POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Rozprawa doktorska

mgr inż. Robert Tomaszewski

Automatyczna synteza bezkolizyjnych sieci jednoukładowych dla systemów wbudowanych

promotor:

dr hab. inż. Stanisław Deniziak

GLIWICE, 2011

Składam serdeczne podziękowania mojemu promotorowi, Stanisławowi Deniziakowi,
za nieocenioną pomoc merytoryczną podczas powstawania niniejszej pracy.
Żonie i Rodzicom pragnę podziękować za wsparcie i cierpliwość.

Spis treści

1 Wstęp

Integrowanie coraz większej liczby funkcji w obecnych systemach wbudowanych powoduje wzrost zapotrzebowania na ich moc obliczeniową. Takie urządzenia jak systemy rozrywki multimedialnej i interaktywnej (przystawki do telewizji oferujące większą funkcjonalność niż tylko odbiór sygnałów wizji i fonii, konsole do gier, przenośne odtwarzacze multimedialne), kamery i aparaty cyfrowe, telefony komórkowe, bankomaty, systemy monitoringowo-alarmowe czy systemy sterujące pracą samochodów dalece przerastają możliwościami swoich protoplastów z lat sześćdziesiątych ubiegłego wieku. Poprawę wydajności systemu wbudowanego najprościej można osiągnąć zwiększając moc obliczeniową jego jednostki przetwarzającej (procesora), na przykład poprzez rozbudowę struktur logicznych. Zaobserwowano jednakże, że podwojenie liczby elementów logicznych prowadzi jedynie do 40% wzrostu wydajności [P99], co jest nieakceptowalne po uwzględnieniu kosztu związanego z zapotrzebowaniem na zasoby układu oraz poborem energii. Ograniczenie wydajności wynika przede wszystkim z niemożności równoległego wykonywania niezależnych od siebie zadań systemu wbudowanego. Rozwiązaniem stało się integrowanie wielu mniej skomplikowanych (często wyspecjalizowanych funkcjonalnie) jednostek przetwarzających, współpracujących w obrębie jednego układu. W tego typu systemach możliwe jest ograniczenie pobieranej mocy dzięki różnicowaniu napięcia i częstotliwości pracy rdzeni przetwarzających [S04] oraz ich selektywnemu wyłączaniu (na czas, gdy dany rdzeń/rdzenie przetwarzające nie wykonują żadnych zadań). Trend projektowania systemów jednoukładowych (ang. *System on Chip*, *SoC*) ewoluował zatem ku systemom pracującym równolegle (ang. *MultiProcessor System on Chip*, *MPSoC*) [S07]. W świetle aktualnego rozwoju technologicznego, architektury rozproszone MPSoC stanowią podstawę projektowania dla nowych, wysokowydajnych systemów wbudowanych [BLMNB07]. Rozwój rozproszonych systemów wbudowanych napotyka jednak na poważne bariery technologiczne:

- efekty submikronowe (ang. *Deep SubMicron*, *DSM*) – we wczesnych układach scalonych VLSI zjawiska takie jak opóźnienia na ścieżkach transmisyjnych czy zakłócenia międzyścieżkowe miały marginalne znaczenie ze względu na mały stopień upakowania elementów logicznych oraz stosunkowo niskie częstotliwości pracy zegara synchronizującego pracę systemu. Dla technologii wytwarzania mniejszych niż 250nm (aluminium) i 180nm (miedź) praca łącz wewnątrzukładowych staje się zauważalnie wolniejsza w stosunku do czasu przełączania bramek i bardziej zawodna [P06]. Dla szybkości transmisji poważną rolę zaczynają odgrywać długości ścieżek łączących elementy układu [HMH01]. Podwyższanie częstotliwości pracy układu w połączeniu

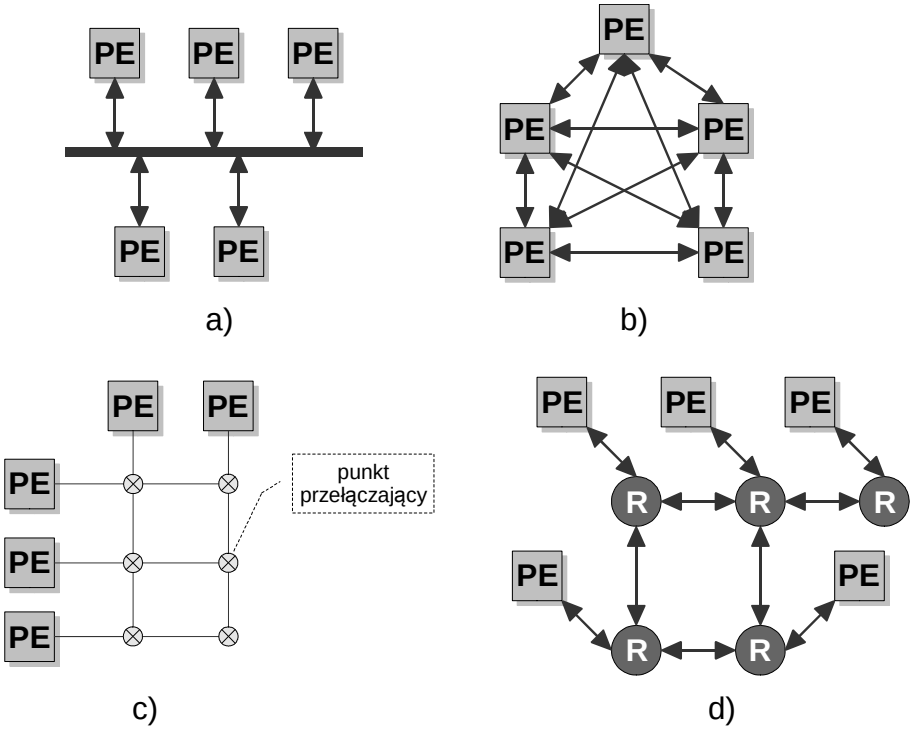
ze wzrostem skali integracji wpływa również na charakterystykę termiczną. Ten czynnik wraz z interferencjami wynikającymi z indukcyjności ścieżek jest przyczyną zakłóceń w pracy systemu. Wszystkie powyższe czynniki powodują, że w submikronowych technologiach wytwarzania układów koszt ścieżek połączeniowych (zasoby, pobierana moc, szybkość propagacji) jest wyższy od kosztu tranzystorów czy elementów logicznych,

- trudności z globalną synchronizacją pracy systemu – miniaturyzacja układów powoduje wzrost rozbieżności pomiędzy czasami przełączania bramek (układów logicznych), a opóźnieniami transmisji sygnałów za pomocą połączeń wewnątrzukładowych (pierwszy maleje, drugi nieoczekiwanie rośnie). W tradycyjnie projektowanych systemach MPSoC elementy przetwarzające połączone są wspólnym, rozgłoszeniowym medium transmisyjnym – magistralą. Wraz ze wzrostem jej długości rośnie również czas propagacji sygnału, w tym zegarowego. Proces technologiczny 35nm i mniejsze sprawia, że transmisja sygnału pomiędzy krańcami układu scalonego zajmuje od kilku do kilkudziesięciu cykli zegarowych (w zależności od częstotliwości pracy zegara) [AHKB00][SK00]. Coraz bardziej rozbudowane drzewa dystrybucji sygnału zegarowego prowadzą do zwiększenia zużycia energii oraz zasobów układu. Jedno z proponowanych rozwiązań to układy asynchroniczne z podziałem na synchroniczne partycje (ang. *Globally Asynchronous Locally Synchronous*, *GALS*) [AEJKR02]. W tego typu rozwiązaniach wydzielone grupy procesorów pracują z własnymi zegarami, natomiast komunikacja między grupami jest asynchroniczna. Trudności w skorelowaniu pracy rozproszonych systemów wbudowanych prowadzą do przesunięcia ciężaru podczas projektowania z części obliczeniowej (jakich elementów przetwarzających użyć?) na część komunikacyjną (jak je połączyć? jak mają wymieniać między sobą informacje?). Jest to również jedna z przyczyn powstania zjawiska określanego jako *design productivity gap*, czyli sytuacji, w której rozwój technologii przewyższa możliwości jej wykorzystania,
- skalowalność architektury komunikacyjnej – jak wspomniano wcześniej większość projektowanych systemów wbudowanych klasy MPSoC korzysta ze wspólnego medium transmisyjnego – magistrali. Taki sposób komunikowania sprawdza się w systemach, gdzie liczba komunikujących się jednostek nie przekracza kilku-kilkunastu. Powyżej tej liczby przepustowość magistrali znacznie spada, uniemożliwiając normalną pracę systemu. Potrzebne są zatem nowe rozwiązania służące łączeniu przesyłających informacje elementów w układach MPSoC [C06].

Reasumując szuka się takich architektur komunikacyjnych dla systemów jednoukładowych, które oferowałyby dużą skalowalność, efektywność pod kątem poboru mocy i zapotrzebowania na zasoby układu oraz eliminowałyby (lub zmniejszały) wpływ opóźnień transmisyjnych w technologiach submikronowych na pracę całego układu.

1.1 Wieloprocesorowe systemy wbudowane – podstawowe architektury

W rozproszonych systemach wbudowanych (lub bardziej ogólnie – w systemach wieloprocesorowych) komunikujące się ze sobą elementy przetwarzające można połączyć na cztery podstawowe sposoby, zobrazowane na Rys.1.1-1. Dla uproszczenia wszystkie przesyłające informacje jednostki systemu wbudowanego oznaczono „PE", natomiast element opisany jako „R" reprezentuje mikroruter (funkcjonalnie zbliżony do przełącznika/rutera stosowanego w sieciach komputerowych).



Rys. 1.1-1 Architektury MPSoC: magistralowa (a), w pełni połączona dedykowanymi łączami (b), z przełącznicą krzyżową (c) oraz sieć jednoukładowa (d).

Najprostszą z nich i powszechnie używaną jest wspólna magistrala (Rys. 1.1-1a). Dostęp do medium komunikacyjnego odbywa się na zasadzie wyłączności (dodatkowy układ – arbiter - przyznaje prawo do komunikacji), a transmisje mają charakter rozgłoszeniowy. Największym problemem jest bardzo słaba skalowalność – umiejscawianie w systemie kolejnych elementów przetwarzających powoduje spadek przepustowości transmisyjnej (rywalizacja o medium). Prócz tego rozbudowa prowadzi do coraz większych trudności w synchronizowaniu pracy wielu jednostek korzystających z tej samej magistrali wskutek opóźnień w propagacji sygnału

zegarowego. Rośnie również pobór mocy powodowany przez wzrost rezystancji ścieżek magistrali oraz pojemności pasożytniczej wnoszonej przez dołączane elementy. Rozwiązaniem jest segmentacja układu poprzez podział magistrali na połączone ze sobą mostkami i pracujące niezależnie odcinki [LR04]. Pozwala to również na zróżnicowanie szybkości działania wydzielonych części (magistrale hierarchiczne). Segmentacja ma jednak swoje granice (komplikacja w zarządzaniu, skalowalność), powyżej których pojawiają się te same problemy, co dla pojedynczej magistrali.

Drugim sposobem budowania infrastruktury komunikacyjnej są dedykowane połączenia pomiędzy przesyłającymi informacje jednostkami (ang. *Point-to-Point*, *P2P*). O ile takie podejście oferuje znakomite parametry, jeśli chodzi o skalowalność oraz przepustowość, o tyle koszt energetyczny oraz zapotrzebowanie na zasoby układu dyskwalifikują je w większości zastosowań MPSoC (duża liczba łącz, często słabo wykorzystywanych podczas pracy systemu, rozbudowane interfejsy dla dedykowanych kanałów komunikacyjnych). Jeśli transmisje każdego z procesorów systemu wbudowanego miałyby charakter rozgłoszeniowy lub byłyby typu każdy-z-każdym, wówczas rezultatem syntezy będzie architektura w pełni połączona (Rys. 1.1-1b). W praktyce infrastruktura komunikacyjna w rozproszonych systemach wbudowanych stanowi projektowaną ad-hoc mieszaninę magistral z rozwiązaniami P2P.

Kolejne podejście w projektowaniu architektur MPSoC, mające na celu poradzenie sobie z ograniczeniami wydajności magistral, to przełącznica krzyżowa zwana również macierzą przełączającą (ang. *crossbar switch*, *crossbar matrix*) (Rys. 1.1-1c). Połączenia między jednostkami PE podzielono na segmenty obsługiwane przez tzw. punkty przełączające. W zależności od źródła i celu danej transmisji przełączniki zestawiają dla niej najkrótszą ścieżkę. Wadami przełącznic są ograniczenia komunikacyjne (zasadniczo transmisje zachodzą pomiędzy elementami przetwarzającymi umiejscowionymi na osi X z tymi, które znajdują się na osi Y) oraz zapotrzebowanie na punkty przełączające (macierz PE o wymiarach MxN wymaga zastosowania M*N punktów). Zmniejszenie liczby przełączników osiągnięto poprzez zastosowanie tzw. sieci Omega. Stały się one jedną z inspiracji projektowych dla ostatniej, najnowszej grupy rozwiązań jaką są sieci jednoukładowe, zwane *Network-on-Chip* (NoC) (Rys. 1.1-1d). Architektura NoC stanowi niejako melanż koncepcji macierzy przełączających oraz sieci komputerowych – zamiast jednego, pełnego stopnia połączeń z rozbudowanym przełącznikiem wykorzystuje się wielostopniową sieć mniejszych, prostszych oraz tańszych (miara – zasoby układu). Komunikaty w takiej mikrosieci są przesyłane ścieżkami determinowanymi połączeniami międzyruterowymi, dlatego możliwy stopień zrównoleglenia transmisji dalece przewyższa ten, który oferują systemy oparte na wspólnej magistrali.

W porównaniu z architekturami korzystającymi z przełącznic krzyżowych NoC oferuje nieporównywalnie większą elastyczność w zestawianiu ścieżek komunikacyjnych oraz topologii połączeń. Podstawowe cechy obu rozwiązań wraz z oceną (zaleta/wada) umieszczono w Tabeli 1.1-1 za [GG00].

Wspólna magistrala	**Ocena**	**Sieć jednoukładowa**	**Ocena**
Każda dołączona jednostka przetwarzająca powoduje wzrost pojemności pasożytniczej magistrali	Wada	Dołączanie jednostek ma charakter lokalny (łącza do rutera), co nie pogarsza parametrów elektrycznych całego układu	Zaleta
Trudność w synchronizacji magistrali rośnie wraz z postępem procesu wytwarzania układów (DSM)	Wada	Ścieżki w sieci NoC są krótkie (typu P2P) i łatwo sterować nimi za pomocą sygnałów zegarowych	Zaleta
Arbitraż dostępu do medium transmisyjnego jest „wąskim gardłem” - jego wpływ rośnie wraz z rozbudową systemu	Wada	Ruting w mikrosieci może być zdecentralizowany – do dyspozycji zawsze jest kilka wirtualnych kanałów komunikacyjnych	Zaleta
Arbiter magistrali jest dostosowany do konkretnej aplikacji	Wada	Ten sam typ rutera może być stosowany w mikrosieciach dowolnych rozmiarów i dla dowolnych aplikacji	Zaleta
Przepustowość maleje wraz z rozbudową systemu	Wada	Przepustowość rośnie wraz z rozbudową systemu	Zaleta
Dzięki wyłącznemu dostępowi do medium komunikacyjnego można przewidzieć opóźnienia transmisji	Zaleta	Symultaniczne transmisje mogą powodować kolizje w mikrosieci co powoduje zatory spowalniające komunikację	Wada
Dopracowana, powszechnie znana metodyka projektowania i szeroki zakres kompatybilności z obecnymi jednostkami przetwarzającymi	Zaleta	Nowe rozwiązanie wymagające poznania przez projektantów i dodatkowego wysiłku przy stosowaniu (dopasowanie rozwiązań dla magistral do realiów mikrosieci)	Wada

Tabela 1.1-1 Porównanie architektur typu wspólna magistrala oraz sieć jednoukładowa [GG00].

Właściwości architektur opartych na magistralach oraz sieciach jednoukładowych zostały skonfrontowane w kilku pracach. Stosując syntetyczne benchmarki w [AMCBR06] porównano

wydajność oraz koszt zasobowo-energetyczny dla kilkunastordzeniowego systemu zbudowanego w oparciu o nowoczesną architekturę magistralową AMBA AHB (wraz z rozszerzeniem *MultiLayer*) [AMBA] z systemem NoC, przez autorów wymienionej pracy nazwanym xPipes [SACRB05]. Prototyp NoC przeważał na polu szybkości działania, przepustowości transmisji i zużycia energii odniesionego do czasu działania całego systemu. Podobnie praca [SRH08] potwierdziła przewagę wydajnościową dekodera MPEG-2 korzystającego z architektury sieci jednoukładowej nad rozwiązaniem AMBA AHB – metodami symulacyjno-analitycznymi wykazano prawie 250% zysk w szybkości przetwarzania obrazu dla systemów taktowanych identycznymi częstotliwościami zegarów. W [LCOM07] zsyntetyzowano prototypowe systemy oparte na architekturach: magistralowej, sieci jednoukładowej oraz P2P dla enkodera MPEG-2. Pod względem zasobochłonności najgorzej wypadła realizacja P2P, natomiast NoC dał rezultat podobny do magistrali. Porównanie wydajności (szybkość przetwarzania oraz przepustowość transmisji) wykazało, że NoC niewiele ustępuje rozwiązaniom P2P. Ostatni parametr - pobór mocy (odniesiony do zakodowanych ramek obrazu) - był dla sieci jednoukładowych zdecydowanie najmniejszy. Obszerne porównanie rozmaitych architektur komunikacyjnych dla platform FPGA zawarto w [MSCL06]. Podsumowując, podstawowe zalety architektury NoC to:

- bardzo dobra skalowalność (teoretycznie nieskończona, a dla niektórych rodzajów NoC niemal liniowa),
- duża przepustowość komunikacyjna przy jednoczesnym wysokim stopniu wykorzystania medium transmisyjnego,
- uniwersalność (ruter) i zarazem możliwość dopasowywania do konkretnej aplikacji (topologia).

Prace badawcze z dziedziny projektowania systemów NoC koncentrują się na kilku aspektach: konstrukcji mikroruterów, protokołach rutingu, topologiach oraz spajającym powyższe w całość szukaniu efektywnych metodologii odwzorowywania aplikacji z danej domeny zastosowań (systemy wbudowane, systemy wieloprocesorowe ogólnego przeznaczenia) w architekturę sieci jednoukładowej, z uwzględnieniem takich parametrów jak koszt rozwiązania, pobór mocy czy spełnienie ograniczeń czasowych aplikacji.

1.2 Zakres pracy

W pracy zostanie zaprezentowana metodologia efektywnego syntezowania bezkolizyjnych sieci jednoukładowych dla systemów wbudowanych. Założono, że charakterystyka komunikacyjna aplikacji jest znana, co pozwala na kosyntezę systemu wbudowanego połączoną z uszeregowaniem zadań i transmisji systemu. Na tej podstawie budowana jest mikrosieć wraz z wyznaczeniem tras dla poszczególnych transmisji. Cel stanowi

uzyskanie efektywnej pod względem kosztu (liczba połączeń, konstrukcja ruterów) oraz spełniającej zadane ograniczenia czasowe architektury NoC.

W rozdziale 2 zaprezentowano przegląd rozwiązań stosowanych w dziedzinie syntezy mikrosieci wraz z opisem nowego paradygmatu komunikacyjnego. Rozdział 3 zawiera motywację rozprawy, zaś rozdział 4 - jej cel i tezę. Niezbędne pojęcia, opis aparatu matematycznego oraz modelu sieci jednoukładowej wykorzystanego w rozprawie zamieszczono w rozdziale 5. Szczegóły działania metodologii oraz algorytmów składających się na jej poszczególne kroki zawarto w rozdziale 6. Rozdział ten zawiera również analizę skuteczności prezentowanej metodologii. Omówienie przeprowadzonych eksperymentów wraz z ich wynikami (porównania z istniejącymi metodami z badanej dziedziny) dla aplikacji syntetycznych oraz rzeczywistych umieszczono w rozdziale 7. Rozdział 8 stanowi podsumowanie rozprawy.

2 Projektowanie sieci jednoukładowych – przegląd

Wraz ze wzrostem popularności architektur MPSoC zaistniała konieczność wypracowania takich wewnątrzukładowych rozwiązań komunikacyjnych, które wspierałyby wysoki stopień zrównoleglenia transmisji wraz z minimalnymi opóźnieniami. Rozwiązania oparte na magistrali, nawet tak zaawansowane jak AMBA AHB [AMBA] czy STMicroelectronics STBus [STM], przestały wystarczać. W marcu 2000 roku przedstawiono koncepcję *Scalable Programmable Integrated Network* (SPIN) [GG00] – pierwszą sieć jednoukładową z ruterami stosującymi przełączanie pakietów. Termin *Network-on-Chip* pojawił się pod koniec roku 2000 w pracy [HJKPO00], gdzie mikrosieć została zaproponowana jako remedium na problem *design productivity gap*. Dally i Towles w połowie 2001 roku [DT01] przedstawili model sieci zorganizowanej w strukturę kraty i korzystającej z ruterów z kanałami wirtualnymi (sposób organizacji i zarządzania buforami) w celu zwiększenia przepustowości. W tym samym roku naukowcy z Philips Research zaprezentowali prototyp mikrorutera będącego w stanie obsłużyć komunikację z ograniczeniami czasowymi (ang. *guaranteed service, GS*) oraz zwykłą, bez ograniczeń (ang. *best effort, BE*) [RGW01]. Na początku roku 2002 idea NoC została przedstawiona jako nowy wzorzec dla projektowania układów MPSoC [BM02]. Obecnie jest to jedna z najintensywniej badanych dziedzin z pogranicza informatyki i mikroelektroniki – świadczy o tym liczba prac publikowanych każdego roku (sama tylko wyszukiwarka IEEEXplore[1] indeksuje 100-200 publikacji rocznie, Google Scholar[2] dla hasła „*Network on Chip*” zwraca ponad 1,3mln wyników zapytania – stan na listopad 2010).

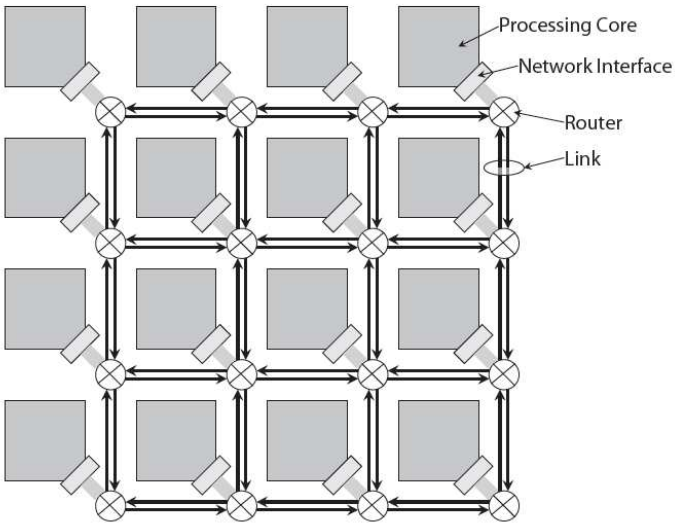
2.1 Sieci mikro i makro – różnice i podobieństwa

Sieci komunikacyjne tradycyjnie obejmują swym zasięgiem telefonię, sieci komputerowe a także superkomputery wieloprocesorowe. W rozprawie będą nazywane makrosieciami w celu odróżnienia ich od sieci integrowanych w pojedynczych układach. Przykład architektury NoC przedstawiono na Rys. 2.1-1. Podstawowymi elementami sieci jednoukładowej są:

- jednostki przetwarzające (ang. *Processing Core* lub *Processing Element, PE*) – przetwarzają dane realizując tym samym funkcje systemu, stanowią nadawców i odbiorców transmisji,
- interfejsy sieciowe (ang. *Network Interface, NI*) – oddzielają część obliczeniową systemu od komunikacyjnej, tworzą pomost pomiędzy protokołem, za pomocą którego porozumiewają się elementy przetwarzające, a protokołem sieci (komunikacja między ruterami),

1 http://ieeexplore.ieee.org/
2 http://scholar.google.pl/

- rutery (ang. *Router*) – wraz z określoną strategią trasowania obsługują przesyłanie informacji na wytyczonej trasie pomiędzy nadawcą i odbiorcą, obsługują sytuacje nadzwyczajne jak spór o zasoby (bufor, łącze),
- łącza (ang. *Link*) – to fizyczne połączenia między ruterami, mogą się składać z jednego lub więcej kanałów fizycznych lub logicznych, zwykle to dwa jednokierunkowe kanały danych (wysyłka/odbiór) z dodatkowymi ścieżkami dla sygnałów kontrolno-sterujących przepływem.



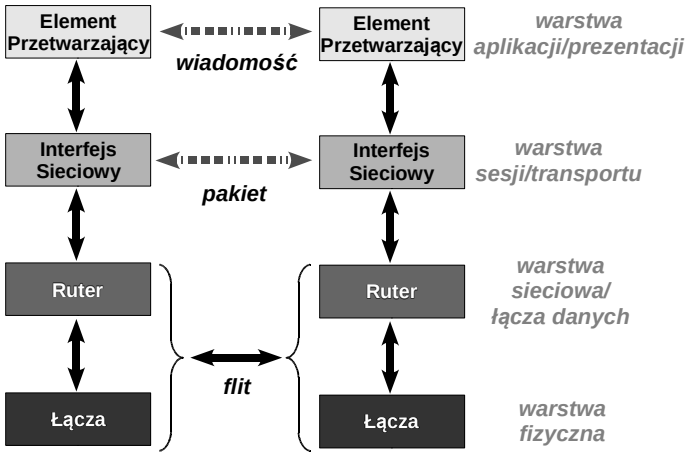
Rys. 2.1-1 Przykład NoC zorganizowanego w topologię regularną (ilustracja z [W08]).

Znaczna część opracowanej teorii dotyczącej działania sieci komunikacyjnych jest wspólna dla mikro- i makrosieci, m.in. teoria kolejkowania wiadomości w ruterach, model opóźnień w sieci, unikanie zakleszczeń czy model programowania systemu działającego w sposób rozproszony/równoległy. Mimo to sieć jednoukładowa, ze względu na swą domenę zastosowań różni się od swojego pierwowzoru w następujących aspektach (przez makrosieć w poniższym porównaniu rozumiana jest sieć komputerowa):

- pobór mocy – projektowanie makrosieci ukierunkowane jest przede wszystkim na osiągnięcie jak największej wydajności - względy energetyczne mają drugorzędne znaczenie; w skali mikro kwestie poboru energii są jednym z priorytetów konstruowania sieci,
- ograniczone zasoby – powszechnie stosowane w makrosieciach (na przykład komputerowych) złożone protokoły rutingu wspomagane pokaźnymi rozmiarami buforów w urządzeniach sieciowych wymagałyby zbyt dużych zasobów układu scalonego, czyniąc jego projekt niewykonalnym; im prostsza konstrukcja mikrorutera tym więcej tego typu elementów można integrować w strukturze półprzewodnikowej, co z kolei pozwala na łączenie ze sobą większej liczby jednostek przetwarzających,

- sposób transmisji – makrosieci serializują dane przed wysłaniem i deserializują je po odebraniu, co powoduje dodatkowy narzut transmisyjny; równoległa natura ścieżek w układach scalonych (szerokości magistral od kilku do kilkuset bitów) eliminuje lub upraszcza te etapy przygotowania przesyłanych informacji,
- wzorce transmisji w systemie – mikrosieci dedykowane są określonym klasom zastosowań, często charakteryzującym się z góry określonym sposobem przetwarzania i komunikacji (tzw. przewidywalny model transmisji); w makrosieciach sytuacja jest nieporównywalnie bardziej dynamiczna – stacje (komputery) mogą być w dowolnej chwili dołączane/odłączane do/z sieci, przydziela się im zmieniające się zadania, rekonfiguracja połączeń też nie nastręcza wielu trudności; sieci jednoukładowe budowane są z określonej liczby jednostek przetwarzających zwykle trwale ze sobą połączonych, każdy procesor w mikrosieci ma zazwyczaj przydzielone na stałe jedno zadanie (lub zadania z określonej klasy zadań).

Paradygmat NoC bazuje na modelu sieci zorganizowanym w sposób przypominający dobrze znany z domeny sieci komputerowych model OSI [T04]. Na Rys. 2.1-2 przedstawiono model NoC z odniesieniem do warstw OSI. Przerywane linie ze strzałkami symbolizują komunikację logiczną, ciągłe – fizyczną.



Rys. 2.1-2 Warstwowy model komunikacji w NoC.

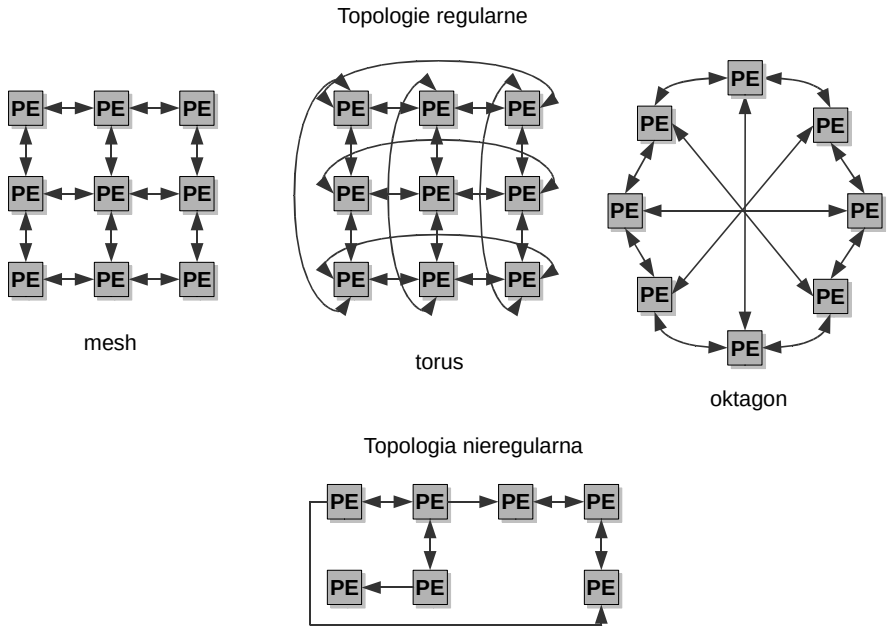
Najmniejszą jednostką danych przesyłanych w jednym cyklu pracy na poziomie warstwy fizycznej w NoC jest flit (ang. ***Fl****ow Control Un****it***). Z flitów zbudowane są pakiety, a te z kolei tworzą komunikaty (wiadomości). Podział na cztery warstwy pozwala również wyodrębnić główne kierunki badań w holistycznym podejściu do projektowania architektur NoC. Warstwa najwyższa obejmuje projektowanie na poziomie systemowym (kosynteza, modelowanie systemu, szeregowanie zadań i wiadomości dla różnych charakterystyk komunikacyjnych: z ograniczeniami czasowymi, bez ograniczeń, mieszanej). W warstwie, w której umieszczono interfejsy sieciowe, określa się metody enkapsulacji transmisji, charakter usługi (połączeniowy,

bezpołączeniowy, z retransmisjami lub bez, itd.) oraz protokół komunikacji dopasowany do modułu przetwarzającego. Warstwa sieciowa i łącza danych to przede wszystkim eksploracja zbioru topologii połączeń, protokołów rutingu, metod przełączania flitów w ruterze, sterowania przepływem i charakteru transmisji (uni-, multi-, broadcast). Na najniższym poziomie – fizycznym – określa się metody synchronizacji oraz kodowania sygnałów.

Metodologia prezentowana w rozprawie koncentruje się na zagadnieniach syntezy komunikacji (efektywnym odwzorowaniu aplikacji w architekturę NoC), a więc związanych z warstwami: sieciową i łącza danych. Punktem wyjścia jest analiza modelu aplikacji umiejscowionego w najwyższej warstwie na Rys. 2.1-2.

2.2 Topologie

Topologia określa sposób połączenia ruterów w sieci jednoukładowej przez co wpływa na rezultat odwzorowania architektury NoC w struktury układu scalonego. Wybór topologii decyduje również o koszcie i wydajności systemu (liczba i długość ścieżek miedzyruterowych w układzie oraz liczba ruterów na trasach nadawca-odbiorca) oraz konstrukcji ruterów (liczba i rodzaj portów). Sposób łączenia ruterów NoC dzieli się na dwie główne kategorie: regularne i nieregularne. Pierwsza grupa jest chętnie wykorzystywana ze względu na przewidywalną skalowalność (wydajność, zajmowane zasoby) oraz wygodę w implementacji w fizycznym układzie (ang. *floorplanning*). Topologie nieregularne natomiast są zwykle lepiej dopasowane do wymagań konkretnej aplikacji i przez to bardziej efektywne pod względem wydajności, wykorzystywanych zasobów oraz poboru mocy. Przykłady topologii regularnych oraz strukturę nieregularną zaprezentowano na Rys. 2.2-1.



Rys. 2.2-1 Przykłady topologii regularnych i nieregularnej.

Najczęściej spotykanym regularnym sposobem łączenia elementów NoC jest topologia kraty (ang. *mesh*) [SKH08]. Jeżeli mikrosieć tworzą jednakowe elementy przetwarzające (chodzi głównie o zajmowane zasoby) wówczas zaletą tego rozwiązania są stałe długości połączeń w układzie i liniowa skalowalność pod względem zajmowanych zasobów. Wadą *mesh*'y jest znaczna średnia odległość pomiędzy procesorami, a co się z tym wiąże – relatywnie duży pobór mocy w trakcie komunikacji. Ponadto w centrum mikrosieci często powstają tzw. *hotspoty*, czyli miejsca (rutery) szczególnie obciążone transmisjami.

Prostą modyfikacją topologii kraty, uzupełnioną o dodatkowe połączenia zmniejszające liczbę przeskoków na trasach jest torus [DT01]. To rozwiązanie, mimo zwiększonego kosztu ścieżek, przydatne jest w szybkich, regularnych mikrosieciach stosujących trasowanie pakietów typu *hot potato* [LZJ06].

Najprostszą regularną topologią jest pierścień, gdzie każda jednostka przetwarzająca połączona jest z dokładnie dwoma sąsiadami – przykład to platforma Proteo [STAN04]. Prosty schemat rutingu oraz sterowania przepływem i mały narzut pod względem zasobów okupione są słabą wydajnością i skalowalnością powyższego podejścia. Rozszerzenie koncepcji pierścienia o dodatkowe łącza dla systemów z ośmioma modułami PE stanowi oktagon [KND02] - maksymalna długość najkrótszych ścieżek pomiędzy dowolnymi parami nadawca-odbiorca wynosi dwa rutery.

Ostatnią grupą topologii regularnych są struktury wywodzące się z drzew. Elementy przetwarzające są liśćmi drzewa, natomiast węzły pośrednie to rutery. Rutery w mikrosieci mogą występować bez przyłączonego do nich żadnego modułu PE – są wówczas jedynie przekaźnikami/rozgałęźnikami ruchu sieciowego (tzw. sieci pośrednie, podobne do wymienionych wcześniej sieci Omega). Takie odmiany topologii jak *fat-tree* [GG00] czy *butterfly tree* [PGJIS05] pozwalają ograniczyć niekorzystne zjawisko powstawania komunikacyjnego *hot-spota* w korzeniu drzewa – odbywa się to kosztem rozbudowy struktury części ruterów (liczba portów) lub zwiększenia ich liczby. Duża dywersyfikacja kanałów komunikacyjnych ma również negatywny wpływ na efekty fizycznej implementacji zaawansowanych topologii drzewiastych – występuje w nich wiele łącz o różnych długościach. Architektury takie charakteryzują się ponadto bardzo niejednorodną konstrukcją ruterów (w sensie liczby i rodzaju portów) [LGMGG09] oraz ich nadmiarem w stosunku do jednostek przetwarzających. Przykład systemu bezkolizyjnego zbudowanego w topologii drzewa z ruterami o znacznie zwielokrotnionej i asymetrycznej liczbie portów (dwa razy więcej portów wyjściowych niż wejściowych) zaprezentowano w [BE06]. Długość tras dla komunikatów predestynuje mikrosieci drzewiaste do zastosowania w takich aplikacjach, gdzie stopień

lokalności transmisji jest bardzo wysoki. Oznacza to, że da się wówczas wydzielić procesory komunikujące się głównie w obrębie danej grupy.

Topologie regularne są preferowane wówczas, gdy układ składa się z takich samych pod względem zajmowanych zasobów jednostek przetwarzających [VHRDW07]. Systemy wbudowane zazwyczaj tworzone są ze zróżnicowanych funkcjonalnie modułów. Prócz tego każdy zajmuje po implementacji inną powierzchnię w układzie. Stąd dla takich systemów lepsze wyniki w zakresie syntezy NoC pod względem wydajności, kosztu zasobów oraz poboru mocy uzyskuje się korzystając z indywidualnie dopasowywanych topologii nieregularnych [MMAAC06][AAMPB08]. Zwykle w nieregularnych sieciach jednoukładowych zarówno liczba portów w ruterach nie podlega żadnemu wzorcowi [DBGBB03][SCK06], jak i liczba łącz i sposób ich wykorzystania w układzie [OMLC06].

Osobną kategorię stanowią powoli wyłaniające się koncepcje projektowania układów z wykorzystaniem topologii 3D. W pracy [FP07] dowiedziono, że wykorzystanie całej, trójwymiarowej przestrzeni układów scalonych może dać lepsze wyniki od rozwiązań planarnych na polu poboru mocy i wydajności – pogorszeniu uległo jedynie zapotrzebowanie na zasoby układu. Pavlidis i Friedmann [PF07] zaproponowali formalny model do analizowania architektur z rodziny NoC 3D – odniesienie stanowi klasyczna topologia *mesh*. Warto zauważyć, że wykorzystanie trzech wymiarów może znacząco ułatwić *floorplanning* topologii nieregularnych (wielowarstwowe układy scalone).

O ile zalety dopasowanych do danej aplikacji topologii nieregularnych są bezdyskusyjne [PS04][MMAAC06][OMLC06][XWHC06][AAMPB08], o tyle trudno przesądzać o przewadze poszczególnych struktur regularnych. Opublikowano wiele prac porównujących parametry systemów zbudowanych w oparciu o różne topologie. Balfour i Dally [BD06] dokonali oszacowania funkcjonalności różnych odmian struktur kraty (*mesh*, torus oraz *mesh* z ruterami obsługującymi cztery moduły PE jednocześnie) z architekturami drzewiastymi. Najbardziej wydajną i efektywną pod względem zasobów i energii topologią okazał się zmodyfikowany *mesh*. W rozwiązaniach NoC z pamięcią dzieloną [MPSV06] topologia kraty okazała się lepsza (moc, szybkość przetwarzania) od pierścienia oraz struktury spidergon (wielowęzłowa odmiana oktagonu) [CLMPS04]. W [JZH06] porównano koszt oraz wydajność architektur zbudowanych z 64 jednostek przetwarzających w topologii pierścienia, kraty 2D i 3D, torusa 2D i 3D, drzewa *fat-tree* oraz hiperkostki. Najwydajniejsze okazały się torusy 2D i 3D oraz *mesh* 3D, natomiast najmniejsze wymagania względem zasobów miał pierścień oraz *mesh* 2D. Zachowanie mikrosieci dla różnych poziomów obciążenia komunikacją przedstawiono w [PGJIS05]. Praca ta potwierdza tezę, iż dla systemów o wysokim stopniu lokalności transmisji najlepszym wyborem

wśród topologii regularnych są drzewa. W [NNC06] porównano topologię kraty z drzewem *fat-tree* dla implementacji dekodera wideo H.264. Wyniki porównań dla topologii *mesh*'a, spidergona oraz pierścienia przeprowadzonych metodami analityczno-symulacyjnymi można znaleźć w [BC06]. W wymienionej pracy użyto syntetycznych generatorów transmisji a porównania ograniczono jedynie do parametrów związanych z wydajnością. W [SSC06] porównano architektury NoC zaimplementowane w różnych układach FPGA z rodziny Virtex [Xilinx] z zastosowaniem topologii pierścienia, gwiazdy, kraty, hiperkostki oraz w pełni połączonej (P2P) dla aplikacji o 8 do 32 elementach przetwarzających. Dla najbardziej rozbudowanych systemów najlepsze wyniki (wydajność, implementacja) dały struktury kraty i hiperkostki.

Syntetyczne zestawienie parametrów kilku podstawowych sposobów łączenia dla architektury MPSoC składającej się z 64 jednostek przetwarzających przedstawiono w Tabeli 2.2-1. Parametr „Stopień bisekcji" oznacza minimalną liczbę łącz, jakie trzeba przeciąć, by otrzymać dwa równe co do ilości węzłów układy (im większa wartość tym lepsza potencjalna przepustowość architektury). „Maksymalna (średnia) liczba przeskoków" na trasie powinna być jak najmniejsza dla uzyskania wysokiej wydajności (małe opóźnienia). Wszystkie parametry z sekcji „Koszt" powinny mieć jak najmniejsze wartości.

Kategoria	**Parametr do porównania**	**Magistrala**	**Pierścień**	**Mesh**	**Torus**	**Hiperkostka**	***Fat-tree***	**P2P**
Wydajność	Stopień bisekcji	1	2	8	16	32	32	1024
	Maksymalna (średnia) liczba przeskoków	1(1)	32(16)	14(7)	8(4)	6(3)	11(9)	1(1)
Koszt	Maks. liczba portów w ruterze	-	3	5	5	7	4	64
	Liczba ruterów	-	64	64	64	64	192	64
	Liczba łącz między węzłami	1	64	112	128	192	320	2016

Tabela 2.2-1 Porównanie parametrów różnych architektur MPSoC dla 64 jednostek PE.

2.3 Realizacja i zarządzanie transmisjami

Topologia tworzy infrastrukturę komunikacyjną, natomiast techniki związane z trasowaniem oraz konstrukcja użytych ruterów determinują rzeczywiste zachowanie systemu związane z transmisjami. Obrana strategia rutingu wraz z mechanizmem kontroli przepływu w mikrosieci mają decydujący wpływ na budowę i zasadę działania rutera. Decyzje projektowe podejmowane na tym etapie mają znaczący wkład do współczynnika wydajność/koszt finalnej architektury NoC.

2.3.1 Ruting

Algorytm trasowania ma na celu wyznaczenie ścieżki od nadawcy do odbiorcy wiadomości. Metody rutingu stosowane w sieciach jednoukładowych można klasyfikować ze

względu na następujące cechy:

1. Miejsce, gdzie podejmowana jest decyzja o wyznaczeniu trasy:
   - metody scentralizowane: trasa jest wyznaczana przez nadawcę i zapisywana w nagłówku wysyłanej wiadomości w postaci osobnych pozycji dla każdego rutera na ścieżce danej transmisji (ang. *source routing*) [LLSY05],
   - metody rozproszone: każdy ruter na trasie komunikatu decyduje o wyborze swojego portu wyjściowego do kontynuowania transmisji; w rutery tego typu wbudowano zaprogramowane tablice rutingu (ang. *table-based routing*) [BMNMV05][DT08a] [DT08b] lub dedykowane układy logiczne pozwalające wyliczyć adres portu wyjściowego dla pakietu (ang. *algorithm-based routing*).
2. Dostosowanie trasy do warunków/natężenia ruchu w mikrosieci:
   - metody deterministyczne (statyczne): każdej parze nadawca-odbiorca przydzielana jest zawsze ta sama trasa,
   - metody adaptacyjne (dynamiczne): podczas wyznaczania kolejnych odcinków trasy brana jest pod uwagę aktualna sytuacja w sieci NoC.
3. Długość wyznaczonej ścieżki:
   - metody minimalne: zawsze wybierana jest jedna z najkrótszych dróg od nadawcy do odbiorcy (innymi słowy każdy przeskok na trasie zmniejsza dystans – liczony w ruterach - do odbiorcy),
   - metody nieminimalne: wyznaczona trasa nie musi być najkrótsza.

Każda z powyższych strategii trasowania i ich kombinacji niesie ze sobą wady i zalety. Decentralizacja decyzji powoduje wzrost kosztu rutera (zasobo- i energochłonne tabele rutingu realizowane w postaci układów pamięci), natomiast trasa zapisana w nagłówku wnosi narzut transmisyjny (większy nagłówek) – bardziej szczegółowo opisano tą kwestię w rozdziale 5.2.1 rozprawy. Z energetycznego oraz wydajnościowego punktu widzenia bardziej pożądane są najkrótsze (minimalne) ścieżki transmisji. Może się jednak zdarzyć sytuacja, gdy przeprowadzenie komunikatu dłuższą trasą pozwoli ominąć nadmiernie obciążone transmisjami fragmenty sieci, a ponadto lepiej wykorzystać pozostałe łącza. Jest to spowodowane tym, że w projektowaniu systemów NoC dąży się do jak najbardziej intensywnego wykorzystania wszystkich połączeń w mikrosieci (ang. *utilization factor*), aby zminimalizować wpływ statycznego poboru mocy na bilans energetyczny systemu (eliminacja sporadycznie używanych łącz). Najbardziej problematyczna jest kwestia wyboru pomiędzy statycznym i dynamicznym wyznaczaniem tras. Mogłoby się wydawać, iż optymalnym wyborem jest system adaptacyjny. Okazuje się jednak, że dla rozbudowanych, wielowęzłowych topologii koszt jego implementacji

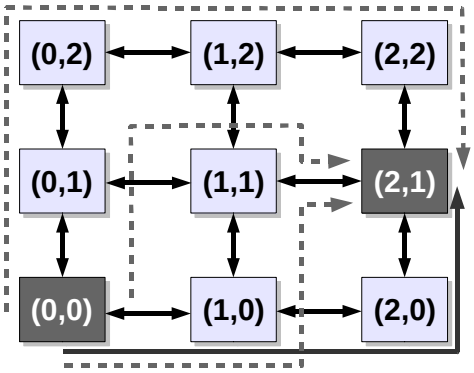
przewyższa korzyści [NTA05], a przy dużym natężeniu transmisji decyzje kierujące ruchem są podejmowane zbyt późno [DT08c]. Jak stwierdzili autorzy platformy HERMES [MCMMO04], ruting dynamiczny ma tendencję do kierowania transmisji do centrum sieci jednoukładowej. Dzieje się tak dlatego, że ruter w trakcie podejmowania decyzji o przesłaniu pakietu do jednego ze swoich wyjść ma ograniczony obraz sytuacji w sieci, zawężony jedynie do stanu zajętości swoich buforów wejściowych/wyjściowych [KPTVD05][WHS06] lub należących do najbliższych sąsiadów [HM04b][ACPP08][BTBTT08]. Inne często wskazywane wady rozwiązań z rutingiem dynamicznym to: rozbudowana (koszt) struktura rutera [SKH08], niekolejne dostarczanie do odbiorcy pakietów tworzących ten sam komunikat (konieczność ponownego złożenia w poprawnej kolejności) [HM03b] i dłuższe opóźnienia przesyłu przez ruter spowodowane koniecznością lokalnego podjęcia decyzji. Dla systemów wbudowanych, gdzie wzorce transmisji są w dużym stopniu przewidywalne (specjalizowana funkcjonalność) dużo lepszym rozwiązaniem jest trasowanie statyczne z ewentualnymi dodatkowymi rozwiązaniami (buforowanie) pozwalającymi uniknąć zatorów w sieci. Prócz natłoku, algorytm rutingu musi radzić sobie jeszcze z dwiema niebezpiecznymi dla stabilnego funkcjonowania systemu sytuacjami:

- zakleszczeniem (ang. *deadlock*) – kilka pakietów wzajemnie blokuje sobie zasoby (porty/ bufory rutera) bezskutecznie czekając na ich zwolnienie; w niektórych sytuacjach pakiet może sam sobie zablokować zasoby, gdy na skutek błędnych przełączeń w ruterach jego ścieżka utworzy cykl,
- błądzeniem (ang. *livelock*) – może się zdarzyć w przypadku stosowania rutingu nieminimalnego, gdy pakiet stale krąży w sieci i nie może zostać prawidłowo przekierowany do odbiorcy; sytuacja niemożliwa w przypadku metod deterministycznych.

Najpopularniejszy algorytm rutingu deterministycznego w sieciach jednoukładowych o strukturze kraty to metoda XY, zwana także *Dimension Ordered Routing* (DOR). Komunikat jest przesyłany trasą biegnącą najpierw wzdłuż osi X, a potem wzdłuż osi Y - wariant trasy pomiędzy modułami o koordynatach (0,0) i (2,1) na Rysunku 2.3.1-1 oznaczony linią ciągłą. Przykład strategii dynamicznej oznaczono liniami przerywanymi - w przypadku niemożności przesłania pakietu wzdłuż wiersza (X) następuje transmisja wzdłuż kolumny (Y).

Istnieją również metody częściowo adaptacyjne (preadaptacyjne) – alternatywne trasy wyznacza się nie w trakcie działania systemu, ale w następstwie analizy jego zachowania są preprogramowane wcześniej, zanim system rozpocznie funkcjonowanie. Tego typu podejście zastosowano również w niniejszej rozprawie. Do takich metod można zaliczyć rozdzielające

ruch w sieci na niezależne ścieżki *Toggle XY* (TXY) [SALR05] czy jego zaawansowaną wersję *Weighted Ordered Toggle* XY (WOT) [GCK07]. W pełni adaptacyjne wersje rutingu XY bazują na modelu zakazującym pewnych przekierowań w ruterach (tzw. *Turn Model* [GN92]), by zapobiec zapętleniom. Przykładowe algorytmy to *West-First*, *North-Last*, *Negative-First* [KS03] czy nieco nowszy *Odd-Even* [C00].



Rys. 2.3.1-1 Ruting XY (linia ciągła) oraz adaptacyjny (linie przerywane) w NoC o topologii kraty.

2.3.2 Przełączanie i kontrola przepływu

Metoda przełączania komunikacji w sieci jednoukładowej definiuje sposób, w jaki dane są przekazywane pomiędzy węzłami oraz jak wygląda etap przekierowania komunikatów pomiędzy portami wejściowymi a wyjściowymi w ruterach.

Pierwsza grupa architektur NoC to systemy z przełączaniem obwodów (ang. *circuit switched*). W tym przypadku zanim zostanie przeprowadzona transmisja musi zostać zestawiony dla niej wyłączny kanał komunikacyjny (faza nawiązywania połączenia). Po udanym zarezerwowaniu trasy, potwierdzonym przez odbiorcę, można przystąpić do transmisji. Dane, bez dodatkowych informacji o rutingu (niski narzut komunikacyjny), są przekazywane od węzła do węzła według zestawionej trasy. Jeśli w transmisji wystąpiłyby zamierzone przerwy, wówczas niemożliwe jest wykorzystanie takich jałowych cykli pracy do przesłania nieobciążonymi łączami pakietów należących do innych komunikatów. Realizacja innych transmisji trasowanych w całości lub w części zarezerwowanymi łączami jest możliwa dopiero po zwolnieniu zestawionej ścieżki – to główna wada rozwiązań z przełączaniem obwodów. Ta metoda gwarantuje żądaną przepustowość dla komunikacji po jej rozpoczęciu, ale prowadzi do nieefektywnego wykorzystania łącz. Ponadto fazy nawiązywania połączenia oraz jego zrywania odbijają się negatywnie na parametrach czasowych układu (opóźnienia). Prosta i tania konstrukcja rutera (minimalne rozmiary buforów lub ich brak) powoduje niewielką elastyczność w kierowaniu ruchem w sieci. Wobec powyższych argumentów rozwiązanie to jest polecane do systemów o ściśle przewidywalnym wzorcu transmisji [WL03][WSRS05].

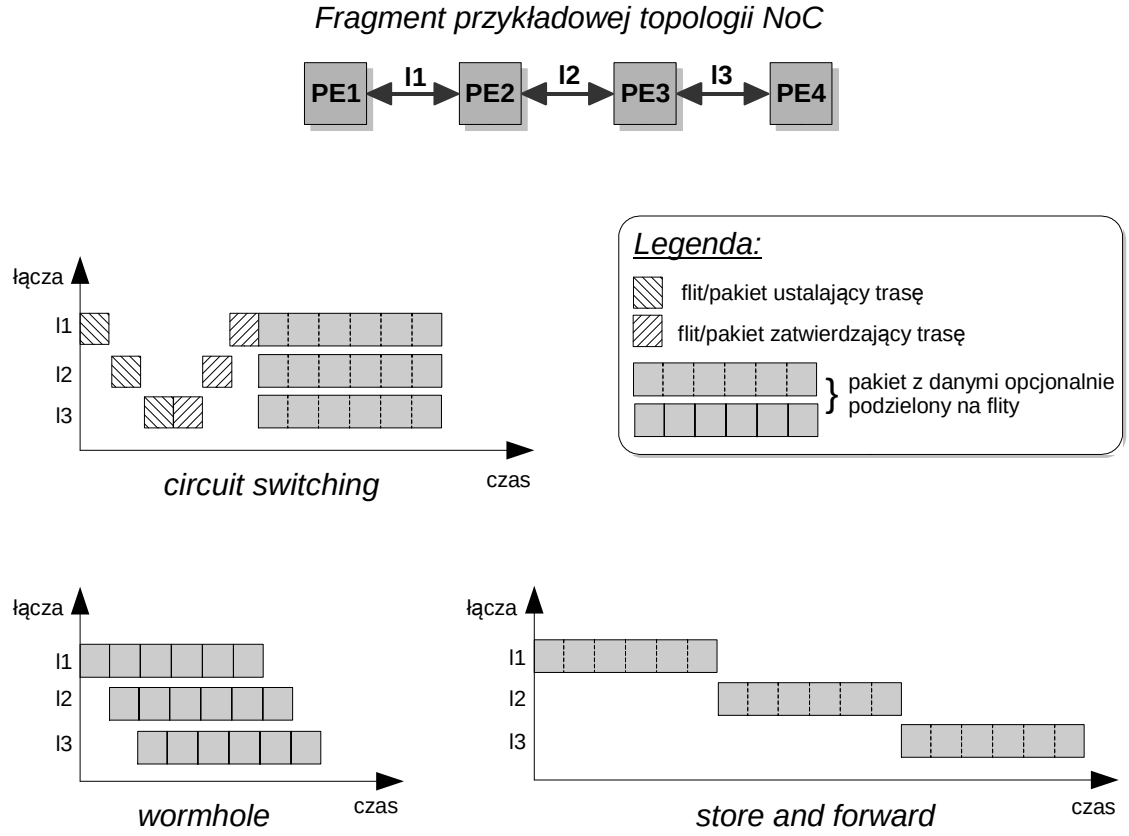
Próbą poprawienia stopnia wykorzystania zasobów (łącza, rutery) w podejściach z przełączaniem obwodów jest koncepcja współużytkowania zestawionych ścieżek w czasie (ang. *Time Division Multiplexing, TDM*). Dzięki temu w przeplatających się cyklach pracy (ang. *time slots*) za pomocą tych samych łącz przesyłane są fragmenty transmisji należących do różnych par nadawca-odbiorca. Przykładowymi platformami korzystającymi z tej metody są Nostrum [MNTJ04] i Æthereal [GDR05]. Są to jednak systemy wymagające pracochłonnego planowania komunikacji - każdy cykl pracy danego rutera powinien zostać przydzielony fragmentowi jakiejś transmisji. Do prawidłowej pracy niezbędna jest również dokładna synchronizacja globalna systemu oraz kosztowne tablice rutingu (w ruterach lub interfejsach sieciowych NoC) przechowujące rezultaty przydziału poszczególnych cykli pracy.

Drugą, najpopularniejszą grupą [SKH08] rozwiązań związanych z przesyłaniem danych pomiędzy węzłami sieci i portami wewnątrz ruterów jest przełączanie pakietów (ang. *packet switching*). W tej metodzie każdy pakiet, będący częścią lub całością komunikatu, posiada informacje o trasowaniu w swoim nagłówku, co pozwala podjąć decyzję o przekierowaniu do portu wyjściowego każdego rutera. Transmisja ma najczęściej charakter bezpołączeniowy, choć istnieją mechanizmy nadające jej cechy połączenia (omówione dalej metody *credit-based* oraz *ACK/NACK*). Brak rezerwowania trasy zwiększa stopień wykorzystania zasobów NoC. Wyróżniono trzy główne techniki przełączania pakietów:

- *store-and-forward* – ruter musi poczekać, aż do jego bufora wejściowego trafi cały pakiet; dopiero po tym i po upewnieniu się, że następny na ścieżce komunikatu węzeł może przyjąć w całości dany pakiet, następuje dalsza transmisja; przykładowe realizacje opisano w [KJSFM02][SBKV05]; główne wady to opóźnienia w przekazywaniu pakietów oraz bardzo wysoki koszt buforów w ruterach (wielkość równa rozmiarowi pakietu), natomiast zaletą jest prostota konstrukcji,
- *virtual cut-through* – modyfikacja poprzedniej metody polegająca na tym, że ruter nie czeka na otrzymanie całego pakietu, lecz zaraz po zdekodowaniu adresu z flita nagłówkowego może przesłać jego i kolejne flity dalej – warunkiem jest odpowiednia ilość wolnego miejsca w buforze kolejnego węzła na trasie; dzięki temu zniwelowano opóźnienia transmisji, ale wada związana z rozmiarami buforów pozostała – w przypadku niemożności natychmiastowego podjęcia transmisji kolejne flity danego pakietu będą trafiały do bufora wejściowego zaraz za nagłówkiem pakietu; przykłady zastosowań można znaleźć w pracach [STAN04][BMNMV05],
- *wormhole* – w tym podejściu zakłada się, że flity należące do pakietu są natychmiast po otrzymaniu i wyznaczeniu adresu przesyłane dalej; powoduje to minimalne opóźnienia

transmisji jak również znaczne oszczędności w konstrukcji rutera (potrzebny jest bardzo mały bufor); wada to zjawisko „zatykania” mikrosieci (ang. *backpressure*) w przypadku niemożności przekazania flitów do kolejnego węzła; mimo powyższego jest to najczęściej stosowana odmiana przełączania pakietów [SKH08], często rozbudowywana o mechanizmy poprawiające niezawodność (opisane w rozdziale 2.3.3 kanały wirtualne).

Na Rys. 2.3.2-1 przedstawiono opóźnienia transmisji dla NoC z przełączaniem obwodów oraz dwóch odmian przełączania pakietów: *store-and-forward* i *wormhole*. Założono, że transmisje odbywają się bez kolizji w sieci oraz dla uproszczenia pominięto opóźnienia wnoszone przez przełączanie w ruterach. Komunikat jest takiej samej wielkości a przesył odbywa się według tej samej trasy (PE1 → PE2 → PE3 → PE4).



Rys. 2.3.2-1 Relacja pomiędzy czasami transmisji przy braku zatorów dla różnych sposobów przełączania transmisji w NoC.

Mechanizmy kontroli przepływu są odpowiedzialne za stwierdzenie, kiedy ruter może rozpocząć transmitowanie danych do sąsiadującego z nim węzła. Zależy to przede wszystkim od stanu zajętości buforów w ruterze-odbiorcy lub module NI docelowej jednostki przetwarzającej. Na Rys. 2.3.2-2 przedstawiono uproszczony schemat połączenia międzyruterowego.

Rys. 2.3.2-2 Dwukierunkowe połączenie międzyruterowe z rozbiciem na magistrale składowe.

Bez metod sterowania przepływem mikrosieć narażona byłaby na dwa niekorzystne zjawiska:

- porzucanie pakietów/flitów – w sytuacji, gdy bufory rutera odbiorcy lub docelowego modułu NI byłyby zapełnione kolejne napływające dane musiałyby być ignorowane (usuwane); rozwiązaniem byłaby wówczas retransmisja całego pakietu lub brakujących flitów (z koniecznością uporządkowania ich w buforze NI odbiorcy) co w naturalny sposób podnosi koszt systemu (narzut czasowy i energetyczny),
- wstrzymywanie transmisji – do momentu zwolnienia odpowiedniej ilości miejsca w buforach odbiorcy (rutera lub modułu NI) pozostałe pakiety/flity należące do danej transmisji byłyby zatrzymywane w ich bieżących lokacjach (bufory ruterów poprzedzających na trasie wiadomości miejsce jej zatoru) narażając mikrosieć na kolejne zatory, kolizje i zakleszczenia.

W architekturach NoC stosowane są najczęściej następujące techniki kontrolowania przepływu:

- *Stop/Go* (zwana także *Stall/Go* lub *On/Off*) – najprostsza z metod oparta na dwóch sygnałach wysyłanych od odbiorcy do nadawcy: *Stop* informującym o zajętości buforów i nakazującym wstrzymanie transmisji oraz *Go* wysyłanym w momencie zwolnienia przestrzeni buforowej i umożliwiającym wznowienie transferu danych; zaletą powyższego podejścia jest znikomy narzut związany z czasem działania oraz wymaganymi zasobami,
- uzgadnianie (ang. *handshake*) [ZS03][ZSS04][SBKV05] – nadawca wysyła sygnał *Valid/Req*, kiedy jest gotowy do wysyłki, natomiast odbiorca odpowiada sygnałem *Ack*, jeśli może prawidłowo odebrać dane (dysponuje wolną przestrzenią buforową), inicjując tym samym transmisję; taka metoda sterowania przepływem pozwala również na budowanie systemów mezochronicznych, plezjochronicznych oraz asynchronicznych,
- *credit-based* [BCGK04] – odbiorca informuje nadawcę o konkretnej ilości wolnego miejsca w buforze przesyłając mu tzw. *credit* – nadawca może wówczas wysłać ilość danych nieprzekraczającą wartości w otrzymanej uprzednio informacji; sytuacja gdy odbiorca przesłał *credit* równy „0” oznacza nakaz wstrzymania transmisji; w pracy [RDPGR05] zastosowano powyższy mechanizm kontroli przepływu na poziomie łącz międzyruterowych oraz na poziomie kanałów komunikacyjnych pomiędzy komunikującymi się jednostkami PE; metoda ta funkcjonuje identycznie jak mechanizm przesuwnego okna (ang. *sliding window*) w protokole TCP [TCP].

Pomiędzy nadawcą i odbiorcą transmisji na poziomie modułów przetwarzających stosowany jest, prócz metody *credit-based*, mechanizm *ACK/NACK* [DBGBB03]. Jego działanie polega na potwierdzaniu prawidłowego odbioru wiadomości (*ACK*) lub wymuszającego

retransmisję sygnalizowaniu błędów (*NACK*). Funkcjonalność ta przypomina działanie flagi potwierdzenia *ACK* (ang. *acknowledge*) w protokole TCP. Porównanie metod kontroli przepływu pod kątem wydajności oraz tolerowania błędów można znaleźć w [PABB05].

2.3.3 Konstrukcja rutera

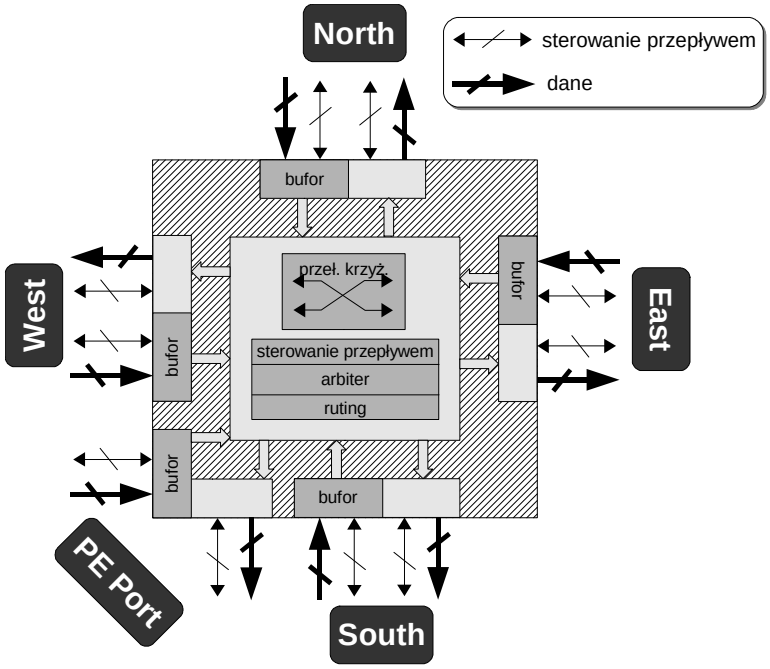
O funkcjonalności i możliwościach sieci jednoukładowej decydują przede wszystkim dwa jej elementy: interfejsy sieciowe NI oraz rutery. Jeśli chodzi o pierwszą grupę to wybór projektowy jest stosunkowo niewielki – można wykorzystać konstrukcję dopasowaną do danego elementu przetwarzającego (lub grupy elementów tworzących część przetwarzającą systemu) lub zastosować rozwiązanie korzystające z ustandaryzowanych protokołów komunikacji na poziomie jednostek PE. Przykład pierwszego rozwiązania można znaleźć w pracy [DT08b], natomiast drugie podejście opisano w [RDPGR05]. Nie da się wskazać, które z powyższych rozwiązań jest bardziej odpowiednie dla mikrosieci, co widać chociażby w zestawieniu różnych cech implementacji NoC uwzględniających między innymi zastosowany interfejs NI [MCMMO04]. W pracy [BM03] potwierdzono natomiast fakt, iż implementacja sprzętowa jest znacznie wydajniejsza od programowej – obsługuje komunikację kilkadziesiąt razy szybciej. Autorzy wymienionej pracy szacują również, że koszt implementacji mierzony zasobami układu byłby wyższy dla wersji programowej modułu NI.

Nieco inaczej wygląda kwestia konstrukcji rutera. Wybory projektowe związane z topologią NoC, metodą rutingu, przełączaniem oraz kontrolą przepływu decydują o budowie zastosowanego w danej sieci jednoukładowej rutera. Niezależnie od konkretnej implementacji każdy tego rodzaju element składa się z kilku podstawowych komponentów przedstawionych na Rys. 2.3.3-1:

- buforów – stosowane w celu przechowania pakietu/flita (zależnie od używanej strategii przełączania w ruterze) przed dalszą transmisją, pomagają unikać zatorów w mikrosieci, mogą służyć również do realizacji szeregowania transmisji [MEP06],
- jednostki sterowania przepływem – implementuje funkcje związane zarówno z przełączaniem transmisji w ruterze jak i kontrolowaniem komunikacji pomiędzy sąsiadującymi węzłami (rozdział 2.3.2),
- jednostki dekodującej adres docelowy (ruting) – służy do wyznaczenia portu wyjściowego rutera, którym dane mają zostać przesłane dalej w mikrosieci,
- arbitra – w przypadku, gdy dwa lub więcej portów wejściowych próbuje przekierować swoje pakiety/flity do tego samego portu wyjściowego, decyduje o kolejności dostępu do spornego zasobu (najpopularniejsze strategie arbitrażu to algorytm karuzelowy *round-robin* oraz oparty na priorytetach portów/transmisji),

- przełącznika krzyżowego – zestawia fizyczne ścieżki transmisji pomiędzy portami wejściowymi a wyjściowymi zgodnie z regułami opisanymi uprzednio, zwykle implementowany w formie zestawu odpowiednio połączonych multiplekserów.

Najczęściej projektuje się systemy NoC korzystając z ruterów o liczbie portów nie przekraczającej 5 (w tym jeden port do podłączenia jednostki przetwarzającej, tzw. port lokalny) [SKH08]. Uwzględniając koszt (energetyczny oraz zapotrzebowanie na zasoby układu) [WPM03][B05][KLPS09] najwięcej uwagi poświęca się kwestiom umiejscowienia i rozmiarów buforów oraz budowy przełącznika krzyżowego.



Rys. 2.3.3-1 Ogólny schemat 5-portowego rutera stosowanego w sieciach jednoukładowych (anglojęzyczne oznaczenia kierunków, buforowane wejście).

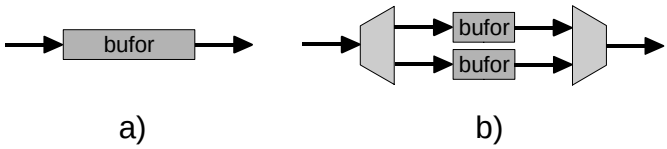
Przełącznik krzyżowy

Porównanie czterech mikroarchitektur będących ulepszeniami/modyfikacjami tradycyjnej macierzy przełączającej zawarto w pracy [WPM03]. Metodami analityczno-symulacyjnymi oszacowano pobór energii, wpływ na wydajność oraz zapotrzebowanie na zasoby układu dla: segmentowanego przełącznika (konstrukcja wzorowana na segmentowanych magistralach), przełącznika pracującego według metody *cut-through* (optymalizacja przesłań do przeciwległych portów), rozwiązania pozwalającego na omijanie bufora wejściowego oraz struktury połączeń zwanego *Express Cube*, zaprojektowanego z myślą o podwyższeniu szybkości komunikacji dzięki dedykowanym kanałom. Redukcję liczby portów przełącznika poprzez jego podział na dwa stopnie przełączające zaproponowano w [KNP06]. Takie rozwiązanie pozwala na zmniejszenie wymagań związanych z zasobami układu kosztem gorszej wydajności (dłuższy transfer pakietu/flita pomiędzy portami rutera). Obniżenie poboru mocy, dzięki częściowemu

wyłączaniu w trakcie pracy systemu, struktury przełącznika opisano w [LLY06]. Metodologię generowania przełącznika dostosowanego do danej aplikacji przedstawiono w [MBM07] - najpierw analizowana jest charakterystyka komunikacyjna aplikacji, by następnie określić, które spośród jednostek przetwarzających realizujących zadania systemu mogą być połączone wspólną magistralą (ich transmisje nie mogą się nakładać na siebie w czasie). Liczba otrzymanych w ten sposób magistral decyduje o liczbie portów jedynego używanego w omawianej metodzie przełącznika (każda magistrala jest podłączona do osobnego portu) – alternatywą dla porównań jest przełącznik o liczbie portów równej liczbie procesorów. Autorzy pracy [MMCCR06] zademonstrowali, że usunięcie z ruterów nieużywanych połączeń ze struktury przełącznika prowadzi do znacznych oszczędności zasobowo-energetycznych (rzędu kilkudziesięciu procent).

Bufory

Pamięci buforowe stanowią nieodłączny element sieci NoC, w znacznej mierze decydując o jej wydajności. Wbudowane są w moduły NI, niekiedy w łącza (przypadki długich, buforowanych połączeń), a przede wszystkim w rutery. Mają największy wpływ zarówno na pobór mocy jak i zasoby zajmowane przez jednostki przełączające ruch w sieci jednoukładowej [KLPS09]. Z tego powodu szuka się rozwiązań minimalizujących rozmiary pamięci buforowych przy zachowaniu wymagań związanych z wydajnością. Brane są pod uwagę dwa aspekty związane z buforami w ruterach: rozmiar oraz umiejscowienie. Ponadto każdy bufor może być pojedynczą strukturą pamięciową (najczęściej kolejką FIFO) lub występować w postaci tzw. kanału wirtualnego (ang. *Virtual Channel, VC*, Rys. 2.3.3-2) [KSJ04].



Rys. 2.3.3-2 Organizacja bufora w ruterze: jednolita (a) i w postaci kanału wirtualnego (b).

Kanał wirtualny (VC) umożliwia współdzielenie pojedynczego kanału fizycznego (łącza między ruterami) pomiędzy kilkoma kanałami logicznymi/transmisjami (na Rysunku 2.3.3-2b są przedstawione dwa kanały VC). Za stosowaniem rozwiązań opartych na buforach VC przemawia kilka zalet [KS03][MTCM05][BM06]:

- eliminacja zakleszczeń – dzięki zwielokrotnieniu liczby zasobów potencjalnie narażonych na wzajemną blokadę przy jednoczesnym zrównolegleniu dostępu do nich,
- zwiększenie stopnia wykorzystania kanałów fizycznych – wzrasta efektywność energetyczna układu poprzez minimalizację jałowych cykli pracy łącz,
- poprawa wydajności – zwielokrotnienie zasobów buforowych zmniejsza liczbę zatorów, co skraca czas transmisji w mikrosieci.

Tego typu podejście jest szczególnie chętnie wykorzystywane w połączeniu z metodą przełączania typu *wormhole*, gdyż eliminuje zjawisko zatorów i poprawia przepustowość w sieci jednoukładowej. Wadą zastosowania kanałów wirtualnych jest znaczny wzrost zużycia energii oraz zasobów układu [MTCM05] – efekt zwielokrotnionej struktury pamięciowej oraz rozbudowane w związku z tym komponenty arbitrażu i sterowania dostępem do niej. W pracy [BWMMS09] porównano pod kątem wydajności oraz efektywności zasobo-energetycznej dwie konstrukcje ruterów z kanałami wirtualnymi z ruterem typu *wormhole* bez VC oraz ruterem działającym w oparciu o przełączanie obwodów. Pod względem kosztu zasobowo-energetycznego najlepsza okazała się konstrukcja *wormhole*, natomiast rutery z kanałami wirtualnymi zaoferowały najwyższą wydajność mierzoną jako opóźnienia transmisji. Metodami szybkiego prototypowania w pracy [SG08] dowiedziono, że zastosowanie przełączników VC może być zbyt kosztowne w stosunku do oferowanych korzyści.

Pod względem umiejscowienia bufora w ruterze, można wyróżnić trzy rodzaje rozwiązań [KS03]: z buforami w portach wejściowych, wyjściowych oraz osadzonymi w strukturze przełączającej rutera (ang. *middle buffering*). Przeważająca liczba implementacji umiejscawia bufory w portach wejściowych i/lub wyjściowych [MCMMO04][SKH08]. Mimo to istnieją podejścia z centralnie umieszczonym buforem. W pracy [ZZHG05] porównano tego rodzaju rozwiązanie z ruterem korzystającym z buforowanego wyjścia z kanałami wirtualnymi. Oba rutery zaprojektowano tak, by oferowały zbliżoną wydajność. Centralizacja bufora pozwoliła na znaczne oszczędności pod względem użytych zasobów układu. W [WSZMB07] autorzy porównali efektywność konstrukcji opartej o jednolite bufory wejściowe, kanały wirtualne oraz umieszczoną centralnie pamięć bufora. Mimo, że ostatnie rozwiązanie wymagało większej liczby zasobów (średnio o 7%) w porównaniu z pojedynczymi buforami wejściowymi to wzrost wydajności przewyższył nawet wynik osiągnięty przez ruter korzystający z kanałów wirtualnych. Rozproszony bufor umieszczony wewnątrz dwublokowej struktury przełączającej rutera przedstawiono w [SRLP09]. Przeprowadzone eksperymenty wykazały ok.20% przewagę na polu wydajności nad ruterem o takiej samej przestrzeni buforowej zaimplementowanej w postaci wejściowych kanałów wirtualnych. Dwustopniowy system bufora centralno-wyjściowego o wydajności przewyższającej podstawowe trzy wersje umiejscowienia pamięci (o tej samej liczbie elementów) zademonstrowano w pracy [HH06]. Nie podano jednakże, jak nowa struktura wpływa na pobór mocy oraz ilość potrzebnych zasobów w układzie (skomplikowane sterowanie i arbitraż dostępu). Autorzy [KS03] zwracają uwagę na wadę rozwiązań korzystających ze scentralizowanego bufora związaną z wymaganiami dotyczącymi parametrów pamięci buforowej. Musi ona oferować wielodostęp przy wysokiej przepustowości

(wiele portów wejściowych/wyjściowych symultanicznie zapisuje/czyta dane). W pracy [BM06] wskazano również na skomplikowaną obsługę centralnego bufora związaną z wielodostępem oraz słabą wydajność w przypadku natłoku transmisyjnego.

Energo- i zasobochłonność buforów skłaniają wielu badaczy do poszukiwania rozwiązań minimalizujących rozmiar tego rodzaju pamięci lub optymalizujących jej wykorzystanie. Dodatkową motywacją jest fakt, że – paradoksalnie - zwiększanie rozmiaru buforów nie prowadzi do spodziewanej poprawy wydajności systemu [KJSFM02]. Optymalizacja wykorzystania to współużytkowanie bufora (buforów) wejściowego [LT06] lub wyjściowego [GG00] przez kilka kanałów fizycznych. Większość rozwiązań zakłada, że w przypadku indywidualnych buforów dla każdego wejścia/wyjścia ich rozmiar jest jednakowy. Hu i Marculescu w pracy [HM04a] zaprezentowali algorytm doboru rozmiaru każdego bufora na podstawie statystycznej analizy komunikacji dla danej aplikacji. Opierając eksperymenty na trzech benchmarkach z dziedziny przetwarzania informacji audio/wideo wykazali, że dla dostosowanej do potrzeb danej aplikacji sieci NoC w topologii *mesh* można tak ustalić wielkość buforów w ruterach, by bez straty wydajności osiągnąć ok.80% redukcję zasobów w porównaniu z rozwiązaniami z jednolitym rozmiarem buforów. Opracowano również technikę pozwalającą całkowicie wyeliminować rozbudowane bufory w ruterze z przełączaniem pakietów bez straty wydajności [GGLD08]. Wyniki porównano z klasyczną konstrukcją typu *wormhole*. Bezbuforowo może również funkcjonować element przełączający stosujący ruting/przełączanie typu *hot potato* [LZJ06][MM09].

2.4 Metody syntezy systemów Network-on-Chip

Systemy rozproszone MPSoC mogą należeć do jednej z dwóch grup: ogólnego przeznaczenia oraz specjalizowanych. Do drugiego rodzaju zaliczają się systemy wbudowane – rozwiązania o ściśle określonej funkcjonalności. Proces projektowania architektury dopasowanej do potrzeb konkretnej aplikacji wymaga ustalenia modelu opisującego funkcje i wymagania tejże, stanowiącego dane wejściowe dla obranej metodologii. Im więcej informacji dostarczy reprezentacja wejściowa, tym lepsza pod względem kosztu będzie zsyntezowana architektura. Decyzje podejmowane podczas projektowania sieci jednoukładowych dotyczą doboru liczby i rodzaju jednostek przetwarzających, wyboru sposobu ich połączenia (topologii), określenia zasad komunikacji (ruting, sterowanie przepływem, szeregowanie) oraz wynikającej z powyższych konstrukcji rutera. Wymienione czynności muszą uwzględniać koszt energetyczny oraz zasobowy docelowej implementacji układu, jak również spełniać ograniczenia czasowe związane z funkcjonowaniem aplikacji.

2.4.1 Modele aplikacji

W syntezie systemów NoC korzysta się z dwóch rodzin modeli do specyfikowania wymagań aplikacji: grafów charakteryzujących system (ang. *Application Characterization Graph, Core Graph – ACG, CG*) oraz grafów zadań (ang. *Task Graph, TG*). W przypadku modelu ACG węzłami grafu są jednostki przetwarzające, zaś krawędzie wskazują na kierunek transmisji między PE. Waga krawędzi to wymagana dla prawidłowego funkcjonowania systemu wydajność transmisji (ang. *required bandwidth*) lub jej wielkość (przesyłana w cyklu zegarowym). Tego typu model jest często wykorzystywany w metodologiach ukierunkowanych na syntezę wydajnych pod względem energetycznym architektur. Celem jest takie zaprojektowanie układu, by moduły PE wymieniające największe ilości danych były jak najbliżej siebie (minimalna liczba ruterów/łącz na trasie to minimalny koszt komunikacji) z zachowaniem ograniczeń wynikających z implementacji (koszt zasobów, szybkość działania). Graf ACG jest zgrubnym modelem aplikacji reprezentowanej w postaci zasobów (procesorów oraz sparametryzowanych pod względem wydajności połączeń). Modelem dużo dokładniejszym pod kątem dostarczanych informacji o specyfice działania systemu jest graf zadań. Węzły TG odzwierciedlają pojedyncze zadania aplikacji (często z podanym czasem wykonania), natomiast krawędzie to transmisje danych między zadaniami (wagi krawędzi odpowiadają ilości przesyłanych danych). Taki opis pozwala na wyrażenie zależności funkcjonalnych i czasowych pomiędzy zadaniami, co szczegółowo przedstawiono w rozdziałach 5.1 i 5.3 rozprawy. Zaletą grafu zadań jest również możliwość jego szeregowania, czyli określenia kolejności w jakiej mają być wykonywane zadania [KCJ94] i transmisje systemu [YG93]. Najczęściej stosuje się szeregowanie listowe – zadaniom przypisuje się priorytety wykonania, bazując na przykład na takich cechach zadań jak liczba i wielkości transmisji czy czas wykonania zadania. Im wyższy priorytet, tym szybciej zostanie ono wykonane. Inne strategie szeregują w pierwszej kolejności zadania o najbardziej rygorystycznych ograniczeniach czasowych (algorytmy *Earliest Deadline First* i *Critical Path based*). Najprostszy wariant to metoda ASAP (ang. *As Soon As Possible*), gdzie każde zadanie ma być zakończone najszybciej jak to możliwe lub funkcjonalnie przeciwna ALAP (ang. *As Last As Possible*).

2.4.2 Odwzorowanie aplikacji w sieć jednoukładową

W zależności od wejściowego modelu aplikacji oraz docelowej topologii NoC metody syntezy możemy podzielić na następujące grupy:

1. Synteza ACG w architektury regularne/nieregularne.
2. Synteza TG w architektury regularne/nieregularne.

Wybór regularnej topologii docelowej znacznie upraszcza proces projektowy, gdyż

decyzje są zawężone do przeszukania pewnego podzbioru rozwiązań. Na przykład ograniczenie do struktur *mesh* polega na znalezieniu odwzorowania procesorów (ACG) lub zadań systemu (TG) w węzły kraty tak, by wartości związane z kosztem były zminimalizowane (najczęściej pobór mocy) przy spełnieniu zadanych ograniczeń (zwykle związanych z szybkością pracy systemu). Podobnie reprezentacja wejściowa aplikacji wpływa na liczbę decyzji projektowych. Przykładowo model ACG pozbawia metodologię możliwości efektywnego szeregowania zdarzeń w systemie (zadań i transmisji) [MBSCW05]. Dotychczas nie opracowano działającej w czasie wielomianowym metody optymalnego (pod kątem kosztu i wydajności) odwzorowania aplikacji w regularną bądź nieregularną architekturę NoC, niezależnie od rodzaju modelu aplikacji.

Dla specyfikacji wejściowej zadanej grafem charakteryzującym system (ACG/CG) najczęściej stosowanym podejściem w projektowaniu systemów NoC jest tzw. rezerwacja pasma [MMAAC06]. Polega ona na takim projektowaniu systemu NoC, by łącza komunikacyjne miały pewien zapas przepustowości na najbardziej niekorzystny przypadek obciążenia transmisjami (rezultat założeń dotyczących semantyki ACG/CG). Osiąga się to poprzez różnicowanie szerokości magistral między ruterami jak również stosowanie różnych częstotliwości (szybkości) pracy dla poszczególnych obszarów sieci. Autorzy pracy [MMAAC06] przeszukują zbiór topologii zaczynając od takiej, gdzie wszystkie jednostki PE są podłączone do jednego wieloportowego rutera a kończąc na rozwiązaniu z indywidualnie przypisanymi do PE modułami przełączającymi. Kolejno generowane topologie podlegają ograniczeniom związanym z poborem mocy, długością łącz i zajmowaną powierzchnią w układzie docelowym. Podobne podejście prezentuje praca [CP08]. W odróżnieniu od poprzedników, Chan i Parameswaran nie poddają analizie zbioru topologii, lecz iteracyjnie próbują ulepszyć pewną topologię początkową utworzoną metodą *floorplanning*'u (rozmieszczenia w fizycznym układzie). Dopuszczają wykorzystanie ruterów o wielu portach lokalnych jak również połączenia bezpośrednie między procesorami. Obie powyższe metody podczas budowania architektury docelowej kierują się rezultatami partycjonowania ACG/CG (podział na podgrafy procesorów silnie powiązanych zależnościami transmisyjnymi). Rozwiązanie przedstawione w [NW08] generuje za pomocą metod symulowanego wyżarzania wstępną, podobną do listy dwukierunkowej, sieć połączeń w formie niepełnego *mesh*'a, do której następnie dodawane są dedykowane łącza między wybranymi węzłami. Dzięki temu stosowane rutery mają jeden port lokalny, a dodatkowe połączenia pozwalają na dywersyfikację ścieżek. Metoda przedstawiona w [SS07] ma na celu minimalizację opóźnień komunikacyjnych, ale proces generowania za pomocą metod symulowanego wyżarzania oraz przeszukiwania z tabu nie bierze pod uwagę aspektów

energetycznych. Rozwiązanie oparte na mikrosieci zbudowanej z identycznych pod względem liczby portów ruterów i sformułowane jako problem liniowego programowania całkowitoliczbowego zaproponowano w [SCK06]. Zaprojektowaną na bazie tej pracy szybką, suboptymalną heurystykę uzupełnioną o rozplanowanie elementów systemu w układzie ci sami autorzy prezentują w [SCK07]. Celem obu metod jest utworzenie topologii z minimalnym poborem energii przez łącza międzyruterowe. Żadne z wyżej wymienionych podejść nie bierze pod uwagę kwestii zatorów w sieci NoC sugerując, że potencjalne konflikty można rozwiązać za pomocą wirtualnych kanałów w ruterach. We wcześniejszej pracy wymienionych autorów [SCK05] problem kolizji komunikatów rozwiązywany jest za pomocą różnicowania przepustowości portów, nawet w obrębie pojedynczych węzłów przełączających. Przedstawiona metodologia najpierw rozmieszcza elementy NoC na płaszczyźnie układu (*floorplanning*), następnie do jednostek przetwarzających przypisuje rutery z określeniem wymagań dotyczących wydajności ich portów, potem sąsiadujące ze sobą rutery są scalane z uwzględnieniem ograniczeń wydajność/koszt a na końcu wyznaczane są najkrótsze ścieżki dla wszystkich wiadomości w mikrosieci. Reasumując metoda bazuje na silnie heterogenicznym projekcie rutera pod względem liczby i przepustowości portów. Pracami, w których podjęto problem odwzorowania ACG/CG w NoC z uwzględnieniem potencjalnych kolizji transmisyjnych są [HP06] i [CM08], jednakże celem obu jest minimalizacja konfliktów a nie ich wyeliminowanie. Ho i Pinkston [HP06] generują topologię zaczynając od podziału jednego, pełnego rutera łączącego wszystkie procesory na wiele mniejszych jednostek – ograniczeniem konstrukcyjnym jest maksymalna liczba portów rutera. Dzięki analizie czasowej ACG/CG są w stanie określić, które komunikaty będą ze sobą kolidowały w czasie. Dla takich komunikatów podczas etapu budowania topologii przydzielane są niepokrywające się trasy (mogą być nieminimalne). Wynikowa architektura zawiera rutery o wielu portach lokalnych (obsługa wielu jednostek PE), jak również węzły połączone zdublowanymi łączami dwukierunkowymi. Chou i Marculesu [CM08] proponują minimalizację kolizji komunikatów poprzez odpowiednie odwzorowanie jednostek PE w węzły topologii kraty. Metoda oparta jest na wyprowadzonej z całkowitoliczbowego programowania liniowego heurystyce minimalizującej koszt transmisji i liczbę pokrywających się odcinków tras dla komunikatów. Prócz jednego wyjątku [HP06] wszystkie wyżej wymienione metody generowania mikrosieci na podstawie ACG/CG zakładają użycie rutingu minimalnego. Wady podejścia opartego na rezerwacji pasma w projektowaniu architektur NoC w porównaniu z analizą czasową możliwych konfliktów komunikacyjnych wykazano również w pracach [MBSCW05] i [LC09]. W [MBSCW05] autorzy dowiedli, że system NoC uwzględniający zależności czasowe pomiędzy transmisjami pozwala skrócić czas

działania aplikacji oraz poprawić jej bilans energetyczny (średnio o kilkadziesiąt procent). Badania ograniczono jednakże tylko do odwzorowania aplikacji w sieć jednoukładową o topologii kraty. Leary i Chatha w pracy [LC09] porównali dedykowane pod względem topologii systemy zaprojektowane według obu zasad – rezerwacji pasma oraz analizy czasowej komunikacji. Przewaga wydajnościowa rozwiązania uwzględniającego kolizje była bezdyskusyjna. Jednakże metoda zastosowana w [LC09] nie jest pozbawiona braków – przede wszystkim zastosowanie rutingu minimalnego (najkrótsze ścieżki) zmniejsza elastyczność trasowania, będącego atutem sieci połączeń. Ponadto rozwiązywanie konfliktów poprzez dublowanie ruterów (typu *multi-local port*) dla spornych fragmentów sieci spowodowało średnio 25% wzrost energo- i zasobochłonności systemu. Dodatkowo powyższe podejście nie wykorzystuje możliwości szeregowania zadań i transmisji.

Mniej liczne propozycje syntezy NoC bazują na grafach zadań jako specyfikacjach wejściowych. W pracy [LK03] przedstawiono algorytm genetyczny przyporządkowujący węzły grafu zadań do połączonych w topologię *mesh* elementów PE. Celem jest minimalizacja czasu przetwarzania systemu. Ruting transmisji odbywa się według najkrótszych ścieżek, metoda szereguje również zadania i transmisje. W [HM05b] celem odwzorowania grafu zadań w topologię kraty jest minimalizacja kosztu energetycznego transmisji i zadań oraz unikanie kolizji transmisyjnych. Metodą realizacji bezkolizyjności jest szeregowanie zadań i transmisji oraz wyznaczanie nieprzecinających się tras dla komunikatów. Autorzy [CYC09] rozszerzają koncept z pracy [HM05b] o etap doboru elementów przetwarzających (kosyntezę). Metodologia oparta na sekwencji algorytmów genetycznych do alokacji zadań na PE, szeregowania zadań i komunikatów oraz wyznaczeniu minimalnych ścieżek transmisyjnych została zaprezentowana w [SK06]. Celem jest minimalizacja poboru mocy przez układ. Graf zadań aplikacji jest odwzorowywany w topologię *mesh*. Autorzy proponują różnicowanie przepustowości łącz międzyruterowych poprzez zmianę ich napięć zasilających. Ponadto postulują odłączanie nieużywanych łącz, aby ograniczyć statyczny pobór mocy układu. Minimalizację globalnego ruchu w mikrosieci obrali za cel autorzy pracy [JP10]. Graf zadań odwzorowują w zadany graf sieci NoC za pomocą metod heurystycznych. W przeciwieństwie do poprzednich prac biorą pod uwagę topologie nieregularne (w tym łącza jednokierunkowe) z heterogenicznymi elementami przetwarzającymi o różnych wymiarach w układzie. Do jednego zasobu PE może być przyporządkowane tylko jedno zadanie, dozwolony ruting – minimalny. Każda z powyższych prac zakłada, że topologia mikrosieci docelowej (regularna bądź nie) jest uprzednio zadana.

Zestawienie najpopularniejszych rozwiązań w dziedzinie odwzorowywania aplikacji w architekturę NoC można znaleźć w pracy [MMCM07]. Autorzy porównali również pod kątem

oszczędności energetycznych oraz czasu uzyskiwania rozwiązania kilka algorytmów probabilistycznych oraz zachłannych. Odwzorowaniu w topologię kraty podlegały aplikacje opisane modelem zbliżonym do grafu zadań (nie stosowano szeregowania zadań). Na podstawie przeprowadzonych eksperymentów autorzy argumentują, że najlepszy współczynnik *jakość/koszt uzyskania rozwiązania* oferują algorytmy zachłanne uzupełnione o jednokrotny przebieg jednego z algorytmów probabilistycznych.

Niezależnie od powyższych technik odwzorowania aplikacji w architekturę NoC kilka prac podjęło problem optymalizacji odwzorowanego w topologię regularną systemu. Zaburzenie regularności topologii poprzez usunięcie zbędnych łącz [WMAK08] i/lub redukcję złożoności portów ruterów [WMAK07] stanowi jedną z metod obniżenia kosztu zasobowo-energetycznego. Natomiast umieszczenie w mikrosieci dedykowanych łącz dla wybranych par komunikujących się jednostek przetwarzających pozwala na podniesienie jej wydajności [OM05][OMLC06].

3 Motywacja

Postęp technologiczny w dziedzinie wytwarzania układów scalonych umożliwia integrowanie w nich coraz większej liczby elementów logicznych. Jednocześnie rosnące zapotrzebowanie na wzrost mocy obliczeniowej systemów cyfrowych doprowadziło do intensywnego rozwoju metod przetwarzania rozproszonego i równoległego. Metody te wspierane są od strony sprzętowej przez architektury wieloprocesorowe, w tym w postaci jednoukładowych systemów wbudowanych MPSoC. W tradycyjnym podejściu część przetwarzająca dane w takim systemie składa się z wielu pracujących równolegle jednostek przetwarzających, obejmujących rdzenie uniwersalnych procesorów oraz wyspecjalizowanych modułów, komunikujących się ze sobą poprzez magistrale (ang. *bus-based systems*). W wielu pracach dowiedziono, iż tak skonstruowany system jest słabo skalowalny, tzn. maksymalna liczba jednostek przetwarzających jest relatywnie mała (kilka do kilkunastu). Przekroczenie tej bariery jest niemożliwe ze względu na wąskie gardło, jakim jest wspólna szyna komunikacyjna i brak możliwości efektywnego rozwiązania konfliktów w dostępie do niej.

Od dekady intensywnie badanym i rozwijanym przez świat nauki i przemysłu trendem w dziedzinie systemów wieloprocesorowych są sieci jednoukładowe (NoC). W architekturze NoC procesory (bardziej ogólnie – elementy przetwarzające) połączone są mikrosiecią wzorowaną na tradycyjnej sieci komputerowej, co pozwala na znaczne zwiększenie liczby kanałów komunikacyjnych. Tak, jak w tradycyjnej sieci komputerowej dopuszczalne są różne topologie połączeń: regularne i nieregularne. Istotne, by cały system NoC mógł być zaimplementowany w jednym (rzadziej kilku) układzie scalonym. Topologie regularne są najczęściej obierane do realizacji układów ogólnego przeznaczenia, takich jak 80 rdzeniowy procesor firmy Intel [VHRDW07], 64 rdzeniowy układ firmy Tilera [Til], kontroler pamięci w układach graficznych AMD Radeon [Rad] czy implementacja technologii AMD HyperTransport (tzw. mostek północny) [CH07].

W efekcie prowadzonych prac badawczych udowodniono, że architektury NoC pozwalają na niemal liniową skalowalność wydajności rzędu kilkudziesięciu połączonych ze sobą jednostek przetwarzających, co czyni je bardzo efektywnymi dla sprzętowej realizacji algorytmów równoległych/rozproszonych intensywnie przetwarzających dane, w tym dla konstruowania systemów wbudowanych (ang. *embedded systems*). Szczególną popularnością cieszą się implementacje w architekturach NoC systemów multimedialnych przetwarzających obraz i dźwięk oraz algorytmów kryptograficznych i DSP. Przykładowe implementacje algorytmów MPEG-2 i MPEG-4, w formie systemów NoC, okazały się kilkukrotnie wydajniejsze od dotychczas stosowanych rozwiązań magistralowych. Problematyka sieci

Network-on-Chip jest stosunkowo nową dziedziną i dla wielu problemów nie znaleziono jeszcze zadowalającego rozwiązania [OHM05][LTH07][MOPJH09]. Jedną z najważniejszych jest kwestia efektywnego odwzorowywania aplikacji w NoC. Powszechnie stosowany do tego celu model aplikacji ACG/CG z powodu swej niedokładności (brak informacji o zależnościach pomiędzy zdarzeniami w systemie oraz o czasie ich wystąpienia) generuje rozwiązania zbyt kosztowne, jeśli chodzi o zasoby i wydajność w porównaniu z modelami uwzględniającymi relacje czasowe (grafy z rodziny TG) [MBSCW05][MCMSR05][MKSC05]. Z kolei wśród istniejących rozwiązań bazujących na grafie TG jako specyfikacji wejściowej brak jest takiego, które w jednej metodologii łączyłoby generowanie taniej topologii połączeń (pobór mocy i wymagane zasoby), alokację zasobów (bezkolizyjne ścieżki dla komunikatów) i szeregowanie zadań oraz transmisji.

4 Cel i teza rozprawy

Niech specyfikacja systemu wbudowanego będzie podana w formie grafu zadań (TG). Zakłada się ponadto, że alokacja jednostek przetwarzających oraz przyporządkowanie zadań do modułów PE zostało dokonane istniejącymi metodami kosyntezy sprzętowo-programowej, generującymi architekturę w pełni połączoną (zbliżoną do rozwiązań P2P z Rys. 1.1-1b). W efekcie znane jest również uszeregowanie zadań i transmisji. Do budowy struktury NoC realizującej funkcje danego systemu wbudowanego będą wykorzystywane rutery o jednym porcie lokalnym do obsługi modułu PE oraz o maksymalnie czterech dwukierunkowych portach służących do połączeń międzyruterowych (Rys. 2.3.3-1). Koszt topologii (zasobowy) zdefiniowany jest jako liczba łącz jednokierunkowych w syntezowanej mikrosieci (i tym samym liczba portów w ruterach), natomiast koszt energetyczny określono jako średnią długość ścieżek dla wszystkich transmisji systemu.

Celem rozprawy jest opracowanie metodologii umożliwiającej znalezienie architektury NoC o minimalnych kosztach topologii i energetycznym, zapewniającej bezkolizyjne przesyłanie pakietów dla systemów działających z wydajnością określoną narzuconymi ograniczeniami czasowymi. Żadne z dotychczas stosowanych rozwiązań nie wykorzystywało informacji o zależnościach pomiędzy transmisjami do jednoczesnego projektowania topologii i szeregowania komunikatów w sieci NoC.

Hipoteza badawcza rozprawy brzmi: **statyczna analiza grafu zadań systemu wbudowanego, z uszeregowanymi i przyporządkowanymi zadaniami do elementów obliczeniowych, umożliwi znalezienie bezkolizyjnej sieci jednoukładowej o minimalnym koszcie topologii, w której koszt energetyczny transmisji będzie najmniejszy z możliwych oraz wszystkie ograniczenia czasowe systemu zostaną spełnione**.

Teza zostanie udowodniona poprzez wprowadzenie modelu matematycznego dla problemu syntezy sieci NoC i wykazanie, że informacje uzyskane na podstawie analizy statycznej grafu zadań są wystarczające, aby ten problem rozwiązać. Efektem opracowania aparatu matematycznego będzie algorytm odwzorowania systemu w topologię NoC, wyznaczania tras oraz szeregowania transmisji, minimalizujący koszt architektury (minimalna liczba ruterów i połączeń) oraz koszt energetyczny (minimalne bezkolizyjne ścieżki dla transmisji). Przeprowadzone eksperymenty, polegające na syntezie przykładowych rzeczywistych systemów wbudowanych oraz grafów zadań wygenerowanych losowo będą miały na celu wykazanie, że opracowana metodologia daje zwykle wyniki dużo lepsze od dotychczas stosowanych metod. Proponowaną w rozprawie metodologię podzielono na trzy etapy postępowania:

1. Analizę grafu zadań pod kątem zależności pomiędzy transmisjami. W tym kroku zakłada się, że zadania są przyporządkowanie do elementów obliczeniowych (alokacja zasobów, przyporządkowanie i uszeregowanie zadań będzie wykonane za pomocą istniejących metod kosyntezy systemów wbudowanych). Następnie na podstawie analizy zostanie utworzony formalny model komunikacji w systemie, wykorzystywany na następnych etapach metodologii.
2. Wybór topologii dla architektury NoC. Etap ten polega na określeniu topologii sieci i rozmieszczeniu w niej elementów obliczeniowych. Na podstawie analizy modelu uzyskanego w poprzednim kroku, zostaną wyznaczone wszystkie grupy transmisji, które zachodzą w tych samych przedziałach czasu. Następnie dla każdej transmisji w ramach jednej grupy zostanie alokowana inna ścieżka w sieci lub komunikat zostanie przeszeregowany.
3. Wyznaczenie tras i szeregowanie transmisji. Trasy dla transmisji z kolizjami zostaną wyznaczone na poprzednim etapie, pozostałe komunikaty będą przesyłane najkrótszymi ścieżkami. W pracy zostanie wykorzystany ruting, w którym trasa jest opisana w nagłówku pakietu, dzięki czemu będzie możliwe wyznaczenie różnych tras dla różnych transmisji pomiędzy tymi samymi elementami obliczeniowymi. Powyższa metoda rutingu umożliwi również zastosowanie prostych ruterów, co dodatkowo obniży koszt implementacji systemu.

Dokonana zostanie również ocena algorytmu poprzez analizę teoretyczną. W jej zakres wejdzie oszacowanie efektywności metody oraz sformułowanie warunków, jakie musi spełnić aplikacja opisana grafem zadań, aby wygenerowanie dla niej architektury NoC, spełniającej założenia metodologii, było możliwe.

5 Synteza sieci jednoukładowych – model formalny

W rozdziale drugim rozprawy zaprezentowano przegląd rozwiązań przyjętych w dziedzinie generowania sieci NoC. Wynika z niego, iż nie istnieje jedna uniwersalna metoda specyfikowania wymagań syntezowanego systemu, gdyż zależy to od domeny jego zastosowań (system wieloprocesorowy ogólnego przeznaczenia, specjalizowany system wbudowany przetwarzający w sposób rozproszony, system czasu rzeczywistego, itp.). Projektant ma do wyboru szerokie spektrum regularnych bądź nieregularnych topologii połączeń oraz musi zadecydować o konstrukcji użytego rutera. To z kolei wpływa na metodę możliwego do zastosowania rutingu (deterministyczny lub adaptacyjny). Wobec tak wielu alternatyw projektowych w niniejszym rozdziale zawarto definicje opisujące formalny model aplikacji odwzorowywanej w system wbudowany. Ponadto omówiono przyjęte założenia dotyczące struktury sieci jednoukładowej, w tym kwestie związane z ruterem. Podano również kryteria estymacji efektywności generowanej architektury *Network-on-Chip*, bazujące na szybkości pracy zaprojektowanego systemu, wymaganych zasobach oraz na jego poborze mocy.

5.1 Graf Zadań – model systemu

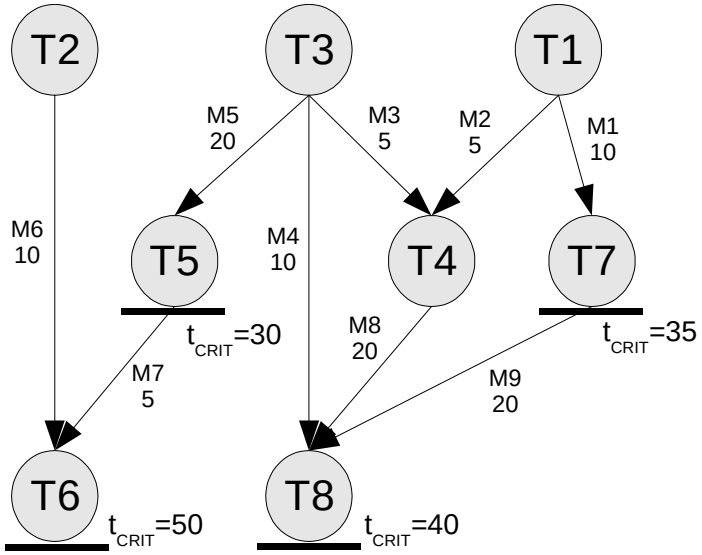
W problematyce związanej z analizą i projektowaniem sieci jednoukładowych dla systemów wbudowanych zorientowanych na przetwarzanie danych oraz charakteryzujących się przewidywalnym wzorcem komunikacji przyjęto posługiwanie się **Grafem Zadań** (ang. *Task Graph*, TG) systemu jako jednym ze sposobów specyfikowania danych wejściowych opisujących funkcje odwzorowywanej w architekturę NoC aplikacji [LK03][HM05a][SK06] [C07][CCM07][JS07][CYC09]. Graf zadań występuje w literaturze anglojęzycznej w różnych odmianach, jak na przykład *Communication Task Graph* [HM05b][YPDMV09] czy *Communication Dependence and Computation Graph* [LT06]. Ten sposób reprezentacji ma swe źródło w zagadnieniach związanych z tematyką szeregowania zadań w wieloprocesorowych systemach równoległych [YG92][GY93][YG93][CKTA94][KCJ94][KA98] oraz w problematyce kosyntezy sprzętowo-programowej dla systemów wbudowanych [SW97] [DRW98][D02].

DEFINICJA 5.1-1 (GRAF ZADAŃ)

Graf Zadań to acykliczny graf skierowany $TG = (\mathbf{T}, \mathbf{M}, c, t_{CRIT})$ *reprezentujący program równoległy, gdzie każdy wierzchołek* $T_i \in \mathbf{T}$ *oznacza pojedyncze zadanie aplikacji, zaś każda krawędź* $(T_i,T_j) = M_{i,j} \in \mathbf{M}$ *to transmisja (zwana również komunikatem lub wiadomością) pomiędzy zadaniami* T_i *oraz* T_j*. Do każdej krawędzi grafu przypisane są wagi* $c(M_{i,j})$*. Ponadto do każdego zadania* T_i *przypisany jest indywidualny parametr czasowy* $t_{CRIT}(T_i)$*, oznaczający*

maksymalny czas, w którym dane zadanie musi zakończyć swoje wykonywanie od momentu rozpoczęcia wykonywania pierwszego zadania w grafie.

Zadanie T_i (ang. *task*) to pewien skończony zbiór obliczeń lub instrukcji, które są wykonywane sekwencyjnie i które nie podlegają zrównolegleniu ani wywłaszczeniu przez inne zadanie. Stopień wejściowy każdego wierzchołka $d_{WE}(T_i)$ oznacza liczbę odbieranych transmisji, które warunkują rozpoczęcie wykonywania zadania T_i. Analogicznie stopień wyjściowy $d_{WY}(T_i)$ oznacza liczbę komunikatów wysyłanych do innych zadań i będących rezultatem wykonania zadania T_i. Waga krawędzi $M_{i,j}$ (ang. *message*) jest liczbą nieujemną odzwierciedlającą ilość przesyłanych danych pomiędzy zadaniami T_i i T_j (wolumen transmisji). Algorytm reprezentowany grafem zadań może być wykonywany cyklicznie. Jeżeli dla danego zadania nie wymaga się spełnienia określonego reżimu czasowego to przyjmuje się, że $t_{CRIT}(T_i) = \infty$. Ograniczenie t_{CRIT} (ang. *deadline*) dla całego grafu stanowi maksimum z danego zbioru limitów czasowych [D02]. Rys. 5.1-1 przedstawia przykładowy Graf Zadań, podkreślenia węzłów symbolizują zadania z nałożonymi ograniczeniami czasowymi.



Rys. 5.1-1 Model aplikacji w postaci Grafu Zadań.

DEFINICJA 5.1-2 (PRZODKOWIE I POTOMKOWIE ZADANIA TI)

*Jeżeli dla dwóch dowolnych wierzchołków T_i, $T_x \in$ **T** istnieje krawędź (transmisja) $M_{x,i} \in$ **M** to wierzchołek (zadanie) T_x i krawędź $M_{x,i}$ nazywamy przodkami zadania T_i. Zbiór wszystkich zadań-przodków zadania T_i oznaczamy **predT**(T_i), zaś zbiór wszystkich transmisji-przodków - **predM**(T_i). Wielkość każdego z powyższych zbiorów wynosi $d_{WE}(T_i)$.*

*Analogicznie, jeżeli dla dwóch dowolnych wierzchołków T_i, $T_x \in$ **T** istnieje krawędź (transmisja) $M_{i,x} \in$ **M** to wierzchołek (zadanie) T_x i krawędź $M_{i,x}$ nazywamy potomkami zadania*

*T_i. Zbiór wszystkich zadań-potomków zadania T_i oznaczamy **succT**(T_i), zaś zbiór wszystkich transmisji-potomków - **succM**(T_i). Wielkość każdego z powyższych zbiorów wynosi $d_{WY}(T_i)$.*

Krawędzie grafu TG wyznaczają kolejność wykonywania zadań oraz wynikające z tego zależności czasowe pomiędzy nimi. Zadanie T_i może rozpocząć wykonywanie (czas rozpoczęcia $t_{START}(T_i)$) dopiero po otrzymaniu wszystkich transmisji ze zbioru ***predM***(T_i), natomiast pierwsza wiadomość $M_{i,x}$ może zostać wysłana dopiero po zakończeniu T_i (czas zakończenia – $t_{STOP}(T_i)$). Pojęcie „czas" jest tutaj terminem umownym, rozumianym w kategoriach przyczynowo-skutkowych jako „przed/po". Dokładne określenie momentów rozpoczęcia/zakończenia zdarzeń związanych z zadaniami oraz transmisjami jest możliwe dopiero po odwzorowaniu Grafu Zadań na konkretną architekturę wykonawczą.

WARUNEK 5.1-1 (ROZPOCZĘCIE ZADANIA)

Jeżeli $d_{WE}(T_i) > 0$ oraz $T_i \in \mathbf{T}$ *to czas startu zadania $t_{START}(T_i)$ musi spełniać następujący warunek:*

$$t_{START}(T_i) \geq \max \{t_{STOP}(M_{x,i}) : M_{x,i} \in \mathbf{predM}(T_i)\} \quad (5.1\text{-}1)$$

Jeżeli $d_{WE}(T_i) = 0$ to $t_{START}(T_i)$ nie jest uwarunkowany zależnościami wynikającymi z grafu TG.

WARUNEK 5.1-2 (ROZPOCZĘCIE TRANSMISJI)

Czas rozpoczęcia transmisji warunkuje następująca formuła:

$$\forall\, M_{i,x} \in \mathbf{succM}(T_i),\ t_{START}(M_{i,x}) \geq t_{STOP}(T_i) \quad (5.1\text{-}2)$$

W przypadku, gdy zadanie generuje więcej niż jedną transmisję model TG nie nakłada żadnych wymagań co do kolejności wysłania wszystkich wiadomości (metoda transmisji zależy od architektury, na której jest implementowany dany Graf Zadań).

5.2 Model architektury Network-on-Chip

Jak podano w rozdziale 2 dominującym trendem w badaniach, projektowaniu i syntezie układów NoC jest konstrukcja oparta na szeregu połączonych ze sobą elementów przetwarzających PE [BM06]. Ich przeznaczeniem jest wykonywanie zadań systemu oraz przesyłanie między sobą komunikatów. Transmitowana wiadomość może zawierać dane lub informacje kontrolno-sterujące. Elementem przetwarzającym może być rdzeń procesora uniwersalnego (ang. *Processor Core*), jak np. NIOS II firmy Altera [Nios], Micro Blaze firmy Xilinx [Mblaze] czy PowerPC w układach Virtex II Pro/4/5, również produkowanych przez firmę Xilinx, procesor sygnałowy DSP, pamięć, czy inny moduł realizujący określone funkcje

(ang. *Intellectual Property, IP*). Wszystkie elementy przetwarzające, zwane dalej w rozprawie **procesorami**, realizują podzielony na zadania równoległy algorytm danego systemu wbudowanego. Wymiana komunikatów zachodzi za pośrednictwem sieci połączeń tworzonej przez rutery. Zgodnie z literaturą [BM06][GCK07] przyjęto, że jeden ruter może być bezpośrednio połączony tylko z jednym modułem PE za pośrednictwem interfejsu sieciowego NI.

DEFINICJA 5.2-1 (WĘZEŁ SIECI JEDNOUKŁADOWEJ, *NETWORK NODE*)

Pojedynczym elementem architektury Network-on-Chip jest węzeł (NN), składający się z procesora (PE), interfejsu sieciowego (NI) oraz rutera. Węzeł posiada następujące własności:

1. *Procesor może wykonywać sekwencyjnie oraz bez wywłaszczania dowolną liczbę zadań ze zbioru **T**.*
2. *W danej chwili PE wykonuje tylko jedno zadanie T_i.*
3. *Kolejność wykonywania zadań musi spełniać wymagania Warunku 5.1-1.*
4. *Transmisje pomiędzy zadaniami przypisanymi do tego samego węzła sieci (komunikacja lokalna) mają zerowy koszt (w sensie czasu transmisji) i nie angażują modułu NI oraz rutera.*
5. *Procesor ma możliwość opóźnienia rozpoczęcia wykonywania zadania do momentu zajścia określonego zdarzenia (szeregowanie zadań).*

Założenia 1 i 2 w powyższej definicji pozwalają na wykorzystanie rozbudowanych możliwości obecnych układów typu SoC [Alt][Xil][SK06][CYC09]. Zerowy koszt transmisji wymieniony w założeniu 4 wynika z faktu realizowania transmisji lokalnych (w obrębie tego samego modułu PE) za pomocą operacji zapisu/odczytu do/z pamięci lokalnej RAM, bez wykorzystania infrastruktury komunikacyjnej systemu wbudowanego (Rys. 5.2-1). Założenie 5 jest wykonalne dzięki mechanizmom systemu operacyjnego (np. Nucleus firmy MentorGraphics [Men], QNX Neutrino firmy QNX Software Systems [Qnx] czy MicroC/OS-II firmy Micrium [Mic]) sterującego pracą modułu PE (procesora) w systemie wbudowanym [EPPD00][HM05b].

Wzorem rozwiązań zaprezentowanych w rozdziale 2, moduł interfejsu sieciowego (NI) służy do konwersji wiadomości otrzymanej od jednostki PE do formatu akceptowalnego na poziomie warstwy fizycznej w sieci. Odbywa się to poprzez podział komunikatu na flity, czyli najmniejsze możliwe do równoległego przesłania między ruterami porcje danych. Czynność ta obejmuje również wpisanie informacji o trasie do flita nagłówkowego (ang. *Head Flit*) zgodnie

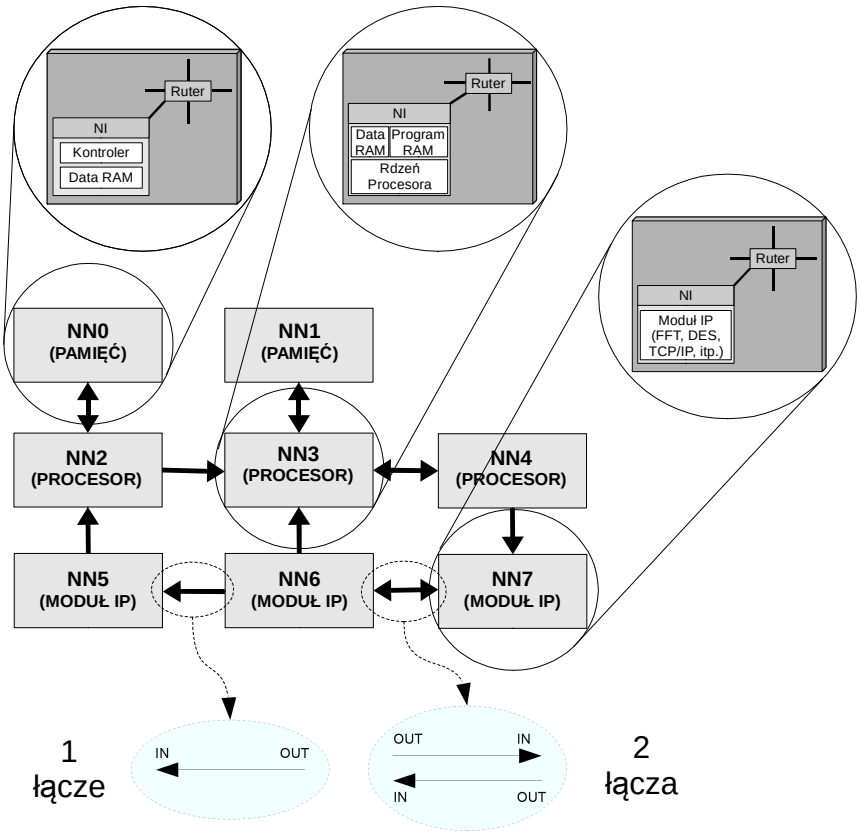
z przyjętą strategią rutingu. Następnie NI przeprowadza transmisję flitów do najbliższego rutera za pomocą tzw. portu lokalnego (ang. *Local Port, LP*). Odbiór transmisji odbywa się na odwrotnej zasadzie – flity przesyłane z portu LP najbliższego rutera są składane w finalny komunikat przekazywany następnie do modułu PE.

DEFINICJA 5.2-2 (INTERFEJS SIECIOWY, *NETWORK INTERFACE*)

Interfejs Sieciowy NI, będący końcówką kanału komunikacyjnego między Węzłami Sieci (NN) ma następujące cechy:

1. *Odbiór i nadawanie w NI może zachodzić równocześnie i niezależnie od siebie.*
2. *Transmisje są niezależne od modułu PE i realizowanych przez niego zadań (w szczególności - PE przygotowuje dane do wysłania jako rezultat wykonywanego zadania, zaś transmisją zajmuje się moduł NI).*
3. *Moduł NI ma możliwość opóźnienia rozpoczęcia transmisji do momentu zajścia określonego zdarzenia (szeregowanie transmisji).*
4. *Moduł NI może w danej chwili wysyłać tylko jeden komunikat M oraz odbierać tylko jeden komunikat M - przeplot odbieranych lub wysyłanych flitów należących do różnych komunikatów nie jest dozwolony.*

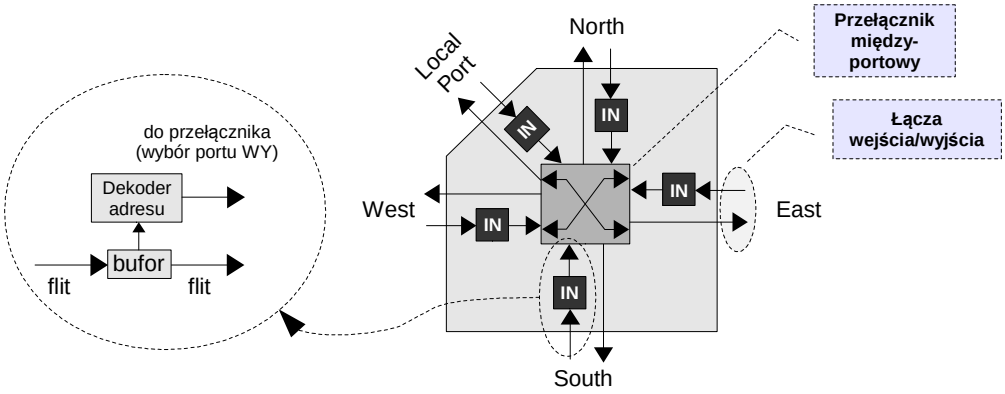
Założenia 2 oraz 3 Definicji 5.2-2 są zgodne z pracami z zakresu szeregowania zadań w systemach równoległych [SS01][SS05] będących inspiracją dla wielu propozycji rozwiązań dotyczących architektur NoC. Założenie 4 wyklucza systemy NoC klasy *Time Division Multiplexing* (TDM) [GDMPR03][RGRDM03][GDR05][HCG07][SHG08], upraszczając konstrukcję NI oraz ruterów. Na Rys. 5.2-1 przedstawiono przykładowy schemat sieci jednoukładowej wzorowany na pracy [BLB07].



Rys. 5.2-1 Architektura NoC zorganizowana w topologię nieregularnej kraty.

5.2.1 Ruter oraz protokoły wyznaczania tras

Elementami sieci jednoukładowych, stanowiącymi niejako podstawę ich działania, są rutery, zwane również przełącznikami. W rozprawie przyjęto model komunikacji sieciowej oparty na przełączaniu pakietów. Schemat budowy rutera przestawiono na Rys. 5.2.1-1.



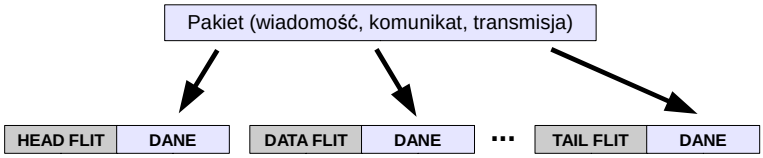
Rys. 5.2.1-1 Model architektury rutera.

Podczas określania funkcji rutera wzięto pod uwagę następujące czynniki:

1. **Liczba portów** – w związku z tym, że większość prac bazuje na topologiach odwzorowywanych w struktury planarne (2D) charakterystyczne dla układów FPGA (krata, torus, pierścień, oktagon, dwuwymiarowe topologie nieregularne) [DBGBB03] [MNTKJ04][GDR05][SKH08], naturalnym jest założenie maksymalnej liczby portów

równej 5 dla każdego rutera (jeden port lokalny i 4 służące do połączeń międzyruterowych (ang. *links*) w kierunkach osi współrzędnych X i Y – *North*, *East*, *South* i *West*). Każdy port jest dwukierunkowy (ang. *full-duplex*), czyli umożliwia jednoczesne nadawanie i odbiór danych. Wszystkie porty mają te same cechy, tzn. szerokość magistrali oraz rozmiar buforów. Szerokość portu jest determinowana przyjętym rozmiarem flita i waha się od kilku [STAN04] do kilkuset bitów [DT01] - minimalny rozmiar wynika z konieczności zmieszczenia informacji o trasie w jednym flicie (nagłówkowym). W związku z tym, że celem rozprawy jest synteza dedykowanych topologii nieregularnych, dopuszcza się redukcję struktury rutera do niezbędnego minimum poprzez eliminację zbędnych portów (ich części wejściowych i/lub wyjściowych) [MMCCR06].

2. **Sposób przełączania flitów w ruterze** – założono model typu *wormhole*, uznawany za najlepszy dla architektur sieci jednoukładowych [BM06][SKH08]. Metoda ta zakłada potokową transmisję podzielonego na flity komunikatu. Każdy flit zawiera informację o swoim typie oraz ładunek. Wyróżniono trzy typy flitów (zilustrowane na Rys. 5.2.1-2):
   - flit nagłówkowy (ang. *Head Flit*) - służy do przenoszenia informacji o trasie i określenia portu wyjściowego w każdym ruterze dla danej transmisji wieloflitowej; po otrzymaniu flita nagłówkowego ruter dekoduje zapisaną w jego części ładunkowej trasę i przy pomocy przełącznicy krzyżowej (ang. c*rossbar switch*) zestawia połączenie pomiędzy swoim portem wejściowym a odpowiednim portem wyjściowym; połączenie międzyportowe jest utrzymywane do momentu otrzymania flita zamykającego,
   - flit danych (ang. *Data Flit*) – przenosi dane użyteczne przesyłanej wiadomości, podąża trasą zestawioną w ruterze przez flit nagłówkowy; większość flitów przesyłanych siecią jednoukładową jest tego typu,
   - flit zamykający (ang. *Tail Flit*) – oznacza koniec potoku flitów tworzącego daną transmisję; w jednej przesyłanej wiadomości jest tylko jeden flit nagłówkowy i jeden zamykający; część ładunkowa flita zamykającego zawiera najczęściej sumę kontrolną całej wiadomości.



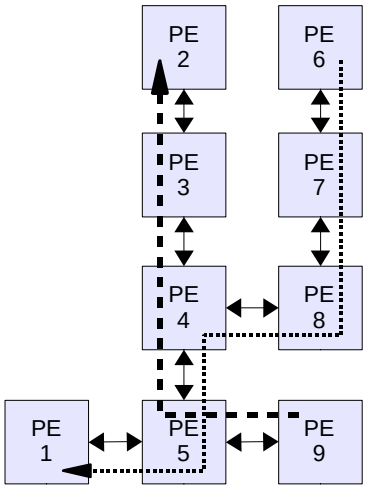
Rys. 5.2.1-2 Relacja pomiędzy komunikatem a flitami.

3. **Transmisje symultaniczne w ruterze** – zgodnie z podejściem zaprezentowanym

w pracach [SBKV05][SV06][SV07] zakłada się możliwość jednoczesnego zestawienia kilku niezależnych połączeń *port wejściowy → port wyjściowy*, co osiągane jest poprzez zastosowanie modułu dekodującego informacje adresowe w każdym porcie wejściowym rutera; w ogólności dla *n* dwukierunkowych portów można zestawić *n* niezależnych połączeń.

4. **Wyznaczanie trasy (ruting)** – przyjęto szeroko stosowany dla systemów wbudowanych o przewidywalnym wzorcu transmisji ruting, gdzie informacja o trasie jaką musi przebyć każda wiadomość jest uprzednio określona i zapisana w nagłówku komunikatu. Nie jest to jednak metoda ściśle deterministyczna, gdyż zakłada, że różne komunikaty pomiędzy węzłem nadawczym a odbiorczym mogą być przesyłane różnymi trasami, niekoniecznie najkrótszymi – z tego względu metoda nosi cechy rutingu adaptacyjnego, zaś etap dostosowania komunikacji do potrzeb danej aplikacji jest dokonywany w momencie projektowania systemu wbudowanego o architekturze NoC [GCK07]. W niniejszej pracy zaproponowano stosowanie rutingu z trasą zapisaną w nagłówku przez nadawcę (ang. *source routing*), opartego na jednej z następujących zasad:
   - cała trasa zapisana w nagłówku - jeżeli topologia jest odwzorowana w strukturę dwuwymiarową „wpisaną” w kartezjański układ współrzędnych to dla każdego rutera na trasie wystarczą 2 bity, by określić jak komunikat ma być przekierowany w danym ruterze w stosunku do portu wejściowego (prześlij na wprost, w lewo, w prawo, do portu lokalnego/modułu PE) [NoCSim] – przykład dla dwóch wiadomości podano na Rys. 5.2.1-3a. Na podobnej zasadzie bazuje ruting wyznaczający port wyjściowy kodowany według zasady „zgodnie z kierunkiem wskazówek zegara” (ang. *clockwise*) [HH06], dla którego przykład podano na Rys. 5.2.1-3b. Dla struktur docelowych innych niż planarne zamiast desygnatorem kierunku można posługiwać się identyfikatorem portu wyjściowego dla każdego rutera – ten sposób jest bardziej uniwersalny od poprzedniego, ale wymaga 3 bitów/hop [LLSY05] - ilustruje to Rys. 5.2.1-3c. Wspólną cechą powyższych rozwiązań jest fakt, iż zawsze na początku flita nagłówkowego znajduje się informacja dla bieżącego rutera uaktualniana po przekierowaniu poprzez przesuw części adresowej o 2 lub 3 bity w lewo,
   - trasowanie oparte na tabelach rutingu umieszczonych w każdym ruterze [DT08a] [DT08b] – w tym podejściu w nagłówku umieszczany jest jedynie identyfikator komunikatu. W każdym ruterze znajduje się niewielka pamięć przechowująca indywidualną tablicę rutingu w postaci *identyfikator komunikatu → identyfikator portu wyjściowego rutera*, gdzie identyfikator portu jest tożsamy z przykładowymi

oznaczeniami z Rys. 5.2.1-3c.



Topologia nieregularna 2D
(dwie transmisje)

Wpis / Port WE	00	01	10	11
East	w lewo	w prawo	prosto	do LP
South	w lewo	w prawo	prosto	do LP
West	w lewo	w prawo	prosto	do LP
North	w lewo	w prawo	prosto	do LP
LP	West	East	North	South

Komunikat 0: 6 (LP) → 7 → 8 → 4 → 5 → 1
Trasa (wpisy w nagłówku): 11 10 01 00 01 11

Komunikat 1: 9 (LP) → 5 → 4 → 3 → 2
Trasa (wpisy w nagłówku): 00 01 10 10 11

a)

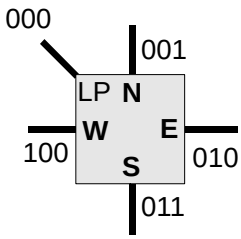
Wpis / Port WE	00	01	10	11
East	South	West	North	do LP
South	West	North	do LP	East
West	North	do LP	East	South
North	do LP	East	South	West
LP	East	South	West	North

Komunikat 0: 6 → 7 → 8 → 4 → 5 → 1
Trasa (wpisy w nagłówku): 00 10 11 00 11 11

Komunikat 1: 9 → 5 → 4 → 3 → 2
Trasa (wpisy w nagłówku): 10 10 01 01 10

b)

Identyfikatory portów rutera



Komunikat 0: 6 (LP) → 7 → 8 → 4 → 5 → 1
Trasa (wpisy w nagłówku): 011 011 100 011 100 000

Komunikat 1: 9 (LP) → 5 → 4 → 3 → 2
Trasa (wpisy w nagłówku): 100 001 001 001 000

c)

Rys. 5.2.1-3 Reguły trasowania źródłowego: z desygnatorami kierunków (a), wg wskazówek zegara (b) oraz z identyfikatorami portów wyjściowych (c).

Wadą rozwiązania z tabelami rutingu jest większe zapotrzebowanie rutera na pamięć, choć można temu zaradzić poprzez zastosowanie technik minimalizujących jej rozmiar [BCGK07]. Jedną z najprostszych metod kompresji jest przechowywanie informacji jedynie o tych komunikatach, które są przesyłane danym ruterem oraz przyjęcie kierunków domyślnych (przykładowo: jeżeli w tabeli rutingu brak trasy dla danego identyfikatora wiadomości, wtedy prześlij ją portem o oznaczeniu 001). Z drugiej strony cała trasa zapisana w nagłówku może wymagać większego rozmiaru flita w przypadku

długich tras (każdy hop, czyli propagacja przez ruter, to kolejna pozycja w nagłówku), gdyż cała trasa musi zmieścić się w jednym flicie - przekłada się to na szerokość magistral łączących porty ruterów. Cechą wspólną wszystkich powyższych metod rutingu jest również fakt, że zawracanie pakietów do tego samego portu w ramach danego rutera nie jest dozwolone. Ze względu na pozostałe cechy rutera (maksymalna liczba portów), w rozprawie zakłada się zastosowanie rutingu z desygnatorami kierunków (Rys.5.2.1-3a). Alternatywnie można się posłużyć zbliżoną funkcjonalnie metodą rutingu „według wskazówek zegara" (Rys.5.2.1-3b).

5. **Bufory danych** – przełączanie typu *wormhole* charakteryzuje się bardzo małym zapotrzebowaniem na przestrzeń buforową – wystarczy bufor o rozmiarze jednego flita umiejscowiony w każdym porcie wejściowym rutera; porty wyjściowe nie wymagają buforowania – przesyłają dane bezpośrednio po otrzymaniu z wejścia do portu wejściowego kolejnego rutera lub modułu NI. W związku z tym, że ryzyko wystąpienia rywalizacji o ten sam port wyjściowy w ruterze pomiędzy kilkoma pakietami jest wyeliminowane poprzez desygnację dedykowanych bezkolizyjnych tras, można również wykorzystać ruter bezbuforowy (ang. *bufferless*, [MM09]). Takie rozwiązanie pozwala na dodatkowe oszczędności w zakresie poboru energii przez sieć jednoukładową oraz w zapotrzebowaniu na zasoby układu (komórki logiczne, bloki pamięci, itp.).
6. **Arbitraż pomiędzy portami** – w związku z tym, że idea rozprawy zakłada eliminację kolizji pomiędzy transmisjami, moduły arbitrów (jak również kanały wirtualne) kierujące ruchem pomiędzy portami wejściowymi i wyjściowymi rutera są zbędne.

Zestaw powyższych zasad składa się na bardzo prostą konstrukcję rutera. Małe zapotrzebowanie na przestrzeń buforową oraz możliwość redukcji liczby portów tworzą energo- i zasobooszczędną strukturę rutera, cechy szczególnie pożądane w przypadku ograniczeń wynikających z implementacji systemu wbudowanego w układach ASIC lub FPGA. Należy zwrócić uwagę na jeszcze jeden istotny aspekt proponowanej architektury – w odróżnieniu od podejścia przyjętego w planowaniu komunikacji w systemach wieloprocesorowych [SS05] czy systemach NoC z podziałem czasu (TDM) i priorytetyzacją komunikatów [GDMPR03] [RGRDM03][GDR05] rutery nie szeregują transmisji. Innymi słowy ruter przesyła komunikat natychmiast po jego otrzymaniu – taka zasada działania zwana jest w anglojęzycznej terminologii *hot potato* [LZJ06][MM09].

5.2.2 Pobór mocy i opóźnienia transmisji

W rozprawie przyjęto stosunkowo zgrubny model architektury NoC (ang. *coarse grained*), którego celem nie jest dokładna analiza właściwości energetycznych czy

wydajnościowych wygenerowanego rozwiązania. Jednakże, ze względu na konieczność porównania proponowanej metody z istniejącymi algorytmami o podobnym przeznaczeniu [MMAAC06][MMCM07] niezbędne jest zdefiniowanie kryteriów pozwalających oszacować jakość otrzymanych rozwiązań. Estymację podzielono na część związaną z zapotrzebowaniem energetycznym oraz szybkością działania zbudowanej architektury, przy czym drugi parametr jest mierzony w cyklach zegarowych.

Koszt energetyczny architektury

Pobór mocy systemu wbudowanego odwzorowanego w sieć jednoukładową jest sumą mocy zużywanej przez rutery *E(r)*, połączenia sieciowe *E(l)* oraz procesory *E(p)* [C07]:

$$E_{NoC} = \sum_{\forall r \in \boldsymbol{RT}} E(r) + \sum_{\forall l \in \boldsymbol{L}} E(l) + \sum_{\forall p \in \boldsymbol{P}} E(p) \qquad (5.2.2\text{-}1)$$

Składowa *E(p)* zależy od wykorzystanych procesorów i jest proporcjonalna do czasu ich pracy. W związku z tym, że porównywane topologie budowane są z tych samych modułów PE, dla danego systemu opisanego grafem TG zakłada się, że składowa *E(p)* jest wprost proporcjonalna do czasu działania systemu NoC [MBSCW05] mierzonego w cyklach zegarowych, czyli do czasu wykonania wszystkich zadań. Przyjmując wartość $E_{ŚR}(P)$ jako średnie zużycie energii przez wszystkie procesory PE przypadające na cykl zegarowy otrzymujemy:

$$E(p) = E_{ŚR}(P) * t \qquad (5.2.2\text{-}2)$$

gdzie *t* oznacza czas działania systemu.

Parametry *E(r)* oraz *E(l)* wiążą się ściśle z architekturą sieci jednoukładowej i definiowane są jako koszt energetyczny przesłania jednego bitu (ang. *bit energy*) przez sieć [HM05a]. A zatem:

$$E(r) + E(l) = E_{bit} = E_{S(bit)} + E_{B(bit)} + E_{W(bit)} + E_{L(bit)} \qquad (5.2.2\text{-}3)$$

$E_{S(bit)}$, $E_{B(bit)}$ i $E_{W(bit)}$ reprezentują koszt przesłania bitu przez ruter i oznaczają odpowiednio składowe związane ze strukturą przełączającą wewnątrz rutera, buforowaniem oraz energią pobieraną przez wewnętrzne połączenia rutera. Intuicyjnie – im więcej portów ma ruter, tym bardziej rozbudowana będzie jego struktura przełączająca (ang. *crossbar*), buforowa oraz połączeniowa. Czynnik $E_{L(bit)}$ to wkład energetyczny wnoszony przez połączenia międzyruterowe. W celu stworzenia modelu energetycznego rutera posłużono się wynikami kilku prac, uzależniającymi pobór mocy od liczby i rodzaju portów (przy założeniu obciążenia rutera małym i średnim natężeniem transmisji). W niżej wymienionych pracach stwierdzono, że:

- buforowane wejście powoduje, że część wejściowa portu zużywa do 5-razy więcej energii niż bezbuforowa część wyjściowa [SCK05],

- każda kolejna para portów (wejście/wyjście, jeden bufor) rutera powoduje wzrost zapotrzebowania na energię o ok. 20% [C07] oraz na zasoby układu o ok. 35%-50% [MMAAC06][SV06].

Na podstawie powyższych obserwacji założono, że pobór mocy przez strukturę ruterów jest proporcjonalny do liczby ich portów wejściowych i wyjściowych:

$$E(r) = \sum_{\forall r_i \in \boldsymbol{RT}} E(r_i) = \sum_{\forall r_i \in \boldsymbol{RT}} E_j * (0.8 * n_{WE}(r_i) + 0.2 * n_{WY}(r_i)) \quad (5.2.2\text{-}4)$$

gdzie E_j to energetyczny koszt jednostkowy przypadający na jeden dwukierunkowy port rutera r_i, natomiast $n_{WE}(r_i)$ i $n_{WY}(r_i)$ oznaczają liczbę portów wejściowych i wyjściowych.

W związku z tym, że wszystkie połączenia międzyruterowe traktowane są jednakowo, ich koszt energetyczny można wyrazić jako:

$$E(l) = E_{L(bit)} = n * E_j(l) \quad (5.2.2\text{-}5)$$

gdzie n to liczba połączeń jednokierunkowych, zaś $E_j(l)$ to koszt pojedynczej jednokierunkowej magistrali łączącej dwa rutery.

DEFINICJA 5.2.2-1 (KOSZT ENERGETYCZNY)

Koszt przesłania w sieci jednoukładowej strumieni bitów (flitów) należących do wszystkich komunikatów danej aplikacji jest określony jako wartość proporcjonalna do sumy trzech czynników: czasu działania systemu, liczby portów wejściowych i wyjściowych w użytych do budowy sieci ruterach oraz liczby połączeń międzyruterowych.

$$E_{NoC} = E_{ŚR}(P) * t + \sum_{\forall r_i \in \boldsymbol{RT}} E_j * (0.8 * n_{WE}(r_i) + 0.2 * n_{WY}(r_i)) + n * E_j(l) \quad (5.2.2\text{-}6)$$

Definicja 5.2.2-1 określa statyczną składową pobieranej mocy i stanowi jedno z kryteriów szacowania jakości otrzymanego rozwiązania. Uzależnienie kosztu energetycznego od wymienionych w definicji czynników wynika z równań (5.2.2-2), (5.2.2-4) i (5.2.2-5). Dwie ostatnie składowe mają również wpływ na koszt topologii NoC (zapotrzebowanie na zasoby układu). Warto zwrócić uwagę, że dodanie do topologii jednego łącza jednokierunkowego wymaga użycia dwóch portów w ruterach: wyjściowego w nadawcy transmisji oraz wejściowego w odbiorcy.

Opóźnienia transmisji (ang. *Message Latency*)

Istotnym parametrem sieci jednoukładowej jest opóźnienie w przesyłaniu komunikatu związane z liczbą ruterów (ang. *hop*) na trasie. Na opóźnienie to składa się czas przesyłu t_l przez łącze prowadzące do portu rutera, w tym portu lokalnego, oraz czas przełączania/przesyłu flita t_r w ruterze. W typowych podejściach pomijających efekt zatorów w sieci [SC05] zakłada się,

że t_l = 1 cykl zegara, zaś t_r = 2 cykle. Zatem dla komunikatu składającego się z n flitów i przebywającego trasę złożoną z h ruterów opóźnienie transmisji t_M przełączanej metodą *wormhole* wynosi:

$$t_M = [(t_l + t_r) * h + t_l] + [(n-1) * t_l] \quad (5.2.2\text{-}7)$$

Pierwszy składnik sumy w nawiasach kwadratowych dotyczy flita nagłówkowego, zaś drugi – pozostałych flitów. Analogicznie składowa dynamiczna mocy pobieranej przez strukturę sieci NoC podczas przesyłania komunikatu (E_M) opisana jest równaniem [SC05]:

$$E_M = (E_r * h) + E_l * n * (h+1) \quad (5.2.2\text{-}8)$$

gdzie E_r to moc pobierana przez ruter podczas transmitowania flita (przełączenie flita nagłówkowego – zestawienie toru transmisji), zaś E_l – koszt energetyczny przesłania flitów przez łącza między portami sąsiadujących ruterów oraz z/do modułów NI.

Równanie (5.2.2-8) zapisane dla wszystkich przesyłanych wiadomości:

$$E_{M\,(NoC)} = \sum_{\forall M_{i,j} \in \boldsymbol{M}} E_{M_{i,j}} \quad (5.2.2\text{-}9)$$

uzupełnia koszt energetyczny określony równaniem (5.2.2-3) o składową dynamiczną pobieranej przez sieć jednoukładową mocy.

W rozprawie zakłada się syntezę systemów przesyłających duże ilości danych, co oznacza, że n w równaniach (5.2.2-7) i (5.2.2-8) jest rzędu kilkuset flitów. Ponadto dla projektowanych struktur sieci liczba hopów przypadających na komunikat rzadko przekracza 3. Stąd:

$$t_M = [(1 + 2) * 3 + 1] + [(n-1) * 1] = 10 + (n-1) \cong n \quad (5.2.2\text{-}10)$$

DEFINICJA 5.2.2-2 (OPÓŹNIENIE PODCZAS TRANSMISJI, CZAS TRANSMISJI)

Dla systemów NoC zorientowanych na przetwarzanie danych czas transmisji jest wyrażony jako $t_M = n$, gdzie „n" jest wielkością transmisji.

W celu oszacowania jakości uzyskanego rozwiązania oraz porównania z innymi metodami wprowadzono w rozprawie jeszcze jeden parametr – średnią ważoną liczbę skoków h_{AVG}. Wartość ta jest liczona w sposób uwzględniający fakt, że większe transmisje (mierzone we flitach) przebywające dłuższą trasę (podaną jako liczba ruterów/hopów) mają większy wkład do końcowego wyniku od transmisji krótkich i/lub przesyłanych krótszymi drogami.

$$h_{AVG} = \frac{\sum_{\forall M_{i,j} \in \boldsymbol{M}} |M_{i,j}| * h(M_{i,j})}{\sum_{\forall M_{i,j} \in \boldsymbol{M}} |M_{i,j}|} \quad (5.2.2\text{-}11)$$

gdzie $|M_{i,j}|$ to rozmiar komunikatu we flitach, zaś $h(M_{i,j})$ to trasa danej wiadomości liczona w hopach/ruterach.

Parametr wyrażający średnią liczbę skoków (hopów) w otrzymanej topologii jest wykorzystywany jako miara jej jakości [MMAAC06][MEKG08], gdyż oprócz informacji o szybkości transmisji (równanie (5.2.2-7)) pozwala oszacować efektywność układu pod kątem zużycia energii (równania (5.2.2-8) i (5.2.2-9)).

Założenia dotyczące miar efektywności otrzymanej architektury NoC są zbliżone do tych, jakie zaproponowano w pracy [MEKG08] i znajdują zastosowanie przy prezentacji wyników eksperymentalnych w dalszej części rozprawy.

Kryteria oceny efektywności topologii

Dla danego systemu wbudowanego realizowanego w architekturze sieci jednoukładowej z zastosowaniem tego samego modelu rutera oraz identycznym przyporządkowaniem zadań systemu do elementów przetwarzających topologia A jest efektywniejsza od topologii B, jeżeli:

a) pozwala wykonać zadania systemu w krótszym czasie (równanie (5.2.2-2)),
b) wykorzystuje rutery o mniejszej liczbie portów (równanie (5.2.2-4)),
c) jest zbudowana przy użyciu mniejszej liczby połączeń międzyruterowych (równanie (5.2.2-5)),
d) przesyła komunikaty krótszymi trasami (równania (5.2.2-8), (5.2.2-9) i (5.2.2-11)).

Kryteria b) i c) stanowią podstawę optymalizacji, gdzie celem jest minimalizacja kosztu układu. Warunki a) oraz d) (w dokładnych modelach systemów, uwzględniających opóźnienia transmisji wynikające z długości pokonywanej trasy) pozwalają na maksymalizację szybkości działania systemu. Natomiast wszystkie cztery kryteria, w stosunku zgodnym z wcześniej wyznaczonym wzorem na pobór mocy, pozwalają minimalizować zapotrzebowanie energetyczne systemu.

5.2.3 Graf Topologii Sieci

Topologiczny model sieci jednoukładowej, potrzebny do wyznaczenia tras dla komunikatów oraz do estymacji parametrów uzyskanej architektury NoC według kryteriów podanych w poprzednim podrozdziale jest opisany za pomocą Grafu Topologii Sieci (ang. *Network Topology Graph, NTG*). Podobna reprezentacja jest często spotykana w pracach związanych z syntezą sieci NoC pod różnymi nazwami, na przykład: *Communication Resource Graph* [MBSCW05], *Architecture Characterization Graph* [HM03a] czy też *NoC Topology Graph* [MM04].

DEFINICJA 5.2.3-1 (GRAF TOPOLOGII SIECI)

*Graf Topologii Sieci to nieważony graf skierowany NTG = (***N**, **L***), gdzie każdy węzeł* $n_i \in$ **N** *reprezentuje węzeł sieci NN, zaś każda krawędź* $l_i \in$ **L** *to jednokierunkowe łącze (magistrala) między dwoma węzłami (ruterami).*

Wagi na krawędziach nie są potrzebne, gdyż wszystkie łącza są jednakowe. Dozwolone jest natomiast występowanie cykli w w grafie NTG. Wynika to z faktu posiadania przez dwa sąsiadujące rutery (węzły NN) osobnych portów nadawczo-odbiorczych i możliwości połączenia ich w obu kierunkach - rutery mogą realizować transmisję typu *full-duplex*. Ograniczenie liczby portów dla połączeń międzyruterowych wyraża następująca zależność:

$$\forall\ n_i \in \mathbf{N},\ (d_{WE}(n_i) \leq 4)\ \text{ i }\ (d_{WY}(n_i) \leq 4) \qquad (5.2.3\text{-}1)$$

Definicja 5.2-1 w połączeniu z Definicją 5.2.3-1 determinuje $|\mathbf{N}| = |\mathbf{P}|$, gdzie **P** jest zbiorem wszystkich procesorów w mikrosieci NoC. Oznacza to, że liczba węzłów sieci musi być równa liczbie procesorów przydzielonych do realizacji zadań systemu wbudowanego (metodologia nie generuje sieci pośrednich opisanych w rozdziale 2.2). Każdy węzeł może być połączony z maksymalnie czterema innymi (w sumie 8 krawędzi), co opisuje poniższa formuła:

$$\forall\ n_i \in \mathbf{N},\ |Adj(n_i)| \leq 4 \qquad (5.2.3\text{-}2)$$

Adj w formule (5.2.3-2) oznacza listę sąsiedztwa danego węzła NTG. Warto zauważyć, że nie ma wymogu, by graf NTG był spójny. Innymi słowy nie wymaga się istnienia ścieżki między dwoma dowolnymi wierzchołkami grafu.

5.3 Atrybutowany Graf Zadań

Model aplikacji opisany Grafem Zadań nie obejmuje informacji o przyporządkowaniu zadań T_i do konkretnych procesorów $P_i \in \mathbf{P}$ oraz transmisji $M_{i,j} \in \mathbf{M}$ do kanałów komunikacyjnych. Skutkuje to brakiem znajomości parametrów czasowych dla zadań i komunikatów, co z kolei wyklucza możliwość ich szeregowania. Wybór odpowiednich elementów przetwarzających (procesorów) oraz kanałów komunikacyjnych do realizacji danego grafu TG nazywa się **alokacją**, natomiast przyporządkowanie zadań do konkretnych procesorów to **odwzorowanie** [D02]. Oba problemy należą do podstawowych zagadnień kosyntezy sprzętowo-programowej. Mimo, iż istnieją prace dotyczące syntezy sieci jednoukładowych rozpatrujące etap odwzorowania [LK03][CYC09] w rozprawie przyjęto rozwiązanie, w którym zarówno alokacja jak i odwzorowanie zostało uprzednio przeprowadzone metodami kosyntezy [D04][DG08], czego efektem jest Atrybutowany Graf Zadań (ang. *Attributed/Annotated Task Graph, ATG*). Metody te umożliwiają przydział do jednego procesora $P \in \mathbf{P}$ grupy zadań $T \in \mathbf{T}$ niepowiązanych zależnościami komunikacyjnymi. Jest to zupełnie odmienne podejście do sposobu partycjonowania grafu TG, rozpatrywanego w pracach z zakresu syntezy architektur NoC lub szeregowania zadań w systemach równoległych. Algorytmy zaprezentowane w pracy [DG08] przyjmują również minimalny, powiązany z ilością przesyłanych danych, czas transmisji między elementami przetwarzającymi, ze względu na konieczność ustalenia kolejności

wykonywania zadań i transmisji. Na tym etapie ignorowana jest możliwość wystąpienia zjawiska kolizji w przesyłanych wiadomościach. Założenie dotyczące istnienia niezależnych kanałów komunikacyjnych wypełnia się w rozprawie poprzez generowanie dedykowanej infrastruktury sieci jednoukładowej, służącej do przesyłania informacji. Niezależność kanałów jest osiągana dzięki wydzieleniu dla każdej transmisji takiego połączenia (trasy), które w danym momencie nie będzie wykorzystywane przez żadną inną transmisję w systemie. Uszeregowanie przetwarzania i komunikacji wykonywane przez algorytm kosyntezy zwane jest dalej w rozprawie „wstępnym", gdyż dopuszcza się możliwość dalszego przeszeregowania celem całkowitego wyeliminowania konfliktów transmisyjnych.

DEFINICJA 5.3-1 (ATRYBUTOWANY GRAF ZADAŃ)

*ATG to acykliczny graf skierowany ATG = (***T**, **M**, **P**, π, ω, τ, t_{START}, t_{CRIT}*) reprezentujący program równoległy, gdzie każdy wierzchołek* $T_i \in$ **T** *oznacza zadanie wykonywane na procesorze* $\pi(T_i)$*, natomiast krawędź grafu* $M_{i,j}$ *to transmisja między zadaniami* T_i *i* T_j*. Waga wierzchołka* $\omega(T_i) = |T_i|$ *opisuje czas wykonywania zadania* T_i*, natomiast wagi krawędzi* $\tau(M_{i,j}) = |M_{i,j}|$ *reprezentują czas przesłania komunikatu. Parametr* $t_{CRIT}(T_i)$ *ma to samo znaczenie, co w przypadku grafu zadań TG, stanowiąc limit czasowy na wykonanie danego zadania. Każdy wierzchołek* T_i *jest dodatkowo opisany informacjami o przydzielonym procesorze* $P \in$ **P** $(\pi(T_i))$ *oraz o czasie rozpoczęcia wykonywania zadania* $t_{START}(T_i)$*. Podobnie każda każda krawędź* $M_{i,j}$ *ma przypisany czas rozpoczęcia transmisji* $t_{START}(M_{i,j})$*.*

Czasy wymienione w Definicji 5.3-1 podane są w cyklach zegarowych, przy czym dla krawędzi $M_{i,j}$ informacja $\tau(M_{i,j})$ oznacza liczbę przesyłanych flitów (zgodnie z Definicją 5.2.2-2). Czas $\omega(T_i)$ wynika z doboru elementu przetwarzającego P i jest determinowany etapem alokacji i odwzorowania. Wartość tego parametru odnosi się do pesymistycznego wariantu przetwarzania zdania T_i na procesorze P (ang. *Worst Case Execution Time, WCET* [LT06]). Przykładowy graf ATG odpowiadający grafowi z Rys. 5.1-1 przedstawia Rys. 5.3-1.

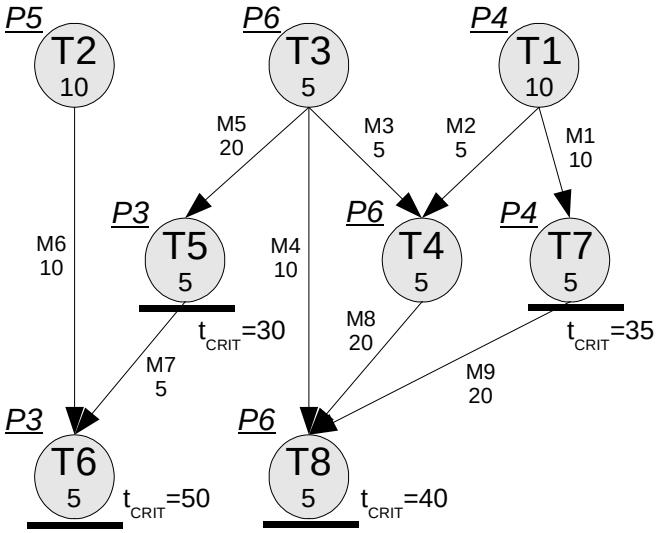
Czasy zakończenia dla zadań $t_{STOP}(T_i)$ i transmisji $t_{STOP}(M_{i,j})$ określają poniższe zależności:

$$t_{STOP}(T_i) = t_{START}(T_i) + \omega(T_i) \tag{5.3-1}$$

$$t_{STOP}(M_{i,j}) = t_{START}(M_{i,j}) + \tau(M_{i,j}) \tag{5.3-2}$$

Ponadto zgodnie z Definicją 5.2-1 punkt 4 (czas transmisji lokalnych):

$$\tau(M_{i,j}) = 0, \text{ dla } \pi(T_i) = \pi(T_j) \tag{5.3-3}$$



Rys. 5.3-1 Graf ATG jako graf TG odwzorowany na konkretną architekturę.

Wstępne uszeregowanie zadań i transmisji w grafie ATG, zakładające brak kolizji, jest pierwszym etapem metody opisywanej dalej w rozprawie. Zarówno ono jak i ewentualne dalsze przeszeregowania dokonywane w trakcie syntezy systemu NoC muszą podlegać regułom opisanym poniżej.

WARUNEK 5.3-1 (OGRANICZENIE PROCESORA – WZAJEMNE WYKLUCZANIE ZADAŃ)

Dla dwóch dowolnych zadań T_i, $T_j \in \mathbf{T}$, gdzie $\pi(T_i) = \pi(T_j)$ muszą zachodzić zależności:

$$[t_{START}(T_i) \geq t_{STOP}(T_j)] \text{ lub } [t_{START}(T_j) \geq t_{STOP}(T_i)] \quad (5.3\text{-}4)$$

Dzięki powyższej regule unika się sytuacji, kiedy dwa zadania przydzielone do tego samego elementu przetwarzającego wykonywane są w tym samym czasie.

Analogicznie do Definicji 5.1-2 - zadania/transmisje bezpośrednio poprzedzające w grafie ATG dane zadanie T_i oznaczamy ***predT***(T_i)/***predM***(T_i), natomiast następujące po nim – ***succT***(T_i)/***succM***(T_i). Dodatkowo dla każdego procesora tworzona jest, będąca jednym z wyników działania algorytmu kosyntezy [DG08], lista zawierająca wszystkie zadania przyporządkowane do danego elementu przetwarzającego. Kolejność zadań na liście wynika z przyjętej na etapie kosyntezy strategii szeregowania, przy czym każde zadanie ma przypisany czas rozpoczęcia i czas wykonania.

DEFINICJA 5.3-2 (USZEREGOWANIE ZADAŃ I TRANSMISJI)

Uszeregowanie σ dla zadań i transmisji określone jest jako funkcja odwzorowująca zbiór zadań i transmisji w przedziały czasu składające się z ciągów następujących po sobie cykli zegarowych, w trakcie których zadania i transmisje są wykonywane.

$$\sigma : (\mathbf{T} \cup \mathbf{M}) \rightarrow \mathbf{S}$$

W szczególności $\sigma(T_i) = \langle t_1, \ldots, t_n \rangle$, gdzie $t_1 = t_{START}(T_i)$, zaś $t_n = t_{STOP}(T_i)$, oznacza ciąg cykli zegarowych przypisany do danego zadania T_i. Liczba elementów $\sigma(T_i)$ wynosi $\omega(T_i) = |T_i|$. Analogicznie $\sigma(M_{i,j})$ oznacza ciąg cykli zegarowych, w trakcie których jest wykonywana transmisja $M_{i,j}$ (gdzie $t_1 = t_{START}(M_{i,j})$, zaś $t_n = t_{STOP}(M_{i,j})$). Liczba elementów $\sigma(M_{i,j})$ wynosi $\tau(M_{i,j}) = |M_{i,j}|$.

WARUNEK 5.3-2 (POPRAWNOŚĆ USZEREGOWANIA)

Uszeregowanie σ dla zadań i transmisji podlega następującemu ograniczeniu:

$$(t_{START}(T_i) + \omega(T_i)) \leq t_{CRIT}(T_i), \text{ gdzie } T_i \in \mathbf{T} \text{ oraz } t_{CRIT}(T_i) \neq \infty \quad (5.3\text{-}5)$$

Czas rozpoczęcia dowolnej transmisji zachodzącej po danym zadaniu musi spełniać poniższe zależności:

$$\forall\, M_{i,x} \in \mathbf{succM}(T_i),\ t_{START}(M_{i,x}) \geq t_{STOP}(T_i)$$

oraz

$$\forall\, M_{i,x}, M_{i,y} \in \mathbf{succM}(T_i),\ [t_{START}(M_{i,x}) \geq t_{STOP}(M_{i,y})] \text{ lub } [t_{START}(M_{i,y}) \geq t_{STOP}(M_{i,x})] \quad (5.3\text{-}6)$$

Uwzględniając Warunek 5.1-1 i 5.3-1 czas rozpoczęcia zadania podlega poniższej zależności:

$$t_{START}(T_i) \geq \max(\{t_{STOP}(T_j) : \pi(T_i) = \pi(T_j) \text{ oraz } t_{START}(T_j) < t_{START}(T_i)\},$$
$$\{t_{STOP}(M_{x,i}) : \forall\, M_{x,i} \in \mathbf{predM}(T_i)\}) \quad (5.3\text{-}7)$$

Nierówność (5.3-5), dotycząca funkcji szeregującej, zapewnia spełnienie wymagań czasowych dla tych zadań aplikacji, dla których takie wymagania zdefiniowano. Zależność (5.3-6) uzupełnia Warunek 5.1-2 założeniami punktu 4 Definicji 5.2-2. W szczególności określa, że każda transmisja z węzła grafu ATG zachodzi w następstwie zakończenia zadania generującego dane do wysyłki. Ponadto czas rozpoczęcia danej transmisji musi uwzględniać czas zakończenia komunikatu, którego wysyłka z tego samego procesora jeszcze trwa (rozpoczęła się wcześniej).

W związku z tym, że Warunek 5.1-1 wynika jedynie z własności Grafu Zadań i nie dotyczy żadnej konkretnej architektury, na którą ów graf miałby być odwzorowany, należy bardziej precyzyjnie zdefiniować czas rozpoczęcia zadań systemu. Formuła (5.3-7) uwzględnia dotychczas podane założenia dotyczące wykorzystywanej architektury sieci jednoukładowej, w tym możliwość sekwencyjnego wykonywania różnych zadań na tym samym module przetwarzającym PE. Zależność (5.3-7) nakłada na szeregowanie zadań ograniczenie wynikające z wzajemnego wykluczania zadań przydzielonych do tego samego procesora oraz

gwarantuje, że zadanie rozpocznie się dopiero po otrzymaniu wszystkich przeznaczonych dla niego komunikatów. Kolejność wykonywania zadań na danym procesorze wynika z pozycji na liście zbudowanej na etapie kosyntezy.

5.4 Kolizje

Każdy komunikat $M_{i,j}$ przesyłany w sieci jednoukładowej między różnymi procesorami P_k i P_l musi mieć przyporządkowany czas rozpoczęcia transmisji $t_{START}(M_{i,j})$ oraz trasę $R_{k,l}(M_{i,j})$. Kolizja, zwana zamiennie konfliktem, ma miejsce w sytuacji, gdy w tej samej chwili czasu t dwie lub więcej wiadomości ma zostać przesłanych tą samą trasą, rozumianą jako jedno lub więcej połączeń między ruterami.

DEFINICJA 5.4-1 (FUNKCJA RUTINGU I TRASA KOMUNIKATU)

Funkcję ρ przyporządkowującą transmisjom trasy nazywa się funkcją rutingu.

$$\rho : \mathbf{M} \rightarrow \mathbf{R}$$

Trasa $\rho(M_{i,j}) = R_{k,l}(M_{i,j}) = \langle l_1, \ldots, l_n \rangle$, gdzie $R_{k,l} \in \mathbf{R}$, jest określona jako ciąg połączeń międzyruterowych, gdzie pierwszy element l_1 oznacza łącze z portu wyjściowego rutera należącego do procesora P_k, natomiast ostatni – l_n – oznacza łącze do portu wejściowego rutera należącego do procesora P_l. Ponadto dla każdego l_i, gdzie (i=1, .., n-1), l_i jest dołączone do portu wejściowego, zaś l_{i+1} do portu wyjściowego tego samego rutera. Długość trasy, podawana jako liczba łącz (n), oznaczona jest $|R_{k,l}(M_{i,j})|$.

Trasa może składać się z wielu łącz opisanych w Definicji 5.4-1 – w ogólności $|R_{k,l}(M_{i,j})| = h - 1$, gdzie h to liczba ruterów na trasie (równanie (5.2.2-7)). Przynależność rutera do danego procesora wynika z Definicji 5.2-1. Dopuszczalna jest również sytuacja, gdy $R_{k,l}(M_{i,j}) \neq R_{k,l}(M_{m,n})$, co w praktyce oznacza, że pomiędzy nadawcą a odbiorcą może istnieć wiele ścieżek, w tym również dłuższe niż minimalne. Dzięki temu zwiększa się poziom elastyczności przy eliminowaniu kolizji poprzez wykorzystanie wszelkich istniejących tras. Ponadto redukcji ulega ryzyko nadmiernego generowania połączeń, co byłoby niekorzystne z zasobowo- i energooszczędnego punktu widzenia (m.in. składowa $E(l)$ w równaniu (5.2.2-1)).

DEFINICJA 5.4-2 (ZBIORY TRANSMISJI TEMPORALNIE KOLIZYJNYCH)

Dla dowolnych dwóch transmisji $M_{i,j}$ i $M_{k,l}$ należących do danego zbioru transmisji temporalnie kolizyjnych $\mathbf{C_x} \subseteq \mathbf{M}$ (x=1, 2, ..., n) spełniona jest zależność:

$$\forall\, M_{i,j}, M_{k,l} \in \mathbf{C_x}, \langle t_{START}(M_{i,j}); t_{STOP}(M_{i,j}) \rangle \cap \langle t_{START}(M_{k,l}); t_{STOP}(M_{k,l}) \rangle \neq \varnothing \qquad (5.4\text{-}1)$$

W grafie, w którym węzłami byłyby komunikaty ze zbiorów $\mathbf{C_x}$, natomiast każda krawędź reprezentowałyby relację nakładania się w czasie (kolizji) połączonych nią węzłów, wszystkie zbiory $\mathbf{C_x}$ utworzyłyby kliki.

DEFINICJA 5.4-3 (KOLIZJA TRAS KOMUNIKATÓW)

Dla dowolnych dwóch transmisji $M_{i,j}$ i $M_{k,l}$ należących do tego samego zbioru $\mathbf{C_x}$ zachodzi zjawisko kolizji tras, jeżeli spełniona jest zależność:

$$R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) \neq \varnothing \qquad (5.4\text{-}2)$$

W sytuacji, gdy zależność (5.4-2) zwróci zbiór niepusty, wynikiem będzie lista łącz, których dotyczy kolizja. Bazując na Definicjach 5.3-2 i 5.4-3 można określić warunki konieczne dla bezkolizyjności systemu o architekturze NoC.

DEFINICJA 5.4-4 (BEZKOLIZYJNA ARCHITEKTURA NOC)

Architektura sieci jednoukładowej utworzona dla danej aplikacji jest komunikacyjnie bezkolizyjna, jeśli dla dowolnych dwóch transmisji $M_{i,j}$, $M_{k,l} \in \mathbf{M}$ spełniona jest zależność:

$$R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) = \varnothing, \text{ gdy } M_{i,j}, M_{k,l} \in \mathbf{C_x} \text{ oraz } \pi(T_i) \neq \pi(T_k) \text{ i } \pi(T_j) \neq \pi(T_l) \qquad (5.4\text{-}3)$$

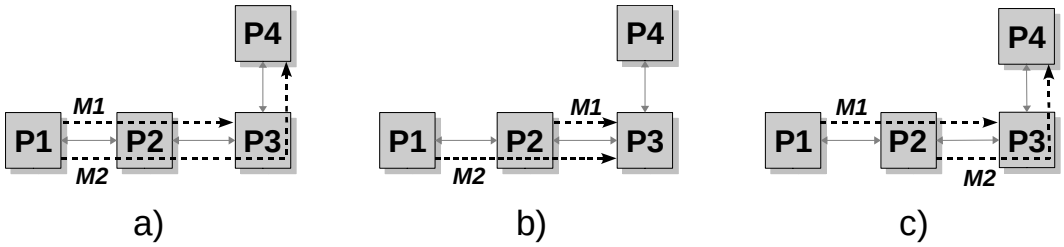
Z Definicji 5.4-4 wynika, że warunkiem wystąpienia konfliktu jest współdzielenie trasy, w całości lub we fragmencie, między zachodzącymi w tym samym czasie transmisjami. Zatem, jeżeli równanie $R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l})$ zwraca zbiór niepusty, architektura jest bezkolizyjna. Definicja wyklucza sytuacje, gdy dwie dowolne transmisje, $M_{i,j}$ i $M_{k,l}$, w tym samym czasie wychodziłyby lub zdążały z/do tego samego procesora. Ten rodzaj konfliktu jest niemożliwy do rozwiązania poprzez wygenerowanie dedykowanej topologii połączeń (architektura narzuca jeden port lokalny dla każdego rutera i jeden ruter obsługujący procesor). Gwarancją bezkolizyjności byłoby wtedy przeszeregowanie transmisji, czyli takie przypisanie czasów rozpoczęcia, by spełniony został warunek $\langle t_{START}(M_{i,j}); t_{STOP}(M_{i,j})\rangle \cap \langle t_{START}(M_{k,l}); t_{STOP}(M_{k,l})\rangle = \varnothing$.

W zależności od wyników zwróconych przez zależność (5.4-2) wyróżniamy trzy rodzaje kolizji [CM08]:

1. Na porcie źródłowym (nadawczym) – zachodzi w przypadku, gdy w (5.4-2) $a=c$ (ten sam procesor próbuje nadać równocześnie więcej niż jeden komunikat).
2. Na porcie docelowym (odbiorczym) – przypadek, gdy w (5.4-2) $b=d$ (więcej niż jedna transmisja jest kierowana do tego samego procesora).
3. Na łączu (na trasie) – wszystkie pozostałe przypadki kolizji.

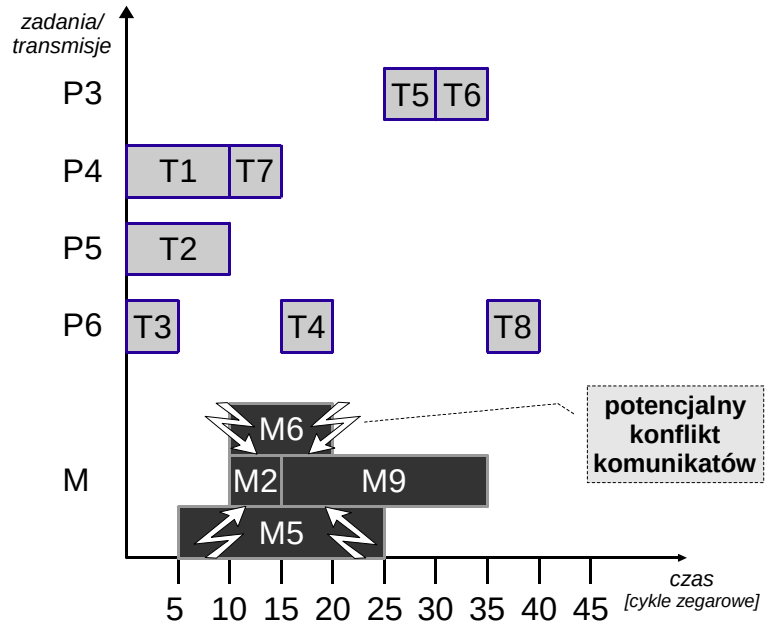
Pierwsze dwa typy kolizji eliminowane są w rozprawie poprzez przeszeregowanie transmisji, natomiast dla trzeciego przypadku tworzona jest topologia połączeń mająca na celu

dywersyfikację tras kolidujących komunikatów. W rozprawie pojęcia kolizji oraz konfliktu transmisji/komunikatów/wiadomości używane są zamiennie. Transmisje między zadaniami przypisanymi do tego samego procesora nie generują kolizji w sieci, gdyż mają zerowy koszt przesłania (równanie (5.3-3)). Schematycznie zjawisko kolizji przedstawiono na Rys. 5.4-1 (przy założeniu, że transmisje M1 i M2 nakładają się na siebie w czasie).



Rys. 5.4-1 Kolizja na porcie źródłowym P1 (a), porcie docelowym P3 (b) i na łączu P2-P3 (c).

Na Rys. 5.4-2 w postaci wykresu Gantta zilustrowano przykładowe wstępne uszeregowanie zadań i komunikatów dla grafu ATG z Rys. 5.3-1, spełniające wymagania dotyczące czasów rozpoczęcia zadań i transmisji. W części wykresu obrazującej uporządkowanie transmisji międzyprocesorowych zaznaczono miejsca potencjalnych kolizji w komunikatach, wynikające z ich wzajemnego nakładania się na siebie w czasie.



Rys. 5.4-2 Wykres Gantta z uszeregowanymi zadaniami oraz komunikatami z zaznaczeniem miejsc potencjalnych konfliktów transmisyjnych.

6 Metoda generowania bezkolizyjnych, dedykowanych sieci jednoukładowych

Bieżący rozdział zawiera formalną definicję i proponowane rozwiązanie problemu generowania dedykowanej dla danej aplikacji topologii sieci jednoukładowej. Podano również schemat metody służącej celowi określonemu tezą rozprawy wraz ze szczegółowym omówieniem jej etapów. Przedstawiony dalej sposób połączenia procesorów, realizujących zadania odwzorowywanej w architekturę NoC aplikacji, wspiera w pierwszym rzędzie zrównoleglenie transmisji oraz minimalizację kosztu energetycznego z zachowaniem zadanych ograniczeń czasowych. W związku z tym, że nie wszystkie konflikty transmisji mogą zostać wyeliminowane poprzez wyznaczenie niekolidujących tras w sieci, niektóre przypadki są rozwiązywane poprzez przeszeregowanie komunikacji. Zatem bezkolizyjność jest osiągana w ostatecznym rozwiązaniu dzięki wyznaczeniu odpowiednich tras dla wszystkich transmisji międzyprocesorowych przy wykorzystaniu minimalnej liczby połączeń międzyruterowych, a także poprzez zróżnicowane czasowo uszeregowanie komunikacji. W niniejszym rozdziale zamieszczono również definicje i opisy dodatkowych struktur danych wspomagających proces tworzenia dedykowanego systemu o architekturze NoC jak również przeanalizowano skuteczność metody.

6.1 Sformułowanie problemu

Sposób postępowania zaprezentowany dalej w rozprawie poszukuje rozwiązania dla dwóch problemów, zaliczanych w literaturze do klasy problemów NP-zupełnych: uszeregowania zadań i transmisji [SS01][HM05b][SS05] oraz odwzorowania aplikacji w topologię połączeń: predefiniowaną [MM04][ACP06] lub dowolną [PCSV03].

Bazując na definicjach i założeniach podanych w poprzednim rozdziale, problem syntezy bezkolizyjnej sieci jednoukładowej, dedykowanej dla danej aplikacji, jest formalnie zdefiniowany w sposób następujący:

Dla danej: aplikacji opisanej Atrybutowanym Grafem Zadań ATG

są szukane: topologia sieci opisana Grafem Topologii Sieci NTG oraz uszeregowanie S dla wszystkich zadań i transmisji z ATG

takie, że:

1. $\forall\ T_i \in \mathbf{T}$, jeśli $t_{CRIT}(T_i) \neq \infty$ to $t_{STOP}(T_i) \leq t_{CRIT}(T_i)$
2. $\forall\ M_{i,j} \in \mathbf{M}$, gdzie $\pi(T_i) \neq \pi(T_j)$, $\exists\ R_{a,b}(M_{i,j}) \neq \varnothing$
3. $\forall\ M_{i,j}, M_{k,l} \in \mathbf{C_x}$, $R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) = \varnothing$
4. $\forall\ P_i \in \mathbf{P}$, $\exists\ n_i(P_i) \in \mathbf{N}$

5. $\forall\ n_i \in \mathbf{N}$, $(d_{WE}(n_i) \leq 4)$ i $(d_{WY}(n_i) \leq 4)$ oraz $|Adj(n_i)| \leq 4$
6. $\forall\ T_i, T_j \in \mathbf{P}$, jeśli $\pi(T_i) = \pi(T_j)$ to $\langle t_{START}(T_i); t_{STOP}(T_i)\rangle \cap \langle t_{START}(T_j); t_{STOP}(T_j)\rangle = \varnothing$
7. $\forall\ M_{i,j}, M_{k,l} \in \mathbf{M}$, jeśli $\pi(T_i) = \pi(T_k)$ lub $\pi(T_j) = \pi(T_l)$ to
   $\langle t_{START}(M_{i,j}); t_{STOP}(M_{i,j})\rangle \cap \langle t_{START}(M_{k,l}); t_{STOP}(M_{k,l})\rangle = \varnothing$
8. E_{NoC} oraz $E_{M(NoC)}$ są jak najmniejsze

Pierwszy warunek z podanego powyżej zbioru gwarantuje takie uszeregowanie zadań systemu, by każde z nich wykonało się w zadanym czasie. Pośrednio implikuje to wykonanie wszystkich zadań w czasie nie przekraczającym globalnego ograniczenia systemu (t_{CRIT}), które stanowi maksimum z całego zbioru ograniczeń nałożonych na zadania. Jeżeli do zadania nie przypisano czasowego limitu wykonania przyjmuje się, że wynosi on nieskończoność i tym samym zwiększa elastyczność algorytmu szeregującego.

Warunki 2 i 3 zapewniają realizację każdej transmisji międzyprocesorowej poprzez przydzielenie jej trasy (listy połączeń między ruterami) oraz zapewniają brak kolizji na łączu dzięki wyznaczeniu niepokrywających się ścieżek dla komunikatów przesyłanych w tym samym czasie (należących do tego samego zbioru $\mathbf{C_x}$). Spełnienie warunku 4 jest niezbędne do prawidłowej budowy struktury sieci, tzn. do upewnienia się, że każdy procesor został w niej umiejscowiony w postaci pojedynczego węzła NN. Jeśli dane zadanie komunikuje się – nadaje i/lub odbiera – z innymi zadaniami, wówczas przydzielony do jego wykonania element PE będzie połączony z co najmniej jednym innym węzłem. Warunek 5 ogranicza liczbę dwukierunkowych portów służących do połączeń międzyruterowych (jedno- lub dwukierunkowych) - każdy węzeł może być połączony z co najwyżej czterema innymi.

Punkty 6 i 7 uzupełniają warunek 1 nałożony na strategię szeregowania zadań i transmisji. W punkcie 6 brane jest pod uwagę spełnienie Warunku 5.3-1 (równanie (5.3-4)), czyli zagwarantowanie wzajemnego wykluczania zadań przyporządkowanych do tego samego procesora. Punkt 7 jest niezbędny w celu wyeliminowania kolizji na portach źródłowych i docelowych. Mimo, że graf ATG jest po etapie kosyntezy wstępnie uszeregowany w sposób wykluczający taki rodzaj kolizji, przeszeregowanie transmisji może ponownie do nich doprowadzić. Ponadto zmiana czasów rozpoczęcia komunikatów może spowodować naruszenie zasady wzajemnego wykluczania zadań wykonywanych na tym samym elemencie przetwarzającym, dlatego taki krok musi uwzględniać obostrzenia wymienione w punkcie 6.

Ostatni warunek określa cel optymalizacji – w generowanej sieci jednoukładowej minimalizacji podlega koszt energetyczny. Osiąga się to poprzez narzucenie limitu czasowego na działanie systemu, wyznaczanie możliwie najkrótszych, bezkolizyjnych tras dla transmisji oraz

poprzez oszczędne gospodarowanie zasobami architektury wyrażające się minimalną liczbą niezbędnych do działania systemu - w ramach narzuconych ograniczeń - połączeń międzyruterowych (minimalizacja kosztu topologii). Redukcja zapotrzebowania energetycznego wynika z przedstawionych w podrozdziale 5.2.2 kryteriów oceny efektywności otrzymanej topologii NoC. W powyższym sformułowaniu problemu ograniczenia 3, 5 i 6 nałożone na rozwiązanie zwane są w literaturze [HM05b] **zgodnością zadań** (punkt 5) i **zgodnością transmisji** (punkty 3 i 6).

Funkcja rutingu ρ przyporządkowująca każdej transmisji M ścieżkę R działa w oparciu o istniejący w grafie NTG zbiór tras. Niezbędne jest zatem znalezienie takiej funkcji, która tworzyłaby dedykowaną topologię pod kątem obecności bezkolizyjnych tras.

DEFINICJA 6.1-1 (FUNKCJA GENEROWANIA TOPOLOGII)

Niech pojedynczy węzeł n_i grafu NTG będzie reprezentowany przez krotkę (P_i, l_i^1, l_i^2, l_i^3, l_i^4), gdzie P_i to element przetwarzający dołączony do portu LP rutera, natomiast l_i^1, l_i^2, l_i^3, l_i^4 to dwukierunkowe porty do połączeń międzyruterowych. Funkcja γ generująca topologię odwzorowuje każde $M_{i,j}$ w ciąg $R_{a,b} = \langle l_1, \ldots, l_n \rangle$ taki, że $l_1 = \langle l_a^x, l_o^p \rangle$, …, $l_n = \langle l_q^r, l_b^y \rangle$, gdzie P_a jest nadawcą, P_b odbiorcą transmisji $M_{i,j}$, zaś $l_i \in \mathbf{L}$.

γ: $\mathbf{M} \rightarrow \mathbf{L}$

Różnica pomiędzy funkcjami ρ (Definicja 5.4-1) i γ polega na tym, że ruting ρ wyznacza ścieżki w przygotowanej (całkowicie lub częściowo) topologii mikrosieci, natomiast zadaniem funkcji γ jest utworzenie grafu NTG poprzez połączenie portów l_x^y należących do różnych ruterów. W związku z ograniczeniami nałożonymi na budowę grafu topologii NoC, zdefiniowanymi w rozdziale 5.2.3, dla systemów złożonych z co najmniej sześciu jednostek PE może nie istnieć bezkolizyjna architektura NoC (w przypadkach grafów z co najwyżej pięcioma procesorami możliwe jest wygenerowanie mikrosieci z połączeniami typu każdy-z-każdym, czyli grafu pełnego, eliminującej wszelkie kolizje dedykowanymi łączami bezpośrednimi).

TWIERDZENIE 6.1-1 (ROZWIĄZYWALNOŚĆ PROBLEMU ODWZOROWANIA APLIKACJI W NOC)

Dla danej aplikacji ATG i uszeregowania **S** *istnieje bezkolizyjna sieć NoC opisana grafem NTG i spełniająca warunki z Definicji 5.4-4 wtedy i tylko wtedy, gdy istnieje funkcja γ taka, że:*

1) $\forall\ \mathbf{C_x}\ \forall R_{a,b}(M_{i,j})\ \forall R_{c,d}(M_{k,l})$, gdzie $M_{i,j}, M_{k,l} \in \mathbf{C_x}$ zachodzi $R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) = \varnothing$

oraz

2) $\forall\ l_i$: $l_i = \langle l_u^w, l_x^y \rangle$ odwzorowanie $\langle l_u^w, l_x^y \rangle \rightarrow l_i$ jest bijekcją.

DOWÓD TWIERDZENIA 6.1-1

„⇒”

Szukana funkcja γ musi spełniać łącznie oba warunki, a zatem załóżmy, że jeden z nich byłby niezrealizowany. Nie spełnienie pierwszego warunku oznacza, że w tym samym zbiorze $\mathbf{C_x}$ znalazłyby się dwie transmisje $M_{i,j}$ *i* $M_{k,l}$ dla których wyznaczone ścieżki pokrywałyby się na co najmniej jednym odcinku l_i (łączu międzyruterowym). Wówczas powyższe komunikaty rywalizowałyby o sporne połączenia w tym samym przedziale czasu powodując kolizję na trasie, co stoi w sprzeczności z Definicją 5.4-4. Niespełnienie drugiego warunku oznacza naruszenie ograniczeń nałożonych na budowę grafu NTG wyrażonych równaniami (5.2.3-1) i (5.2.3-2). W szczególności, jeżeli odwzorowanie $\langle l_u^w , l_x^y \rangle \rightarrow l_i$ nie byłoby wzajemnie jednoznaczne, wówczas wystąpiłaby jedna z poniższych sytuacji:

- z danego portu (przyłącza) l_u^w lub l_x^y wychodziłaby więcej niż jedna krawędź l_i skutkując niemożnością wyznaczenia trasy zgodnie z Definicją 5.4-1 (niejednoznaczność wyznaczenia danego odcinka trasy),
- krawędź l_i łączyłaby więcej niż dwa przyłącza l_u^w i l_x^y (hiperkrawędź) – graf NTG stałby się hipergrafem (sprzeczność z Definicją 5.2.3-1).

„⇐”

Załóżmy, że funkcja γ spełnia wyżej wymienione warunki, natomiast sieć NoC nie spełnia założeń twierdzenia. W przypadku, gdy topologia nie jest bezkolizyjna, wtedy zgodnie z Definicją 5.4-4 muszą istnieć $M_{i,j}$, $M_{k,l} \in \mathbf{C_x}$ takie, że $R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) \neq \varnothing$ – wówczas nie będzie spełniony pierwszy warunek. Niech NTG nie spełnia warunków zdefiniowanych w rozdziale 5.2.3, wtedy musi istnieć węzeł n_x połączony z co najmniej 5 innymi węzłami. Ponieważ każdy węzeł ma tylko cztery porty, musiałby istnieć port l_x^a dołączony do dwóch innych węzłów n_y i n_z poprzez krawędzie: $l_i = \langle l_x^a, l_y^b \rangle$ i $l_j = \langle l_x^a, l_z^c \rangle$. A zatem odwzorowanie portów w krawędzie nie będzie wzajemnie jednoznaczne, nawet jeśli $l_i = l_j$. ¤

Złożoność problemu polegającego na znalezieniu funkcji jest zatem złożeniem dwóch problemów:

- szukania rozłącznych tras w grafie NTG (problem NP-zupełny [GJ79]),
- odwzorowania l_x^y w l_i, gdzie przestrzeń rozwiązań jest określona przez liczbę możliwych grafów o maksymalnym stopniu węzła równym 4.

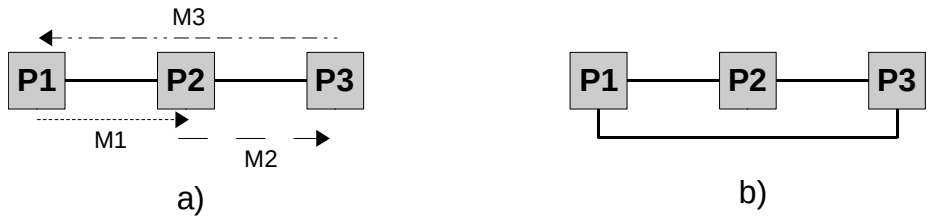
Dla grafu o *n* węzłach, gdzie każdy ma 4 wyjścia, które można połączyć z 4 wejściami innego węzła/węzłów, połączeń można dokonać na (*4*n*)! sposobów. Mimo wyeliminowania

przypadków, gdzie niektóre krawędzie są incydentne z jednym i tym samym węzłem nadal można zaobserwować, że poszukiwanie w takiej przestrzeni rozwiązań grafu o najmniejszym koszcie zasobowo-energetycznym może w najgorszym razie prowadzić do konieczności sprawdzenia wszystkich możliwych grafów, będących wynikiem permutacji połączeń międzywęzłowych. Oznacza to, że złożoność problemu odwzorowania l_x^y w l_i jest rzędu $O(n!)$.

Z powyższego wynika, iż kwestia znalezienia funkcji γ cechuje wysoki stopień złożoności, a zatem dla jej rozwiązania celowe jest szukanie metody heurystycznej.

6.2 Optymalizacja kosztu topologii NoC

Kwestie syntezy NoC komplikuje kryterium – tworzone przez dwie składowe: minimalizację liczby połączeń (wpływ na statyczną składową pobieranej mocy) i długości tras (składowa dynamiczna zużywanej energii) - według którego poszukiwane jest rozwiązanie optymalne. Sprzeczność obu celów minimalizacji ukazuje prosty przykład z Rys. 6.2-1. Jeżeli pożądana byłaby synteza sieci NoC o minimalnej liczbie połączeń dla komunikującej się grupy procesorów, wystarczyłoby zbudować topologię dowolnego drzewa wolnego (Rys. 6.2-1a). Dołożenie wymogu istnienia najkrótszych ścieżek (np. o długości równej 1 hop) dla wszystkich transmisji już sprawę utrudnia – niezbędne może okazać się wygenerowanie kolejnych połączeń prowadzące do systemu w pełni połączonego (o ile pozwoli na to wybrana do implementacji architektura NoC – Rys. 6.2-1b).



Rys. 6.2-1 Cele optymalizacji topologii NoC: liczba łącz (a), długość tras (b).

W rozdziale 5.2.2 zdefiniowano koszt energetyczny architektury NoC. Z równania (5.2.2-6) wynika, iż elementami mikrosieci, które wpływają na statyczną składową pobieranej mocy są łącza międzyruterowe oraz związane z nimi porty ruterów. W związku z tym, że elementy przetwarzające PE alokowane są przez kosyntezę niezależnie od architektury docelowej, koszt topologii NoC określa liczba połączeń między ruterami rzutująca jednocześnie na liczbę wymaganych portów w poszczególnych elementach przełączających. Ze względu na ograniczenia dotyczące konstrukcji rutera (rozdział 5.2.1), w dalszych rozważaniach dotyczących szukania rozwiązania brane są pod uwagę grafy topologii NTG o stopniu nie większym niż 4 oraz takie aplikacje ATG, gdzie nie ma żadnych kolizji na portach.

TWIERDZENIE 6.2-1 (GRAF NTG O MINIMALNYM KOSZCIE TOPOLOGII)

Dowolny graf NTG, w którym $\forall l_i \in \mathbf{L}$ *istnieje transmisja* $M_{i,j}$ *taka, że w grafie* $NTG' = NTG - \{l_i\}$ *nie istnieje żadna ścieżka* $R_{a,b}(M_{i,j})$ *jest grafem o minimalnym koszcie topologii* NTG_{min}.

DOWÓD TWIERDZENIA 6.2-1

Niech graf będzie budowany według następującej metody:

$\forall M_{i,j} \in \mathbf{M}$ wykonaj:

a) jeśli dla $M_{i,j}$ $\exists R_{a,b}(M_{i,j}) \neq \emptyset$ – nic nie rób,

b) jeśli dla $M_{i,j}$ nie istnieje $R_{a,b}(M_{i,j}) \neq \emptyset$ to: NTG $\leftarrow$ NTG $\cup$ $\{l_m\}$, gdzie $l_m = \langle l_u^x, l_v^y \rangle$: $R_{a,u} \subset$ NTG i $R_{b,v} \subset$ NTG

Z powyższych zasad wynika, iż krawędź w NTG jest dodawana tylko wtedy, gdy dla danej transmisji nie istnieje żadna ścieżka w istniejącej topologii. To gwarantuje co najwyżej jedną ścieżkę pomiędzy dowolną parą węzłów. Zatem usunięcie dowolnej krawędzi spowodowałoby wyeliminowanie ścieżki dla co najmniej jednej transmisji. Stąd wynika, że tak tworzony graf charakteryzuje minimalny koszt topologii wyrażony liczbą krawędzi (łącz). ¤

Z przedstawionej powyżej zasady generowania NTG_{min} wynika, iż topologia będzie miała postać lasu. Dokonując statycznej analizy ATG można wyznaczyć wszystkie transmisje międzyprocesorowe, a następnie na ich podstawie utworzyć NTG_{min} według zasad podanych w dowodzie Twierdzenia 6.2-1. Tak utworzona topologia nie bierze jednak pod uwagę kolizji komunikatów. W związku z tym, że w NTG_{min} pomiędzy dowolną parą węzłów istnieje dokładnie jedna ścieżka, w celu usunięcia kolizji należy zdywersyfikować trasy komunikatów poprzez rozbudowę topologii (dodatkowe dedykowane połączenia).

TWIERDZENIE 6.2-2 (BEZKOLIZYJNY GRAF NTG O MINIMALNYM KOSZCIE TOPOLOGII)

Bezkolizyjny graf NTG ma minimalny koszt topologii ($NTG_{CollFree\text{-}min}$) *wtedy i tylko wtedy, gdy istnieje taki* $\mathbf{E_x} \subset \mathbf{E}$, *gdzie* **E** *to zbiór wszystkich krawędzi* $NTG_{CollFree\text{-}min}$, *że* $NTG_{CollFree\text{-}min} - \mathbf{E_x} = NTG_{min}$ *oraz* $|\mathbf{E_x}|$ *jest najmniejsze z możliwych.*

DOWÓD TWIERDZENIA 6.2-2

Niech $NTG_{CollFree}$ będzie bezkolizyjnym grafem NTG. Dla każdej spójnej składowej tego grafu można wyznaczyć drzewo rozpinające [CLR98]. Drzewa te utworzą las, w którym pomiędzy dowolnymi węzłami jest co najwyżej jedna ścieżka. Zatem las ten tworzy NTG_{min} takie, że $NTG_{min} \subseteq NTG_{CollFree}$. Niech zbiór krawędzi wykluczonych przez algorytm szukania

drzew rozpinających to zbiór $\mathbf{E_x}$. Z własności zbiorów wynika, że jeśli $\mathbf{A} = \mathbf{B} \cup \mathbf{C}$, gdzie $\mathbf{B} \cap \mathbf{C} = \varnothing$ to $\mathbf{A_{min}} = \mathbf{B_{min}} \cup \mathbf{C_{min}}$, zatem jeśli $NTG_{CollFree} = NTG_{min} \cup \mathbf{E_x}$ to $NTG_{CollFree\text{-}min} = NTG_{min} \cup \mathbf{E_{x\,min}}$.

Z powyższych faktów wynika, że każdy bezkolizyjny graf $NTG_{CollFree}$ da się przedstawić jako sumę pewnego grafu NTG_{min} oraz zbioru krawędzi $\mathbf{E_x}$, zatem dla $NTG_{CollFree\text{-}min}$, $|\mathbf{E_x}|$ będzie najmniejsze z możliwych. ¤

Analizując graf ATG można wyznaczyć nie tylko zbiór transmisji międzyprocesorowych, służący do utworzenia grafu NTG_{min}, ale również zbiory $\mathbf{C_x}$ (Definicja 5.4-2). Potrzebne do tego informacje o komunikatach (czas rozpoczęcia oraz czas trwania) zawarte są w ATG. Przeszukując zbiory $\mathbf{C_x}$ można określić, które z komunikatów kolidują na ścieżce (Definicja 5.4-3) w grafie NTG i rozbudować go poprzez wygenerowanie dodatkowych krawędzi (zbiór $\mathbf{E_x}$).

Schemat przykładowego algorytmu służącego do budowy $NTG_{CollFree\text{-}min}$ wygląda następująco:

1. Wyznacz wszystkie transmisje międzyprocesorowe $M_{i,j}$ oraz zbiory $\mathbf{C_x}$.
2. Wygeneruj NTG_{min}.
3. Dla każdego $\mathbf{C_x}$ wyznacz wszystkie możliwe zbiory krawędzi eliminujących kolizje komunikatów należących do $\mathbf{C_x}$.
4. Dla sumy zbiorów krawędzi wyznaczyć minimalny zbiór eliminujący wszystkie kolizje dla wszystkich $\mathbf{C_x}$ – będzie to $\mathbf{E_{x\,min}}$.
5. Wykonaj $NTG_{min} \cup \mathbf{E_{x\,min}}$ – rezultatem będzie $NTG_{CollFree\text{-}min}$.
6. Powtórz kroki 2-4 dla wszystkich możliwych NTG_{min}.
7. Z otrzymanych grafów $NTG_{CollFree\text{-}min}$ wybierz ten, dla którego $|\mathbf{L}|$ jest najmniejsze.

Powyższy sposób postępowania gwarantuje znalezienie – o ile istnieje – najlepszego rozwiązania. Wadą algorytmu jest wysoka złożoność obliczeniowa wynikająca z konieczności przeszukania dużej przestrzeni rozwiązań. Ponadto istotny dla metody krok 4 to NP-zupełny problem minimalnego pokrycia macierzy, w której wiersze odpowiadają kolizjom, a kolumny zbiorom krawędzi [CLR98].

Minimalizacja kosztu energetycznego obejmuje również składową dynamiczną pobieranej mocy E_M, związaną z długością tras komunikatów (równania (5.2.2-8) i (5.2.2-9)). Koszt energetyczny najprościej wyrazić jako sumę długości ścieżek dla wszystkich transmisji, czyli $\sum|\mathbf{R_i}|$.

Do znalezienia grafu NTG_{min} o minimalnym koszcie energetycznym E_M wystarczy lista transmisji międzyprocesorowych uzyskana poprzez analizę ATG. Na podstawie tej listy można utworzyć graf G_{min}, w którym węzły odpowiadają procesorom, a krawędzie transmisjom.

Następnie każdą spójną składową grafu G_{min} można przekształcić w graf pełny. W ten sposób można otrzymać graf G_{max} charakteryzujący się tym że $\forall\ NTG_{min} \subset G_{max}$. Zatem znalezienie drzewa rozpinającego grafu G_{max}, dla którego $\sum|\mathbf{R_i}|$ jest najmniejsze, prowadzi do znalezienia NTG_{min} o minimalnym koszcie energetycznym.

Analogicznie $NTG_{CollFree\text{-}min}$ o minimalnym koszcie energetycznym E_M można wyznaczyć poprzez znalezienie $NTG_{CollFree\text{-}min}$ dla którego $\sum|\mathbf{R_i}|$ jest najmniejsze z możliwych. Algorytm ten wygląda następująco:

1. Wyznacz wszystkie możliwe grafy NTGCollFree-min.
2. Dla każdego $NTG_{CollFree\text{-}min}$ policz wartość $\sum|R_i|$.
3. Wybierz $NTG_{CollFree\text{-}min}$ z najmniejszą wartością $\sum|R_i|$.

Graf wybrany w kroku 3 reprezentuje architekturę NoC wygenerowaną dla danej aplikacji ATG, zminimalizowaną pod kątem kosztu (liczby łącz międzyruterowych), w której koszt energetyczny jest najmniejszy z możliwych. Algorytm wymaga informacji o transmisjach oraz kolizjach, które można otrzymać na podstawie statycznej analizy grafu ATG (odpowiednie algorytmy zostaną przedstawione w rozdziałach 6.4 i 6.5).

Wobec dużej złożoności obliczeniowej algorytmu szukania $NTG_{CollFree\text{-}min}$ o minimalnym koszcie energetycznym, w rozprawie zaproponowano metodę heurystyczną, którą charakteryzuje mała złożoność obliczeniowa (duża szybkość działania) i jednoczesne minimalizowanie długości tras komunikatów kosztem braku gwarancji co do uzyskania minimalnej topologii (topologia zrównoważona pod względem kosztu i poboru energii).

6.3 Schemat metody

Jak zaprezentowano w rozdziale 2.4 istnieje wiele metod, według których przy tym samym celu optymalizacji (koszt, pobór mocy) może być budowany system NoC. Liczba sposobów, na które można połączyć ze sobą *n* procesorów zależy od tego, czy docelowa sieć jednoukładowa ma być regularna czy nie, czy dozwolone są łącza jednokierunkowe między modułami PE, czy zastosowane rutery mogą mieć więcej niż jeden port lokalny i więcej niż cztery porty do połączeń międzyprocesorowych, czy para ruterów może być połączona więcej niż jednym łączem (zaangażowanie więcej niż jednego portu w każdym ruterze), itd..

Dla rozwiązania problemu odwzorowania aplikacji w topologię NoC spełniającą warunki postawione w rozdziale 6.1 zaproponowano wieloetapową metodę (Rys. 6.3-3). Poszczególne czynności wykonywane w jej ramach są następujące:

1. Efektem kosyntezy, wykonanej metodami opisanymi w pracy [DG08] jest graf ATG z przyporządkowanymi procesorami do poszczególnych zadań. Jak wspomniano wcześniej użyty algorytm kosyntezy, oprócz alokacji i odwzorowania, dokonuje również

uszeregowania zadań i transmisji, podczas którego zakłada minimalny czas komunikacji wynikający z ilości przesyłanych danych. Kosynteza szereguje transmisje w sposób wykluczający kolizje na portach źródłowych i docelowych. Konflikty na trasie są niemożliwe do wyeliminowania w tej fazie metody, gdyż nie ma jeszcze informacji o topologii połączeń procesorów, a tym samym nie można określić ścieżek dla poszczególnych transmisji. Kolejność wykonywania zadań i transmisji w grafie ATG podlega ograniczeniom Warunku 5.3-1 (zgodność zadań) oraz Warunku 5.3-2. Jednocześnie system zachowuje zdolność do wykonania wszystkich zadań w założonych czasach t_{CRIT}.

2. Pierwszą czynnością jest utworzenie Listy Transmisji Międzyprocesorowych (ang. *Interprocessor Transmission List, ITL*) służącej wyodrębnieniu komunikacji zachodzącej między parami procesorów. Tylko taka komunikacja wiąże się z kosztem przesłania w sieci i może powodować konflikty. Lista ITL jest szczegółowo opisana w podrozdziale 6.4. Węzły listy ITL to transmisje międzyprocesorowe z przypisanymi czasami rozpoczęcia, z wagami określającymi wolumen transmisji wyrażony we flitach (odzwierciedlający czas potrzebny na przesłanie danego komunikatu), z parami komunikujących się procesorów oraz z trasami, jakie każdy komunikat pokonuje w wygenerowanej topologii sieci (na wstępie każda transmisja ma przypisaną pustą trasę).
3. Na podstawie listy ITL tworzona jest Lista Kolizji Temporalnych (ang. *Time Collision List, TCL*), na której każda pozycja składa się z pary nakładających się w czasie transmisji wraz z opisem źródła i przeznaczenia dla każdej z nich oraz parametrami czasowymi (czasy startu i trwania komunikatów oraz czas kolizji).
4. W przypadku, gdy dokonano przeszeregowania zasadne jest przeszukiwanie listy TCL pod kątem wystąpień kolizji na porcie nadawczym lub odbiorczym – takie kolizje nie mogą być usunięte poprzez wyznaczenie alternatywnych tras, gdyż wynikają z ograniczeń modułów NI (Definicja 5.2-2, punkt 4). Jedynym rozwiązaniem jest przeszeregowanie komunikatów, polegające na opóźnieniu jednego z nich. Wymuszone przeszeregowanie często wiąże się ze zmianą czasów w grafie ATG, gdyż zmiana czasu rozpoczęcia komunikatu może pociągnąć za sobą zmianę czasów rozpoczęcia zadań i transmisji znajdujących się na ścieżce zstępującej z danej krawędzi. Graf ATG jest zatem aktualizowany i następuje powrót do pkt. 2 niniejszej metody, ponownie tworzącej listę ITL (przeliczenie parametrów czasowych) oraz listę TCL.
5. Jeżeli przeszeregowanie dokonane w pkt. 4 doprowadziło do naruszenia Warunku 5.3-2

(czas działania przynajmniej jednego z zadań systemu przekroczył ograniczenie t_{CRIT}), przeszeregowuje się drugą z kolidujących w danej parze transmisję. Jeżeli taka próba również zakończy się niepowodzeniem następuje powrót do etapu kosyntezy. Jednocześnie przekazywana jest informacja o zadaniu-nadawcy drugiego z kolidujących komunikatów wraz z czasem o jaki przekroczono ograniczenia czasowe w ATG. Zadanie to oraz zadania poprzedzające je w grafie TG podczas ponownej kosyntezy będą miały ustanowione bardziej rygorystyczne ograniczenie czasowe t_{CRIT} (skrócone o przekazany czas).

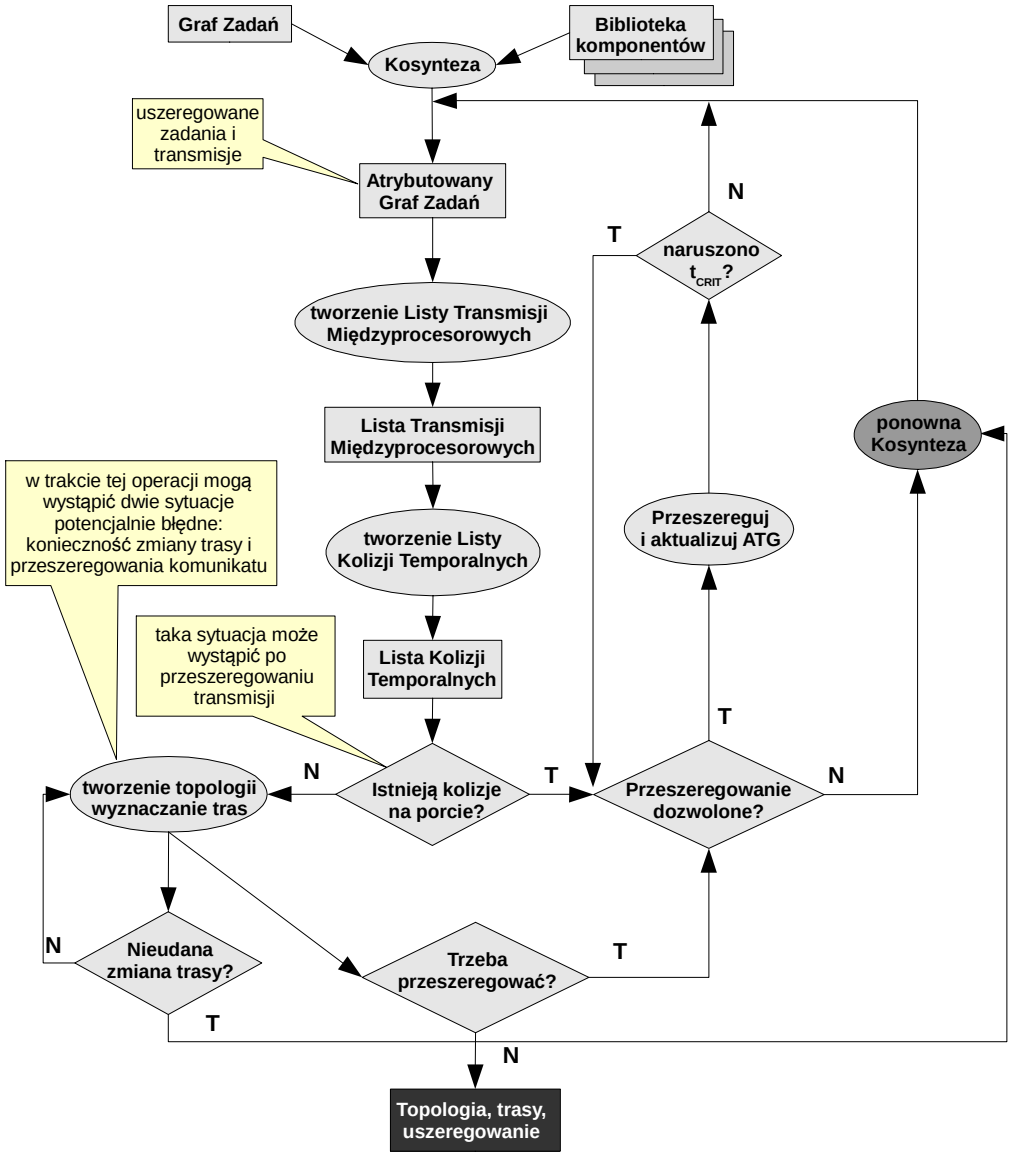
6. Po usunięciu kolizji na portach następuje etap generowania topologii połączeń dla pozostałych kolizji. Założenia dotyczące czynności na tym etapie są następujące:
   a) jeżeli dla danych transmisji nakładających się w czasie już przypisano trasy (usunięto inne kolizje, w których brały udział rozpatrywane komunikaty) dokonywane jest sprawdzenie czy nie są to kolidujące trasy - jeśli tak, podejmowana jest próba modyfikacji (często – pogorszenia, wydłużenia) trasy dla krótszej z transmisji, a w przypadku niepowodzenia – dla dłuższej; zmiana wyznaczonych tras wymaga sprawdzenia bezkolizyjności uprzednio usuniętych konfliktów; niepowodzenie ustanowienia tras w istniejącej topologii prowadzi do próby utworzenia bezpośredniego połączenia pomiędzy nadawcą i odbiorcą dłuższego komunikatu, a jeśli architektura na to nie pozwala (określają to zależności (5.2.3-1) i (5.2.3-2)) próba dodania łącza bezpośredniego wykonywana jest dla krótszej z kolidujących wiadomości; niepowodzenie wszystkich powyższych kroków skutkuje ponowną kosyntezą - jednocześnie przekazywana jest informacja o zadaniu-nadawcy drugiego (dłuższego) z komunikatów, dla którego nie udało się zmienić trasy, kolejna kosynteza wykonywana jest według zasad opisanych w pkt. 5 niniejszego opisu (zamiast czasu o jaki przekroczono t_{CRIT} przekazuje się czas trwania danej kolizji potrzebny algorytmowi szeregującemu),
   b) jeżeli dla danej transmisji nie wyznaczono dotąd trasy wówczas sprawdza się, czy w już utworzonej topologii istnieje najkrótsza droga dla komunikatu; to samo dotyczy drugiej kolidującej transmisji; w przypadku kolizji tras krótszy komunikat będzie miał wyznaczoną inną niż najkrótsza drogę,
   c) niepowodzenie poprzedniego kroku skutkuje sprawdzeniem zależności (5.2.3-1) i (5.2.3-2) – jeden ruter może mieć nie więcej niż 4 łącza wejściowe i 4 wyjściowe biegnące do maksymalnie 4 innych ruterów; spełnienie powyższych warunków oznacza rozbudowę topologii, czyli wygenerowanie kolejnego, dedykowanego

połączenia międzyruterowego dla pary nadawca-odbiorca,

d) jeżeli poprzednie kroki okazały się niewystarczające dla danej pozycji na liście TCL (nie da się rozwiązać danego konfliktu za pomocą dedykowanej topologii) podejmowana jest próba przeszeregowania komunikatów opisana w pkt. 4 i 5.

7. Po przetworzeniu wszystkich pozycji z listy TCL sprawdza się, czy lista ITL zawiera węzły nieuwzględnione w TCL, czyli transmisje niekolidujące z żadnymi innymi. Jeśli tak, wyznacza się najkrótszą drogę dla każdej brakującej transmisji (może to również oznaczać konieczność wstawienia dodatkowego połączenia).

Efektem końcowym algorytmu jest graf topologii NTG, trasy dla wszystkich transmisji oraz ostateczne uszeregowanie zadań i transmisji w grafie ATG.



Rys. 6.3-1 Sieć działań metodologii generującej bezkolizyjne i dedykowane sieci jednoukładowe.

6.4 Wyodrębnienie transmisji międzyprocesorowych

Do budowy dedykowanej sieci połączeń niezbędna jest wiedza o komunikatach

przesyłanych między różnymi procesorami, w tym o takich, które kolidują w czasie. Stąd pierwszym krokiem jest analiza grafu prowadząca do utworzenia Listy Transmisji Międzyprocesorowych ITL.

DEFINICJA 6.4-1 (LISTA TRANSMISJI MIĘDZYPROCESOROWYCH)

Lista Transmisji Międzyprocesorowych to lista jednokierunkowa, w której każdy węzeł $n_{ITL} = (M_{i,j}, t_{START}(M_{i,j}), |M_{i,j}|, P_{NAD}, P_{ODB}, R_{k,l})$ *odpowiada transmisji* $M_{i,j} \in \mathbf{M}$*, spełniającej warunek* $\pi(T_i) \neq \pi(T_j)$*. Pozycje na liście są uporządkowane niemalejąco według czasów rozpoczęcia* $t_{START}(M_{i,j})$ *i zawierają informacje o wielkości komunikatu* $|M_{i,j}|$*, o komunikujących się procesorach: nadawczym* $P_{NAD} = \pi(T_i)$ *i odbiorczym* $P_{ODB} = \pi(T_j)$ *oraz o wyznaczonej trasie* $R_{k,l}(M_{i,j})$*.*

Szkic algorytmu tworzącego listę ITL przedstawiono w ramce Algorytm 1.

Algorytm 1 *– Generowanie listy ITL*

Wejście: graf ATG

Wyjście: lista ITL

```
0:    ITL := NULL;
1:    dla każdej Mi,j w ATG wykonaj
2:          jeśli P(Ti) <> P(Tj) to
3:                WstawSort(ITL, NodeITL(Mi,j,tSTART(Mi,j),|Mi,j|,PNAD,PODB,NULL))
```

Schemat działania Algorytmu 1 zakłada kolejne przetworzenie wszystkich krawędzi z grafu ATG (linia 1). Jeżeli stwierdzono obecność komunikacji międzyprocesorowej (linia 2) to aktualizowana jest lista ITL. Procedura *WstawSort* z linii 3 ma na celu uzupełnienie struktury ITL o nowy element, przy miejsce jego wstawienia wyznaczane jest na podstawie wartości $t_{START}(M_{i,j})$ (porządek sortowania listy - niemalejący). Warto zauważyć, że trasy przypisane do pozycji na liście ITL są wstępnie puste (*NULL*) – będą uzupełniane podczas budowania Grafu Topologii Sieci.

6.5 Lista kolizji i przeszeregowanie transmisji

Po wyodrębnieniu komunikacji międzyprocesorowej można przystąpić do sporządzenia wykazu wiadomości nakładających się na siebie w czasie. Jest to bardzo ważny etap całej metody, gdyż w jego wyniku otrzymuje się strukturę listową służącą do ustalenia kolidujących ze sobą wiadomości przesyłanych między procesorami. Strukturę tą nazwano Listą Kolizji Temporalnych (ang. *Time Collision List*, *TCL*).

DEFINICJA 6.4-1 (LISTA KOLIZJI TEMPORALNYCH)

Lista Kolizji Temporalnych to lista jednokierunkowa, na której każda pozycja

K = (n_{ITL}^i, n_{ITL}^j, t_{COLL}, przeszeregowano, jest_trasa) zawiera następujące informacje:

1. *Indeksy ITL kolidujących transmisji: n_{ITL}^i, n_{ITL}^j.*
2. *Czas trwania kolizji t_{COLL}.*
3. *Znacznik informujący o podjętej próbie przeszeregowania transmisji - `przeszeregowano`.*
4. *Znacznik informujący o udanym rozwiązaniu konfliktu poprzez dedykowaną topologię (wyznaczone niekolidujące trasy) – `jest_trasa`.*

Informacja o podjętej próbie przeszeregowania transmisji ma zapobiec zapętleniu algorytmu na skutek nieudanych przeszeregowań, tj. takich, które nie spełniają Warunku 5.3-2. Samo przeszeregowanie zmienia jedynie czasy rozpoczęcia zadań i transmisji i nie ingeruje w już zbudowaną topologię połączeń, natomiast może zmienić zawartość Listy Kolizji Temporalnych. Zatem znacznik odnotowujący fakt rozwiązania konfliktu poprzez wygenerowanie dedykowanych tras zapobiega powtórnemu szukaniu ścieżek dla komunikatów w grafie NTG. Sytuacja taka mogłaby mieć miejsce, gdyż szukanie tras jest zachłannym algorytmem konstrukcyjnym z powrotami (rozdział 6.6). Algorytm generujący listę TCL zaprezentowano w ramce Algorytm 2.

***Algorytm 2** – Budowanie listy TCL*

Wejście: Lista Transmisji Międzyprocesorowych ITL, poprzednia wersja listy TCL (jeśli istnieje)

Wyjście: Lista Kolizji Temporalnych TCL

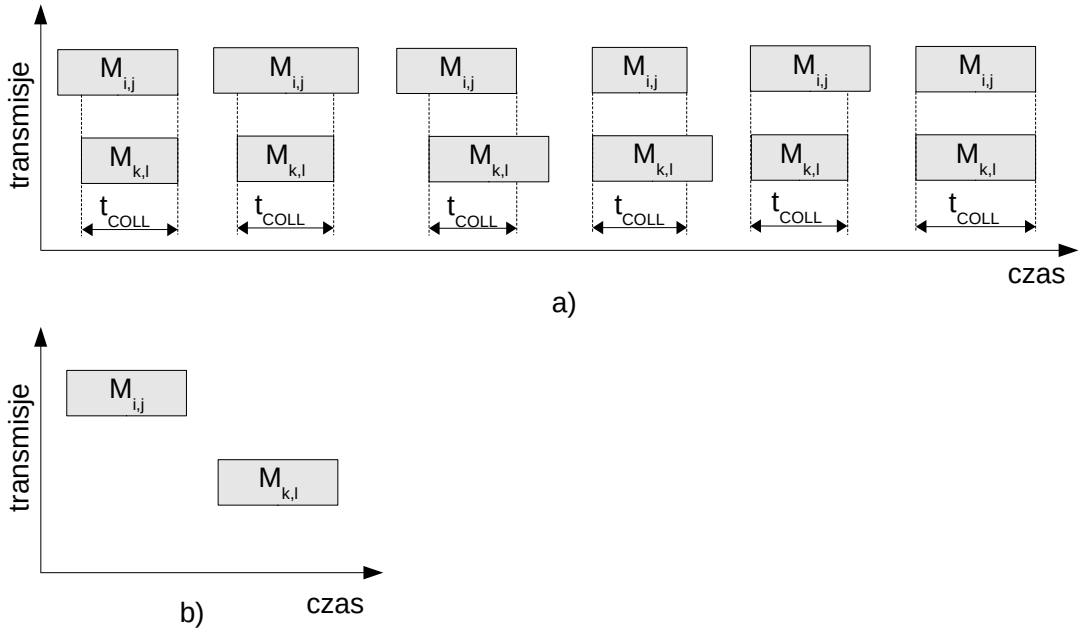
```
1:    jeśli TCL <> NULL to OLD_TCL := TCL
2:    TCL := NULL
3:    M_{i,j} := ITL.pierwszy_element;
4:    M_{k,l} := ITL.element_po_M_{i,j};
5:    dopóki M_{k,l} <> NULL wykonuj
6:          dopóki t_STOP(M_{i,j}) > t_START(M_{k,l}) wykonuj
7:                t_COLL := min {t_STOP(M_{i,j}),t_STOP(M_{k,l})} – max {t_START(M_{i,j}),t_START(M_{k,l})}
8:                jeśli (M_{i,j}, M_{k,l}) brak w TCL to
9:                      jeśli |M_{i,j}|>=|M_{k,l}| to TCL := TCL ∪ Node_TCL(M_{i,j}, M_{k,l},
                              t_COLL, 0, FALSE)
10:                     jeśli nie to TCL := TCL ∪ Node_TCL(M_{k,l}, M_{i,j}, t_COLL, 0,
                              FALSE)
11:                     jeśli OLD_TCL <> NULL to Aktualizuj(TCL)
12:               M_{k,l} := ITL.element_po_M_{k,l};
13:         M_{i,j} := ITL.element_po_M_{i,j};
```

14: $M_{k,l}$:= ITL.element_po_$M_{i,j}$;

Algorytm 2 najpierw sprawdza, czy tworzenie listy jest wykonywane pierwszy raz, czy jest skutkiem przeszeregowania (linia 1). Następnie dokonuje sekwencyjnego porównania parami ułożenia czasowego kolejnych transmisji znajdujących się na liście ITL (tj. pierwszy element z drugim, trzecim, itd., następnie drugi z trzecim, czwartym, itd., potem trzeci z czwartym, piątym, itd.). Liczba porównań dla *n*-elementowej listy ITL określona jest jako:

$$\frac{n(n-1)}{2} \tag{6.5-1}$$

Warunek w linii 6 ma za zadanie stwierdzenie faktu nakładania się transmisji w czasie. Wszystkie możliwe sposoby temporalnych relacji między dwoma komunikatami z listy ITL zobrazowano na Rys. 6.5-1. Część a) ilustracji odpowiada kolizji wiadomości, natomiast część b) przedstawia niekolidujące wiadomości.



Rys.6.5-1 Ułożenie w czasie transmisji kolidujących (a) oraz niekolidujących (b).

W przypadku wykrycia kolizji wykonywany jest kod z linii 7-11, rozpoczynający się obliczeniem czasu trwania kolizji. Następnie, jeśli wykryta kolizja jest nowa (nie ma jej na liście TCL), wówczas algorytm dodaje parę skonfliktowanych komunikatów do listy TCL w kolejności wynikającej z czasu ich trwania – jako pierwsza dodawana jest wiadomość dłuższa. Powyższa zasada znajduje swe uzasadnienie podczas szukania tras dla komunikatów w trakcie tworzenia dedykowanej topologii (rozdział 6.6). Z energetycznego punktu widzenia (równania (5.2.2-7), (5.2.2-8) i (5.2.2-11)) pożądane jest, aby dłuższe wiadomości były przesyłane krótszymi trasami, zatem dla nich w pierwszej kolejności szukane są najkrótsze ścieżki w sieci jednoukładowej. Linia 11 pseudokodu wykonywana jest w przypadku przeszeregowania transmisji, np. podczas usuwania kolizji na portach, opisanego dalej w niniejszym podrozdziale. W takim przypadku nie

dodaje się nowej pozycji *K* do listy TCL, lecz aktualizuje parametry istniejącej. Aktualizacja polega na porównaniu nowej listy TCL z jej poprzednią kopią (OLD_TCL) pod kątem odnalezienia wspólnych pozycji (konfliktów). Jeśli porównanie dla danej kolizji zakończy się sukcesem to przepisywana jest stara wartość znacznika *K.jest_trasa*, co zapobiega powtórnej próbie szukania ścieżek w dotychczas wygenerowanej topologii połączeń. Ponadto z nowej kopii listy TCL usuwane są te pozycje, dla których dokonano udanego przeszeregowania (*K.przeszeregowano <> 0*), opisanego w kolejnym akapicie.

Usuwanie kolizji na portach nadawczych i odbiorczych

W związku z punktem 4 Definicji 5.2-2, kolizje na portach nie mogą być usunięte za pomocą dedykowanej topologii, zatem należy je przeszeregować. Mimo, iż zakłada się, że algorytm kosyntezy szereguje komunikację w sposób eliminujący ten typ konfliktów, to mogą się one pojawić po przeszeregowaniu w ramach omawianej metodologii. Czynności służące usunięciu kolizji na portach prezentuje ramka Algorytm 3. Przy podejmowaniu decyzji, którą z kolidujących transmisji opóźnić kierowano się kryterium preferującym przemieszczenie komunikatów rozpoczynających się później i/lub dłużej trwających (o większym wolumenie). Można również zastosować bardziej złożony warunek wyboru transmisji do przeszeregowania – do opóźnienia może być wybierana ta z kolidujących transmisji, która w mniejszym stopniu pogarsza czasy zakończenia zadań z ustanowionymi ograniczeniami czasowymi (ale bez ich przekroczenia) i/lub która nie znajduje się na ścieżce krytycznej ATG.

Algorytm 3 *– Usuwanie kolizji na portach*

Wejście: Lista Kolizji Temporalnych TCL

Wyjście: zaktualizowane listy ITL i TCL oraz graf ATG

```
1:    dla każdej pozycji K.(M_{i,j},M_{k,l}) w TCL wykonaj
2:          jeśli P_NAD(T_i) = P_NAD(T_k) LUB P_ODB(T_j) = P_ODB(T_l) to
3:                jeśli K.przeszeregowano <> 2 to
4:                      K.przeszeregowano := K.przeszeregowano + 1
5:                      jeśli  K.przeszeregowano = 1 to
6:                            jeśli [t_START(M_{i,j}) > t_START(M_{k,l})] LUB
                              [t_START(M_{i,j}) = t_START(M_{k,l}) ORAZ |M_{i,j}| > |M_{k,l}|] to
7:                                  t_START-SET(M_{i,j}) := t_START(M_{k,l}) + |M_{k,l}|
8:                            jeśli nie to t_START-SET(M_{k,l}) := t_START(M_{i,j}) + |M_{i,j}|
9:                      jeśli  K.przeszeregowano = 2 to
10:                           jeśli [t_START(M_{i,j}) < t_START(M_{k,l})] LUB
                              [t_START(M_{i,j}) = t_START(M_{k,l}) ORAZ |M_{i,j}| < |M_{k,l}|] to
11:                                 t_START-SET(M_{i,j}) := t_START(M_{k,l}) + |M_{k,l}|
12:                           jeśli nie to t_START-SET(M_{k,l}) := t_START(M_{i,j}) + |M_{i,j}|
```

```
13:                    Aktualizuj_i_sprawdź(ATG)
14:              jeśli nie to Kosynteza(Tₖ)
```

Każda pozycja *K* w liście TCL poddawana jest testowi logicznemu opisanemu w linii 2, mającemu na celu wykrycie kolizji na porcie. Linia 3 pseudokodu zapobiega zapętleniu algorytmu na skutek nieskutecznego naprzemiennego przeszeregowywania kolidujących transmisji. Kroki 5-8 dokonują pierwszej próby zmiany kolejności komunikatów według uprzednio opisanych kryteriów. Jeśli takie podejście zakończy się porażką (naruszono któreś z ograniczeń t_{CRIT}), wykonywana jest druga próba, odwracająca zasady decydujące o przeszeregowaniu i opisana liniami 9-12. Po każdej czynności przypisania nowego czasu rozpoczęcia danej transmisji niezbędna jest aktualizacja grafu ATG. Niepowodzenie obu prób, wykryte w linii 3, prowadzi do ponownej kosyntezy z ustanowionymi nowymi ograniczeniami czasowymi dla zadań systemu jak opisano w rozdziale 6.3. Jeżeli natomiast po którymkolwiek z przeszeregowań sprawdzenie ograniczeń t_{CRIT} da wynik pozytywny, pozycja *K* dla której *K.przeszeregowano* <> *0* jest natychmiast usuwana z listy TCL na etapie aktualizacji zawartości tej struktury. Aktualizacja grafu ATG polega na ponownym przypisaniu czasów rozpoczęcia dla zadań i transmisji przy czym dla jednego komunikatu w grafie czas rozpoczęcia jest narzucony ($t_{START\text{-}SET}(M_{i,j})$ lub $t_{START\text{-}SET}(M_{k,l})$). Uszeregowanie sprowadza się zatem do przeliczenia i w rezultacie zwiększenia czasu rozpoczęcia zadań i transmisji tworzących podgraf zaczynający się krawędzią, dla której wyznaczono czas rozpoczęcia. Pseudokod algorytmu sprawdzającego naruszenie wymagań Warunku 5.3-2 dla zadań po przeszeregowaniu zawiera ramka Algorytm 4.

Algorytm 4 *– Sprawdzenie ograniczeń wynikających z czasów* t_{CRIT}

Wejście: przeszeregowana transmisja $M_{x,i}$ graf ATG

Wyjście: TRUE – jeśli ograniczenia niespełnione, FALSE – w przeciwnym przypadku

```
1:    dla każdego węzła Tᵢ podgrafu o początku w succT(Mₓ,ᵢ) wykonaj
2:          jeśli (tSTART(Tᵢ)+|Tᵢ|) > tCRIT(Tᵢ) to
                  zwróć TRUE
3:    zwróć FALSE
```

6.6 Budowanie topologii sieci

Ostatnim, i w sumie najistotniejszym, krokiem w całej metodologii zaprezentowanej na Rys. 6.3-1 jest utworzenie dedykowanej topologii połączeń w celu rozwiązania kolizji na trasie oraz realizacji wszystkich transmisji międzyprocesorowych. Topologia budowana jest poprzez generowanie krawędzi w grafie NTG złożonym wstępnie z samych węzłów. Ponadto lista ITL jest sukcesywnie uzupełniana informacjami o trasach wyznaczanych dla poszczególnych transmisji.

Po uszeregowaniu (etap kosyntezy) dla każdego zadania T_i można wyznaczyć tzw. zapas czasowy t_{SLACK} (ang. *slack time*), będący różnicą pomiędzy nałożonym na zadanie ograniczeniem czasowym a wynikającym z uszeregowania czasem zakończenia wykonywania:

$$t_{SLACK}(T_i) = t_{CRIT}(T_i) - t_{STOP}(T_i) \quad (6.6\text{-}1)$$

Prawidłowe uszeregowanie gwarantuje, że $t_{SLACK}(T_i) \geq 0$. Zjawisko kolizji powoduje, że żadna ze skonfliktowanych transmisji nie będzie mogła się zakończyć (całkowita blokada zasobów) lub jedna z nich zakończy się z opóźnieniem w wyniku oczekiwania na zwolnienie zasobów komunikacyjnych. W drugim przypadku mogą ulec wydłużeniu czasy zakończenia zadań i komunikatów tworzących podgraf rozpoczynający się przesuniętą w czasie transmisją. Im większe przesunięcie (dłuższa kolizja), tym większe potencjalne pogorszenie czasów zakończenia zadań, wpływających na wartość t_{SLACK}. W szczególności dla zadania T_j, otrzymującego dane od transmisji $M_{i,j}$ opóźnionej w wyniku kolizji, zapas czasu dla przeszeregowania $t_{SLACK'}$ wynosi:

$$t_{SLACK'}(T_j) = t_{SLACK}(T_j) + t_{SLACK}(M_{i,j}) - t_{COLL}(T_j) \quad (6.6\text{-}2)$$

gdzie

$$t_{SLACK}(M_{i,j}) = t_{START}(T_j) - t_{STOP}(M_{i,j}) \quad (6.6\text{-}3)$$

oznacza zapas czasowy pomiędzy końcem transmisji a rozpoczęciem zadania otrzymującego dane od tej transmisji. Wartość $t_{SLACK'} < 0$ oznacza niespełnienie ograniczenia czasowego t_{CRIT} dla danego zadania.

DEFINICJA 6.6-1 (PRIORYTET KOLIZJI)

Parametr zwany priorytetem kolizji, wprowadzony dla potrzeb algorytmu generującego sieć połączeń w NTG, definiuje się następująco:

$$\forall K \in TCL, \text{ gdzie } M_{i,j}, M_{k,l} \in K, priorytet_K = min(t_{SLACK'}(T_j), t_{SLACK'}(T_l)) \quad (6.6\text{-}4)$$

Z Definicji 6.6-1 wynika, że im większe pogorszenie zapasu czasowego do zakończenia zadań otrzymujących dane od skolidowanych transmisji, tym mniejsza wartość priorytetu.

OBSERWACJA 6.6-1 (KOLEJNOŚĆ ROZSTRZYGANIA KOLIZJI)

Rozwiązywanie kolizji w kolejności niemalejącej wartości „priorytet$_K$" prowadzi do systemu o wyższej wydajności.

Na podstawie Obserwacji 6.6-1 oraz modelu energetyczno-szybkościowego sieci jednoukładowej (równania (5.2.2-7), (5.2.2-8) i (5.2.2-11)) przed przystąpieniem do tworzenia topologii połączeń międzyruterowych lista z kolizjami TCL jest sortowana niemalejąco według wartości priorytetów. Kompletny pseudokod ostatniego kroku zawarto w ramce Algorytm 5.

***Algorytm 5** – Generowanie dedykowanej bezkolizyjnej topologii NoC*

Wejście: posortowana Lista Kolizji Temporalnych TCL, graf ATG, opcjonalnie częściowo zbudowany Graf Topologii Sieci NTG

Wyjście: Graf Topologii Sieci NTG, uzupełniona o wpisy z trasami i opcjonalnie zmienione czasy rozpoczęcia Lista Transmisji Międzyprocesorowych ITL

```
1:    dla każdego P_i w ATG wykonaj węzeł_P_i := DodajW(NTG, P_i)
2:    dla każdej pozycji K.(M_i,j,M_k,l) w TCL wykonaj
3:          jeśli K.jest_trasa = FALSE to
4:                jeśli K.R_a,b(M_i,j) <> NULL ORAZ K.R_c,d(M_k,l) <> NULL to
5:                      jeśli K.R_a,b(M_i,j) ∩ K.R_c,d(M_k,l) = ∅ to
6:                            K.jest_trasa := TRUE
7:                      jeśli nie to
8:                            R'(M_k,l) := Znajdź_inną_trasę(NTG, M_k,l, R_a,b)
9:                            dopóki R'(M_k,l) <> NULL wykonuj
10:                               jeśli Powoduje_kolizje(TCL,R'(M_k,l)) = TRUE
                                  to R'(M_k,l) := Znajdź_inną_trasę(NTG,M_k,l,R_a,b)
11:                               jeśli nie to
12:                                   Aktualizuj(ITL, R'(M_k,l))
13:                                   K.jest_trasa := TRUE
14:                                   przejdź do kroku 2
15:                           krawędź_NTG := DodajK(NTG, węzeł_P_a, węzeł_P_b)
16:                           jeśli krawędź_NTG <> NULL to
17:                                 R'(M_i,j) := krawędź_NTG
18:                                 Aktualizuj(ITL, R'(M_i,j))
19:                                 K.jest_trasa := TRUE
20:                                 przejdź do kroku 2
21:                           jeśli nie to powtórz kod z linii 8-14 dla
                              (M_i,j,R_c,d) oraz 15-20 dla (węzeł_P_c,węzeł_P_d, M_k,l)
22:                           Kosynteza(T_i)
23:               jeśli nie to
24:                     jeśli K.R_c,d(M_k,l) <> NULL ORAZ K.R_a,b(M_i,j) = NULL to
25:                           R'(M_i,j):=Szukaj_najkrótszej_trasy(NTG, M_i,j, R_c,d)
26:                           jeśli R'(M_i,j) <> NULL to
27:                                 Aktualizuj(ITL, R'(M_i,j))
28:                                 K.jest_trasa := TRUE
29:                           jeśli nie to
30:                                 R'(M_i,j):= Znajdź_inną_trasę(NTG, M_i,j, R_c,d)
31:                                 jeśli R'(M_i,j) <> NULL to
32:                                       Aktualizuj(ITL, R'(M_i,j))
33:                                       K.jest_trasa := TRUE
```

```
34:                                   jeśli nie to
35:                                         krawędź_NTG := DodajK(NTG,węzeł_Pa,
                                                              węzeł_Pb)
36:                                         jeśli krawędź_NTG <> NULL to
37:                                               R'(Mi,j) := krawędź_NTG
38:                                               Aktualizuj(ITL, R'(Mi,j))
39:                                               K.jest_trasa := TRUE
40:                                         jeśli nie to przejdź do kroku 43
41:                         jeśli K.Ra,b(Mi,j) <> NULL ORAZ K.Rc,d(Mk,l) = NULL to
                                  powtórz kod z linii 24-32 dla (Mk,l, Ra,b)
                                  i z linii 33-39 dla (węzeł_Pc, węzeł_Pd, Mk,l)
42:                         jeśli nie to
                                  powtórz kod z linii 24-32 i 33-39 najpierw dla
                                  argumentów (Mi,j, NULL) i (węzeł_Pa,węzeł_Pb,Mi,j),
                                  a potem (Mk,l, Ra,b) i (węzeł_Pc,węzeł_Pd,Mk,l)
43:                   przejdź do kroku 2
44:                   wykonaj linie 3-14 z Algorytmu 3, cały Algorytm 2, zresetuj
                      wskaźnik TCL na początek listy i przejdź do kroku 2
45:   dla każdej pozycji Mi,j w ITL wykonaj
46:         jeśli R(Mi,j) = NULL to
47:               R'(Mi,j) := Szukaj_najkrótszej_trasy(NTG, Mi,j, NULL)
48:               jeśli R'(Mi,j) <> NULL to Aktualizuj(ITL, R'(Mi,j))
49:               jeśli nie to
50:                     krawędź_NTG : = DodajK(NTG, węzeł_Pa, węzeł_Pb)
51:                     jeśli krawędź_NTG <> NULL to
52:                           R'(Mi,j) := krawędź_NTG
53:                           Aktualizuj(ITL, R'(Mi,j))
54:                     jeśli nie to Kosynteza(Ti)
55:   Rafinacja(ITL, TCL, NTG)
```

Opis działania podprogramów wywoływanych w ramach Algorytmu 5:

- *Aktualizuj(ITL, R(M))* – podprogram uzupełniający o trasę *R* wpis dla pozycji *M* na liście ITL.
- *DodajK(NTG, P)* – funkcja łącząca krawędzią dwa węzły grafu NTG, zwraca *NULL* w przypadku niepowodzenia operacji.
- *DodajW(NTG, P)* – funkcja umieszczająca procesor *P* w grafie NTG (w postaci węzła).
- *Kosynteza(T)* – powrót do algorytmu kosyntezy, szczegóły podano w rozdziale 6.3.
- *Powoduje_kolizje(TCL, R(M))* – funkcja zwracająca *TRUE*, jeżeli nowo wyznaczona trasa *R* dla transmisji *M* powoduje kolizje dla uprzednio rozwiązanych konfliktów danej transmisji *M* z innymi komunikatami na liście TCL.

- *Rafinacja(ITL, TCL, NTG)* – ustala w finalnej topologii połączeń dla każdego komunikatu najkrótszą możliwą ścieżkę.
- *Szukaj_najkrótszej_trasy(NTG, M, R)* – funkcja szukająca najkrótszej ścieżki w grafie NTG między węzłami przesyłającymi komunikat *M*, rozłącznej względem ścieżki *R*; jeżeli *R = NULL* szukana jest dowolna najkrótsza ścieżka; w przypadku niepowodzenia operacji funkcja zwraca *NULL*.
- *Znajdź_inną_trasę(NTG, M, R)* – funkcja szukająca dowolnej ścieżki (niekoniecznie najkrótszej) w grafie NTG pomiędzy węzłami transmitującymi komunikat *M*; szukana ścieżka musi być rozłączna względem ścieżki *R*; jeżeli *R = NULL* szukana jest dowolna ścieżka; zwraca znalezioną trasę lub *NULL* w przypadku braku trasy.

Linie pseudokodu zaczynające się od słów „*powtórz kod(...)*” oznaczają, że w miejscu ich wystąpienia należy umieścić wymienione linie kodu z instrukcjami korzystającymi z podanych argumentów. Algorytm po kolei przetwarza pozycje z posortowanej listy TCL. Jeżeli dana kolizja byłaby już rozwiązana poprzez dedykowaną topologię (warunek w linii 3 byłby niespełniony), wówczas wczytywana jest kolejna pozycja. W przeciwnym przypadku w kroku 4 sprawdza się, czy uczestniczące w konflikcie wiadomości mają już przypisane jakieś trasy (na liście ITL). Jeśli tak i są to ścieżki niekolidujące (nie mają wspólnych odcinków *l*), pozycja *K* oznaczana jest jako przetworzona pomyślnie (linia 6).

W liniach 7-13 pseudokodu próbuje się zmienić kolidującą trasę dla krótszej transmisji. W przypadku niepowodzenia (brak możliwości znalezienia rozłącznej trasy lub znalezione trasy powodują inne konflikty) następuje próba wygenerowania bezpośredniego łącza dla dłuższego komunikatu. W razie porażki algorytm powtarza powyższe kroki dla dłuższej transmisji (linie 8-13 – wyznaczenie nowej ścieżki) i ewentualnie dla krótszej (linie 15-20 – wygenerowanie nowego, bezpośredniego połączenia między procesorem nadawczym i odbiorczym). Próba dodania dedykowanego połączenia respektuje warunki nałożone zależnościami (5.2.3-1) i (5.2.3-1). Nowe trasy, szukane w liniach 8-13, mogą być nieoptymalne pod względem liczby hopów, tzn. dłuższe. Jeśli szukanie ścieżek oraz dodawanie nowego łącza się nie uda, metoda wykonuje ponowną kosyntezę uzupełnioną o dodatkowe informacje (czas kolizji - szczegóły podano w rozdziale 6.3).

W liniach 24-42 algorytm szuka najkrótszej (lub nie, w przypadku niepowodzenia) trasy w sytuacji, gdy dla obu (linia 42) lub dla jednego z komunikatów nie przydzielono żadnej ścieżki *R(M)* (warunki w liniach 24 i 41). Brak jakichkolwiek tras spełniających wymóg rozłączności prowadzi do próby wygenerowania bezpośredniego łącza dla jednej lub obu skonfliktowanych transmisji (drugi przypadek znajduje zastosowanie tylko dla sytuacji, gdy żaden z komunikatów

nie miał dotąd przypisanej ścieżki - linia 42). Niepowodzenie w szukaniu trasy dla któregokolwiek z komunikatów w liniach 24-42 prowadzi do przeszeregowania. Jeśli zakończy się ono sukcesem następuje budowa listy TCL (z uwzględnieniem starej kopii) i algorytm zaczyna pracę od linii 2. Istotnym jest, iż zbudowana dotychczas topologia połączeń NTG zostaje zachowana. Po przetworzeniu całej listy TCL, w liniach 45-54 przypisywane są trasy dla pozostałych transmisji międzyprocesorowych to znaczy takich, które nie biorą udziału w żadnej kolizji, ale występują na liście ITL. Ta czynność również może okazać się niewykonalna o czym świadczy przedostatnia linia pseudokodu. W związku z tym, że mikrosieć jest stopniowo rozbudowywana w trakcie działania algorytmu komunikat o trasie przypisanej na początku działania metodologii może być ostatecznie przesłany więcej niż jedną drogą w NTG. Ostatni podprogram – *Rafinacja* – ma na celu sprawdzenie, czy dla każdej transmisji przypisano najkrótszą bezkolizyjną ścieżkę. Jeżeli próba skrócenia trasy prowadziłaby do naruszenia bezkolizyjności (którakolwiek pozycja *K* z listy TCL utraciłaby status rozwiązanej), wówczas pozostawia się oryginalny wpis ze ścieżką. Pseudokod procedury rafinacji tras zamieszczono w ramce Algorytm 6. Lista ITL jest sortowana nierosnąco według czasów trwania transmisji, gdyż ze względu na koszt energetyczny transmisji, wynikający z zależności (5.2.2-8) i (5.2.2-11), najpierw szuka się najkrótszych ścieżek dla najdłuższych komunikatów. Jeżeli Algorytm 5 zakończy się bez przekierowania do ponownej kosyntezy to graf NTG będzie zawierał topologię połączeń dla dedykowanej sieci NoC, lista ITL – trasy i końcowe uszeregowanie transmisji, natomiast graf ATG – końcowe uszeregowanie zadań i transmisji.

Algorytm 6 *– Rafinacja tras komunikatów*

Wejście: posortowana Lista Transmisji Międzyprocesorowych ITL, Lista Kolizji Temporalnych TCL, Graf Topologii Sieci NTG

Wyjście: zaktualizowane trasy w Liście Transmisji Międzyprocesorowych ITL

```
1:    dla każdej M(i,j) w ITL wykonaj
2:          R'(M(i,j)) := Szukaj_najkrótszej_trasy(NTG, M(i,j), NULL)
3:          jeśli R(a,b)(M(i,j)) <> R'(M(i,j)) to
4:                jeśli Powoduje_kolizje(TCL,R'(M(i,j))) = TRUE to
                        przejdź do kroku 1
5:                jeśli nie to Aktualizuj(ITL, R'(M(i,j)))
```

Metoda generowania topologii (Algorytm 5) w pierwszej kolejności próbuje wykorzystać istniejącą infrastrukturę połączeń do wyznaczenia bezkolizyjnych tras – stąd podprogramy *Szukaj_najkrótszej_trasy* oraz *Znajdź_inną_trasę* wywoływane na początku każdego wariantu rozwiązywanej kolizji. Wygenerowanie kolejnego łącza (*DodajK*) odbywa się w drugiej kolejności, spowodowanej brakiem rozłącznych ścieżek dla skonfliktowanych transmisji.

Bezpośrednie połączenie oznacza przypisanie trasy minimalnej jednemu z komunikatów, toteż ze względu na koszt energetyczny transmisji wykonywane jest na rzecz dłuższej (a w razie niepowodzenia – krótszej) z wiadomości. Dalsze obniżenie poboru mocy przez transmisje w systemie NoC wykonywane jest na etapie rafinacji rozwiązania (Algorytm 6). Nie ingeruje się wówczas w zbudowaną strukturę połączeń a jedynie próbuje zoptymalizować trasy komunikatów poprzez wyznaczenie najkrótszych ścieżek transmisji bez naruszania wymogu bezkolizyjności. A zatem heurystyka w pierwszej kolejności minimalizuje koszt zasobów (liczbę połączeń) poprzez eksplorację tras, a następnie pobór energii przez transmisje (szukanie najkrótszych tras).

Algorytm generowania systemu NoC dla danej aplikacji zadanej grafem ATG zaprezentowany w rozprawie należy do klasy algorytmów konstrukcyjnych z powrotami. W algorytmach tego typu istnieje ryzyko zapętlenia się. Przypadek taki ma miejsce, gdy w wyniku nawrotów nastąpi powrót do wcześniej rozpatrywanego rozwiązania częściowego.

TWIERDZENIE 6.6-1 (ZBIEŻNOŚĆ METODY)

Algorytm generowania systemu NoC dla danej aplikacji zadanej grafem ATG, przedstawiony na Rys. 6.3-1, jest zbieżny.

DOWÓD TWIERDZENIA 6.6-1

Powroty następują w przypadkach, gdy nie jest możliwe wyznaczenie niekolidujących tras dla transmisji z kolejnej rozpatrywanej kolizji $K.(M_{i,j},M_{k,l})$. Wówczas może wystąpić jedna z poniższych sytuacji:

1. Przeszeregowanie transmisji $M_{i,j}$ lub $M_{k,l}$ nie pozwala na usunięcie ich kolizji.
2. Kolizję można wyeliminować poprzez przeszeregowanie jednej z kolidujących transmisji.

W pierwszym przypadku wykonywana jest ponowna kosynteza, ale z dodatkowym ograniczeniem dla zadania T_i równym: $t_{CRIT}(T_i) = t_{STOP}(T_i) + t_{SLACK'}(T_j)$ (gdzie $t_{SLACK'}(T_j)<0$ oznacza przekroczenie $t_{CRIT}(T_j)$ spowodowane przeszeregowaniem transmisji $M_{i,j}$). Przy kolejnych iteracjach czas $t_{CRIT}(T_i)$ będzie zmniejszany aż do chwili, gdy algorytm znajdzie rozwiązanie lub czas $t_{CRIT}(T_i)$ osiągnie wartość dla której nie istnieje uszeregowanie zadań takie że $t_{STOP}(T_i) \leq t_{CRIT}(T_i)$.

W drugim przypadku wystąpienia nawrotu zapamiętywany jest bieżący stan rozwiązania (graf NTG oraz lista TCL) będący podstawą dla kolejnych iteracji metody. Zatem, jeżeli A oznacza topologię NTG zbudowaną w kroku *x-1* metody przy uszeregowaniu **S** oraz wyznaczonych trasach **R**, A' - topologię powstałą w kroku *x* przy uszeregowaniu **S'** oraz

wyznaczonych trasach **R'**, zaś Δ_l to zbiór połączeń międzyruterowych (krawędzi grafu NTG) dodanych w kroku *x* to dla każdego *A'* prawdziwe jest równanie:

$A' = A + \Delta_l$, gdzie jeżeli $\Delta_l = 0$, to $\mathbf{S'} \neq \mathbf{S}$ lub $\mathbf{R'} \neq \mathbf{R}$

Maksymalna liczba kroków *x* w metodzie wynika z maksymalnej możliwej liczby kolizji, którą określa wzór (6.5-1).

W związku z tym, że nie wraca się do rozwiązań już rozpatrywanych (topologii *A* przy danym uszeregowaniu **S** oraz wyznaczonych trasach **R**) oraz maksymalna liczba kroków jest wartością skończoną metodologia jest zbieżna. ¤

Ruting deterministyczny preadaptacyjny w utworzonej strukturze sieci jednoukładowej gwarantuje brak zakleszczeń (ang. *deadlock freedom*). Zakleszczenie w sieci NoC występuje wówczas, gdy dwie (lub więcej) transmisje oczekuje jednocześnie na zwolnienie zajmowanych przez drugą stronę zasobów (kanałów komunikacyjnych, czyli połączeń międzyruterowch lub buforów w portach rutera). Sytuacja taka może wystąpić, gdy spełnione są łącznie oba poniższe warunki:

- transmisje zachodzą w tym samym przedziale czasu,
- transmisje współdzielą ten sam kanał komunikacyjny w postaci całości lub fragmentu trasy *R*.

Generowana w rozprawie architektura NoC ma wśród założeń konstrukcyjnych warunek:

$\forall\ M_{i,j}, M_{k,l} \in \mathbf{C_x}, R_{a,b}(M_{i,j}) \cap R_{c,d}(M_{k,l}) = \varnothing$

co powoduje wyeliminowanie sytuacji sprzyjających zakleszczeniom (transmisje zachodzące w tym samym przedziale czasu mają przydzielone odrębne, niepokrywające się trasy).

6.7 Skuteczność metody

W związku z faktem, iż do odwzorowania aplikacji w architekturę NoC zgodnie z warunkami sformułowanymi w rozdziale 6.1 wykorzystywana jest metoda heurystyczna, nie gwarantuje się znalezienia rozwiązania optymalnego. Może również wystąpić taki przypadek, kiedy metodologia nie znajdzie rozwiązania, mimo iż ono istnieje. W niniejszym rozdziale zostaną przeanalizowane przypadki, dla których wygenerowanie sieci jednoukładowej zaproponowaną metodologią nie będzie możliwe. Na wstępie należy wyodrębnić te kroki w procesie budowania sieci jednoukładowej, których wykonanie może się zakończyć sukcesem lub porażką, skutkującą koniecznością ponownej kosyntezy.

Pierwszą czynnością, jaka jest wykonywana na rzecz pary skonfliktowanych transmisji jest znalezienie dla każdej z nich niezależnej trasy. W najbardziej niekorzystnym przypadku

w aplikacji zbudowanej z *n* procesorów w jednej chwili komunikaty mogą nadawać wszystkie (z zastrzeżeniem, że odbiorcy komunikatów są różni – brak konfliktów na portach docelowych). W dowolnym grafie maksymalna liczba rozłącznych ścieżek jest równa liczbie krawędzi. Dla grafu o stopniu 4 zbudowanego z *n* węzłów maksymalna liczba krawędzi jest równa *2*n*.

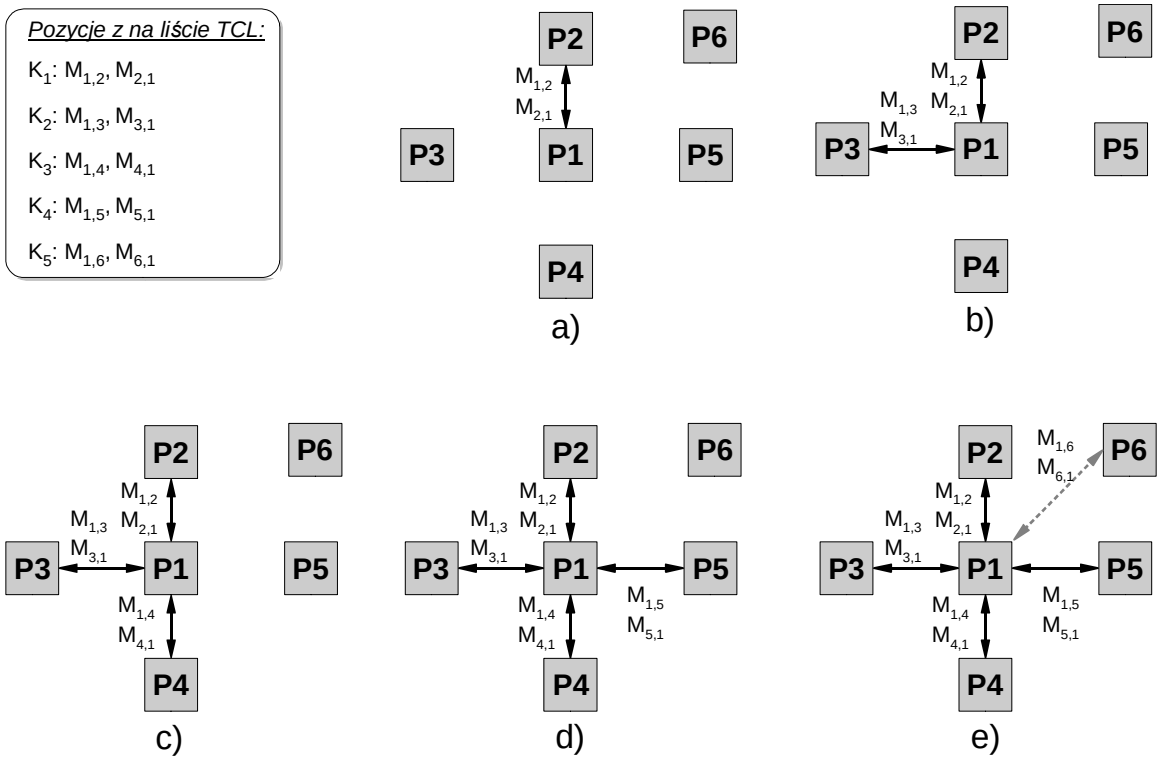
Kolejnym krokiem w metodologii, wykonywanym po nieudanym wyznaczeniu niekolidujących tras jest wygenerowanie kolejnego, dedykowanego połączenia pomiędzy nadawcą i odbiorcą. Sytuacja taka ma miejsce w jednym z poniższych przypadków:

1. Moduły nadawcy i odbiorcy są już umieszczone w NTG (uczestniczyły w innych kolizjach), ale nie ma między nimi dedykowanego łącza, gdyż do tej pory ich transmisje nie powodowały kolizji.
2. Modułów nadawcy i/lub odbiorcy nie ma jeszcze w NTG, gdyż żadna z ich transmisji nie powodowała dotąd żadnych kolizji.
3. Pomiędzy komunikującymi się węzłami istnieją ścieżki, ale kolidują one z innymi trasami w ramach transmisji z tego samego zbiór $\mathbf{C_x}$.

Dodawanie łącza międzywęzłowego podlega ograniczeniom zależności (5.2.3-1). Wynika z niej, że do danego modułu PE wolno podłączyć bezpośrednio co najwyżej cztery inne moduły. Problem może się pojawić, gdy w wyniku wyżej wymienionych ograniczeń nie da się wyeliminować kolizji poprzez modyfikację topologii sieci NOC. Taki przypadek będzie miał miejsce w następującej sytuacji: :

a) dla dowolnego procesora P_x w ATG istnieje co najmniej pięć innych modułów PE wysyłających/odbierających do/od niego transmisje,
b) transmisje te znajdą się po sobie na liście TCL (wejdą w kolizje z innymi komunikatami i kolizje te będą uszeregowane nierosnąco względem ich czasu trwania),
c) wymienione w ppkt.a) procesory nie komunikowały się między sobą przed wysłaniem/odebraniem wiadomości do/od P_x.

Sytuację taką obrazuje Rys. 6.7-1, na którym przedstawiono kolejne etapy powstawania topologii. Dla uproszczenia założono, że kolizje o których mowa w ppkt.b) zachodzą na skutek transmisji bilateralnych. Ilustracja pokazuje wpływ kolejności transmisji na budowę NTG. Przykładowo, gdyby P3 spowodował konflikt komunikacyjny z P2 zanim wywoła to samo zjawisko z P1, metoda dodałaby łącze bezpośrednie P3 – P2, natomiast późniejsza komunikacja P3 z P1 biegłaby trasą P1 – P2 – P3. Taka kolejność zdarzeń umożliwiłaby bezpośrednie połączenie P1 – P6.



Rys. 6.7-1 Brak możliwości wygenerowania łącza P1←→P6.

Reasumując – metoda nie znajdzie rozwiązania, gdy wyeliminowanie kolizji z pięciu zbiorów $\mathbf{C_x}$ będzie wymagało dodania nowego połączenia do tego samego węzła zgodnie z przedstawioną analizą. Powyższą wadę można wyeliminować poprzez umożliwienie dodawania do NTG nie tylko łącz bezpośrednich, ale również pośrednich, na przykład do procesorów sąsiadujących z komunikującą się parą jednostek PE. Przypadek opisany w bieżącym rozdziale występuje w praktyce rzadko – w trakcie badań z brakiem możliwości wygenerowania dedykowanego połączenia zetknięto się tylko raz (dla jednego z wariantów kosyntezy aplikacji FFT). Rozwiązaniem dla tego typu sytuacji jest usunięcie jednego z wymienionych zbiorów $\mathbf{C_x}$ poprzez przeszeregowanie lub ponowną kosyntezę.

7 Wyniki eksperymentalne

Często spotykaną cechą nowych dziedzin badań jest brak ogólnie przyjętych metod porównywania jakości otrzymywanych rozwiązań, czyli tzw. *benchmarków*. Problem ten dotyczy również projektowania sieci jednoukładowych. Sytuację dodatkowo komplikuje fakt tworzenia mikrosieci NoC przez wiele będących przedmiotem badań elementów składowych, jak na przykład topologię, protokoły rutingu i sterowania przepływem czy routery. Kryteria porównań mogą być zatem diametralnie różne – są to m.in. szybkość działania systemu, maksymalna przepustowość, koszt zasobowy i koszt energetyczny. W ostatnich latach środowisko badaczy podjęło prace nad ustandaryzowaniem sposobów porównywania propozycji z zakresu projektowania sieci NoC [GIPJS07], ale do tej pory nie opublikowano zestawu ogólnie przyjętych testów. Niemniej jednak w literaturze przedmiotu można zaobserwować pewne wspólne kierunki szacowania jakości rozwiązań NoC na drodze eksperymentalnej. W bieżącym rozdziale opisano sposób przeprowadzenia badań zastosowany w rozprawie oraz podano wyniki dla wygenerowanych syntetycznych grafów zadań oraz dla przykładowych systemów rzeczywistych.

7.1 Metodyka badań

Kryteria porównywania różnych rozwiązań w dziedzinie odwzorowania aplikacji w NoC podano w podrozdziale 5.2.2. Należą do nich: czas wykonania zadań systemu, zapotrzebowanie na zasoby mierzone jako liczba łącz jednokierunkowych (oraz powiązanych z nimi portów ruterów) i przeciętna długość tras komunikatów. Zgodnie z podejściami spotykanymi w literaturze przedmiotu rozwiązania uzyskane metodologią opisywaną w rozprawie dla danego zbioru aplikacji porównano z kilkoma rodzajami systemów:

- architekturami o losowo wygenerowanej topologii kraty z rutingiem XY [HM03a] [NC05][LT06][ZZM06][TOPD08]; pod uwagę wzięto struktury kwadratowe (*mesh* NxN) oraz prostokątne (*mesh* MxN) – pierwszy rodzaj najczęściej występuje w pracach z dziedziny NoC, natomiast drugi został wybrany ze względu na nieco bardziej oszczędne gospodarowanie zasobami w niektórych przypadkach (przykładowo dla systemu zbudowanego z 5 modułów PE najmniejszy możliwy kwadratowy *mesh* miałby wymiary 3x3, co dla realizacji rutingu XY wymusiłoby zastosowanie 4 ruterów bez przyłączonych do nich jednostek i potencjalnie wydłużyłoby trasy komunikatów; ten sam przypadek odwzorowany w kratę prostokątną miałby wymiary 3x2 i wymagałby tylko jednego nadmiarowego rutera do budowy architektury),
- architekturami NoC wygenerowanymi metodą bazującą na pracach [MMCM07]

[TOPD08], lokującą jak najbliżej siebie procesory wymieniające największe ilości danych - celem jest zwykle obniżenie kosztu komunikacyjnego oraz przyspieszenie pracy systemu; metodę tę nazwano LCF za [MMCM07] (ang. *Largest Communication First*); inne prace bazujące na zasadzie rezerwacji pasma transmisyjnego, na przykład [MMAAC06], również mają na celu w pierwszej kolejności bezpośrednie łączenie modułów intensywnie komunikujących się ze sobą,

- dla części aplikacji syntetycznych oraz dla wszystkich badanych systemów rzeczywistych oszacowano również czas działania dla realizacji w architekturze opartej na wspólnej magistrali.

Architekturę NoC wygenerowano metodą LCF w sposób następujący:

- graf ATG został przekształcony do postaci grafu charakteryzującego system (ACG, CG) – zadania przydzielone do tych samych modułów PE pogrupowano i przeanalizowano ich zależności komunikacyjne,
- następnie posortowano nierosnąco wszystkie transmisje międzyprocesorowe (wolumeny transmisji zachodzących pomiędzy tą samą parą jednostek przetwarzających zostały zsumowane) tworząc strukturę podobną do Listy Kolizji Temporalnych (TCL),
- na bazie rezultatu poprzedniego kroku generowano topologię dodając dwukierunkowe łącza między komunikującymi się procesorami; jeżeli pomiędzy daną parą modułów PE istniała ścieżka, łącze nie było dodawane (minimalizacja kosztu zasobów),
- na końcu sprawdzono, czy uzyskana topologia pozwala wyeliminować wszystkie kolizje transmisyjne poprzez wyznaczenie niepokrywających się minimalnych ścieżek; jeśli trasowanie kończyło się niepowodzeniem kolizja była usuwana dedykowanym jednokierunkowym łączem generowanym dla większej z kolidujących wiadomości (stąd w rozprawie w tabelach z wynikami metoda zwana jest „LCF+łącza”),
- niepowodzenie poprzedniego kroku (zbyt dużo połączeń dla danego rutera – zależności (5.2.3-1) i (5.2.3-2)) skutkowałoby koniecznością przeszeregowania jednego z komunikatów; sytuacja taka nie wystąpiła w żadnym z badanych przypadków.

W związku z tym, że jednym z kryteriów porównań był czas działania systemu, zrezygnowano z nałożenia na grafy zadań ograniczeń czasowych. Założenie $t_{CRIT} = \infty$ dla każdego z zadań implikuje również $t_{SLACK} = \infty$ (równanie 6.6-1), stąd przyjęto, że wartość priorytetu opisanego zależnością (6.6-4) zależy tylko od wartości t_{COLL} (odwrotnie proporcjonalnie – im wyższy czas kolizji tym mniejsza wartość priorytetu). Czynnikami sterującymi jakością rozwiązań były zatem koszty zasobowe (liczba połączeń) oraz energetyczne (liczba połączeń oraz długości tras dla komunikatów). W eksperymentach nie wykonano również

etapu rafinacji tras (Algorytm 6) ze względu na porównanie z topologiami kraty – w ich przypadku nie ma możliwości poprawy ścieżki po odwzorowaniu aplikacji w topologię. Dla części aplikacji syntetycznych oraz dla systemów rzeczywistych poszerzono zakres porównań. Szczegóły każdorazowo podano w miejscu prezentacji wyników. Wszystkie eksperymenty przeprowadzono metodami analitycznymi.

7.2 Syntetyczne grafy zadań

Za pomocą narzędzia wzorowanego na aplikacji TGFF [DRW98] wygenerowano 16 rozbudowanych, syntetycznych grafów ATG, zróżnicowanych pod względem parametrów obliczeniowo-transmisyjnych. Taką technikę tworzenia danych wejściowych do badań można spotkać w wielu pracach z zakresu syntezy NoC, jak choćby [LK03][HM05b][LT06][NW08] [CYC09]. Parametry aplikacji przedstawiono w Tabeli 7.2-1. Liczba PE alokowanych do realizacji zadań jest rezultatem kosyntezy. Przedostatnia kolumna wskazuje na prawdopodobieństwo wystąpienia kolizji w danym systemie (im mniejszy współczynnik tym wyższe prawdopodobieństwo), natomiast ostatnia informuje o poziomie natłoku komunikacyjnego (im wyższy współczynnik tym większy natłok, „Czas kolizji" to sumaryczny czas trwania wszystkich kolizji z listy TCL). Natłok rozumiany jest jako ilość jednocześnie transmitowanych wiadomości w mikrosieci.

Aplikacja	Liczba zadań	Liczba transmisji	Liczba PE	Czas obliczeń/ czas transmisji	Czas kolizji/ czas transmisji
Graf1	22	21	6	0,54	0,35
Graf2	27	25	7	0,69	0,64
Graf3	27	27	5	0,36	0,48
Graf4	20	17	5	0,66	0,3
Graf5	24	17	8	1,32	0,41
Graf6	22	22	6	1	0,4
Graf7	21	24	7	0,58	0,5
Graf8	29	25	9	1,22	0,4
Graf9	22	15	6	1,83	0,25
Graf10	29	29	8	1,06	0,52
Graf11	33	42	9	0,56	0,24
Graf12	34	31	12	1,07	0,21
Graf13	37	41	14	0,92	0,88
Graf14	33	26	12	1,5	0,39
Graf15	32	28	12	1,1	0,72
Graf16	35	46	10	1,01	0,84

Tabela 7.2-1 Parametry badanych aplikacji syntetycznych.

Dla pierwszej grupy grafów (Graf1 - Graf10) przeprowadzono podstawowy zakres porównań, natomiast dla pozostałych aplikacji (Graf11 – Graf16) poszerzono eksperymenty o dodatkowe metody. W Tabeli 7.2-2 zebrano wyniki dotyczące pogorszenia czasu wykonania zadań systemu dla kilku rozwiązań. WWK (Według Wielkości Kolizji – rezultat rezygnacji dla potrzeb eksperymentów z ograniczeń czasowych dla zadań) oznacza metodę proponowaną

w rozprawie, natomiast *mesh* NxN i MxN to losowo wygenerowane struktury kraty. Systemem referencyjnym jest graf ATG otrzymany w wyniku kosyntezy (bez utworzonej architektury). Podejście WWK dało rezultaty średnio dwa razy lepsze od losowych struktur regularnych, a w trzech przypadkach czas wykonania nie uległ pogorszeniu po wygenerowaniu topologii dla aplikacji po kosyntezie. Wyniki dla rozwiązania „LCF+łącza" były identyczne jak dla WWK (rezultat usunięcia kolizji dedykowanymi łączami).

Aplikacja	**WWK [%]**	***mesh* NxN [%]**	***mesh* MxN [%]**
Graf1	7,08	12,39	12,39
Graf2	10,77	19,84	38,25
Graf3	7,79	13,25	16,09
Graf4	8,66	20,14	13,52
Graf5	9,86	15,44	20,8
Graf6	5,02	7,14	10,55
Graf7	0	10,17	9,34
Graf8	0	8	8
Graf9	0	3,64	0
Graf10	10,9	17,18	15,2
średnio	*6,01*	*12,72*	*14,41*

Tabela 7.2-2 Pogorszenie czasów wykonania dla różnych architektur NoC względem wyniku kosyntezy.

W Tabeli 7.2-3 zamieszczono rezultaty badań informujące o stopniu pogorszenia zasobochłonności (liczba połączeń użytych do budowy topologii) poszczególnych architektur oraz ich wydajności czasowo-energetycznej (parametr hop_{AVG}). Druga kolumna zawiera wartości odniesienia uzyskane dla podejścia WWK, zaś pozostałe – stopień pogorszenia (procentowo) względem systemu referencyjnego. Wartość ujemna oznacza poprawienie danego parametru.

Aplikacja	***WWK* hop_{AVG} (łącza)**	**Pogorszenie arch. NoC: hop_{AVG} (łącza) [%]**		
		***mesh* NxN**	***mesh* MxN**	**LCF+łącza**
Graf1	2,28 (9)	19,72 (62,5)	17,09 (35,71)	4,2 (18,18)
Graf2	2,58 (9)	19,63 (62,5)	15,13 (55)	-5,31 (35,71)
Graf3	2,17 (9)	16,86 (62,5)	15,89 (35,71)	2,69 (-12,5)
Graf4	2,54 (6)	10,88 (75)	12,41 (57,14)	0 (45,45)
Graf5	2,46 (10)	29,91 (58,33)	19,34 (50)	-3,36 (28,57)
Graf6	2,94 (9)	-1,03 (62,5)	-3,16 (35,71)	-24,05 (10)
Graf7	2,35 (12)	25,16 (50)	25,87 (40)	1,26 (0)
Graf8	2,43 (13)	11,31 (45,83)	11,31 (45,83)	-8,48 (27,78)
Graf9	2,47 (8)	22,57 (66,67)	6,44 (42,86)	0 (27,27)
Graf10	2,3 (13)	15,44 (45,83)	22,03 (35)	4,96 (13,33)
średnio	*2,45 (-)*	*17,04 (59,17)*	*14,24 (43,3)*	*-2,81 (19,38)*

Tabela 7.2-3 Pogorszenie parametru hop_{AVG} oraz liczby użytych łącz jednokierunkowych w porównaniu z zaprezentowanym podejściem (NoC WWK).

Przed interpretacją wyników z Tabeli 7.2-3 warto przypomnieć, iż metody LCF mają na celu przede wszystkim minimalizację poboru mocy związanej z kosztem komunikacji i maksymalizację wydajności przetwarzania, czyli dążą do zmniejszenia wartości hop_{AVG}. Zatem średnie pogorszenie o niecałe 3% dla rozwiązań WWK względem LCF należy uznać za wynik dobry. Jedynie w czterech przypadkach system LCF poprawił wymieniony parametr. Grafy

zadań, dla których metoda prezentowana w rozprawie dała gorsze rezultaty, cechowały się występowaniem długich transmisji niewchodzących w kolizje z innymi komunikatami, kolizjami pomiędzy relatywnie krótkimi transmisjami i/lub dużą liczbą komunikatów wymienianych pomiędzy tymi samymi parami procesorów. Średnia oszczędność pod względem użytych do konstrukcji topologii połączeń dla WWK wynosi od niemal 20% (LCF) do prawie 60% (losowy *mesh* kwadratowy).

Dla pozostałych sześciu aplikacji syntetycznych poszerzono zakres porównań o kilka dodatkowych rozwiązań. W szczególności uwzględniono architekturę opartą na wspólnej magistrali (tylko w zestawieniu czasów wykonania zadań systemu dla poszczególnych podejść) oraz rozwiązania zwane WKK (Według Kolejności Kolizji) i „LCF+przeszeregowanie". Pierwsze – WKK – stanowi modyfikację metodologii rozprawy, gdzie lista TCL jest posortowana nie w kolejności wielkości kolizji, ale w kolejności ich wystąpienia w trakcie działania aplikacji. „LCF+przeszeregowanie" natomiast w przeciwieństwie do wariantu „LCF+łącza" usuwa kolizje poprzez przeszeregowanie transmisji zamiast generowania dla nich dedykowanych połączeń (rezultatem jest bardziej oszczędne gospodarowanie zasobami systemu).

W Tabeli 7.2-4 zestawiono wartości związane z pogorszeniem czasu wykonania zadań ATG względem rezultatu kosyntezy (system bez odwzorowania w konkretną architekturę). Podejścia WWK i WKK dały te same wyniki, wydłużając działanie aplikacji średnio o ok.1,5% (w dwóch przypadkach wydajność była identyczna z systemem referencyjnym). Nieco lepszy średni wynik – 1% - dała metoda „LCF+łącza". Pozostałe architektury spowolniły pracę odwzorowanej w NoC aplikacji o 8-10%, natomiast rozwiązania oparte na wspólnej magistrali dały – zgodnie z przewidywaniami – najsłabsze rezultaty. Pogorszyły czas działania od niemal 20 do ponad 50%.

Aplikacja	**Wspólna magistrala [%]**	***Network-on-Chip* [%]**				
		WWK, WKK	***mesh* NxN**	***mesh* MxN**	**LCF+ łącza**	**LCF+ przeszeregowanie**
Graf11	20,9	0,43	3,54	8,33	0,43	3,91
Graf12	19,38	0,42	0,42	0,42	0,42	0,42
Graf13	39,53	4,28	15,8	12,58	4,28	9,69
Graf14	30,69	4,33	29,85	27,19	0,05	12,41
Graf15	51,09	0	0	0	0	30,79
Graf16	52,81	0	0	0	0,93	2,8
średnio	*35,73*	*1,58*	*8,27*	*8,09*	*1,02*	*10,01*

Tabela 7.2-4 Pogorszenie czasów wykonania dla różnych architektur NoC względem wyniku kosyntezy – badania poszerzone.

Tabela 7.2-5 zawiera porównanie jakości rozwiązań dla różnych topologii NoC. Kryteria porównania to średnia ważona liczba skoków oraz liczba potrzebnych do budowy danej

topologii łącz jednokierunkowych. Systemem referencyjnym jest proponowany w niniejszej pracy NoC typu WWK. Modyfikacja WKK stanowi na tym polu rozwiązanie tylko nieznacznie gorsze. Trasy komunikatów są tu średnio o ok.3% dłuższe (podobnie jak poprzednio wartość ujemna dla danej pozycji oznacza poprawę względem systemu wzorcowego), zaś topologia wymaga średnio niecałe 2% więcej łącz międzyruterowych. Obie grupy systemów typu *mesh* wypadają znacznie gorzej - pogorszenie jest średnio kilkudziesięcioprocentowe. Ostatnie dwie grupy rozwiązań - LCF - wypadły nieco lepiej, jeśli chodzi o długość wyznaczonych tras (średni zysk w długości tras to około 3,4% i 2,5%). Taki wynik nie zaskakuje, gdyż są to metody mające na celu minimalizację ruchu w mikrosieci. Natomiast zapotrzebowanie na zasoby układu w obu ostatnich podejściach było większe od systemu WWK i wyniosło średnio odpowiednio ok. 22% oraz 15%. W drugim przypadku spadek jakości rozwiązania jest mniejszy, gdyż do usunięcia kolizji metoda korzysta z przeszeregowania zamiast z dedykowanych, dodatkowych połączeń między ruterami.

Aplikacja	**WWK hop_{AVG} (łącza)**	**Pogorszenie arch. NoC: hop_{AVG} (łącza) [%]**				
		WKK	***mesh* NxN**	***mesh* MxN**	**LCF+ łącza**	**LCF+ przeszereg.**
Graf11	2,75 (14)	-2,23 (-7,69)	10,42 (41,67)	30,38 (46,15)	-11,79 (17,65)	-9,56 (12,5)
Graf12	2,44 (17)	5,43 (10,53)	44,8 (64,58)	37,11 (50)	5,43 (26,09)	5,43 (22,73)
Graf13	2,69 (23)	7,88 (-4,55)	39,55 (52,08)	39,41 (42,5)	-8,91 (17,86)	-5,08 (11,54)
Graf14	2,44 (17)	5,79 (-6,25)	46,37 (64,58)	37,76 (50)	-6,55 (32)	-0,41 (22,73)
Graf15	2,11 (22)	1,4 (8,33)	53,42 (54,17)	45,62 (35,29)	2,31 (21,43)	-1,93 (0)
Graf16	2,73 (16)	0,36 (11,11)	18,02 (66,67)	21,33 (38,46)	-0,74 (15,79)	-3,41 (0)
średnio	*2,53 (-)*	*3,11 (1,91)*	*35,43 (57,29)*	*35,27 (43,73)*	*-3,37 (21,8)*	*-2,49 (11,58)*

Tabela 7.2-5 Pogorszenie parametru hop_{AVG} oraz liczby użytych łącz jednokierunkowych w porównaniu z zaprezentowanym podejściem (NoC WWK) - badania poszerzone.

7.3 Przykładowe systemy rzeczywiste

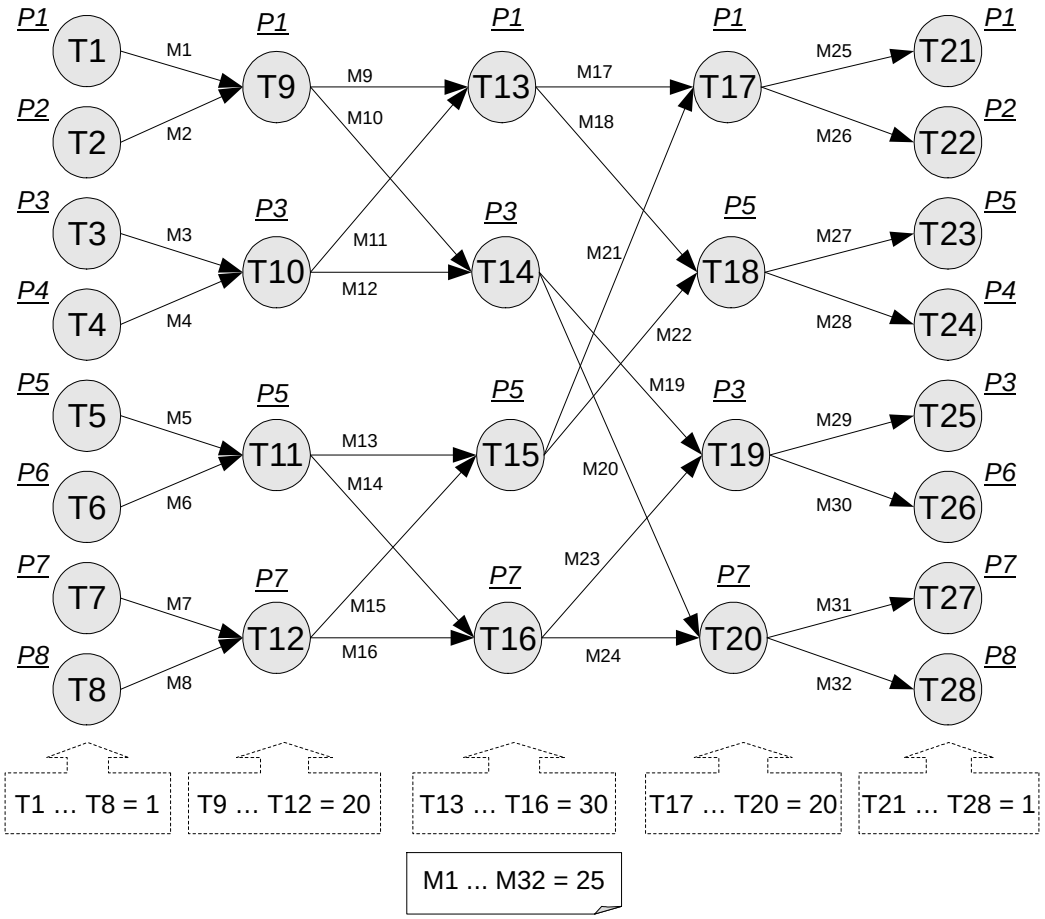
W związku z tym, że wyniki uzyskane na bazie syntetycznych aplikacji nie stanowią podstawy wnioskowania o zastosowaniach praktycznych proponowanej w rozprawie metody, przeprowadzono analogiczne jak w poprzednim rozdziale eksperymenty dla aplikacji rzeczywistych. Do tego celu posłużono się grafami zadań kompresji do formatu graficznego JPEG [VN06], implementacją algorytmu FFT (ang. *Fast Fourier Transform*) [CKTA94] oraz przykładem systemu multimedialnego MMS (ang. *MultiMedia System*) zapisującego/odtwarzającego sygnał audio/wideo w formatach MP3 i H.263 [HM05a].

Aplikacja	**Liczba zadań**	**Liczba transmisji**	**Liczba PE**	**Czas obliczeń/ czas transmisji**	**Czas kolizji/ czas transmisji**
FFT	28	16	8	0,74	1,25
JPEG	8	9	8	0,21	0,45
MMS	40	32	16	0,11	0,39

Tabela 7.3-1 Parametry badanych aplikacji rzeczywistych.

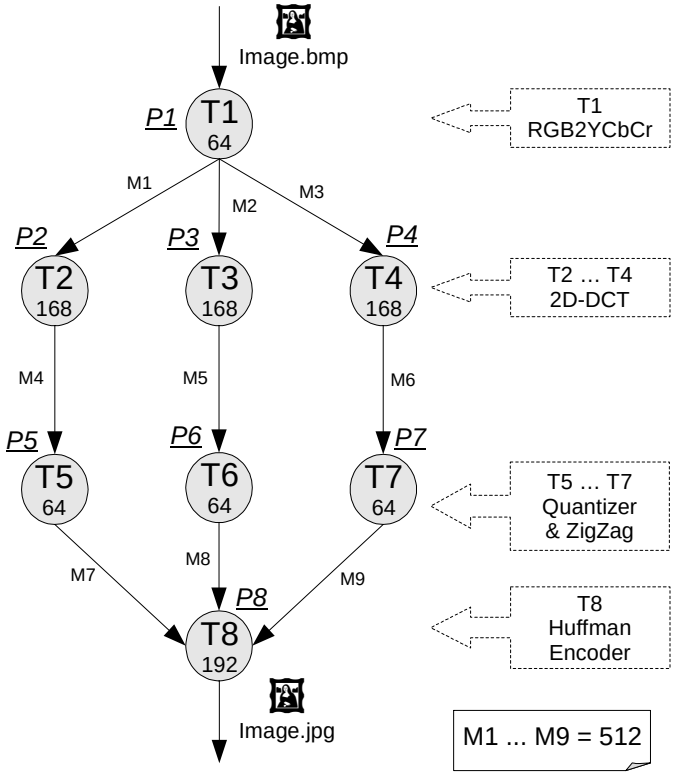
Grafy zadań wyżej wymienionych aplikacji charakteryzuje duża liczba równoległych, niezależnych od siebie ścieżek, co predestynuje te systemy do realizacji w postaci architektur przetwarzających w sposób równoległy. Parametry aplikacji zawarto w Tabeli 7.3-1.

Aplikacja FFT jest często spotykana jako funkcja składowa wielu bardziej złożonych systemów. Niemniej jednak, ze względu na mnogość operacji elementarnych (zadań i transmisji) doskonale nadaje się do testowania wydajności metody prezentowanej w rozprawie. Atrybutowany Graf Zadań dla wyżej wymienionego algorytmu przedstawiono na Rys. 7.3-1. Zarówno ATG jak i parametry czasowe zadań i transmisji zaczerpnięto z [CKTA94].



Rys. 7.3-1 Graf ATG algorytmu FFT (na dole ilustracji czasy zadań i transmisji).

Funkcję kompresji obrazu do formatu JPEG można spotkać w wielu urządzeniach w rodzaju aparatów czy kamer cyfrowych. Jako, że liczba zadań jest relatywnie niewielka, na etapie kosyntezy zdecydowano o alokacji typu jeden element przetwarzający do jednego zadania. Graf ATG dla kodera obrazu metodą JPEG zobrazowano na Rys. 7.3-2. Parametry czasowe zadań i transmisji dla modelu zaczerpnięto z pracy [VN06].

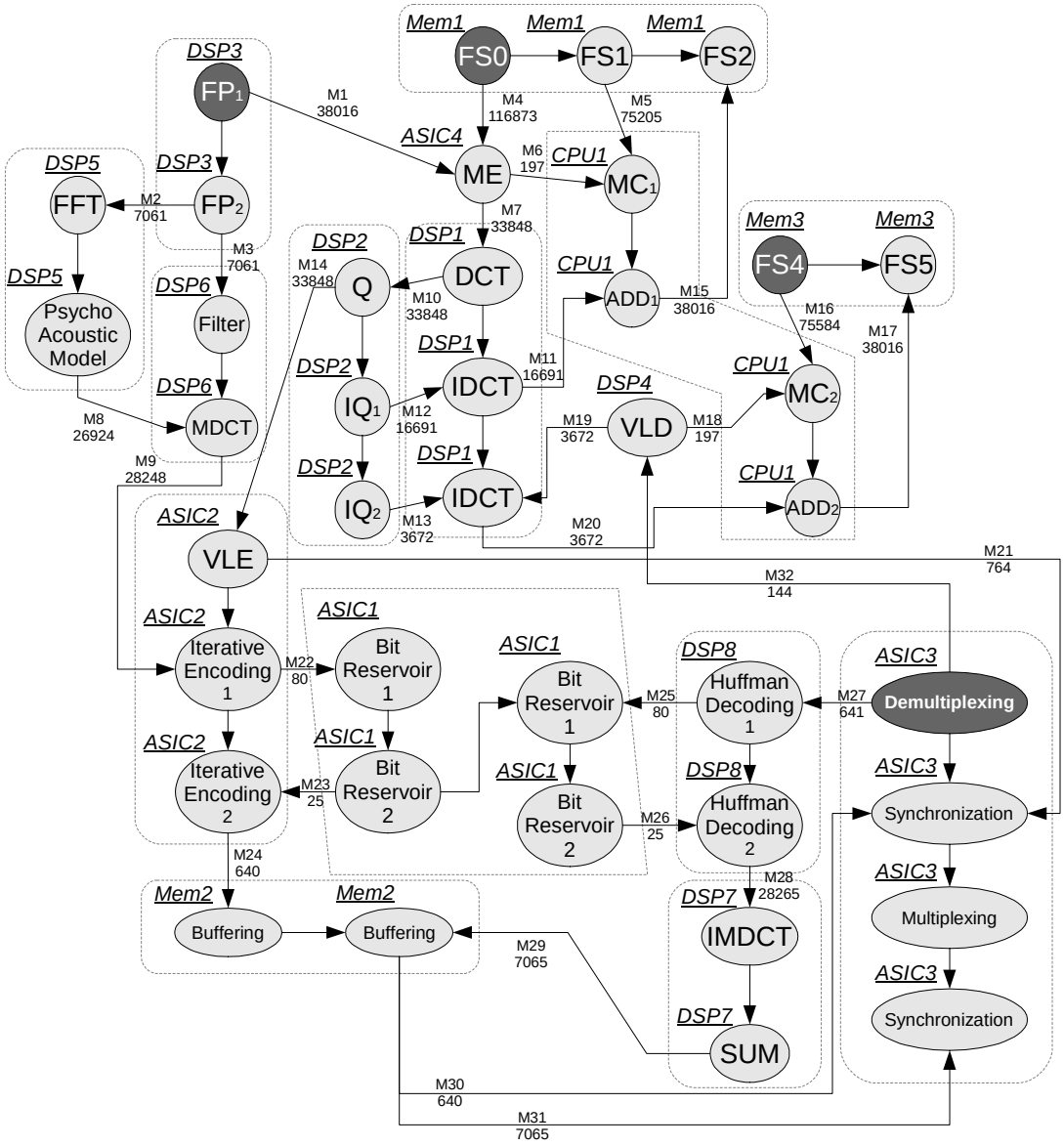


Rys. 7.3-2 Graf ATG kompresji obrazu do formatu JPEG (po prawej stronie rysunku czasy zadań i transmisji).

Najbardziej obszernym przykładem aplikacji rzeczywistej jest system odtwarzania i zapisu danych (MMS) w formatach H.263 (wideo) oraz MP3 (dźwięk). Przykładowo tego typu funkcje realizują nagrywarki DVD. Graf ATG z przydzielonymi do elementów przetwarzających zadaniami zaczerpnięto z pracy [HM05a], natomiast czasy dla węzłów oszacowano na podstawie szeregu prac z dziedziny przetwarzania strumieni audio/wideo. Rezultat zobrazowano na Rys. 7.3-3 (czasy zadań ze względu na czytelność ilustracji pominięto, linią przerywaną obwiedziono zadania przydzielone do tego samego modułu PE).

W Tabeli 7.3-2 zebrano wyniki eksperymentów szacujących czas działania aplikacji odwzorowanych w poszczególne architektury. Dla kompresora JPEG rozwiązania WWK, „LCF+łącza” oraz „LCF+przeszeregowanie” dały tą samą topologię NoC o identycznych parametrach wydajnościowych, energetycznych i zasobowych. Jest to rezultat założeń dla kosyntezy (przydział „jeden PE → jedno zadanie”) i specyfiki grafu (m.in. symetrii ścieżek i jednakowych transmisji). Dla wszystkich systemów rzeczywistych czas działania sieci jednoukładowych wygenerowanych metodą „LCF+łącza” był taki sam jak dla systemu WWK (zerowe pogorszenie), zatem to rozwiązanie pominięto w tabeli. Warto zwrócić uwagę, iż dla aplikacji MMS nie badano wariantu *mesh* MxN. Jest to skutek kosyntezy – zadania systemu multimedialnego alokowano w 16 elementach przetwarzających, dlatego generowanie dla niego topologii prostokątnej kraty byłoby sprzeczne z podejściem najczęściej spotykanym w literaturze

przedmiotu (*mesh* kwadratowy, w tym przypadku 4x4).



Rys. 7.3-3 Graf ATG systemu multimedialnego.

Aplikacja	**Wspólna magistrala [%]**	***Network-on-Chip* [%]**			
		WWK	***mesh* NxN**	***mesh* MxN**	**LCF+ przeszeregowanie**
FFT	57,21	0	12,69	22,52	0
JPEG	37,34	0	12,81	5,22	12,69
MMS	37,63	0	23,07	-	0
średnio	*44,06*	*0*	*16,19*	*13,87*	*4,23*

Tabela 7.3-2 Pogorszenie czasów wykonania dla różnych architektur NoC względem wyniku kosyntezy – aplikacje rzeczywiste.

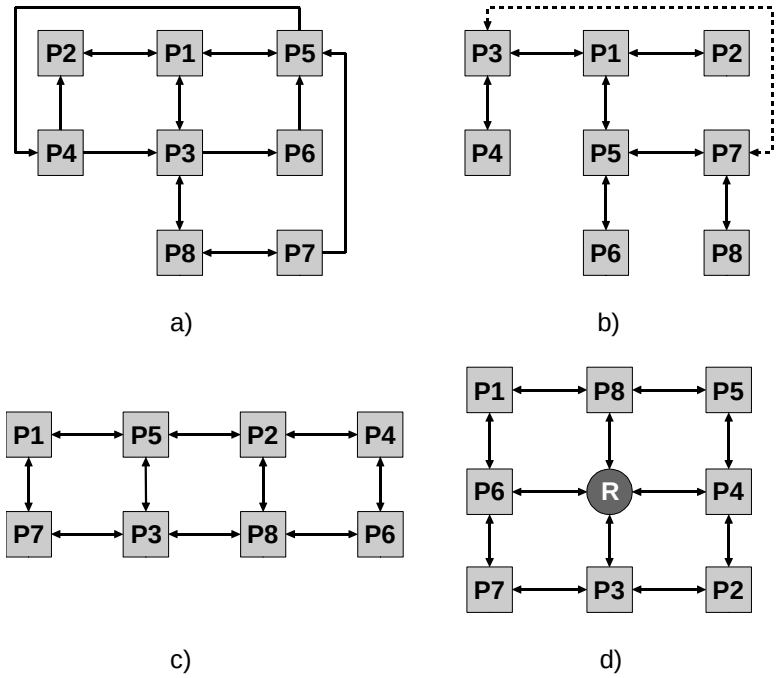
Kolejna tabela - 7.3-3 - zawiera rezultaty badań informujące o stopniu pogorszenia zasobochłonności oraz wydajności czasowo-energetycznej dla badanych architektur. Na tym polu podejście WWK dało najlepsze wyniki przewyższając nawet rozwiązania z rodziny LCF (*mesh* MxN dla aplikacji MMS nie był badany z powodów podanych wcześniej). W jednym

przypadku - „LCF+przeszeregowanie” dla aplikacji FFT – uzyskano rozwiązanie bardziej oszczędne pod względem użytych łącz o ok.14% od metody WWK. Prawdopodobna przyczyna to sposób przydziału zadań do procesorów skutkujący wymianą dużych ilości danych pomiędzy tymi samymi parami modułów PE. Wadą takiego wariantu rozwiązania okazało się natomiast 20% pogorszenie parametru hop_{AVG}.

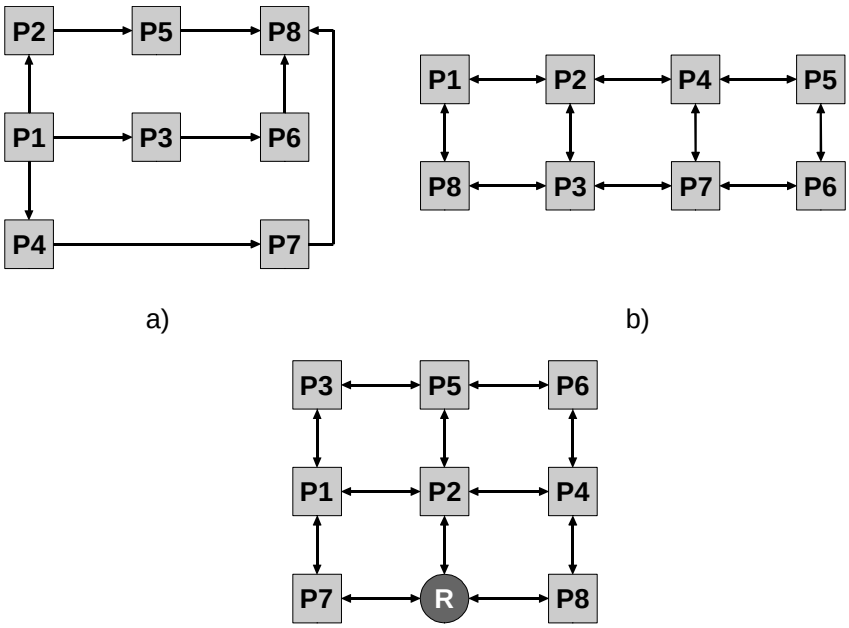
Aplikacja	**WWK** hop_{AVG} **(łącza)**	**Pogorszenie arch. NoC:** hop_{AVG} **(łącza) [%]**			
		***mesh* NxN**	***mesh* MxN**	**LCF+ łącza**	**LCF+ przeszereg.**
FFT	2 (16)	44,9 (33,33)	30,56 (20)	11,11 (0)	20 (-14,29)
JPEG	2 (9)	30,8 (62,5)	35,69 (55)	0 (0)	0 (0)
MMS	2,01 (23)	57,95 (52,08)	- (-)	7,37 (25,81)	7,37 (23,33)
średnio	*2 (-)*	*44,55 (49,31)*	*33,12 (37,5)*	*6,16 (8,6)*	*9,12 (3,02)*

Tabela 7.3-3 Pogorszenie parametru hop_{AVG} oraz liczby użytych łącz jednokierunkowych w porównaniu z zaprezentowanym podejściem (NoC WWK) - aplikacje rzeczywiste.

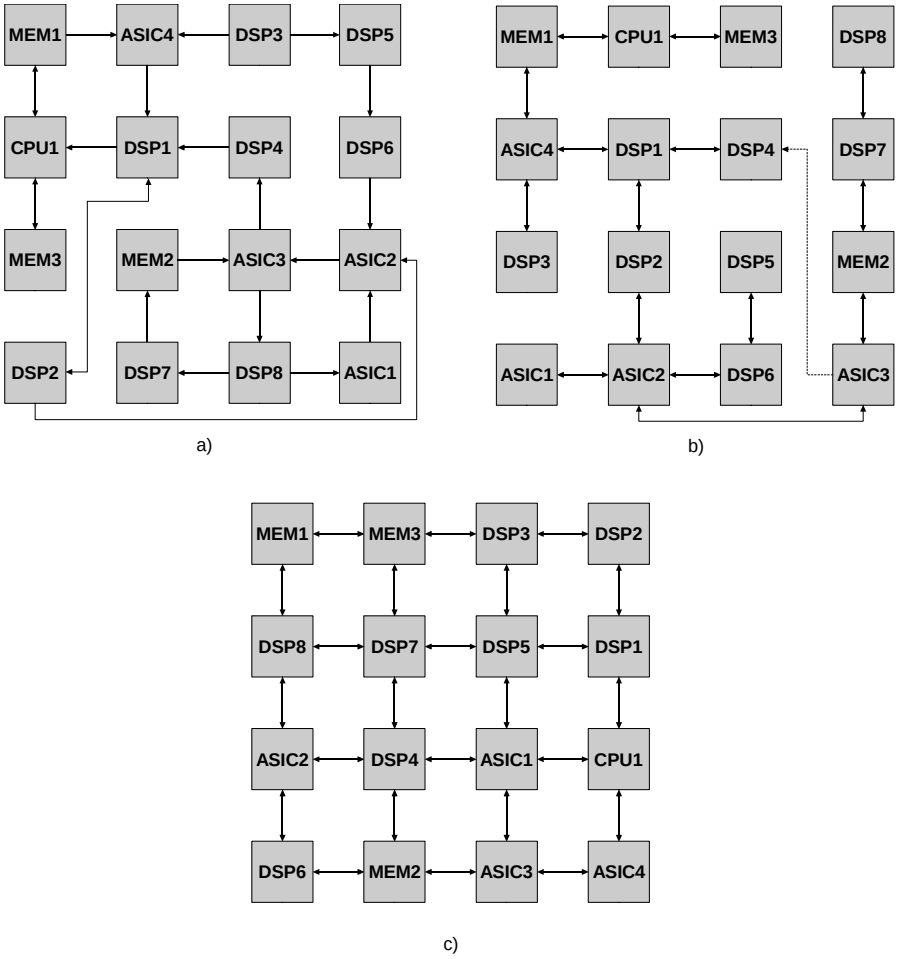
W celu zobrazowania uzyskanych topologii NoC dla badanych systemów realnych na kolejnych trzech ilustracjach – 7.3-4, 7.3-5 oraz 7.3-6 – przedstawiono zsyntezowane mikrosieci (rozplanowanie planarne zostało wykonane metodami ręcznymi). Na pierwszym z wymienionych rysunków zamieszczono wyniki dla algorytmu FFT, na drugim - dla kompresora JPEG, zaś na trzecim – dla systemu multimedialnego. Węzły oznaczone jako „R” w topologiach kraty zawierają jedynie ruter i musiały zostać dodane, aby możliwa była realizacja trasowania metodą XY (pełny *mesh* kwadratowy lub prostokątny). Łącza oznaczone linią przerywaną na Rys. 7.3-4b i 7.3-6b reprezentują dedykowane ścieżki służące do usuwania kolizji w metodzie LCF (w wariancie „LCF+łącza”).



Rys. 7.3-4 Wynikowe topologie dla algorytmu FFT uzyskane metodą: WWK (a), LCF (b), losowo budowanym mesh'em MxN (c) i losowo budowanym mesh'em NxN (d).



Rys. 7.3-5 Wynikowe topologie dla kompresji JPEG uzyskane metodą: WWK/LCF (a), losowo budowanym mesh'em MxN (b) i losowo budowanym mesh'em NxN (c).



Rys. 7.3-6 Wynikowe topologie dla systemu multimedialnego uzyskane metodą: WWK (a), LCF (b), losowo budowanym mesh'em NxN (c).

8 Podsumowanie

W rozprawie zaprezentowano metodologię generowania bezkolizyjnych, dedykowanych sieci jednoukładowych (*Network-on-Chip*) dla systemów wbudowanych. Architektury NoC zdobywają coraz większe uznanie środowisk akademickich oraz przemysłu ze względu na znakomite możliwości konstruowania systemów przetwarzających w sposób równoległy (MPSoC). Przykłady praktycznych implementacji podane w rozdziale 3 uzupełnia fakt prezentacji przez firmę Altera w styczniu 2011 wstępnej wersji narzędzia zwanego Qsys [Qsys] służącego do projektowania w oparciu o układy FPGA i infrastrukturę komunikacyjną bazującą na mikrosieci.

W związku z tym, że funkcjonalność aplikacji jest z góry określona założono, iż specyfikacja wejściowa zadana jest w formie grafu zadań – modelu szczególnie szeroko rozpowszechnionym w pracach z dziedziny kosyntezy. Dzięki temu możliwe było również ścisłe powiązanie etapu budowania architektury komunikacyjnej (NoC) z algorytmem kosyntezy. Czynnikami sterującymi krokami metody, w kolejności ich priorytetów, są:

- bezkolizyjność – wyznaczenie niepokrywających się tras w mikrosieci dla transmisji zachodzących w tych samych przedziałach czasu,
- minimalizacja kosztu zasobowego – architektura NoC powinna funkcjonować w oparciu o jak najmniejszą liczbę połączeń międzyruterowych,
- minimalizacja kosztu energetycznego – wyznaczane są możliwie najkrótsze, bezkolizyjne ścieżki dla komunikatów.

W uzupełnieniu powyższych elementów schematu projektowego, metodologia tworzy architekturę spełniającą - narzucone na czas działania zadań systemu - ograniczenia czasowe. Proponowaną w rozprawie metodę podzielono na trzy etapy postępowania (szczegółową sieć działań przedstawiono w rozdziale 6.3):

1. Analizę grafu zadań pod kątem zależności pomiędzy transmisjami. Bazując na jej wynikach utworzono formalny model komunikacji w systemie, wykorzystywany na następnych etapach metodologii.
2. Budowę dedykowanej topologii dla architektury NoC i rozmieszczenie w niej elementów obliczeniowych. Rezultat poprzedniego etapu to wyznaczenie wszystkich grup transmisji, które zachodzą w tych samych przedziałach czasu. Następnie dla każdej transmisji w ramach jednej grupy alokowano inną ścieżkę w sieci (w przypadku niemożności bezkolizyjnej alokacji dany komunikat był przeszeregowywany).
3. Wyznaczenie tras i szeregowanie transmisji. Trasy dla transmisji z kolizjami otrzymano w wyniku działania metody na etapie 2, dla pozostałych transmisji międzyprocesorowych

wyznaczono trasy najkrótsze. W pracy wykorzystano ruting statyczny preadaptacyjny (trasowanie dostosowane w fazie projektowania), w którym trasy są opisane w nagłówkach pakietów. Dzięki temu możliwe było wyznaczenie różnych tras dla różnych transmisji pomiędzy tymi samymi elementami obliczeniowymi. Taka metoda rutingu umożliwiła również zastosowanie prostych ruterów, co pozwoliło obniżyć koszt implementacji systemu.

W celu potwierdzenia skuteczności metody dokonano jej analizy teoretycznej, a ponadto na drodze eksperymentalnej dowiedziono jej przewagi nad typowymi podejściami prezentowanymi aktualnie w literaturze przedmiotu. Należy przy tym pamiętać, iż w eksperymentach zrezygnowano z etapu rafinacji tras ze względu na ujednolicenie kryteriów porównań z innymi metodologiami (losowo wygenerowane topologie *mesh*). Powoduje to, że osiągnięty wynik dla parametru hop_{AVG} mógłby być dodatkowo poprawiony po wykonaniu Algorytmu 6.

Główne oryginalne osiągnięcia autora rozprawy to:

- Opracowanie modelu matematycznego dla potrzeb odwzorowania systemu wbudowanego w bezkolizyjną sieć jednoukładową.
- Stworzenie metody całkowicie eliminującej konflikty transmisyjne.
- Projekt bardzo prostej i efektywnej konstrukcji rutera (ruting deterministyczny z buforem jednoflitowym, bez kanałów wirtualnych, z metodą przełączania pakietów typu *wormhole*).
- Algorytm generowania topologii NoC dla danego uszeregowania systemu, spełniającej ograniczenia szybkości pracy systemu i zrównoważonej pod kątem kosztu (liczby łącz międzyruterowych) oraz pod kątem energetycznym (długości tras komunikatów).
- Powiązanie metod kosyntezy z tworzeniem architektury komunikacyjnej dla rafinacji rozwiązania.

Tematyka syntezy architektur NoC jest relatywnie nowa i stale intensywnie badana. W związku z powyższym przyszłe prace autora rozprawy obejmują:

- opracowanie metody rozplanowania planarnego dla grafu NTG (prace w toku),
- badania prototypów wygenerowanych proponowaną metodologią i zaimplementowanych w układach FPGA (prace w toku),
- rozszerzenie metody na systemy o mniej przewidywalnym wzorcu komunikacyjnym (m.in. reprezentowanymi przez warunkowy graf zadań) przy wykorzystaniu modelu rutera z kanałami wirtualnymi.

9 Wykaz skrótów i symboli

2D	dwuwymiarowy (planarny)
3D	trójwymiarowy (przestrzenny)
ACG, CG	graf charakteryzujący system, ang. *Application Characterization Graph, Core Graph*
ASAP	metoda szeregowania zadań i transmisji, bezzwłoczna, ang. *As Soon As Possible*
ATG	Atrybutowany Graf Zadań, ang. *Attributed/Annotated Task Graph*
$\mathbf{C_x}$	zbiór transmisji potencjalnie kolizyjnych
$c(M_{i,j})$	koszt transmisji $M_{i,j}$ powiązany z jej wolumenem
DSM	technologie submikronowe, ang. *Deep SubMicron*
$d_{WE}(T_i)/d_{WY}(T_i)$	stopień wejściowy/wyjściowy węzła T_i
$E(l)$	moc pobierana przez łącza międzyruterowe
$E(p)$	moc pobierana przez elementy przetwarzające mikrosieci
$E(r)$	moc pobierana przez rutery
$E_{B(bit)}$	koszt energetyczny związany z buforowaniem w ruterze
E_{bit}	koszt energetyczny przesłania jednego bitu przez mikrosieć
$E_j(l)$	koszt pojedynczej jednokierunkowej magistrali łączącej dwa rutery
E_l	koszt energetyczny przesłania flitów przez łącza między portami sąsiadujących ruterów oraz z/do modułów NI
$E_{L(bit)}$	moc pobierana przez łącza międzyruterowe
E_M	moc pobierana w trakcie transmisji wiadomości M
E_{NoC}	moc pobierana przez system zrealizowany w postaci sieci jednoukładowej
E_r	moc pobierana przez ruter podczas transmitowania flita (przełączenie flita nagłówkowego – zestawienie toru transmisji)
$E_{S(bit)}$	koszt energetyczny związany z działaniem struktury przełączającej wewnątrz rutera
$E_{W(bit)}$	energia pobierana przez wewnętrzne połączenia rutera
flit	jednostka przepływu danych, ang. *Flow Control Unit*
GALS	układy asynchroniczne z podziałem na synchronicznie funkcjonujące partycje, ang. *Globally Asynchronous Locally Synchronous*
$h(M_{i,j})$	liczba ruterów na trasie komunikatu $M_{i,j}$, ang. *hop*
h_{AVG}	średnia ważona liczba skoków
IP	specjalizowany funkcjonalnie moduł przetwarzający dane, ang. *Intellectual Property*
ITL	Lista Transmisji Międzyprocesorowych, ang. *Interprocessor Transmission List*
L	zbiór wszystkich łącz w NTG

LCF	bazująca na ilości transmitowanych informacji metoda odwzorowania aplikacji w NoC, ang. *Largest Communication First*
l_i	łącze międzyruterowe
LP	port lokalny, którym ruter jest połączony z modułem NI, ang. *Local Port*
M	zbiór wszystkich transmisji w grafie zadań
$M_{i,j}$	transmisja pomiędzy zadaniami *i* oraz *j* aplikacji (krawędź w grafie zadań)
MPSoC	wieloprocesorowy system jednoukładowy, ang. *MultiProcessor System on Chip*
N	zbiór wszystkich węzłów n_i (lub NN) w NTG
NI	interfejs sieciowy modułu PE, ang. *Network Interface*
n_i, NN	węzeł sieci jednoukładowej (para moduł PE-ruter), ang. *Network Node*
n_{ITL}	pojedynczy element listy ITL
NoC	sieć jednoukładowa, ang. *Network-on-Chip*
NTG	graf topologii sieci jednoukładowej, ang. *Network Topology Graph*
$NTG_{CollFree-min}$	bezkolizyjny graf NTG o minimalnym koszcie topologii
NTG_{min}	graf NTG o minimalnym koszcie topologii
$n_{WE}(r_i)/n_{WY}(r_i)$	liczba portów wejściowych/wyjściowych rutera r_i
P	zbiór wszystkich procesorów tworzących daną architekturę NoC
PE	element przetwarzający, ang. *Processing Element*
P_i	element przetwarzający, procesor
predT(T_i)/**predM**(T_i)	zbiór zadań-przodków/transmisji-przodków zadania T_i w grafie zadań
R	zbiór wszystkich tras w NTG
$R_{k,l}(M_{i,j})$	trasa dla komunikatu $M_{i,j}$ między procesorami P_k i P_l
RT	zbiór wszystkich ruterów w danym systemie NoC
S	zbiór przedziałów czasu, w trakcie których wykonywane są zadania i transmisje systemu po uszeregowaniu
SoC	system jednoukładowy, ang. *System on Chip*
succT(T_i)/**succM**(T_i)	zbiór zadań-potomków/transmisji-potomków zadania T_i w grafie zadań
T	zbiór wszystkich zadań w grafie zadań
$t_{CRIT}(T_i)$	ograniczenie czasu zakończenia wykonania dla zadania T_i
TCL	Lista Kolizji Transmisyjnych, ang. *Transmission Collision List*
t_{COLL}	czas trwania pojedynczej kolizji na liście TCL
TDM	koncepcja współużytkowania połączeń w czasie, ang. *Time Division Multiplexing*
TG	graf zadań, ang. *Task Graph*
T_i	zadanie aplikacji (węzeł grafu zadań)
t_l	czas transmisji jednego flita łączem międzyruterowym

t_M	czas transmisji komunikatu M
t_r	czas transmisji jednego flita przez ruter (przełączenie)
t_{SLACK}	dla zadania: różnica pomiędzy ograniczeniem czasowym a rzeczywistym czasem zakończenia wykonywania zadania, tzw. zapas czasowy dla transmisji: zapas czasowy pomiędzy końcem transmisji a rozpoczęciem zadania otrzymującego dane od tej transmisji
$t_{SLACK'}$	wynikająca z kolizji transmisyjnej wartość, o jaką może nastąpić ewentualne pogorszenie czasu zakończenia zadania
$t_{START}(M_{i,j})/t_{STOP}(M_{i,j})$	czas rozpoczęcia/zakończenia realizowania transmisji $M_{i,j}$
$t_{START}(T_i)/t_{STOP}(T_i)$	czas rozpoczęcia/zakończenia wykonywania zadania T_i
WKK	sposób generowania topologii NoC, Według Kolejności Kolizji
WWK	sposób generowania topologii NoC, Według Wielkości Kolizji
VC	kanał wirtualny, ang. *Virtual Channel*
γ	funkcja generowania topologii (grafu NTG), tworząca zbiór **L**
π	funkcja przyporządkowująca zadania ze zbioru **T** do procesorów ze zbioru **P**
ρ	funkcja szukająca tras w grafie NTG dla transmisji ze zbioru **M**
σ	funkcja szeregująca zadania i transmisje, przyporządkowująca im przedziały czasów, w których są wykonywane
$\tau(M_{i,j})$	czas przesyłania komunikatu $M_{i,j}$ (graf ATG)
$\omega(T_i)$	czas wykonania zadania T_i (graf ATG)

10 Wykaz ilustracji

11 Wykaz tabel

12 Bibliografia

[AAMPB08] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, G. De Micheli, "Network-on-Chip design and synthesis outlook", INTEGRATION - the VLSI journal, 41(3), pp.340–359, 2008

[ACP06] G. Ascia, V. Catania, M. Palesi, "A Multi-objective Genetic Approach to Mapping Problem on Network-on-Chip", Journal of Universal Computer Science, 12(4), pp.370-394, 2006

[ACPP08] G. Ascia, V. Catania, M. Palesi, D. Patti, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip", IEEE Transactions on Computers, 57(6), pp.809-820, 2008

[AEJKR02] A. Allan, D. Edenfeld, J.W. Joyner, A.B. Kahng, M. Rodgers, Y. Zorian, "2001 technology roadmap for semiconductors", IEEE Computer, 35(1), pp.42–53, 2002

[AHKB00] V. Agarwal, M.S. Hrishikesh, S.W. Keckler, D. Burger, "Clock rate versus IPC: the end of the road for conventional microarchitectures", Proceedings of the 27th Annual International Symposium on Computer Architecture, pp.248–259, 2000

[Alt] Altera Corp., Katalogi firmowe, http://www.altera.com

[AMBA] ARM Ltd. The Advanced Microcontroller Bus Architecture (AMBA), http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

[AMCBR06] F. Angiolini, P. Meloni, S. Carta, L. Benini, L. Raffo, "Contrasting a NoC and a traditional interconnect fabric with layout awareness", Proceedings of the Conference on Design, Automation and Test in Europe, pp.124–129, 2006

[B05] S. Bhat, "Energy models for networks-on-chip components", Master’s thesis, Technische University Eindhoven, Netherlands, 2005

[BC06] L. Bononi, N. Concer, "Simulation and Analysis of Network on Chip Architectures: Ring, Spidergon and 2D Mesh", Proceedings of Design, Automation and Test in Europe, pp.154-159, 2006

[BCGK04] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, "QNoC: QoS architecture and design process for network on chip", Journal of Systems Architecture, 50(2-3, Special Issue on Network on Chip), pp.105-128, 2004

[BCGK07] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, "Routing Table Minimization for Irregular Mesh NoCs", Proceedings of the conference on Design, Automation and Test in Europe, pp.942-947, 2007

[BD06] J. Balfour, W.J. Dally, "Design Tradeoffs for Tiled CMP On-Chip Networks", Proceedings of the 20th annual international conference on Supercomputing, pp.187-198, 2006

[BE06] A. Bouhraoua, M.E. Elrabaa, "An Efficient Network-on-Chip Architecture Based on the Fat-Tree (FT) Topology", Proceedings of the International Conference on Microelectronics, pp.28–31, 2006

[BLB07] J.H. Bahn, S.E. Lee, N. Bagherzadeh, "On Design and Analysis of a Feasible Network-on-Chip (NoC) Architecture", Proceedings of the International Conference on Information Technology, pp.1033-1038, 2007

[BLMNB07] K. De Bosschere, W. Luk, X. Martorell, N. Navarro, M. O’Boyle, D. Pnevmatikatos, A.Ramirez, P. Sainrat, A. Seznec, P. Stenstrom, O. Temam, "High-Performance Embedded Architecture and Compilation Roadmap",

Transactions on HiPEAC I, LNCS 4050, pp.5–29, 2007

[BM02] L. Benini, G. De Micheli, "Networks on chips: A new SoC paradigm", IEEE Computer, 35(1), pp.70–78, 2002

[BM03] P. Bhojwani, R. Mahapatra, "Interfacing cores with on-chip packet-switched networks", Proceedings of 16th International IEEE Conference on VLSI Design, pp.382–387, 2003

[BM06] T. Bjerregaard, S. Mahadevan, "A survey of research and pratices of network-on-chip", ACM Computing Surveys, 38(1), pp.71-121, 2006

[BMNMV05] T.A. Bartic, J.Y. Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, R. Lauwereins, "Topology adaptive network-on-chip design and implementation", IEEE Proceedings on Computer Digitital Technologies, 152(4), pp.467-472, 2005

[BTBTT08] R. Ben-Tekaya, A. Baganne, K. Torki, R. Tourki, "Design and Implementation of MIC@R Router for On-Chip Networks", International Design and Test Workshop, pp.17-21, 2008

[BWMMS09] A. Banerjee, P.T. Wolkotte, R.D. Mullins, S.W. Moore, G.J.M. Smit, "An Energy and Performance Exploration of Network-on-Chip Architectures", IEEE Transactions on Very Large Scale Iintegration (VLSI) Systems, 17(3), pp.319-329, 2009

[C00] G.M. Chiu, "The odd-even turn model for adaptive routing", IEEE Transactions on Parallel Distributed Systems, 11(7), pp.729–738, 2000

[C06] T. Claasen, "An industry perspective on current and future state-of-the-art in system-on-chip (SoC) technology", Proceedings of the IEEE, 94(6), pp.1121–1137, 2006

[C07] J. Chan, "Energy-aware Synthesis for Networks on Chip Architectures", PhD dissertation, University of New South Wales, Australia, 2007

[CCM07] E. Carvalho, N. Calazans, F. Moraes, "Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs", Proceedings of the 18th IEEE/IFIP International Workshop on Rapid System Prototyping, pp.34-40, 2007

[CH07] P. Conway, B. Hughes, "The AMD Opteron northbridge architecture", IEEE Micro, 27(2), pp.10–21, 2007

[CKTA94] C.L. McCreary, A.A. Khan, J.J. Thompson, M.E. McArdle, "A Comparison of Heuristics for Scheduling DAGS on Multiprocessors", Proceedings of Eighth International Parallel Processing Symposium, pp.446-451, 1994

[CKTA94] C.L. McCreary, A.A. Khan, J.J. Thompson, M.E. McArdle, "A Comparison of Heuristics for Scheduling DAGS on Multiprocessors", Proceedings of Eighth International Parallel Processing Symposium, pp.446-451, 1994

[CLMPS04] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, A. Scandurra, "Spidergon: A Novel On-Chip Communication Network", Proceedings of International Symposium on System-on-Chip, pp.15-18, 2004

[CLR98] T.H. Cormen, C.E. Leiserson, R.L. Rivest, „Wprowadzenie do algorytmów”, Wydawnictwa Naukowo-Techniczne, Warszawa, 1998

[CM08] C.L. Chou, R. Marculescu, "Contention-aware application mapping for Network-on-Chip communication architectures", Proceedings of International Conference on Computer Design, pp.164-169, 2008

[CP08] J. Chan, S. Parameswaran, "NoCOUT: NoC Topology Generation with Mixed Packet-Switched and Point-to-Point Network", Proceedings of the Asia and South Pacific Design Automation Conference, pp.265–270, 2008

[CYC09] Y.J Chen, C.L. Yang, Y.S. Chang, "An architectural co-synthesis algorithm for energy-aware Network-on-Chip design", Journal of Systems Architecture, 55, pp.299-309, 2009

[D02] R.P. Dick, "Multiobjective Synthesis of Low-power Real-time Distributed Embedded Systems", PhD dissertation, Princeton University, USA, 2002

[D04] S. Deniziak, "Cost-efficient synthesis of multiprocessor heterogeneous systems", Control and Cybernetics, Vol.33, No. 2, pp.341-355, 2004

[DBGBB03] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, L. Benini, "Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs", Proceedings of the 21st International Conference on Computer Design, pp.536-539, 2003

[DG08] S. Deniziak, A. Górski, "Hardware/Software Co-Synthesis of Distributed Embedded Systems Using Genetic Programming", Lecture Notes in Computer Science, vol.5216, Springer-Verlag, pp.83-93, 2008

[DRW98] R. P. Dick, D. L. Rhodes , W. Wolf, "TGFF: task graphs for free", Proceedings of the 6th international workshop on Hardware/software codesign, pp.97-101, 1998

[DT01] W.J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", Proceedings of the 38th Conference on Design Automation, pp.684–689, 2001

[DT08a] S. Deniziak, R. Tomaszewski, "Rapid Prototyping of NoC Architectures from a SystemC Specification", Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp.104-109, 2008

[DT08b] S. Deniziak, R. Tomaszewski, "The Environment for Mapping SystemC Multi-module Specifications onto NoC Architectures", The Open Cybernetics and Systemics Journal, vol.2, pp.142-152, 2008

[DT08c] S. Deniziak, R. Tomaszewski, "Adaptive Routing Protocols Validation in NoC Systems via Rapid Prototyping", Proceedings of the IEEE Human System Interaction, pp.115-120, 2008

[EPPD00] P. Eles, Z. Peng, P. Pop, A. Doboli, "Scheduling with bus access optimization for distributed embedded systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8(5), pp.472-491, 2000

[FP07] B. Feero and P. Pande, "Performance evaluation for three-dimensional networks-on-chip", Proceedings of the IEEE Computer Society Annual Symposium on VLSI, pp.305–310, 2007

[GCK07] R. Gindin, I. Cidon, I. Keidar, "NoC-Based FPGA: Architecture and Routing", Proceedings of the First International Symposium on Networks-on-Chip, pp.253-264, 2007

[GDMPR03] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavko, A. Radulescu, E. Rijpkema, E. Waterlander, P. Wielage, "Networks on chip", rozdz. "Guaranteeing the quality of services in networks on chip", Kluwer Academic Publishers, pp.61–82, 2003

[GDR05] K. Goossens, J. Dielissen, A. Radulescu, "Aethereal Network on Chip: Concepts,

Architectures, and Implementations", IEEE Design and Test, 22(5), pp.414–421, 2005

[GG00] P. Guerrier. A. Greiner, "A generic architecture for on-chip packet-switched interconnections", Proceedings of the conference on Design, Automation and Test in Europe, pp.250–256, 2000

[GGLD08] C. Gomez, M.E. Gomez, P. Lopez, J. Duato, "An Efficient Switching Technique for NoCs with Reduced Buffer Requirements", Proceedings of International Conference on Parallel and Distributed Systems, pp.713-720, 2008

[GIPJS07] C. Grecu, A. Ivanov, P. Pande, A. Jantsch, E. Salminen, U. Ogras, R. Marculescu, "Towards open network-on-chip benchmarks", Proceedings of First International Symposium on Networks-on-Chip, pp.205, 2007

[GJ79] M.R. Garey, D.S. Johnson, „Computers and Interactability: a Guide to the Theory of NP-Completeness”, W.H. Freeman and Co., New York, USA, 1979

[GN92] C.J. Glass, L.M. Ni, "The turn model for adaptive routing", Proceedings of International Symposium on Computer Architecture, pp.278–287, 1992

[GY93] A. Gerasoulis, T. Yang, "On the granularity and clustering of directed acyclic task graphs", IEEE Transactions on Parallel and Distributed Systems, 4(6), pp.686-701, 1993

[HCG07] A. Hansson, M. Coenen, K. Goossens, "Channel Trees: Reducing Latency by Sharing Time Slots in Time-Multiplexed Networks on Chip", Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis, pp.149-154, 2007

[HH06] P.T. Huang, W. Hwang, "2-Level FIFO Architecture Design for Switch Fabrics in Network-on-Chip", Proceedings of IEEE International Symposium on Circuits and Systems, 2006

[HJKPO00] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, D. Lindqvist, "Network on chip: An architecture for billion transistor era", Proceeding of the IEEE NorChip Conference, pp.166-173, 2000

[HM03a] J. Hu, R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints", Proceedings of the 2003 Asia and South Pacific Design Automation Conference, pp.233-239, 2003

[HM03b] J. Hu, R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures", Proceedings of the conference on Design, Automation and Test in Europe, pp.688-693, 2003

[HM04a] J. Hu, R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design", Proceedings of IEEE/ACM International Conference on Computer-Aided Design, pp.354–361, 2004

[HM04b] J. Hu, R. Marculescu, "DyAD—Smart Routing for Networkson-Chip", Proceedings of the Design Automation Conference, pp.260-263, 2004

[HM05a] J. Hu, R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 24(4), pp.551-562, 2005

[HM05b] J. Hu, R. Marculescu, "Communication and Task Scheduling of Application-Specific Networks-on-Chip", IEEE proceedings of Computers and Digital Techniques, pp.643-651, 2005

[HMH01] R. Ho, K. Mai, M. Horowitz, "The future of wires", Proceedings of the IEEE, 89(4), pp. 490–504, 2001

[HP06] W.H. Ho, T.M. Pinkston, "A design methodology for efficient application-specific on-chip interconnects", IEEE Transactions on Parallel and Distributed Systems, 17(2), pp.174-190, 2006

[JMBM04] A. Jalabert, S. Murali, L. Benini, G. De Micheli, "×pipescompiler: A tool for instantiating application specific networks on chip", Proceedings of the conference on Design, Automation and Test in Europe - Volume 2, pp.884–889, 2004

[JP10] W. Jang, D.Z. Pan, "A3MAP: Architecture-Aware Analytic Mapping for Networks-on-Chip", Proceedings of Asian and South Pacific Design Automation Conference, pp.523-528, 2010

[JS07] R.K. Jena, G.K. Sharma, "A Multi-Objective Evolutionary Algorithm Based Optimization Model for Network-on-Chip Synthesis", Proceedings of the International Conference on Information Technology, pp.977-982, 2007

[JZH06] D. N. Jayasimha, B. Zafar, Y. Hoskote, "On-Chip Interconnection Networks: Why They are Different and How to Compare Them", Intel Technical Report, http://blogs.intel.com/research/terascale/ODI_why-different.pdf, 2006

[KA98] Y.K. Kwok, I. Ahmad, "Benchmarking the Task Graph Scheduling Algorithms", Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium, pp.531-537, 1998

[KCJ94] A.A. Khan, C.L. McCreary, M.S. Jones, "A Comparison of Multiprocessor Scheduling Heuristics", Proceedings of the 1994 International Conference on Parallel Processing - Volume 02, pp.243-250, 1994

[KJSFM02] S. Kumar, A. Jantsch, J.P. Soininen, M. Forsell, M. Millberg, J. Oberg, A. Tiensyrja, A. Hemani, "A network-on-chip architecture and design methodology", Proceedings of the Computer Society Annual Symposium on VLSI, pp.117–124, 2002

[KLPS09] A.B. Kahng, B.Li, L.S. Peh. K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration", Proceedings of the Conference on Design, Automation and Test in Europe, pp.423-428, 2009

[KND02] F. Karim, A. Nguyen, S. Dey, "An interconnect architecture for networking systems on chips", IEEE Micro, 22(5), pp.36–45, 2002

[KNP06] J. Kim, C. Nicopoulos, D. Park, "A gracefully degrading and energy efficient modular router architecture for on-chip networks", Proceedings of the IEEE International Symposium on Computer Architecture, pp.4–15, 2006

[KPTVD05] J.Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C.R. Das, "A Low Latency Router Supporting Adaptivity for On-Chip Interconnects", Proceedings of Design Automation Conference, pp.559-564, 2005

[KS03] N. Kavaldjiev, G.J. Smit, "A survey of efficient on-chip communications for SoC", Proceedings of 4th PROGRESS Symposium on Embedded Systems, pp.129–140, 2003

[KSJ04] N. Kavaldjiev, G.J. Smit, P.G. Jansen, "A Virtual Channel Router for On-chip Networks", Proceedings of IEEE International SoC Conference, pp.289-293, 2004

[LC09] G. Leary, K.S. Chatha, "Automated technique for design of NoC with minimal communication latency", Proceedings of CODES+ISSS, pp.471-480, 2009

[LCOM07] H. Lee, N. Chang, U. Ogras, R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches", ACM Transactions on Design Automation of Electronic Systems, 12(3), pp. 1–20, 2007

[LGMGG09] D. Ludovici, F. Gilabert, S. Medardoni, C. Gomez, M.E. Gomez, P. Lopez, G.N. Gaydadjiev, D. Bertozzi, "Assessing Fat-Tree Topologies for Regular Network-on-Chip Design under Nanoscale Technology Constraints", Proceedings of Design, Automation and Test in Europe, pp.562-565, 2009

[LK03] T. Lei, S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture", Proceedings of the Euromicro Symposium on Digital Systems Design, pp.180-187, 2003

[LLSY05] S.J. Lee, K. Lee, S.J. Song, and H.J. Yoo, "Packet-Switched On-Chip Interconnection Network for System-On-Chip Applications", in IEEE Trans. on Circuits and Systems—II: EXPRESS BRIEFS, vol. 52, no. 6, pp.308-312, 2005

[LLY06] K. Lee, S.J. Lee, H.J. Yoo, "Low-power network-on-chip for highperformance SoC design," IEEE Transactions on Very Large Scale Integration Systems, 14(2), pp.148–160, 2006

[LR04] K. Lahiri, A. Raghunathan, "Power analysis of system-level on-chip communication architectures", Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, pp.236–241, 2004

[LT06] L.F. Leung, C.Y. Tsui, "Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands", Proceedings of the 44th annual Design Automation Conference, pp.128-131, 2007

[LT06] L.F. Leung, C.Y. Tsui, "Optimal Link Scheduling on Improving Best-Effort and Guaranteed Services Performance in Network-on-Chip Systems", Proceedings of the 43rd annual Design Automation Conference, pp.833-838, 2006

[LTH07] Y. Liu, Y. Tan, J. Hang, "Key Problems on Network-on-Chip", 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, pp.549-552, 2007

[LZJ06] Z. Lu, M. Zhong, A. Jantsch, "Evaluation of on-chip networks using deflection routing", Proceedings of the 16th ACM Great Lakes Symposium on VLSI, pp.296-301, 2006

[Mblaze] Micro Blaze processor specification, http://www.xilinx.com/tools/microblaze.htm

[MBM07] S. Murali, L. Benini, G. De Micheli, "An Application-Specic Design Methodology for On-Chip Crossbar Generation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(7), pp.1283-1296, 2007

[MBSCW05] C. Marcon, A. Borin, A. Susin, L. Carro, F. Wagner, "Time and Energy Efficient Mapping of Embedded Applications onto NoCs", Proceedings of the 2005 Asia and South Pacific Design Automation Conference, pp.33-38, 2005

[MCMMO04] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip", Integration, the VLSI Journal, 38(1), pp.69–93, 2004

[MCMSR05] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, F. Hessel, "Exploring NoC Mapping Strategies: An Energy and Timing Aware Technique", Proceedings of the Design, Automation and Test in Europe, pp.502-507, 2005

[MEKG08] A.A. Morgan, H. Elmiligi, M.W. El-Kharashi, F. Gebali, "Networks-on-Chip topology generation techniques: Area and delay evaluation", Proceedings of 3rd International Design and Test Workshop, pp.33-38, 2008

[Men] Nucleus OS specification, http://www.mentor.com/embedded-software/nucleus/

[MEP06] S. Manolache, P. Eles, Z. Peng, "Buffer space optimisation with communication synthesis and traffic shaping for NoCs", Proceedings of the Design Automation and Test in Europe Conference, pp.1-6, 2006

[MHKEO99] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, H. Tenhunen, "Globally asynchronous locally synchronous architecture for large high-performance ASICs", Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2, pp.512–515, 1999

[Mic] MicroC OS III specification, http://micrium.com/page/products/rtos/os-iii

[MKSC05] C. Marcon, M. Kreutz, A. Susin, N. Calazans, "Models for Embedded Application Mapping onto NoCs: Timing Analysis", Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping, pp.17-23, 2005

[MM04] S. Murali, G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures", Proceedings of the Design Automation and Test in Europe Conference, pp. 896-901, 2004

[MM09] T. Moscibroda, O. Mutlu, "A Case for Bufferless Routing in On-Chip Networks", Proceedings of International Symposium on Computer Architecture, pp.196-207, 2009

[MMAAC06] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, L. Raffo, "Designing Application-Specific Networks on Chips with Floorplan Information", Proceedings of IEEE/ACM International Conference on Computer-Aided Design, pp.355-362, 2006

[MMCCR06] P. Meloni, S. Murali, S. Carta, M. Camplani, L. Raffo, G. De Micheli, "Routing aware switch hardware customization for networks on chips", Proceedings of the 1st International Conference on Nano-Networks and Workshops, pp.1–5, 2006

[MMCM07] C.A.M. Marcon, E.I. Moreno, N.L.V. Calazans, F.G. Moraes, "Evaluation of Algorithms for Low Energy Mapping onto NoCs", pp.389-392, 2007

[MNTJ04] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip", Proceedings of Design Automation and Test in Europe Conference, 2, pp.890-895, 2004

[MNTKJ04] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, "The Nostrum backbone – a communication protocol stack for networks on chip", Proceedings of the 17th International Conference on VLSI Design, pp.693-696, 2004

[MOPJH09] R. Marculescu, U.Y. Ogras, L.S. Peh, N. Enright Jerger, Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives", IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 28(1), pp.3-21, 2009

[MPSV06] M. Monchiero, G. Palermo, C. Silvano, O. Villa, "Exploration of Distributed

Shared Memory Architectures for NoC-based Multiprocessors", Journal of Systems Architecture: the EUROMICRO Journal, 53(10), pp.719-732, 2007

[MSCL06] T.S.T. Mak, P. Sedcole, P.Y.K. Cheung, W. Luk, "On-FPGA communication architectures and design factors", Proceedings of the International Conference on Field-Programmable Logic and Its Applications, pp.1-8, 2006

[MTCM05] A. Mello, L. Tedesco, N. Calazans, F. Moraes, "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC", Proceedings of 18th Symposium on Integrated Circuits and Systems Design, pp.178-183, 2005

[NC05] V.-D. Ngo, H.-W. Choi, "An optimum mapping of IPs for On-Chip Network design based on the minimum latency constraint", Proceedings of IEEE Region 10 TENCON Conference, pp.1-5, 2005

[Nios] NIOS II processor specification, http://www.altera.com/products/ip/processors/nios2/ni2-index.html

[NK93] L.M. Ni, P.K. McKinley, "A survey of wormhole routing techniques in direct networks", Computer, 26(2), pp.62-76, 1993

[NNC06] V.D. Ngo, H.N. Nguyen, H.W. Choi, "Realizing Network on Chip Design of H.264 Decoder Based on Throughput Aware Mapping", Proceedings of First International Conference on Communications and Electronics, pp.337-342, 2006

[NoCSim] NoCSim project documentation, http://research.cs.tamu.edu/codesign/nocsim

[NTA05] C. Neeb, M. Thul, N. Andwehn, "Network-on-chip-centric approach to interleaving in high throughput channel decoders", Proceedings of International Symposium on Circuits and Systems, pp.1766–1769, 2005

[NW08] C. Neeb, N. Wehn, "Designing efficient irregular networks for heterogeneous systems-on-chip", Journal of Systems Architecture: the EUROMICRO Journal, 54(3-4), pp.384-396, 2008

[OHM05] U.Y. Ogras, J. Hu, R. Marculescu, "Key research problems in NoC design: A holistic perspective", Proceedings of the International Conference on Hardware/software Codesign and System Synthesis, pp.69-74, 2005

[OM05] U.Y. Ogras, R. Marculescu, "Application-specific network-on-chip architecture customization via longrange link insertion", Proceedings of International Conference on Computer Aided Design, pp.246–253, 2005

[OMLC06] U.Y. Ogras, R. Marculescu, H.G. Lee, N. Chang, "Communication architecture optimization: making the shortest path shorter in regular networks-on-chip", Proceedings of Design, Automation and Test in Europe Conference, pp.712-717, 2006

[P06] J.W. McPherson, "Reliability challenges for 45nm and beyond", Proceedings of the 43rd Design Automation Conference, pp. 176–181, 2006

[P99] F.J. Pollack, "New microarchitecture challenges in the coming generations of CMOS process technologies", Proceedings of the 32nd annual ACM/IEEE International Symposium on Microarchitecture, p.2, 1999

[PABB05] A. Pullini, F. Angiolini, L. Benini, D. Bertozzi, "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes", Proceedings of the 18th annual symposium on Integrated circuits and system design, pp.224-229, 2005

[PCSV03] A. Pinto, L.P. Carloni, A.L. Sangiovanni-Vincentelli, "Efficient Synthesis of

Networks On Chip", Proceedings of the 21st International Conference on Computer Design, pp.146-150, 2003

[PF07] V.F. Pavlidis, E.G. Friedman, "3-d topologies for networks-on-chip", IEEE Transactions on VLSI Systems, 15(10), pp.1081–1090, 2007

[PGJIS05] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures", IEEE Transactions on Computers, 54(8), pp.1025-1040, 2005

[PS04] G. Palermo, C. Silvano, "PIRATE: A Framework for Power/Performance Exploration of Network-On-Chip Architectures", Lecture Notes in Computer Science, vol.3254, pp.521–531, 2004

[Qnx] Neutrino OS specification, http://www.qnx.com/products/neutrino-rtos/index.html

[Qsys] Qsys System-integration Tool, http://www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qts-qsys.html?GSA_pos=9&WT.oss_r=1&WT.oss=qsys

[Rad] ATI Radeon HD 2900 Graphics Technology specification, http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-2000/hd-2900/Pages/ati-radeon-hd-2900-specifications.aspx

[RDPGR05] A. Radulescu, J. Dielissen, S.G. Pestana, O.P. Gangwal, E. Rijpkema, P. Wielage, K. Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration", IEEE Transactions on Computer-Aided Design ofIntegrated Circuits and Systems, 24(1), pp.4-17, 2005

[RGRDM03] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip", Proceedings of the conference on Design, Automation and Test in Europe - Volume 1, pp.350-355, 2003

[RGW01] E. Rijpkema, K. Goossens, P. Wielage, "A router architecture for networks on silicon", Proceedings of Progress 2001, 2nd Workshop on Embedded Systems, pp.181-188, 2001

[S04] K. Skadron, "Hybrid architectural dynamic thermal management", Proceedings of Design, Automation and Test in Europe Conference, pp.10-15, 2004

[S07] S. Borkar, "Thousand Core Chips – A Technology Perspective", Proceedings of Design Automation Conference, pp.746–749, 2007

[SACRB05] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, G.D. Micheli, "×pipes Lite: A synthesis oriented design library for networks on chips", Proceedings of the Design, Automation and Test in Europe, pp. 1188-1193, 2005

[SALR05] D. Seo, A. Ali, W.T. Lim, N. Rafique, "Near-optimal worst-case throughput routing for two-dimensional mesh networks", Proceedings of IEEE 32nd International Symposium on Computer Architecture, pp.432–443, 2005

[SBKV05] B. Sethuraman, P. Bhattacharya, J. Khan, R. Vemuri, "LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip", Proceedings of the 15th ACM Great Lakes symposium on VLSI, pp.452-457, 2005

[SC05] K. Srinivasan, K.S. Chatha, "SAGA: synthesis technique for guaranteed throughput NoC architectures", Proceedings of the Asia and South Pacific

Design Automation Conference, pp.489-494, 2005

[SCK05] K. Srinivasan, K.S. Chatha, G. Konjevod, "An automated technique for topology and route generation of application specific on-chip interconnection networks",Proceedings of International Conference on Computer Aided Design, pp.231-237, 2005

[SCK06] K. Srinivasan, K.S. Chatha, G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures", IEEE Transactions on VLSI Systems, 14(4), pp.407–420, 2006

[SCK07] K. Srinivasan, K.S. Chatha, G. Konjevod, “Application Specific Network-on-Chip Design with Guaranteed Quality Approximation Algorithms", Proceedings of Asia and South Pacific - Design Automation Conference, pp.184-190, 2007

[SG08] G. Schelle, D. Grunwald, "Exploring FPGA Network on Chip implementations across various application and network loads", Proceedings of International Conference on Field Programmable Logic and Application, pp.41-46, 2008

[SHG08] F.A. Samman, T. Hollstein, M. Glesner, "Planar Adaptive Router Microarchitecture for Tree-Based Multicast Network-on-Chip", Proceedings of the First International Workshop on Network on Chip Architectures, pp.6-14, 2008

[SK00] D. Sylvester, K. Keutzer, "A global wiring paradigm for deep submicron design", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 19(2), pp.242–252, 2000

[SK06] D. Shin, J. Kim, "Communication Power Optimization for Network-on-Chip Architectures", Journal of Low Power Electronics, 2(2), pp.1-12, 2006

[SKH08] E. Salminen, A. Kulmala, T.D. Hamalainen, "Survey of Network-on-chip Proposals", white paper, OCP-IP, http://www.ocpip.org/socket/whitepapers, 2008

[SRH08] R.A. Shafik, P. Rosinger, B.M. Al-Hashimi, "MPEG-based Performance Comparison between Network-on-Chip and AMBA MPSoC", Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp.1-6, 2008

[SRLP09] V. Soteriou, R.S. Ramanujam, B. Lin, L.S. Peh, "A High-Throughput Distributed Shared-Buffer NoC Router", IEEE Computer Architecture Letters, 8(1), pp.21-24, 2009

[SS01] O. Sinnen, L. A. Sousa, "Comparison of contention-aware list scheduling heuristics for cluster computing", Proceedings of the 2001 International Conference on Parallel Processing Workshops, pp.382-387, 2001

[SS05] O. Sinnen, L.A. Sousa, "Communication Contention in Task Scheduling", IEEE Transactions on Parallel and Distributed Systems, 16(6), pp.503-515, 2005

[SS07] M.B. Stuart, J. Sparso, "Custom Topology Generation for Networkon-Chip", Proceedings of NorChip Conference, pp.1-4, 2007

[SSC06] M. Saldana, L. Shannon, P. Chow, "The routability of multiprocessor network topologies in FPGAs", Proceedings of the 2006 International Workshop on System-level Interconnect Prediction, pp.49-56, 2006

[STAN04] D. Siguenza-Tortosa, T. Ahonen, J. Nurmi, "Issues in the development of a practical NoC: the Proteo concept", Integration, the VLSI Journal, 38(1), pp.95–105, 2004

[STM] STMicroelectronics, The STBus interconnect documentation, http://www.st.com/internet/com/technical_resources/technical_literature/user_manual/cd00176920.pdf

[SV06] B. Sethuraman, R. Vemuri, "optiMap: A Tool for generating efficient NoC Architectures using Multi-Port Routers for FPGAs", Proceedings of the conference on Design, Automation and Test in Europe, pp.947-952, 2006

[SV07] B. Sethuraman, R. Vemuri, "A Force-directed Approach for Fast Generation of Efficient Multi-Port NoC Architectures", Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference: Embedded Systems, pp.419-426, 2007

[SW97] J. Staunstrup, W. Wolf, "Hardware/Software Co-Design: Principles and Practice", Springer/Kluwer Academic Publishers, 1997

[T04] A.S. Tanenbaum, "Sieci komputerowe", Wydawnictwo Helion, Gliwice, 2004

[TCP] Transmission Control Protocol, dokument RFC793, http://tools.ietf.org/html/rfc793

[Til] Tilera Processor Family, Katalogi firmowe, http://www.tilera.com/products/processors.php

[TOPD08] R. Tornero, J.M. Orduña, M. Palesi, J. Duato, "A Communication-Aware Topological Mapping Technique for NoCs", Lecture Notes in Computer Science, 5168/2008, pp.910-919, 2008

[VHRDW07] S. Vangal, J. Howard, G. Ruhl, S. Dighel, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, N. Borkar, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS", Proceedings of IEEE International Solid-State Circuits Conference, pp.98-589, 2007

[VN06] M.P. Véstias, H.C. Neto, "Co-Synthesis of a Configurable SoC Platform based on a Network on Chip Architecture", Proceedings of the 2006 Asia and South Pacific Design Automation Conference, pp.48-53, 2006

[W08] P.T. Wolkotte, "Exploration within the Network-on-Chip Paradigm", PhD dissertation, University of Twente, Netherlands, 2008

[WHS06] D. Wu, B.M. Al-Hashimi, and M.T. Schmitz, "Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection," Proceedings Asia and South Pacific Design Automation Conference, pp.36-41, 2006

[WL03] D. Wiklund, D. Liu, "SoCBUS: Switched network on chip for hard real time systems", Proceedings of the IEEE International Parallel and Distributed Processing Symposium, pp.78a, 2003

[WMAK07] D. Wang, H. Matsutani, H. Amano, M. Koibuchi, "A temporal correlation based port combination methodology for networks-on-chip on reconfigurable systems", pp.383-388, 2007

[WMAK08] D. Wang, H. Matsutani, H. Amano, M. Koibuchi, "A link removal methodology for Networks-on-Chip on reconfigurable systems", Proceeding of International Conference on Field Programmable Logic and Application, pp.269-274, 2008

[WPM03] H. Wang, L.S. Peh, S. Malik, "Power-driven design of router microarchitectures in on-chip networks", Proceedings of 36th Annual IEEE/ACM International Symposium on Microarchitecture, pp.105-116, 2003

[WSRS05] P.T. Wolkotte, G.J.M. Smit, G.K. Rauwerda, L.T. Smit, "An energy-efficient reconfigurable circuit-switched network-on-chip", Proceedings of the IEEE International Parallel and Distributed Processing Symposium, pp.155, 2005

[WSZMB07] N. Wang, A. Sanusi, P. Zhao, S. Mohamed. M.A. Bayoumi, "PMCNOC: A pipelining multi-channel central caching network-on-chipcommunication architecture design", SiPS, pp.487-492, 2007

[Xil] Xilinx Inc., Katalogi firmowe, http://www.xilinx.com

[XWHC06] J. Xu, W.Wolf, J.Henkel, S.Chakradhar, "A Design Methodology for Application-Specific Networks-on-Chip", ACM Transactions on Embedded Computing Systems, 5(2), pp.263–280, 2006

[YG92] T. Yang, A. Gerasoulis, "PYRROS: Static Scheduling and Code Generation for Message Passing Multiprocessors", Proceedings of the 6th international conference on Supercomputing, pp.428-437, 1992

[YG93] T. Yang, A. Gerasoulis, "List Scheduling with and without Communication Delays", Parallel Computing, 19(12), pp.1321-1344, 1993

[YPDMV09] K. Yalamanchilil, A. Pasalapudi, A. Dargar, S. Mehrotra, H. Ved, "Evaluation of Performance Optimal Tree Based Application Specific Network on Chip Architectures", Proceedings of IEEE International Advance Computing Conference, pp.620-623, 2009

[ZS03] C.A. Zeferino, A.A. Susin, "SoCIN: a parametric and scalable network-on-chip", Proceedings of Symposium on Integrated Circuits and Systems Design, pp.169-174, 2003

[ZSS04] C.A. Zeferino, F.G.M.E. Santo, A.A. Susin, "ParIS: a parameterizable interconnect switch for networks-on-chip", Proceedings of Symposium on Integrated Circuits and Systems Design, pp.204-209, 2004

[ZZHG05] H. Zimmer, S. Zink, T. Hollstein, M. Glesner, "Buffer-architecture exploration for routers in a hierarchical Network-on-Chip", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, pp.171-174, 2005

[ZZM06] W. Zhou, Y. Zhang, Z. Mao, "An application specific NoC mapping for optimized delay", Proceedings of International Conference on Design and Test of Integrated Systems in Nanoscale Technology, pp.184-188, 2006