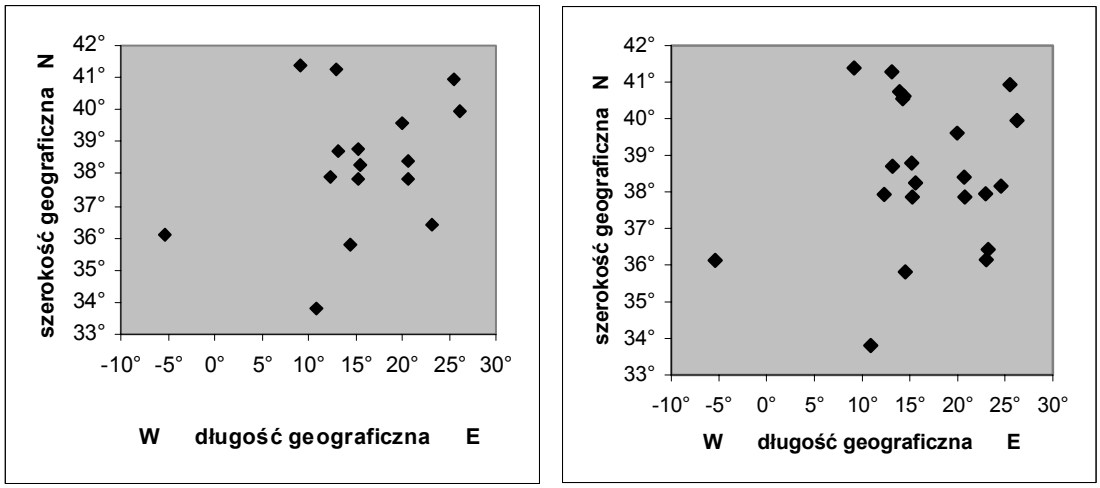
Celem przeprowadzonych symulacji było znalezienie optymalnej trasy komiwojażera dla 6 przykładów. Pierwsze dwa przykłady są zbiorem 10 miast. Przykład 1 jest standardowym zbiorem porównawczym metod wykorzystujących sieci neuronowe. Pierwotnie został zastosowany w pracy [107]. Przykład 2 pochodzi z pracy [92]. Rysunek 59 przedstawia położenie 10 miast leżących w kwadracie jednostkowym dla przykładów 1 oraz 2.



(a) (b)

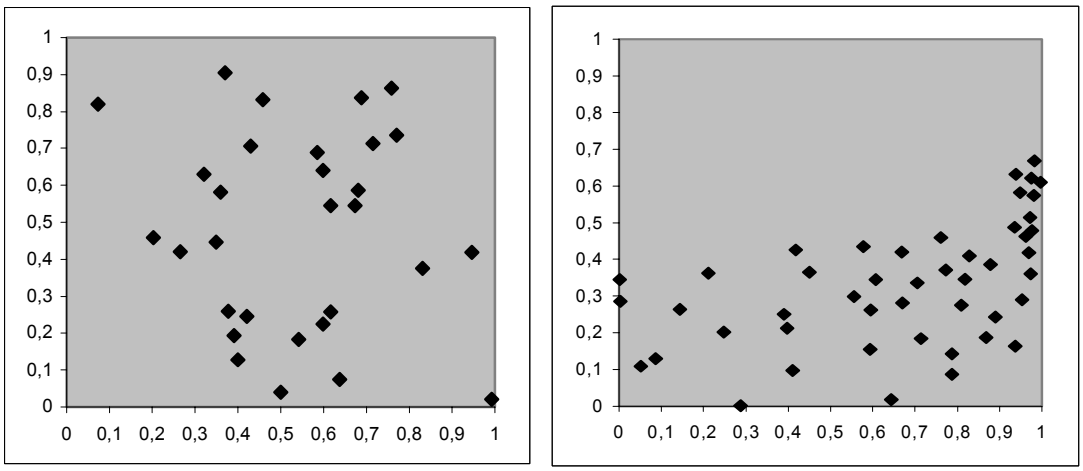
Rys. 59. Położenie miast w kwadracie jednostkowym: a) dla przykładu 1, b) dla przykładu 2

Przykłady 3, 4 oraz 6 zaczerpnięto z biblioteki problemu komiwojażera [122]. W bibliotece znajdują się również optymalne trasy dla tych przykładów razem z kosztem rozwiązań. Przykłady 3 oraz 4 (*ulysses16.tsp* oraz *ulysses22.tsp*) opisują podróż Odyseusza, która składa się odpowiednio z 16 oraz 22 etapów. Poszczególne etapy podróży wyznaczają współrzędne geograficzne punktów na Ziemi, ujemne wartości długości geograficznej odpowiadają długości geograficznej W. Przykład 6 (*att48.tsp*) tworzą stolice 48 stanów w USA. Dla powyższych przykładów podczas symulacji zastosowano takie samo skalowanie odległości między miastami jak w pracy [132]. W wyniku skalowania optymalna trasa dla przykładu 3 wynosi 2,36316, dla przykładu 4 2,41279 oraz 4,32452 dla przykładu 6. Przykład 5 jest zbiorem 30 miast, pierwotnie został zastosowany w pracy [107]. Rysunek 60 przedstawia współrzędne geograficzne punktów na Ziemi dla przykładów 3 oraz 4. Rysunek 61 przedstawia położenie miast w kwadracie jednostkowym dla przykładów 5 oraz 6.



(a) (b)

Rys. 60. Współrzędne geograficzne punktów na Ziemi: a) dla przykładu 3, b) dla przykładu 4



(a) (b)

Rys. 61. Położenie miast w kwadracie jednostkowym: a) dla przykładu 5, b) dla przykładu 6

Każdy z przykładów został rozwiązany z użyciem klasycznej sieci Hopfielda z oryginalną modyfikacją funkcji kosztu oraz z użyciem sieci z oryginalną modyfikacją sygnałów wejściowych. Dla obydwu metod zbadano dwa sposoby inicjalizacji neuronów. Symulacje przeprowadzono z zastosowaniem programu napisanego w języku C++ [128]. Zastosowano takie same rozwiązania oraz wartości parametrów, jak w przypadku minimalizacji długości połączeń w rozdziale 5. Wartości stałych w funkcji kosztu sieci Hopfielda dobrano tak, aby prawdopodobieństwo uzyskania prawidłowego rozwiązania było bliskie 100%, co umożliwia porównanie wyników otrzymanych różnymi metodami. Tabela 29 przedstawia wyniki otrzymane dla przykładów 1-6. Dla wszystkich przykładów wykonano 100 prób. Tabela 29 zawiera następujące wyniki: średni koszt otrzymanych rozwiązań, koszt najlepszego otrzymanego rozwiązania, koszt optymalnego rozwiązania oraz stosunek średniego kosztu do kosztu optymalnego rozwiązania.

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
1	średni koszt	2,7343	2,6907	2,6907	2,6907
	koszt najlepszego otrzymanego rozw.	2,6907	2,6907	2,6907	2,6907
	koszt optymalnego rozwiązania	2,6907	2,6907	2,6907	2,6907
	śr. koszt / koszt opt. rozw.	1,0162	1	1	1
2	średni koszt	2,8170	2,7818	2,7818	2,7818
	koszt najlepszego otrzymanego rozw.	2,7818	2,7818	2,7818	2,7818
	koszt optymalnego rozwiązania	2,7818	2,7818	2,7818	2,7818
	śr. koszt / koszt opt. rozw.	1,0127	1	1	1
3	średni koszt	3,0272	2,5108	2,4939	2,4996
	koszt najlepszego otrzymanego rozw.	2,5323	2,3811	2,3951	2,3942
	koszt optymalnego rozwiązania	2,3632	2,3632	2,3632	2,3632
	śr. koszt / koszt opt. rozw.	1,2810	1,0625	1,0553	1,0577
4	średni koszt	3,4939	2,6718	2,6177	2,5909
	koszt najlepszego otrzymanego rozw.	2,8337	2,4522	2,4649	2,4522
	koszt optymalnego rozwiązania	2,4128	2,4128	2,4128	2,4128
	śr. koszt / koszt opt. rozw.	1,4481	1,1073	1,0849	1,0738

Nr przykł.	Wynik	Klasyczna metoda		Metoda z modyfikacją sygnałów wejściowych	
		inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01	inicjalizacja -0,3÷0,3	inicjalizacja -0,01÷0,01
5	średni koszt	6,1397	4,9936	4,7480	4,6218
	koszt najlepszego otrzymanego rozw.	5,2880	4,3781	4,2875	4,2786
	koszt optymalnego rozwiązania	4,2774	4,2774	4,2774	4,2774
	śr. koszt / koszt opt. rozw.	1,4354	1,1674	1,1100	1,0805
6	średni koszt	6,5588	5,5125	5,2576	5,1185
	koszt najlepszego otrzymanego rozw.	5,5005	4,6505	4,6151	4,5281
	koszt optymalnego rozwiązania	4,3245	4,3245	4,3245	4,3245
	śr. koszt / koszt opt. rozw.	1,5167	1,2747	1,2158	1,1836

Tabela 29. Wyniki otrzymane dla poszczególnych przykładów z użyciem klasycznej i zmodyfikowanej sieci Hopfielda

Analiza wyników zawartych w tabeli 29 pozwala wyciągnąć wniosek, że wyniki otrzymane metodą z modyfikacją sygnałów wejściowych są lepsze od wyników osiągniętych klasyczną siecią Hopfielda. Wyniki otrzymane obydwoma metodami dla przykładów 1 oraz 2 są bardzo zbliżone, a koszt rozwiązań jest równy lub bliski kosztowi optymalnego rozwiązania. Począwszy od przykładu 3 jest widoczna przewaga metody z modyfikacją sygnałów wejściowych, przewaga ta wzrasta wraz ze wzrostem wymiaru problemu. Przeprowadzone symulacje potwierdzają również, że inicjalizacja neuronów sygnałami wejściowymi bliskimi 0 (sygnały wyjściowe neuronów bliskie 0,5), przyczynia się do wzrostu skuteczności klasycznej sieci Hopfielda, pod względem jakości otrzymywanych rozwiązań. Wyniki uzyskane metodą z modyfikacją sygnałów wejściowych dla obydwu sposobów inicjalizacji neuronów są jednak lepsze od wyników otrzymanych klasyczną siecią, niezależnie od sposobu inicjalizacji neuronów.

Wyniki otrzymane podczas symulacji dla przykładów 1-6 są zdecydowanie lepsze od wyników przedstawionych w pracach innych badaczy, poświęconych problemowi komiwojażera. W niniejszej pracy szczególną uwagę zwraca skuteczność sieci Hopfielda dla problemów o liczbie miast równej 10, dla których sieć Hopfielda dla 100% prób generowała optymalne rozwiązanie. Takich rezultatów nie udało się osiągnąć w innych pracach z użyciem klasycznego modelu sieci Hopfielda. W wielu pracach średni koszt rozwiązań otrzymanych klasyczną siecią Hopfielda był znacznie większy od kosztu optymalnego rozwiązania [92, 107, 130-132]. W niektórych pracach rezultaty zbliżone do wyników otrzymanych w niniejszym rozdziale osiągnięto z użyciem sieci Hopfielda jedynie w wyniku zastosowania bardzo skomplikowanych rozwiązań [92, 132, 153].

Rozdział 10
Zakończenie

Model ciągły sieci Hopfielda został zaproponowany w 1985r. przez J. J. Hopfielda i D. W. Tanka [107]. Pierwsze próby zastosowania sieci Hopfielda w problemach optymalizacyjnych dotyczyły przede wszystkim problemu komiwojażera. Początkowe zainteresowanie badaczy siecią Hopfielda stopniowo zmniejszało się, ponieważ kolejni badacze otrzymywali rozwiązania o znacznie gorszej jakości niż w pracy [107]. Potwierdziły to liczne późniejsze prace [92, 108, 130-132]. Niska jakość otrzymywanych rozwiązań była również powodem niewielu zastosowań sieci Hopfielda w projektowaniu układów VLSI [116]. Badania przedstawione w poprzednich rozdziałach dowodzą, że sieć Hopfielda może dostarczać rozwiązań o bardzo dobrej jakości, szczególnie dla problemów o małym wymiarze. Metody poprawy skuteczności sieci Hopfielda oraz bardzo dobre rezultaty wykorzystania sieci Hopfielda jako jednostki przetwarzającej w systemie równoległego przetwarzania, otwierają nowe możliwości zastosowania sieci Hopfielda w projektowaniu układów VLSI oraz w innych problemach optymalizacyjnych.

10.1. Możliwości i zalety zastosowania sieci Hopfielda w projektowaniu topografii systemów VLSI

Rozwiązania przedstawione w niniejszej pracy są przeznaczone do sprzętowej realizacji. Sprzętowa realizacja algorytmu rozmieszczania komórek logicznych umożliwiłaby znaczne skrócenie czasu obliczeń, w porównaniu do czasu wykonywania programów rozmieszczania na tradycyjnych komputerach. Sprzętowe rozwiązanie problemu rozmieszczania umożliwiłoby opracowywanie i realizację nowych algorytmów przetwarzania układu rekonfigurowalnego FPGA w sposób dynamiczny, podczas pracy układu. W tym przypadku czas potrzebny na przygotowanie i realizację nowego algorytmu przetwarzania odgrywa decydującą rolę. Rekonfigurowalne układy FPGA posiadają zdolność do wymiany danych konfiguracyjnych w czasie pracy układu. Układy ze zdolnością rekonfiguracji w „locie” znajdują coraz szersze zastosowanie w telekomunikacji, systemach wizyjnych, robotyce, systemach obliczeniowych. Istotnym ograniczeniem zastosowania układów rekonfigurowalnych jest konieczność wcześniejszego przygotowania algorytmu lub algorytmów, realizowanych z użyciem tych układów. Dla każdego algorytmu musi być wcześniej wykonana synteza logiczna, rozmieszczanie komórek logicznych oraz trasowanie połączeń. Dzięki temu jest możliwe otrzymanie danych konfiguracyjnych układu dla danego algorytmu przetwarzania. Ze względu na

bardzo długi czas niezbędny do opracowania nowego algorytmu przetwarzania, układy rekonfigurowalne mogą używać tylko wcześniej opracowanych algorytmów przetwarzania. Nie jest możliwe tworzenie nowych algorytmów przetwarzania podczas pracy układu. Rozmieszczanie komórek jest etapem, który wymaga najdłuższych obliczeń podczas opracowywania nowego algorytmu przetwarzania. Bardzo dobre rezultaty rozmieszczania komórek logicznych w układach FPGA, otrzymane z użyciem równoległego przetwarzania oraz jednostek przetwarzających w postaci sieci Hopfielda, umożliwiają zastosowanie powyższego rozwiązania do rozmieszczania komórek logicznych w układach rekonfigurowalnych. Sprzętowe rozwiązanie problemu rozmieszczania komórek logicznych umożliwiłoby znaczne skrócenie czasu obliczeń oraz opracowywanie i realizację nowych algorytmów przetwarzania układu rekonfigurowalnego w sposób dynamiczny, podczas pracy układu.

Rozwiązanie wykorzystujące równoległe przetwarzanie oraz jednostki przetwarzające w postaci sieci Hopfielda może być również zastosowane w układach rekonfigurowalnych poddawanych częściowej rekonfiguracji. Sieć Hopfielda umożliwia wykonanie rozmieszczania z ustalonym położeniem wybranych komórek logicznych oraz w przypadku, gdy pewne części podłoża układu nie mogą być wykorzystane do rozmieszczania komórek logicznych. Zastosowanie sieci Hopfielda do rozmieszczania komórek logicznych, umożliwiłoby znaczne skrócenie czasu częściowej rekonfiguracji układu.

Rozwiązanie oparte na równoległym przetwarzaniu oraz jednostkach przetwarzających w postaci sieci Hopfielda można również wykorzystać w układach rekonfigurowalnych do wykonywania rozmieszczania na poziomie obiektów sprzętowych (ang. hardware object). Zastosowanie biblioteki obiektów sprzętowych umożliwia budowę dużych systemów z mniejszych komponentów znajdujących się w bibliotece. Użycie sieci Hopfielda wymagałoby jednak podziału obiektów sprzętowych na elementarne części o jednakowych rozmiarach, rozmiar tych części powinien być wielokrotnością rozmiaru pojedynczej komórki logicznej. Podobne rozwiązanie zostało zastosowane w pracy [121]. Następnie, części obiektów sprzętowych mogą być rozmieszczane. Obrys obiektu sprzętowego po rozmieszczeniu wszystkich jego części może nie być prostokątem. W układach rekonfigurowalnych (układy FPGA) można dopuścić takie rozwiązanie, ponieważ zapewnia lepsze wykorzystanie powierzchni układu i umożliwia rozmieszczenie obiektów sprzętowych na prostokątnym obszarze o mniejszych rozmiarach. W przeciwnym razie, należy wyznaczyć położenie prostokątnych obiektów sprzętowych w wyniku uśrednienia położenia elementarnych części poszczególnych obiektów. Ewentualne zachodzenie obiektów sprzętowych na siebie można usunąć wykorzystując metodę heurystyczną, która została zastosowana w programie MPG-MS [102].

Sieć Hopfielda oraz równoległe przetwarzanie z jednostkami przetwarzającymi w postaci sieci Hopfielda mogą być również zastosowane podczas projektowania innych układów VLSI. Stale wzrastające rozmiary projektowanych układów scalonych sprawiają, że programy służące do rozmieszczania modułów w układzie wymagają bardzo długiego czasu na wykonanie obliczeń. Zastosowanie dedykowanego sprzętu umożliwiłoby wykonywanie znacznej części obliczeń w sposób sprzętowy, co umożliwiłoby znaczne skrócenie czasu obliczeń. Powyższe rozwiązanie może znaleźć

zastosowanie podczas projektowania wszystkich rodzajów układów FPGA oraz układów o topografii wierszowej: matryc bramkowych GA, układów Sea-of-Gates oraz układów Standard Cell z komórkami logicznymi o jednakowych rozmiarach. Sieć Hopfielda może być również zastosowana podczas rozmieszczania szczegółowego w układach o topografii wierszowej, w przypadku, gdy rozmieszczenie globalne modułów zostało wykonane inną metodą rozmieszczania. Podczas rozmieszczania szczegółowego modułów w układach o topografii wierszowej wykonuje się korekcję rozmieszczenia małych grup modułów. Takie rozwiązanie jest stosowane w programach Capo i Feng Shui. Korekcja jest wykonywana wzdłuż wierszy w tzw. oknie (ang. window). Okno jest przesuwane wzdłuż wszystkich wierszy od początku do końca wiersza. Celem korekcji jest minimalizacja całkowitej długości połączeń w oknie. Po każdym wykonaniu korekcji rozmieszczenia modułów, okno jest przesuwane o połowę jego szerokości. Sieć Hopfielda może być również zastosowana do powyższej korekcji rozmieszczenia modułów [43, 45, 146].

Rozmieszczanie modułów z użyciem sieci Hopfielda w bardzo dużych układach jest możliwe w wyniku zastosowania algorytmu min-cut do podziału układu na części, w których następnie może być wykonane rozmieszczanie z użyciem sieci Hopfielda lub z użyciem równoległego przetwarzania z jednostkami przetwarzającymi w postaci sieci Hopfielda. Innym rozwiązaniem może być łączenie modułów w klastry, które umożliwia zmniejszenie złożoności problemu rozmieszczania podczas wyznaczania rozmieszczenia globalnego modułów [2, 148]. Powyższe rozwiązania są powszechnie stosowane podczas rozmieszczania modułów w bardzo dużych układach VLSI, w których nie jest możliwe bezpośrednie zastosowanie danej metody rozmieszczania.

Przedstawiony schemat rozmieszczania modułów z użyciem równoległego przetwarzania może być również zastosowany dla jednostek przetwarzających, które wykorzystują inną metodę rozmieszczania.

10.2. Podsumowanie

W niniejszej pracy zbadano możliwości zastosowania sieci Hopfielda do optymalizacji topografii systemów wielkiej skali integracji. Wprowadzono oryginalne modyfikacje w klasycznym modelu sieci Hopfielda, które umożliwiają efektywne zastosowanie sieci w problemach optymalizacyjnych. W wyniku wprowadzonych modyfikacji, sieć Hopfielda umożliwiła dla problemu komiwojażera otrzymanie bardzo dobrych rezultatów, które nie zostały osiągnięte w pracach innych badaczy, poświęconych temu problemowi. Dokonano przeglądu metod rozmieszczania modułów, stosowanych podczas projektowania topografii układów VLSI. Wprowadzono oryginalne modyfikacje w sieci Hopfielda, które umożliwiają zastosowanie sieci w rzeczywistych układach z ustalonym położeniem końcówek całego układu, ustalonym położeniem wybranych modułów oraz w przypadku, gdy pewne części podłoża nie mogą być wykorzystane do rozmieszczania modułów. Oryginalnie zastosowano sieć Hopfielda do równoczesnego rozmieszczania modułów i końcówek całego układu. Oryginalne jest również rozwiązanie, umożliwiające minimalizację

długości połączeń w układach, w których węzły posiadają dowolną liczbę końcówek. Wykorzystano również oryginalnie sposób korekcji estymacji długości połączeń dla węzłów zawierających wiele końcówek. Bardzo dobre wyniki rozmieszczania modułów osiągnięto w wyniku oryginalnego zastosowania równoległego przetwarzania, z jednostkami przetwarzającymi w postaci sieci Hopfielda. Przedstawiono liczne możliwości zastosowania zaproponowanych rozwiązań.

Wadą sieci Hopfielda w przypadku zastosowania sieci do rozwiązywania problemów optymalizacyjnych jest szybko wzrastająca liczba neuronów oraz połączeń między neuronami, wraz ze wzrostem wymiaru problemu. Powyższa sytuacja występuje również w przypadku użycia sieci Hopfielda do rozmieszczania modułów w układach VLSI. Rozwiązaniem tego problemu jest zaproponowana metoda, polegająca na zastosowaniu równoległego przetwarzania, przez wiele równocześnie pracujących jednostek przetwarzających. Każda z jednostek przetwarzających jest siecią Hopfielda, która wykonuje rozmieszczanie w danej części układu. Powyższe rozwiązanie umożliwia znaczne zmniejszenie liczby neuronów niezbędnych do wykonania rozmieszczenia modułów i liczby połączeń między neuronami, w porównaniu do rozwiązania, w którym jedna sieć Hopfielda wykonywałaby rozmieszczenie wszystkich modułów układu. Rezultaty otrzymane z wykorzystaniem równoległego przetwarzania są bardzo dobre, ponieważ są porównywalne lub nawet lepsze od wyników uzyskanych z użyciem programu VPR, który jest zaliczany do wiodących programów rozmieszczania oraz trasowania połączeń dla układów FPGA. Przedstawione rozwiązanie może być również zastosowane w innych problemach optymalizacyjnych.

Innym ograniczeniem sieci Hopfielda jest możliwość rozmieszczania jedynie modułów o takich samych rozmiarach. Powyższe ograniczenie nie występuje w przypadku rozmieszczania komórek logicznych w układach FPGA, których zastosowanie stale wzrasta. Powyższe ograniczenie nie występuje również w matrycach bramkowych GA, układach Sea-of-Gates oraz układach Standard Cell z komórkami logicznymi o jednakowych rozmiarach.

Badania przeprowadzone w niniejszej pracy dowodzą, że sieć Hopfielda może dostarczać rozwiązań o bardzo dobrej jakości. Sieć Hopfielda można zastosować do minimalizacji długości połączeń w układach VLSI, w których węzły posiadają dowolną liczbę końcówek. Minimalizacja długości połączeń w układach VLSI może być wykonana z użyciem równoległego przetwarzania, z jednostkami przetwarzającymi w postaci sieci Hopfielda, a powyższe rozwiązanie umożliwia otrzymanie rozmieszczeń o bardzo dobrej jakości. Teza postawiona w niniejszej pracy została udowodniona.

Przedstawione możliwości zastosowania sieci Hopfielda w optymalizacji topografii systemów VLSI są podstawowymi kierunkami dalszych prac. Chociaż niniejsza praca jest poświęcona problemowi optymalizacji topografii systemów VLSI, to zaproponowane rozwiązania można również zastosować w innych problemach optymalizacyjnych.

Dodatek
Zawartość płyty CD

Płyta CD załączona do niniejszej pracy zawiera:

1. Katalog **Kody źródłowe programów** – kody źródłowe i dokumentacja programów wykorzystanych w pracy, opis ich wykorzystania podczas rozmieszczania modułów oraz opis formatu danych w stosowanych plikach danych. Kody źródłowe programów zostały napisane w języku C++ oraz skompilowane i zlinkowane z użyciem pakietu Borland C++ Builder 6.0 Personal.
    a. Katalog **Współczynniki korygujące** – kod źródłowy oraz dokumentacja programu **korekcja.exe**
    b. Katalog **Konwersja pliku .net** – kod źródłowy oraz dokumentacja programu **net.exe**
    c. Katalog **vpr_do_hopfield** – kod źródłowy oraz dokumentacja programu **vpr_do_hopfield.exe**
    d. Katalog **Symulacja sieci Hopfielda** – kod źródłowy oraz dokumentacja programu **hopfield.exe**
    e. Katalog **Równoległe przetwarzanie** – kod źródłowy oraz dokumentacja programu **rownolegle.exe**
    f. Katalog **Wizualizacja** – kod źródłowy oraz dokumentacja programu **Wizualizacja.exe**
    g. Katalog **Format plików danych** – opis formatu danych w stosowanych plikach danych
    h. Plik **opis.txt** – opis wykorzystania programów podczas rozmieszczania modułów
2. Katalog **Symulacje** – pliki z danymi oraz programy umożliwiające przeprowadzenie symulacji i obliczeń przedstawionych w poszczególnych rozdziałach pracy.
    a. Katalog **Rozdział 5.2**
    b. Katalog **Rozdział 6.2**
    c. Katalog **Rozdział 7.2.2**
    d. Katalog **Rozdział 7.3**
    e. Katalog **Rozdział 8.3**
    f. Katalog **Rozdział 8.3.1**
    g. Plik **opis.txt** – opis wykorzystania programów podczas rozmieszczania modułów

Bibliografia

1. M. J. S. Smith: Application-Specific Integrated Circuits. Addison Wesley Longman, 1997
2. A. Kos: Modelowanie hybrydowych układów mocy i optymalizacja ich konstrukcji ze względu na rozkład temperatury. Kraków, Wydawnictwa AGH, 1994
3. B. T. Preas, M. J. Lorenzetti (red.): Physical Design Automation of VLSI Systems. Menlo Park, Benjamin-Cummings, 1988
4. T. Ohtsuki (red.): Layout Design and Verification. Elsevier Science Publishers B. V. (North-Holland), 1986
5. M. M. Vai: VLSI Design. CRC Press, 2001
6. C. Sechen: VLSI Placement and Global Routing Using Simulated Annealing. Boston, Kluwer Academic Publishers, 1988
7. D. F. Wong, H. W. Leong, C. L. Liu: Simulated Annealing for VLSI Design. Kluwer Academic Publishers, 1988
8. W. Wolf: Modern VLSI Design: a systems approach. Englewood Cliffs, New Jersey, PTR Prentice Hall, 1994
9. K. Shahookar, P. Mazumder: VLSI Cell Placement Techniques. ACM Computing Surveys, 1991, vol. 23, pp. 143-220
10. V. Betz, J. Rose: VPR: A New Packing, Placement and Routing Tool for FPGA Research. Proc. 7th International Workshop on Field Programmable Logic and Applications, London, 1997, pp. 213-222, http://www.eecg.toronto.edu/~vaughn/papers/fpl97.pdf
11. J.-G. Kim, Y.-D. Kim: A Linear Programming-Based Algorithm for Floorplanning in VLSI Design. IEEE Transactions on Computer-Aided Design, 2003, vol. 22, pp. 584-592
12. M. Mogaki, C. Miura, H. Terai: Algorithm for Block Placement with Size Optimization Technique by the Linear Programming Approach. Proc. of the IEEE International Conference on Computer-Aided Design, 1987, pp. 80-83
13. Y.-T. Lai, S. M. Leinwand: Algorithms for Floorplan Design Via Rectangular Dualization. IEEE Transactions on Computer-Aided Design, 1988, vol. 7, pp. 1278-1289
14. S. Wimer, I. Koren, I. Cederbaum: Optimal Aspects Ratios of Building Blocks in VLSI. IEEE Transactions on Computer-Aided Design, 1989, vol. 8, pp. 139-145
15. K. Chong, S. Sahni: Optimal Realisations of Floorplans. IEEE Transactions on Computer-Aided Design, 1993, vol. 12, pp. 793-801
16. G. K.-H. Yeap, M. Sarrafzadeh: A Unified Approach to Floorplan Sizing and Enumeration. IEEE Transactions on Computer-Aided Design, 1993, vol. 12, pp. 1858-1867
17. W. Shi: A Fast Algorithm for Area Minimization of Slicing Floorplans. IEEE Transactions on Computer-Aided Design, 1996, vol. 15, pp. 1525-1532

18. P. S. Dasgupta, S. Sur-Kolay, B. Bhattacharya: A Unified Approach to Topology Generation and Optimal Sizing of Floorplans. IEEE Transactions on Computer-Aided Design, 1998, vol. 17, pp. 126-135
19. P. Pan, C. L. Liu: Area Minimization for Floorplans. IEEE Transactions on Computer-Aided Design, 1995, vol. 14, pp. 123-132
20. H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani: VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair. IEEE Transactions on Computer-Aided Design, 1996, vol. 15, pp. 1518-1524
21. H. Murata, K. Fujiyoshi, M. Kaneko: VLSI/PCB Placement with Obstacles Based on Sequence Pair. IEEE Transactions on Computer-Aided Design, 1998, vol. 17, pp. 60-68
22. K. Fujiyoshi, H. Murata: Arbitrary Convex and Concave Rectilinear Block Packing Using Sequence-Pair. IEEE Transactions on Computer-Aided Design, 2000, vol. 19, pp. 224-233
23. S Nakatake, K. Fujiyoshi, H. Murata, Y. Kajitani: Module Packing Based on the BSG-Structure and IC Layout Applications. IEEE Transactions on Computer-Aided Design, 1998, vol. 17, pp. 519-530
24. E. F. Y. Young, C. C. N. Chu, Z. C. Shen: Twin Binary Sequences: A Nonredundant Representation for General Nonslicing Floorplan. IEEE Transactions on Computer-Aided Design, 2003, vol. 22, pp. 457-469
25. Y. Feng, D. P. Mehta, H. Yang: Constrained Floorplanning Using Network Flows. IEEE Transactions on Computer-Aided Design, 2004, vol. 23, pp. 572-580
26. D. Sitaram, Y. Zheng, K. L. Shepard: Full-Chip Three-Dimensional Shapes-Based RLC Extraction. IEEE Transactions on Computer-Aided Design, 2004, vol. 23, pp. 711-727
27. A. Kos, Z. Nagórny: Estymacja długości połączeń w układach VLSI. IV Krajowa Konferencja Elektroniki, Darłówko Wschodnie, czerwiec 2005, pp. 159-164
28. M. A. Breuer (red.): Automatyczne projektowanie maszyn cyfrowych. Warszawa, PWN, 1976
29. D. S. Rao, J. D. Provence: An integrated approach to routing and via minimization. Information Processing Letters, 1991, vol. 39, pp. 257-263
30. T. C. Hu, E. S. Kuh (red.): VLSI Circuit Layout: Theory and Design. New York, IEEE Press, 1985
31. C. J. Alpert, G.-J. Nam, P. G. Villarrubia: Effective Free Space Management for Cut-Based Placement via Analytical Constraint Generation. IEEE Transactions on Computer-Aided Design, 2003, vol. 22, pp. 1343-1353
32. B. Krishnamurthy: An Improved Min-Cut Algorithm for Partitioning VLSI Networks. IEEE Transactions on Computers, 1984, vol. 33, pp. 438-446
33. A. E. Dunlop, B. W. Kernighan: A Procedure for Placement of Standard-Cell VLSI Circuits. IEEE Transactions on Computer-Aided Design, 1985, vol. 4, pp. 92-98
34. P. R. Suaris, G. Kedem: An Algorithm for Quadrisection and Its Application to Standard Cell Placement. IEEE Transactions on Circuits and Systems, 1988, vol. 35, pp. 294-302

35. A. E. Caldwell, A. B. Kahng, I. L. Markov: Design and Implementation of Move-Based Heuristics for VLSI Hypergraph Partitioning. ACM Journal of Experimental Algorithmics, 2000, vol. 5, artykuł 5 http://www.eecs.umich.edu/~imarkov/pubs/jour/j004.pdf
36. C. J. Alpert, J.-H. Huang, A. B. Kahng: Multilevel Circuit Partitioning. IEEE Transactions on Computer-Aided Design, 1998, vol. 17, pp. 655-667
37. G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar: Multilevel Hypergraph Partitioning: Applications in VLSI Domain. IEEE Transactions on VLSI Systems, 1999, vol. 7, pp. 69-79
38. G. Karypis, V. Kumar: Multilevel *k*-way Hypergraph Partitioning. Proc. of the Design Automation Conference, 1999, pp. 343-348, http://www-users.cs.umn.edu/~karypis/publications/Papers/PDF/khmetis.pdf
39. University of Minnesota, USA; George Karypis's Home Page, http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.html
40. A. E. Caldwell, A. B. Kahng, I. L. Markov: Can Recursive Bisection Alone Produce Routable Placements? Design Automation Conference, 2000, pp. 477-482, http://www.eecs.umich.edu/~imarkov/pubs/conf/c015.pdf
41. A. E. Caldwell, A. B. Kahng, I. L. Markov: Improved Algorithms for Hypergraph Bipartitioning. Proc. Asia and South Pacific Design Automation Conference, 2000, pp. 661-666, http://www.eecs.umich.edu/~imarkov/pubs/conf/c013.pdf
42. A. E. Caldwell, A. B. Kahng, I. L. Markov: Iterative Partitioning with Varying Node Weights. VLSI Design, 2000, vol. 11, pp. 249-258 http://www.eecs.umich.edu/~imarkov/pubs/jour/j006.pdf
43. A. E. Caldwell, A. B. Kahng, I. L. Markov: Optimal Partitioners and End-Case Placers for Standard-Cell Layout. IEEE Transactions on Computer-Aided Design, 2000, vol. 19, pp. 1304-1313
44. M. Wang, X. Yang, M. Sarrafzadeh: Dragon2000: Standard-Cell Placement Tool for Large Industry Circuits. International Conference on Computer-Aided Design, 2000, pp. 260-263, http://er.cs.ucla.edu/Dragon/papers/dragon.pdf
45. A. R. Agnihotri, S. Ono, C. Li, M. C. Yildiz, A. Khatkhate, C.-K. Koh, P. H. Madden: Mixed Block Placement via Fractional Cut Recursive Bisection. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 748-761
46. M. C. Yildiz, P. H. Madden: Improved Cut Sequences for Partitioning Based Placement. Proc. of the Design Automation Conference, 2001, pp. 776-779, http://vlsicad.cs.binghamton.edu/pubs/Yildiz01dac.pdf
47. M. C. Yildiz, P. H. Madden: Global Objectives for Standard Cell Placement. Proc. of the Great Lakes Symposium on VLSI, 2001, pp. 68-72, http://vlsicad.cs.binghamton.edu/pubs/Yildiz010068.pdf
48. A. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, P. H. Madden: Fractional Cut: Improved Recursive Bisection Placement. Proc. of the International Conference on Computer-Aided Design, 2003, pp. 307-310 , http://vlsicad.cs.binghamton.edu/pubs/Agnihotri03.pdf
49. D. J.-H. Huang, A. B. Kahng: Partitioning-Based Standard-Cell Global Placement with an Exact Objective. Proc. International Symposium on Physical Design, 1997, pp. 18-25, http://citeseer.ist.psu.edu/71921.html

50. VLSI CAD Bookshelf Slots and Entries, A. B. Kahng, I. L Markov, http://vlsicad.eecs.umich.edu/BK/Slots/slots/WirelengthdrivenStandardCellPlacement.html
51. P. H. Madden: Reporting of Standard Cell Placement Results. IEEE Transactions on Computer-Aided Design, 2002, vol. 21, pp. 240-247
52. S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, P. H. Madden: Benchmarking for Large-Scale Placement and Beyond. IEEE Transactions on Computer-Aided Design, 2004, vol. 23, pp. 472-486
53. O. Liu, M. Marek-Sadowska: A Study of Netlist Structure and Placement Efficiency. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 762-772
54. C.-C. Chang, J. Cong, M. Romesis, M. Xie: Optimality and Scalability Study of Existing Placement Algorithms. IEEE Transactions on Computer-Aided Design, 2004, vol. 23, pp. 537-549
55. M. Gajęcki, A. Kos: A Problem of Optimization of Topography of VLSI Circuit. Proc. of the XIXth National Conference on Circuit Theory and Electronic Networks, Kraków-Krynica (Poland), October 1996, pp. II/307-312
56. X. Yang, B.-K. Choi, M. Sarrafzadeh: Timing-Driven Placement using Design Hierarchy Guided Constraint Generation. Proc. of the International Conference on Computer-Aided Design, 2002, pp. 177-184, http://er.cs.ucla.edu/Dragon/papers/timingdragon.pdf
57. R. Nair, C. L. Berman, P. S. Hauge, E. J. Yoffa: Generation of Performance Constraints for Layout. IEEE Transactions on Computer-Aided Design, 1989, vol. 8, pp. 860-874
58. H. Youssef, R.-B. Lin, E. Shragowitz: Bounds on Net Delays for VLSI Circuits. IEEE Transactions on Circuits and Systems, 1992, vol. 39, pp. 815-824
59. H. Ren, D. Z. Pan, D. S. Kung: Sensitivity Guided Net Weighting for Placement-Driven Synthesis. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 711-721
60. Q. Liu, B. Hu, M. Marek-Sadowska: Individual Wire-Length Prediction With Application to Timing-Driven Placement. IEEE Transactions on VLSI Systems, 2004, vol. 12, pp. 1004-1014
61. T. Łuba, K. Jasiński, B. Zbierzchowski: Specjalizowane układy cyfrowe w strukturach PLD i FPGA. Warszawa, Wydawnictwa Komunikacji i Łączności, 1997
62. T. Łuba, B. Zbierzchowski: Komputerowe projektowanie układów cyfrowych. Warszawa, Wydawnictwa Komunikacji i Łączności, 2000
63. A. Korytowski, M. Ziółko: Metody optymalizacji z ćwiczeniami laboratoryjnymi. Kraków, Wydawnictwa AGH, 1992
64. S. Sutanthavibul, E. Shragowitz, J. B. Rosen: An Analytical Approach to Floorplan Design and Optimization. IEEE Transactions on Computer-Aided Design, 1991, vol. 10, pp. 761-769
65. S. Goto: An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout. IEEE Transactions on Circuits and Systems, 1981, vol. 28, pp. 12-18
66. W. Findeisen, J. Szymanowski, A. Wierzbicki: Teoria i metody obliczeniowe optymalizacji. Warszawa, PWN, 1977

67. W. Grabowski: Programowanie matematyczne. Warszawa, Państwowe Wydawnictwo Ekonomiczne, 1982
68. I. Dziubiński, T. Świątkowski (red.): Poradnik matematyczny. Warszawa, PWN, 1982
69. S. Osowski: Sieci neuronowe w ujęciu algorytmicznym. Warszawa, WNT, 1996
70. J. M. Kleinhans, G. Sigl, F. M. Johannes, K. J. Antreich: GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization. IEEE Transactions on Computer-Aided Design, 1991, vol. 10, pp. 356-365
71. G. Sigl, K. Doll, F. M. Johannes: Analytical Placement: A Linear or a Quadratic Objective Function? Design Automation Conference, 1991, pp. 427-432, http://www.sigda.org/Archives/ProceedingArchives/Dac/
72. K. Doll, F. M. Johannes, K. J. Antreich: Iterative Placement Improvement by Network Flow Methods. IEEE Transactions on Computer-Aided Design, 1994, vol. 13, pp. 1189-1200
73. R. Sedgewick: Algorithms in C, Part 5: Graph Algorithms. Addison Wesley Professional, 2002
74. R. Sedgewick: Algorytmy w C++. Część 5. Grafy. Warszawa, Wydawnictwo RM, 2003
75. H. Eisenmann, F. M. Johannes: Generic Global Placement and Floorplanning. Proc. of the Design Automation Conference, 1998, pp. 269-274, http://www.sigda.org/Archives/ProceedingArchives/Dac/
76. B. Obermeier, H. Ranke, F. M. Johannes: Kraftwerk - A Versatile Placement Approach. Proc. of the International Symposium on Physical Design, 2005, pp. 242-244, http://www.sigda.org/Archives/ProceedingArchives/Ispd/
77. N. Viswanathan, C. C.-N. Chu: FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 722-733
78. F. Mo, A. Tabbara, R. K. Brayton: A Force-Directed Macro-Cell Placer. Proc. International Conference on Computer-Aided Design, 2000, pp. 177-180, http://www.sigda.org/Archives/ProceedingArchives/Iccad/
79. T. F. Chan, J. Cong, T. Kong, J. R. Shinnerl: Multilevel Optimization for Large-Scale Circuit Placement. Proc. of the International Conference on Computer-Aided Design, 2000, pp. 171-176, http://cadlab.cs.ucla.edu/~cong/papers/iccad00_placement.pdf
80. T. F. Chan, J. Cong, T. Kong, J. R. Shinnerl, K. Sze: An Enhanced Multilevel Algorithm for Circuit Placement. Proc. of the International Conference on Computer-Aided Design, 2003, pp. 299-306, http://ballade.cs.ucla.edu/~cong/papers/mpl2.pdf
81. C.-K. Cheng, E. S. Kuh: Module Placement Based on Resistive Network Optimization. IEEE Transactions on Computer-Aided Design, 1984, vol. 3, pp. 218-225
82. R.-S. Tsay, E. S. Kuh, C.-P. Hsu: PROUD: A Fast Sea-Of-Gates Placement Algorithm. Proc. of the Design Automation Conference, 1988, pp. 318-323

83. L. Sha, T. Blank: ATLAS - A Technique for Layout using Analytic Shapes. Proc. of the International Conference on Computer-Aided Design, 1987, pp. 84-87, http://www.sigda.org/Archives/ProceedingArchives/Iccad/Last20/Papers/1987/
84. B. Hu, M. Marek-Sadowska: Multilevel Fixed-Point-Addition-Based VLSI Placement. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 1188-1203
85. A. B. Kahng, Q. Wang: Implementation and Extensibility of an Analytic Placer. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 734-747
86. H. H. Chan, I. L. Markov: Practical Slicing and Non-slicing Block-Packing without Simulated Annealing. Proc. of the Great Lakes Symposium on VLSI, 2004, pp. 282-287, http://vlsicad.eecs.umich.edu/BK/BloBB/PAPERS/p037-chan.pdf
87. VLSI CAD Bookshelf Slots and Entries, A. B. Kahng, I. L Markov, http://vlsicad.eecs.umich.edu/BK/Slots/slots/BlockPacking.html
88. J. Rutkowski: Heuristic network partitioning algorithm using the concept of loop index. IEE Proceedings, Pt G, 1984, vol. 131, pp. 203-208
89. J. Rutkowski, L. Zielinski: Using Evolutionary Techniques for Chosen Optimization Problems Related to Analog Circuits Design. Proc. of the ECCTD, Kraków (Poland), 2003, pp. 313-316
90. S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi: Optimization by Simulated Annealing. Science, 1983, vol. 220, no. 4598, pp. 671-680
91. D. T. Pham, D. Karaboga: Intelligent Optimisation Techniques. Springer-Verlag London Limited, 2000
92. J. Mańdziuk: Sieci neuronowe typu Hopfielda. Teoria i przykłady zastosowań. Warszawa, Akademicka Oficyna Wydawnicza EXIT, 2000
93. J. A. Freeman, D. M. Skapura: Neural networks: algorithms, applications, and programming techniques. Addison-Wesley Publishing Company, 1991
94. T. Khanna: Foundations of neural networks. Addison-Wesley Publishing Company, 1990
95. C.Sechen, A. Sangiovanni-Vincentelli: The TimberWolf Placement and Routing Package. IEEE Journal of Solid-State Circuits, 1985, vol. 20, pp. 510-522
96. C. Sechen, K.-W. Lee: An Improved Simulated Annealing Algorithm for Row-Based Placement. Proc. of the International Conference on Computer-Aided Design, 1987, pp. 478-481, http://www.sigda.org/Archives/ProceedingArchives/Iccad/
97. C. Sechen, D. Braun, A. Sangiovanni-Vincentelli: ThunderBird: A Complete Standard Cell Layout Package. IEEE Journal of Solid-State Circuits, 1988, vol. 23, pp. 410-420
98. W. Swartz, C. Sechen: New Algorithms for the Placement and Routing of Macro Cells. Proc. of the International Conference on Computer-Aided Design, 1990, pp. 336-339, http://www.sigda.org/Archives/ProceedingArchives/Iccad/
99. W.-J. Sun, C. Sechen: Efficient and Effective Placement for Very Large Circuits. IEEE Transactions on Computer-Aided Design, 1995, vol. 14, pp. 349-359
100. Strona internetowa firmy InternetCAD.com, http://www.internetcad.com

101. C.-C. Chang, J. Cong, Z. Pan, X. Yuan: Multilevel Global Placement With Congestion Control. IEEE Transactions on Computer-Aided Design, 2003, vol. 22, pp. 395-409
102. C.-C. Chang, J. Cong, X. Yuan: Multi-level Placement for Large-Scale Mixed-Size IC Design. Proc. of the Asia and South Pacific Design Automation Conference, 2003, pp. 325-330, http://cadlab.cs.ucla.edu/~cong/papers/aspdac03.pdf
103. A. Marquardt, V. Betz, J. Rose: Timing-Driven Placement for FPGAs. Proc. International Symposium on FPGA, 2000, pp. 203-213, http://www.sigda.org/Archives/ProceedingArchives/Compendiums/papers/fpga/confsym.htm
104. P. Maidee, C. Ababei, K. Bazargan: Timing-Driven Partitioning-Based Placement for Island Style FPGAs. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 395-406
105. VLSI CAD Bookshelf Slots and Entries, A. B. Kahng, I. L Markov, http://vlsicad.eecs.umich.edu/BK/Slots/slots/FPGALayout.html
106. F. Balasa, S. C. Maruvada, K. Krishnamoorthy: On the Exploration of the Solution Space in Analog Placement With Symmetry Constraints. IEEE Transactions on Computer-Aided Design, 2004, vol. 23, pp. 177-191
107. J. J. Hopfield, D. W. Tank: “Neural” computation of decisions in optimization problems. Biological Cybernetics, 1985, vol. 52, pp. 141-152
108. R. Tadeusiewicz: Sieci neuronowe. Warszawa, Akademicka Oficyna Wydawnicza, 1993
109. J. Hertz, A. Krogh, R. G. Palmer: Wstęp do teorii obliczeń neuronowych. Warszawa, WNT, 1995
110. J. Żurada, M. Barski, W. Jędruch: Sztuczne sieci neuronowe. Warszawa, Wydawnictwo Naukowe PWN, 1996
111. M. Glesner, W. Pöchmüller: Neurocomputers. London, Chapman & Hall, 1994
112. C.-X. Zhang, D. A. Mlynski: Mapping and Hierarchical Self-Organizing Neural Networks for VLSI Placement. IEEE Transactions on Neural Networks, 1997, vol. 8, pp. 299-314
113. A. Hemani, A. Postula: Cell Placement by Self-Organisation. Neural Networks, 1990, vol. 3, pp. 377-383
114. P. Bratek, A. Kos: Complex Optimisation of Topology of VLSI Circuits with Self-Organising Neural Nets. Proc. of the MIXDES 1996 Mixed Design of Integrated Circuits and Systems, Łódź (Poland), June 1996, pp. 78-83
115. P. Bratek, A. Kos: Self-Organising Neural Computations for Optimisation of IC Topography in Thermal Aspect. Proc. of the XIXth National Conference on Circuit Theory and Electronic Networks, Kraków-Krynica (Poland), October 1996, pp. II/627-632
116. Y. Takefuji, K.-C. Lee, H. Aiso: An artificial maximum neural network: a winner-take-all neuron model forcing the state of the system in a solution domain. Biological Cybernetics, 1992, vol. 67, pp. 243-251
117. A. Kos, Z. Nagórny: Estymacja długości połączeń w układach VLSI. Elektronika, 2005, rok 46, nr 11, pp. 47-49
118. K. Shahookar, P. Mazumder: A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization. IEEE Transactions on Computer-Aided Design, 1990, vol. 9, pp. 500-511

119. R. M. Kling, P. Banerjee: ESP: Placement by Simulated Evolution. IEEE Transactions on Computer-Aided Design, 1989, vol. 8, pp. 245-256
120. S. N. Adya, I. L. Markov: Fixed-Outline Floorplanning: Enabling Hierarchical Design. IEEE Transactions on VLSI Systems, 2003, vol. 11, pp. 1120-1135
121. S. N. Adya, I. L. Markov: Consistent Placement of Macro-Blocks Using Floorplanning and Standard-Cell Placement. Proc. of the International Symposium on Physical Design, 2002, pp. 12-17, http://www.sigda.org/Archives/ProceedingArchives/Ispd/
122. G. Reinelt: Traveling salesman problem library. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/
123. M. G. Wrighton, A. M. DeHon: Hardware-Assisted Simulated Annealing with Application for Fast FPGA Placement. Proc. of the International Symposium on FPGA, 2003, pp. 33-42, http://www.sigda.org/Archives/ProceedingArchives/Compendiums/papers/fpga/confsym.htm
124. K. Wiatr: Akceleracja obliczeń w systemach wizyjnych. Warszawa, WNT, 2003
125. VLSI CAD Bookshelf 2, A. B. Kahng, I. L Markov, http://vlsicad.eecs.umich.edu/BK/
126. H. Mączka, P. Dziurdzia, A. Kos: Neural Algorithm for Minimisation of Total Length of Connections in VLSI Circuits. Proc. of the XIXth National Conference on Circuit Theory and Electronic Networks, Kraków-Krynica (Poland), October 1996, pp. II/319-324
127. A. Kos, Z. Nagórny: Minimalizacja długości połączeń w układach elektronicznych z wykorzystaniem sieci Hopfielda. Kwartalnik Elektroniki i Telekomunikacji, 2005, tom 51, z. 1, pp. 55-72
128. Z. Nagórny, A. Kos: Optymalizacja z wykorzystaniem zmodyfikowanej sieci Hopfielda. Kwartalnik Elektroniki i Telekomunikacji, 2005, tom 51, z. 2, pp. 255-275
129. A. Kos, Z. Nagórny: A Modified Hopfield Neural Network for VLSI Placement. Proc. of the MIXDES 2005 Mixed Design of Integrated Circuits and Systems, Kraków (Poland), June 2005, vol. 1, pp. 33-38
130. B. Kamgar-Parsi, B. Kamgar-Parsi: On Problem Solving with Hopfield Neural Networks. Biological Cybernetics, 1990, vol. 62, pp. 415-423
131. G. V. Wilson, G. S. Pawley: On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank. Biological Cybernetics, 1988, vol. 58, pp. 63-70
132. S. Z. Li: Improving Convergence and Solution Quality of Hopfield-Type Neural Networks with Augmented Lagrange Multipliers. IEEE Transactions on Neural Networks, 1996, vol. 7, pp. 1507-1516
133. VLSI CAD Bookshelf Slots and Entries, A. B. Kahng, I. L Markov, Rectilinear Steiner Minimum Tree Slot, http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/
134. P. Winter: Steiner Problem in Networks: A Survey. Networks, 1991, vol. 17, pp. 129-167
135. C. Ebeling, L. McMurchie, S. A. Hauck, S. Burns: Placement and Routing Tools for the Triptych FPGA. IEEE Transactions on VLSI Systems, 1995, vol. 3, pp. 473-482
136. A. Drozdek, D. L. Simon: Struktury danych w języku C. Warszawa, WNT, 1996

137. University of Toronto, Kanada, Guy Lemieux's Home Page, przykłady układów FPGA w formacie SEGA: ftp://ftp.eecg.toronto.edu/pub/software/SEGA/SEGA-1.1.tar.gz, kod źródłowy programu do konwersji z formatu SEGA do BLIF: http://www.eecg.toronto.edu/~lemieux/sega/sega2blif.c
138. T.-Y. Ho, Y.-W. Chang, S.-J. Chen, D.-T. Lee: Crosstalk- and Performance-Driven Multilevel Full-Chip Routing. IEEE Transactions on Computer-Aided Design, 2005, vol. 24, pp. 869-878
139. K. Ueda, T. Komatsubara, T. Hosaka: A Parallel Processing Approach for Logic Module Placement. IEEE Transactions on Computer-Aided Design, 1983, vol. 2, pp. 39-47
140. S. A. Kravitz, R. A. Rutenbar: Placement by Simulated Annealing on a Multiprocessor. IEEE Transactions on Computer-Aided Design, 1987, vol. 6, pp. 534-549
141. A. Casotto, F. Romeo, A. Sangiovanni-Vincentelli: A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells. IEEE Transactions on Computer-Aided Design, 1987, vol. 6, pp. 838-847
142. S. Devadas, A. R. Newton: Topological Optimization of Multiple Level Array Logic: On Uni and Multi-processors. Proc. of the International Conference on Computer-Aided Design, 1986, pp. 38-41, http://www.sigda.org/Archives/ProceedingArchives/Iccad/
143. J. S. Rose, D. R. Blythe, W. M. Snelgrove, Z. G. Vranesic: Fast, High Quality VLSI Placement On An MIMD Multiprocessor. Proc. of the International Conference on Computer-Aided Design, 1986, pp. 42-45, http://www.sigda.org/Archives/ProceedingArchives/Iccad/
144. J. A. Chandy, S. Kim, B. Ramkumar, S. Parkes, P. Banerjee: An Evaluation of Parallel Simulated Annealing Strategies with Application to Standard Cell Placement. IEEE Transactions on Computer-Aided Design, 1997, vol. 16, pp. 398-410
145. M. Haldar, A. Nayak, A. Choudhary, P. Banerjee: Parallel Algorithms For FPGA Placement. Proc. of the Great Lakes Symposium on VLSI, 2000, pp. 86-94, http://www.sigda.org/Archives/ProceedingArchives/Compendiums/papers/glsvlsi/confsym.htm
146. F. H. Khundakjie, P. H. Madden, N. B. Abu-Ghazaleh, M. C. Yildiz: Parallel Standard Cell Placement on a Cluster of Workstations. IEEE International Conference on Cluster Computing, 2001, pp. 85-94, http://vlsicad.cs.binghamton.edu/pubs/Khundakjie010529.pdf
147. P. K. Chan, M. D. F. Schlag: Parallel Placement for Field-Programmable Gate Arrays. Proc. of the International Symposium on FPGA, 2003, pp. 43-50, http://www.sigda.org/Archives/ProceedingArchives/Compendiums/papers/fpga/confsym.htm
148. Y. Sankar, J. Rose: Trading Quality for Compile Time: Ultra-Fast Placement for FPGAs. Proc. of the International Symposium on FPGA, 1999, pp. 157-166, http://www.sigda.org/Archives/ProceedingArchives/Compendiums/papers/fpga/confsym.htm
149. Z. Nagórny, A. Kos: Projektowanie topografii systemów VLSI. Cz. 1. Style i etapy projektowania, rozmieszczanie modułów. Kwartalnik Elektroniki i Telekomunikacji, 2006, tom 52, z. 3, pp. 451-468

150. Z. Nagórny, A. Kos: Projektowanie topografii systemów VLSI. Cz. 2. Algorytm min-cut. Kwartalnik Elektroniki i Telekomunikacji, 2006, tom 52, z. 3, pp. 469-488
151. Z. Nagórny, A. Kos: Projektowanie topografii systemów VLSI. Cz. 3. Metody analityczne. Kwartalnik Elektroniki i Telekomunikacji, 2006, tom 52, z. 4, pp. 669-695
152. Z. Nagórny, A. Kos: Projektowanie topografii systemów VLSI. Cz. 4. Symulowane wyżarzanie, sieci neuronowe. Kwartalnik Elektroniki i Telekomunikacji, 2006, tom 52, z. 4, pp. 697-727
153. H. Qu, Z. Yi, H. Tang: Improving local minima of columnar competitive model for TSPs. IEEE Transactions on Circuits and Systems I, Regular Papers, 2006, vol. 53, pp. 1353-1362