WYBRANE ZAGADNIENIA INFORMATYKI TECHNICZNEJ

Modelowanie i optymalizacja

pod redakcją naukową Ireneusza Mrozka

Politechnika Białostocka

WYBRANE ZAGADNIENIA INFORMATYKI TECHNICZNEJ
Modelowanie i optymalizacja

pod redakcją naukową
Ireneusza Mrozka

Politechnika Białostocka
OFICYNA WYDAWNICZA POLITECHNIKI BIAŁOSTOCKIEJ
BIAŁYSTOK 2021

Recenzenci:
prof. dr hab. inż. Alexander Barkalov, Uniwersytet Zielonogórski
assoc. prof. dr Alexander Ivaniuk, Belarusian State University of Informatics and Radioelectronics
prof. dr hab. inż. Dariusz Kania, Politechnika Śląska
assoc. prof. dr. Igor Murashko, Pavel Sukhoi State Technical University of Gomel
dr hab. Agnieszka Nowak - Brzezińska, prof. UŚ, Uniwersytet Śląski
dr inż. Grzegorz Rubin, Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży
dr inż. Andrzej Sawicki, Państwowa Wyższa Szkoła Informatyki i Przedsiębiorczości w Łomży
dr hab. Marek Sikora prof.PŚ, Politechnika Śląska
prof. dr hab. inż. Vyachelsav Yarmolik, Belarusian State University of Informatics and Radioelectronics

Redaktor naukowy dyscypliny informatyka techniczna i telekomunikacja:
prof. dr hab. Jarosław Stepaniuk

Redakcja i korekta językowa:
Janina Demianowicz

Skład, grafika:
Oficyna Wydawnicza Politechniki Białostockiej

Okładka:
Marcin Dominów
Zdjęcie na okładce: TheDigitalArtist
https://pixabay.com/pl/illustrations/sie%c4%87-chmura-obliczeniowa-dane-4851119/

ISBN 978-83-66391-95-6
ISBN 978-83-66391-96-3 (eBook)
DOI: 10.24427/978-83-66391-96-3

Druk: Agencja Reklamowa TOP Agnieszka Łuczak

Oficyna Wydawnicza Politechniki Białostockiej
ul. Wiejska 45C, 15-351 Białystok
e-mail: oficyna.wydawnicza@pb.edu.pl
www.pb.edu.pl

Spis treści

Wstęp

Informatyka jest rozległą dziedziną obejmującą zarówno badania teoretyczne, skupiające się w głównej mierze nad algorytmami i ich optymalizacją, jak i rozwiązania praktyczne związane z projektowaniem, implementacją i wdrażaniem systemów obliczeniowych w warstwie sprzętowej i programowej. Niniejsza monografia jest zbiorem pięciu oryginalnych prac naukowych przedstawiających osiągnięcia młodych i dojrzałych naukowców z obszaru informatyki. Poszczególne rozdziały dotyczą zagadnień związanych z eksploracją danych, zbiorów przybliżonych, modelowaniem układów o charakterze dynamicznym, technikami testowania pamięci RAM i projektowaniem układów cyfrowych.

Rozdział Katarzyny Borowskiej *Obliczenia granularne w przetwarzaniu danych niezbalansowanych* jest poświęcony problemowi eksploracji danych w przypadku zbiorów nizbalansowanych. Autorka zauważyła, iż najnowsze badania wykazały, że zagadnienie to nie powinno być rozpatrywane jedynie w kontekście deficytu informacji o jednej z klas. W rzeczywistości spadek jakości standardowych klasyfikatorów w zetknięciu z tego typu zbiorami danych wiąże się przede wszystkim z wysoką złożonością rozkładu danych. Złożoność ta wzrasta, gdy w charakterystyce danych pojawiają się dodatkowe trudności takie, jak dekompozycja klas, niejednoznaczność strefy brzegowej oraz występowanie szumu. Wymaga to zastosowania dedykowanych rozwiązań, w szczególności uwzględniających lokalną charakterystykę rozkładu. Dlatego zaproponowano w pracy algorytm RGA, który wykorzystuje granule informacyjne i zbiory przybliżone do ukierunkowanego modyfikowania zbioru uczącego oraz proponuje automatyzację doboru parametrów.

Praca Andrzeja Chmielewskiego *Tolerancyjny algorytm V-Detector* ma swoje źródło w sztucznych systemach immunologicznych. Autor analizuje działanie algorytmu V-Detector, wykorzystywanego standardowo do detekcji anomalii w wielowymiarowych zbiorach danych. W tekście zaproponowano nowe podejście do budowy detektorów, inspirowane ideą tolerancyjnych zbiorów przybliżonych. Umożliwia ono zwiększenie efektywności wykrywania pojawiających się anomalii i częściowo rozwiązuje również problem skalowalności, który pojawia się przy standardowym podejściu.

Wiktor Jakowluk w rozdziale *Dobór optymalnego pobudzenia w zadaniu identyfikacji układu dynamicznego w czasie swobodnym* przeprowadza dobór optymalnego sygnału wejściowego w czasie swobodnym, który następnie jest wykorzystany w zadaniu estymacji parametrów modelu układu dynamicznego. Funkcjonał celu Autor

sformułował w postaci funkcji Bolzy z ograniczeniem na D-efektywność oraz energię sygnału sterującego. Przedstawiona metoda może być stosowana dla ogólnej klasy systemów i została zweryfikowana przykładami numerycznymi.

Ireneusz Mrozk w pracy *Dwuprzebiegowe testy krokowe z regularnym indeksem inkrementacji* skupia się nad techniką testowania pamięci RAM opartą na dwóch przebiegach testu krokowego. Aby zminimalizować złożoność niezbędną do generowania sekwencji adresowych zaproponowano metodę generowania drugiej sekwencji na bazie pierwszej na podstawie przyjętego indeksu inkrementacji W pracy dokonano dogłębnej analizy warunków, które muszą spełniać sekwencje adresowe w takiej sesji testowej, aby uzyskać jak największe pokrycie uszkodzeń. Określono optymalną wartość indeksu inkrementacji prowadzącą do uzyskania maksymalnego pokrycia uszkodzeń.

W ostatnim rozdziale monografii *Metodyka ASMD-FSMD projektowania układów cyfrowych na FPGA z wykorzystaniem języka Verilog* Valery Salauyou omawia metodykę projektowania układów cyfrowych opartą na automacie skończonym ze ścieżką przetwarzania danych (finite state machine with datapath – FSMD), w którym działanie układu opisano w postaci diagramu blokowego automatu ze ścieżką przetwarzania danych (algorithmic state machine with datapath – ASMD). Metodyka ta jest nazywana metodyką ASMD-FSMD. W pracy wykazano, iż metodyka ASMD-FSMD, w porównaniu z tradycyjną, pozwala na obniżenie kosztów realizacji z 28,6% do 39,7% oraz zwiększenie wydajności poszczególnych projektów do 17,6%. Ponadto, zastosowanie metodyki ASMD-FSMD może znacznie skrócić czas projektowania i zwiększyć niezawodność projektów.

Ireneusz Mrozek
Białystok, listopad 2021

DOI: 10.24427/978-83-66391-96-3_1

Rozdział 1
Obliczenia granularne w przetwarzaniu danych niezbalansowanych

Katarzyna Borowska
Wydział Informatyki, Politechnika Białostocka

Streszczenie: Dane niezbalansowane od ponad dwóch dekad stanowią jeden z najbardziej interesujących problemów eksploracji danych. Najnowsze badania wykazały, że zagadnienie to nie powinno być rozpatrywane jedynie w kontekście deficytu informacji o jednej z klas. W rzeczywistości spadek jakości standardowych klasyfikatorów w zetknięciu z tego typu zbiorami danych wiąże się przede wszystkim z wysoką złożonością rozkładu danych. Złożoność ta wzrasta, gdy w charakterystyce danych pojawiają się dodatkowe trudności, takie jak dekompozycja klas, niejednoznaczność strefy brzegowej oraz występowanie szumu. Wymaga to zastosowania dedykowanych rozwiązań, w szczególności uwzględniających lokalną charakterystykę rozkładu. Niniejsza praca opisuje algorytm RGA, który wykorzystuje granule informacyjne i zbiory przybliżone do ukierunkowanego modyfikowania zbioru uczącego oraz proponuje automatyzację doboru parametrów. Przeprowadzone eksperymenty dowiodły wysokiej skuteczności zaproponowanej metody w porównaniu z podobnymi technikami.

Słowa kluczowe: dane niezbalansowane, przetwarzanie wstępne danych, obliczenia granularne, granule informacyjne, zbiory przybliżone, SMOTE

Wprowadzenie

Rozwój technologii jest obecnie szybki jak nigdy dotąd. Spektakularne wynalazki, o których dawniej nikt nawet nie marzył, stają się częścią naszej codzienności. Roboty, autonomiczne pojazdy, asystenci głosowi, zaawansowana diagnostyka medyczna, telechirurgia, biodruk 3D, inteligentne domy, rozszerzona rzeczywistość to tylko nieliczne przykłady dziedzin, które w bardzo spektakularny sposób rozwinęły się w ostatnim czasie. Każda z nich korzysta z szeroko pojmowanej sztucznej inteligencji. Ta zaś opiera się przede wszystkim na zgromadzonych danych. Wydobywanie cennej wiedzy z danych jest złożonym procesem, wymagającym eksperckiej wiedzy, doświadczenia oraz odpowiednio przygotowanych i wdrożonych technik. Niestety, nawet najbardziej kompetentni badacze napotykają w trakcie eksploracji danych na problemy,

które niejednokrotnie wynikają z samej natury danych [22]. Wśród takich problemów szczególnie interesujący i dominujący pod względem częstości występowania jest problem danych niezbalansowanych. Dane tego typu charakteryzują się znaczną dysproporcją liczebności próbek reprezentujących poszczególne klasy. W praktyce oznacza to, że nie udało się pozyskać dostatecznych informacji na temat jednej z klas – liczba odnotowanych obserwacji należących do tej klasy (nazywanej mniejszościową lub pozytywną) jest diametralnie mniejsza niż liczba pozostałych obserwacji (z klasy nazywanej większościową lub negatywną) [9]. Co więcej, to właśnie ta klasa, która występuje w zbiorze danych w niedomiarze, jest zazwyczaj przedmiotem szczególnego zainteresowania. Sam fakt niewielkiej liczby próbek reprezentujących jedną z klas nie stanowiłby tak znaczącego problemu, gdyby nie tendencja narzędzi używanych standardowo w klasyfikacji do uogólniania, czyli podejmowania decyzji na korzyść klas, których reprezentacja jest największa [9, 26]. Ponadto, na poziom złożoności problemu wpływa również rozkład danych. Okazuje się, że nie tylko deficyt danych z jednej z klas może skutkować negatywnym wpływem na proces uczenia maszynowego, ale również sama charakterystyka danych, w szczególności jej poziom skomplikowania, może doprowadzić do degradacji wyników [9, 17, 26]. Te dwa czynniki: niezbalansowanie danych oraz ich złożoność stanowią główne przyczyny komplikacji w procesie uczenia maszynowego. Wśród proponowanych rozwiązań problemu można wyróżnić metody modyfikujące zbiór uczący, zmodyfikowane klasyfikatory oraz techniki hybrydowe, łączące obydwa podejścia [9, 10, 15, 26].

W ramach niniejszej pracy opisana zostanie metoda, która proponuje zastosowanie obliczeń granulanych [21] we wstępnym przetwarzaniu danych niezbalansowanych, aby uzyskać zbalansowany pod względem liczebności zbiór próbek oraz doprowadzić do zmniejszenia złożoności dystrybucji danych. Biorąc pod uwagę indywidualny charakter rozkładów danych z różnych dziedzin oraz ich wewnętrzną dystrybucję, opracowany algorytm pozwala na zachowanie elastyczności przez możliwość automatycznego doboru parametrów.

1.1. Istota problemu danych niezbalansowanych

1.1.1. Źródła problemu

W zagadnieniu danych niezbalansowanych głównym wyznacznikiem powagi problemu jest liczba oraz rodzaj dziedzin, które są reprezentowane przez zbiory danych dotknięte tym zjawiskiem. Okazuje się, że dane tego typu charakteryzują zagadnienia z bardzo wielu obszarów otaczającej nas rzeczywistości, również tych, które uznawane są za wyjątkowo istotne. W szczególności są to: diagnostyka medyczna, wykrywanie wszelkich anomalii, takich jak wycieki ropy naftowej, detekcja wad produkcyjnych, wykrywanie przestępstw finansowych oraz włamań, zarządzanie ryzykiem,

kategoryzacja tekstu oraz analiza sentymentu [9, 12, 15, 16, 28]. W większości tych dziedzin to poprawne rozpoznawanie obiektów z klasy, która jest mniej liczna, stanowi główny cel analizy [28]. Spadek jakości algorytmu względem mniej licznej klasy w wielu dziedzinach może prowadzić do dramatycznych skutków, takich jak zbyt późne wdrożenie (lub całkowity brak wdrożenia) odpowiednich do danego problemu rozwiązań (np. niepodjęcie leczenia we wczesnym stadium choroby nowotworowej [12]). Potwierdza to, jak ważna jest świadomość istnienia niniejszego problemu oraz odpowiednie reagowanie we wczesnych etapach eksploracji danych.

Podstawowym krokiem, który pozwala wykrywać potencjalny problem danych niezbalansowanych, jest zbadanie wskaźnika *imbalanced ratio* (IR), [24]:

$$IR = \frac{N^-}{N^+}, \tag{1.1}$$

gdzie N^- stanowi liczbę obiektów z klasy większościowej, zaś N^+ oznacza liczbę obiektów z klasy mniejszościowej.

Nie zdefiniowano jednoznacznego progu określającego sytuację, w której można nazwać analizowaną dystrybucję danych niezbalansowaną [26]. Zarówno proporcje między klasami takie, jak 100:1, 1000:1, czy 1000000:1 mogą wskazywać na konieczność szczególnego traktowania danych oraz uwzględnienia w dalszych pracach tej dysproporcji [28].

Oprócz wspomnianej weryfikacji wskaźnika IR, niezwykle ważne jest uwzględnienie dodatkowych trudności, które mają bezpośredni wpływ na złożoność rozkładu danych. Ostatnie szczegółowe badania potwierdzają, że to właśnie te komplikacje, w połączeniu z dysproporcją liczebności próbek z poszczególnych klas, stanowią główne źródło problemu [9, 12, 14, 17, 18, 24, 26, 28, 29]. Zaburzona charakterystyka danych oraz ich niezbalansowanie ma ogromne znaczenie w procesie uczenia klasyfikatorów. Większość badań pokazuje w takich przypadkach ukierunkowanie modelu do podejmowania decyzji na korzyść klasy większościowej. Próbki z klasy mniejszościowej, ich cechy szczególne, wyróżniające właściwości, nie mogą zostać prawidłowo rozpoznane i w rezultacie zostają zdominowane przez bardziej liczne obserwacje z klasy większościowej. Wśród wspomnianych dodatkowych komplikacji można wyróżnić:

- Dekompozycję klas – zbiór danych składa się z rozdrobnionych klastrów, skupiających bardzo niewielkie zbiory próbek z poszczególnych klas [12, 14, 18, 24, 28, 29]. Zjawisko to jest szczególnie niekorzystne w przypadku klasy mniejszościowej, której liczebność nawet bez podziału na klastry jest zbyt mała, aby standardowe klasyfikatory miały możliwość wyznaczenia charakterystyki wyróżniającej obiekty reprezentujące daną klasę [14]. Podział na rozproszone podzbiory dodatkowo utrudnia rozpoznawanie klasy mniejszościowej.
- Niejednoznaczność strefy brzegowej – występuje w sytuacji, gdy istnieją w rozkładzie danych obszary, które koncentrują bardzo podobne do siebie obiekty z różnych klas. Stąd klasyfikatory nie mają możliwości wyznaczenia reguł odpowiednio

rozróżniających próbki między klasami, gdyż odbywałoby się to kosztem zdolności uogólniania [9]. Przeprowadzone badania wskazują, że im większy jest poziom nakładania się klas w obszarze brzegowym, tym większa podatność algorytmu uczenia na problem niezbalansowania danych [28].
- Występowanie szumu – istnienie specyficznych przykładów, które mimo, że reprezentują klasę mniejszościową, posiadają cechy specyficzne dla klasy większościowej. Obiekty tego typu w przestrzeni cech zajmują miejsca oddalone od próbek z tej samej klasy, jednocześnie znajdując się w otoczeniu przykładów z klasy przeciwnej [24]. Próbki tego rodzaju mogą stanowić błędne dane, jednak nie można wykluczyć, że są to bardzo rzadkie przypadki [29].

Opisane trudności, zwiększające złożoność danych, powinny zawsze być rozpatrywane równolegle z problemem danych niezbalansowanych.

1.1.2. Parametry jako zagadnienie zwiększające złożoność problemu

Oprócz wspomnianych istotnych komplikacji, które można nazwać wewnętrznymi, gdyż wynikają z samej natury danych niezbalansowanych oraz złożoności ich rozkładów, występują również trudności niejako zewnętrzne, związane z narzędziami, które potencjalnie mogłyby stanowić skuteczne rozwiązanie problemu danych niezbalansowanych. Otóż, nawet te narzędzia, których wyniki w zetknięciu z niezbalansowanymi zbiorami danych są obiecujące, niezwykle często nie spełniają kryterium oczekiwanej elastyczności. W istocie pozwalają na uzyskiwanie rezultatów o wysokiej jakości, jednak zazwyczaj jest to poprzedzone żmudną pracą nad odpowiednim doborem parametrów, które są adekwatne jedynie do danego, konkretnego przypadku. Jednocześnie te same parametry najprawdopodobniej nie znajdą zastosowania przy pracy nad innym zbiorem danych (w szczególności takim, który będzie miał zupełnie inną charakterystykę rozkładu). W ten sposób można wyszczególnić kolejne, niekiedy pomijane oraz poniekąd ukryte, źródło problemów w przetwarzaniu danych niezbalansowanych: trudności w doborze parametrów.

Wiele technik przedstawianych jako środki zaradcze na niezbalansowanie danych wymaga konfiguracji różnych parametrów. Wśród nich można wyróżnić między innymi wylistowane poniżej parametry:
- liczba najbliższych sąsiadów,
- miary odległości,
- proporcje i wartości progowe pozwalające określić złożoność rozkładu danych,
- liczba klastrów,
- docelowa liczebność klasy mniejszościowej,
- wagi cech,
- wagi obiektów reprezentujących poszczególne klasy,
- próg określający podobieństwo obiektów,
- tryby przetwarzania dobrane w zależności od poziomu złożoności rozkładu danych.

Występowanie wymienionych parametrów jest uzależnione od wybranej metody, jednakże dobór nawet pojedynczych z nich może wpływać znacząco na wyniki. Brak doświadczenia lub wiedzy dziedzinowej może skutkować zaprzepaszczeniem skuteczności danego algorytmu przy nieodpowiednio dobranych parametrach. Ponadto, nawet przy dysponowaniu tymi pożądanymi cechami również istnieje ryzyko błędnej oceny odnośnie zbiorów danych o wysokiej złożoności rozkładu. Nie należy więc lekceważyć tych dodatkowych trudności i warto zautomatyzować proces dobierania parametrów celem uzyskania większej pewności co do tego, czy możliwości algorytmów są wykorzystywane w pełni w ramach prac nad danymi o konkretnej charakterystyce.

1.2. Przegląd dotychczasowych rozwiązań

Niniejsza praca obejmuje propozycję rozwiązania problemu danych niezbalansowanych należącą do technik modyfikujących dane (przetwarzania wstępnego). Stąd również w tym rozdziale zostaną przedstawione wyłącznie techniki pokrewne do przedstawionego algorytmu. Nie sposób jednak nie wspomnieć, że oprócz tej kategorii metod, istnieje wiele innych rozwiązań ukierunkowanych na dostosowanie algorytmów klasyfikacji na poprawne wykrywanie próbek z klasy mniejszościowej lub narzędzia hybrydowe, łączące modyfikacje klasyfikatorów z modyfikacjami danych [9]. Nie umniejszając skuteczności wymienionych alternatywnych metod, w niektórych zastosowaniach mogą okazać się niewystarczająco elastyczne ze względu na konieczność ingerencji w mechanizmy działania algorytmów. Analiza ta będzie więc dotyczyła wybranych metod niezależnych od klasyfikatorów.

1.2.1. Synthetic Minority Oversampling Technique (SMOTE)

Rozpatrując dostępne techniki niwelowania problemu danych niezbalansowanych, oparte na modyfikacjach w zbiorze uczącym, niewątpliwie należy wyróżnić algorytm SMOTE [6]. Metoda ta zawdzięcza swój sukces idei generowania nowych próbek danych z klasy mniejszościowej bazując na podobieństwie między istniejącymi przykładami reprezentującymi tę klasę [9]. W porównaniu z losowym tworzeniem obiektów z klasy mniejszościowej, stanowiących wierne kopie istniejących już obserwacji, algorytm ten można nazwać przełomowym. Niestety, biorąc pod uwagę dodatkowe trudności, wynikające ze złożoności rozkładów danych, nawet tak skuteczna metoda jak SMOTE okazuje się niepozbawiona wad [26, 28]. Generowanie nowych próbek z jednej z klas, bez uwzględniania charakterystyki konkretnego zbioru danych, może prowadzić do wprowadzenia niejasności oraz niespójności, a tym samym do zwiększenia złożoności rozkładu danych. W szczególności umiejscowienie pojedynczych próbek z klasy mniejszościowej w obszarach zajmowanych głównie przez obserwacje z klasy większościowej, stanowi jedynie wprowadzanie niepotrzebnego szumu, który

nic nie wniesie w procesie klasyfikacji oraz nie poprawi wyników (a może wręcz przeciwnie – zaburzyć je). Stąd w wielu pracach postanowiono skupić się na udoskonaleniu tak obiecującej metody [8].

1.2.2. Modyfikacje SMOTE

Pierwsza grupa rozwiązań opartych na SMOTE obejmuje techniki, które zakładają generowanie nowych, syntetycznych próbek z klasy mniejszościowej jedynie w określonych obszarach przestrzeni cech. Metoda Borderline-SMOTE [11] ogranicza tworzenie nowych obiektów jedynie do granicy między przykładami z obydwu klas. Autorzy sugerują, że to właśnie obszar graniczny jest punktem newralgicznym w procesie klasyfikacji i wzmocnienie reprezentacji próbek z klasy mniejszościowej w tym rejonie polepszy zdolność algorytmów do poprawnego klasyfikowania. MSMOTE [13] z kolei przyjmuje odmienne podejście. Na początku przydziela próbki do trzech grup w zależności od ich sąsiedztwa w przestrzeni cech. Podział ten decyduje o sposobie tworzenia nowych obiektów z klasy mniejszościowej: w przypadku obiektów leżących na granicy między klasami syntetyczna próbka tworzona jest na podstawie połączenia cech analizowanej aktualnie obserwacji oraz jedynie jej najbliższego sąsiada. Pozostałe obiekty podlegają identycznemu procesowaniu jak w przypadku klasycznej metody SMOTE. Przykłady uznane za szum nie biorą udziału w przetwarzaniu. Podobny mechanizm działania przyjmuje metoda Safe-Level SMOTE [5]. Próbki z klasy mniejszościowej są kategoryzowane pod względem poziomów bezpieczeństwa, które reprezentują. Poziomy te odzwierciedlają sąsiedztwo, w jakim znajduje się dana próbka – jeśli będzie to otoczenie złożone głównie z obiektów z klasy przeciwnej (większościowej), wtedy obserwacja uznawana jest za należącą do zmniejszonego poziomu bezpieczeństwa. Nowe próbki generowane są jedynie z pobliżu obszarów złożonych z obiektów o wysokich poziomach bezpieczeństwa (czyli obszarach składających się głównie z obserwacji z klasy mniejszościowej).

1.2.3. Metody filtrujące

Druga grupa rozwiązań nawiązujących do techniki SMOTE wzbogaca przetwarzanie oparte na ukierunkowanym generowaniu syntetycznych próbek o dodatkowy krok czyszczenia danych. Metoda SPIDER [18] w trybie „Weak" opiera się na wzmacnianiu liczebności przykładów z klasy mniejszościowej, które zostały uznane za bezpieczne (bazując na ich położeniu w przestrzeni cech). Tryb „Relabel" wprowadza fazę zmian etykiet obiektów z klasy większościowej, które zostały uznane za szum. W trybie „Strong" zarówno bezpieczne, jak i uznane za szum próbki z klasy mniejszościowej są wzmacniane (biorą udział w przetwarzaniu wstępnym analogicznym do metody SMOTE). *SMOTE-RSB*$_{\star}$ [23] zakłada generowanie syntetycznych przykładów algorytmem SMOTE. Dodatkowo, opierając się na teorii zbiorów przybliżonych, wprowadza

etap czyszczenia nowoutworzonych danych, które nie należą do dolnej aproksymacji. Metoda SMOTE + Tomek-links [1] po kroku tworzenia nowych obiektów, usuwa z obydwu klas przykłady, które wprowadzają niejasności do rozkładu danych. Wykorzystuje do tego pary najbliżej sąsiadujących obserwacji, tzw. Tomek-links, które zawierają próbki z dwóch różnych klas. Wszystkie tak zidentyfikowane pary są usuwane. SMOTE-IPF [24] do usuwania wprowadzających szum danych wykorzystuje filtr IPF. Filtr ten bazuje na modelach klasyfikatora C4.5 utworzonych na partycjach z podzielonego źródłowego zbioru danych. W przypadku, gdy ponad połowa z tak zbudowanych klasyfikatorów nie jest w stanie poprawnie sklasyfikować danej próbki, taka obserwacja jest usuwana. Filtrowaniu podlegają wszystkie przykłady: zarówno te pierwotnie występujące w zbiorze danych, jak i wygenerowane w procesie przetwarzania wstępnego metodą SMOTE. Algorytm VISROT [4] łączy podejście ukierunkowanego tworzenia nowych próbek (przez generowanie ich jedynie w wybranych obszarach przestrzeni cech – tych, które można uznać za bezpieczne, czyli oddalone od przykładów z klasy większościowej) z etapem czyszczenia danych. Wszystkie nowe, syntetyczne przykłady, które są uznane za szum, zostają usuwane.

1.3. Algorytm RGA

Głównym celem powstania algorytmu RGA (Rough-Granular Computing Algorithm) [2] jest zapewnienie elastyczności w połączeniu z efektywnością działania przy wydobywaniu wiedzy z danych niezbalansowanych. Opisywane podejście wykorzystuje wyłącznie proces przetwarzania wstępnego danych, nie ingerując w żaden sposób w mechanizmy działania algorytmów klasyfikacji. Stąd zachowana jest niezależność metody od wybranego klasyfikatora. Algorytm opiera się na wybiórczym i ukierunkowanym generowaniu nowych próbek z klasy mniejszościowej oraz dodatkowym kroku filtrowania. Etap filtrowania został ograniczony jedynie do syntetycznych próbek, aby nie stracić żadnych potencjalnie istotnych informacji z oryginalnego zbioru danych. Dodatkowo istotną cechą wyróżniającą metodę RGA jest możliwość automatycznego doboru parametrów do konkretnego problemu poprzez iteracyjne powtarzanie kroku przetwarzania wstępnego z różnymi kombinacjami wartości parametrów.

1.3.1. Wyznaczenie granul

Algorytm rozpoczyna się od identyfikacji granul informacyjnych [21, 27]. Identyfikacja ta opiera się na założeniu, że obserwacje mogą być traktowane jako podobne, jeśli znajdują się w bliskiej odległości w przestrzeni cech. Za pomocą metody kNN wyznaczane są granule skupiające instancje z klasy mniejszościowej wraz z sąsiadującymi próbkami z obydwu klas. Kluczowe jest w tym kroku określenie miary podobieństwa obiektów. Jako że rzeczywiste dane przeważnie wymagają możliwości przetwarzania

zarówno jakościowych, jak i ilościowych atrybutów, stosowana jest dedykowana metryka: Heterogeneous Value Distance Metric (HVDM), [30]. Pozwala ona skutecznie określić podobieństwo obserwacji bazując na ich sąsiedztwie, czyli bliskim położeniu w przestrzeni cech. Bliskie położenie wskazuje na zbliżone wartości poszczególnych atrybutów rozpatrywanych obserwacji.

1.3.2. Badanie stopnia zawierania granul

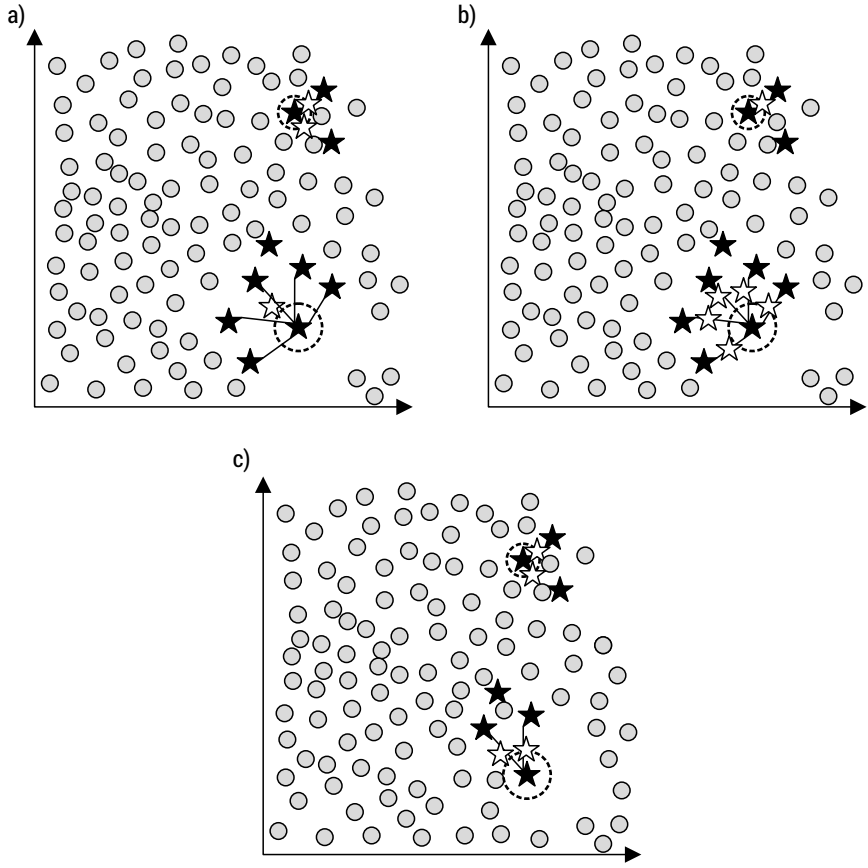
Wykonanie kroku formowania granul informacyjnych, skupiających próbki z klasy mniejszościowej otoczone podobnymi obiektami z obydwu klas, pozwala na podział problemu na mniejsze zadania [21]. Dzięki temu granule mogą zostać przetworzone w sposób adekwatny do specyfiki danych, które zawierają. Proces ten stanowi kluczowy etap działania algorytmu, identyfikujący złożoność rozkładu danych i determinujący mechanizm postępowania, który powinien zostać zastosowany wobec danych określonego typu. Tym samym uzyskiwana jest elastyczność, pozwalająca na skuteczną pracę ze zbiorami danych o różnej charakterystyce.

Proces dobierania odpowiedniej metody przetwarzania do typu granuli obejmuje ustalenie, w jakim stopniu określona granula zawiera się w granuli definiującej całą klasę mniejszościową. W rezultacie każdej z granul przydzielana jest etykieta wyznaczająca dalszy proces postępowania. Poniższe punkty zawierają opis poszczególnych etykiet:

- SAFE – wysoki stopień zawierania granuli informacyjnej oznacza, że znajduje się ona w obszarze jednolicie zajmowanym przez obiekty z klasy mniejszościowej. Obszar taki można uznać za bezpieczny – algorytm klasyfikacji nie powinien napotkać trudności w rozpoznawaniu próbek tego typu. Wskaźnikiem umożliwiającym przydzielenie granuli tej etykiety jest proporcja zawierających się w niej obserwacji z dwóch klas: jeśli większość stanowią przykłady z klasy mniejszościowej i jednocześnie próbka położona najbliżej centralnego punktu granuli należy do klasy mniejszościowej, granula otrzymuje etykietę SAFE.
- BOUNDARY – niski stopień zawierania jest wynikiem znacznej reprezentacji próbek z klasy większościowej w danej granuli. Tak określona granula znajduje się w mniej jednorodnym obszarze, stanowiącym granicę między klasami w przestrzeni cech. Występowanie granul tego rodzaju oznacza, że próbki z obydwu klas są wymieszane. Doprecyzowując, granula jest oznaczana jako BOUNDARY, gdy przynajmniej połowa jej elementów reprezentuje klasę większościową lub próbka położona najbliżej elementu centralnego granuli należy do klasy większościowej.
- NOISE – zerowy stopień zawierania granuli zachodzi w sytuacji, gdy żaden z obiektów tworzących granulę, z wykluczeniem centralnego przykładu, nie należy do klasy mniejszościowej. Jako że jedynie centralna próbka reprezentuje klasę mniejszościową, należy ją rozpatrywać w kategoriach bardzo rzadkiego przypadku. Otoczenie sąsiadującymi obserwacjami z klasy większościowej będzie

przysparzało znacznych trudności w procesie uczenia standardowymi klasyfikatorami. Granula oznaczona tą etykietą niesie informacje, które charakteryzują przede wszystkim klasę większościową.

Wymienione etykiety odzwierciedlają, w jakim stopniu dana granula definiuje specyficzne cechy klasy mniejszościowej, jak bardzo są one charakterystyczne dla tej klasy i tym samym, czy będzie powodowała trudności w procesie uczenia. Na podstawie tak dokonanego podziału granul podejmowana jest decyzja odnośnie trybu dalszego przetwarzania danych. W zależności od przyjętego trybu działania, algorytm generuje w określonych obszarach przestrzeni cech odpowiednią liczbę syntetycznych próbek (rysunek 1.1).



RYSUNEK 1.1. Centralne punkty granul zostały oznaczone linią przerywaną. Wygenerowane przykłady z klasy mniejszościowej oznaczono znakiem gwiazdy bez wypełnienia. Koła z szarym wypełnieniem reprezentują próbki z klasy większościowej. Proces generowania syntetycznych próbek w zależności od przyjętego trybu działania: (a) tryb *LowComplexity*, (b) tryb *HighComplexity*, (c) tryb *noSAFE*

Wybór sposobu przetwarzania wstępnego danych niezbalansowanych odbywa się poprzez weryfikację wartości parametrycznej definiującej procentowy udział granul oznaczonych etykietą BOUNDARY w zbiorze wszystkich granul stworzonych wokół obiektów z klasy mniejszościowej. Wartość progowa określa, z jakim typem problemu będzie trzeba się zmierzyć w procesie przygotowania danych do uczenia maszynowego. Eksperymenty przeprowadzone w [12] udowadniają, że co najmniej 30% przykładów z klasy mniejszościowej będących na granicy klas (gdzie próbki są wymieszane) może doprowadzić do znacznego pogorszenia skuteczności klasyfikacji. Algorytm RGA umożliwia określanie wartości progowej jako parametru: *complexity_th*.

Detekcja niewielu granul typu BOUNDARY w zbiorze danych umożliwia zastosowanie trybu *LowComplexity*. Dane tego typu niosą dużo informacji charakterystycznych dla danej klasy, pozwalając odróżniać od siebie obiekty z poszczególnych klas. Przykłady z klasy mniejszościowej są przeważnie otoczone próbkami reprezentującymi tę samą klasę. Stąd można wnioskować, że granule te położone są w relatywnie jednorodnych obszarach. Poniższy wzór definiuje kryterium uznania zbioru danych za odpowiedni do przetwarzania w tym trybie:

$$\frac{card\left(\left\{x \in X_{d=+} : Label(x) = BOUNDARY\right\}\right)}{card\left(X_{d=+}\right)} < complexity\_th, \tag{1.2}$$

gdzie: $card(U)$ to liczba elementów zbioru U, x stanowi pojedynczy obiekt, $X_{d=+}$ oznacza przykłady z klasy mniejszościowej, zaś $Label(x)$ jest etykietą przypisaną obiektowi x.

Po ustaleniu metody działania algorytmu jako *LowComplexity*, przetwarzanie wstępne granul poszczególnych rodzajów odbywa się w dostosowany do ich właściwości sposób. Dokładny opis tego mechanizmu znajduje się poniżej:

- $Label(x) = SAFE$: w większości występujące granule oznaczone jako bezpieczne nie wymagają znacznego wsparcia nowymi syntetycznymi próbkami – tworzone jest wyłącznie po jednej próbce na granulę posiadającą tę etykietę. Nowy obiekt opisany jest przez zbiór cech stanowiących kombinację właściwości rozpatrywanego centralnego przykładu danej granuli oraz najbliższego sąsiada.
- $Label(x) = BOUNDARY$: większość syntetycznych próbek generowana jest w tych granicznych obszarach. Dzięki temu można uzyskać wzmocnienie roli obserwacji z klasy mniejszościowej w tym trudnym rejonie przestrzeni cech. Jako że nie dysponujemy wieloma przypadkami tego typu, utworzenie wielu egzemplarzy instancji z klasy mniejszościowej w obszarze nakładania się klas pozwoli klasyfikatorowi na wygenerowanie poprawnych reguł. Nowe próbki tworzone są w bliższej odległości od centralnego przykładu danej granuli, aby zapobiegać zbytniemu zwiększaniu złożoności rozkładu przez nadmierne wymieszanie klas.
- $Label(x) = NOISE$: nowe przykłady nie są generowane.

Przewaga granul oznaczonych jako BOUNDARY, czyli:

$$\frac{card\left(\left\{x \in X_{d=+} : Label(x) = BOUNDARY\right\}\right)}{card\left(X_{d=+}\right)} \geq complexity\_th, \tag{1.3}$$

implikuje konieczność przyjęcia bardziej restrykcyjnych reguł generowania syntetycznych próbek. Wiąże się to z potencjalnymi znacznymi komplikacjami w procesie uczenia w wyniku wysokiej złożoności problemu. Przyjmowany jest więc tryb *HighComplexity*, który zakłada dalsze przetwarzanie danych w następujący sposób:

- $Label(x) = SAFE$: większość nowych próbek generowana jest w tym uznanym za bezpieczny obszarze. Zakładając, że granule tego typu skupiają przykłady posiadające specyficzne dla klasy mniejszościowej cechy i stanowiące proste do wyuczenia wzorce, utworzenie głównej puli syntetycznych przykładów właśnie w tych rejonach nie spowoduje wzrostu złożoności problemu, a jednocześnie umożliwi klasyfikatorom lepsze rozeznanie odnośnie właściwości charakteryzujących klasę mniejszościową.
- $Label(x) = BOUNDARY$: liczba obserwacji tego typu jest podwajana poprzez tworzenie pojedynczych nowych przykładów, wykorzystując kombinację cech centralnej próbki granuli oraz jej najbliżej położonego sąsiada – przy czym nowy przykład będzie wykazywał większe podobieństwo do centralnej próbki granuli. Tworzenie wielu nowych obiektów w obszarze granicznym w trybie *HighComplexity* mogłoby doprowadzić do nadmiernego efektu nakładania się danych różnych klas. Między dwoma obiektami z klasy mniejszościowej może występować obiekt z klasy większościowej. Stąd nowa próbka mogłaby być zbyt podobna do reprezentanta klasy negatywnej. Aby zapobiec takiej sytuacji, w tym niejednoznacznym obszarze generowane są wyłącznie pojedyncze nowe przykłady z klasy mniejszościowej.
- $Label(x) = NOISE$: nowe przykłady nie są generowane.

Ostatni przypadek dotyczy szczególnej sytuacji, w której żadne granule typu bezpiecznego (SAFE) nie zostały rozpoznane:

$$\left\{x \in X_{d=+} : Label(x) = SAFE\right\} = \varnothing, \tag{1.4}$$

Wybierany jest wtedy tryb *noSAFE*. Określa on dane o szczególnie dużej złożoności. Brak bezpiecznych obszarów oznacza bardzo wysoki poziom wymieszania danych pochodzących z różnych klas. Stosowana jest w takim wypadku dedykowana metoda przetwarzania, zgodna z opisem poniżej:

- $Label(x) = BOUNDARY$: wszystkie syntetyczne obiekty z klasy mniejszościowej generowane są w obszarach granicznych. Nowe przykłady nie mogą być tworzone w bezpiecznych rejonach, gdyż takie nie istnieją. W tej szczególnej sytuacji jedynie wzmocnienie reprezentacji próbek z klasy mniejszościowej w obszarach wymieszania klas może poprawić skuteczność klasyfikacji.
- $Label(x) = NOISE$: nowe przykłady nie są generowane.

Granule typu NOISE zostały wykluczone z przetwarzania, gdyż nie ma pewności odnośnie przyczyny ich występowania. Mogą stanowić zarówno błędnie zapisane dane, jak i niespotykane przypadki. Zakładając możliwość prawdziwości każdej z tych potencjalnych przyczyn, tego typu próbki nie powinny być usuwane, ale jednocześnie nie warto poddawać ich przetwarzaniu, aby nie doprowadziły do nadmiernego zwiększenia złożoności rozkładu, wprowadzając szum.

1.3.3. Filtrowanie danych

W celu zapewnienia jak najwyższej jakości tworzonych obiektów klasy mniejszościowej, wprowadzony został etap usuwania niejednoznaczności. Po wygenerowaniu nowych próbek weryfikowana jest ich przynależność do dolnej aproksymacji zbioru zawierającego elementy z klasy mniejszościowej. Jeśli nowy obiekt nie należy do dolnej aproksymacji, jest usuwany ze zbioru syntetycznych próbek.

Przyjmując U jako uniwersum, relacja nierozróżnialności $IND \subseteq U \times U$ w teorii zbiorów przybliżonych identyfikuje przykłady, które są opisywane identycznymi informacjami [20, 27]. Jednym z głównych pojęć opartych na tym twierdzeniu jest dolna aproksymacja zbioru X (stanowiącego dowolny podzbiór uniwersum U):

$$\left\{x \in U : [x]_{IND} \subseteq X\right\}, \tag{1.5}$$

Jest to zbiór wszystkich próbek, które mogą być jednoznacznie sklasyfikowane jako należące do X w odniesieniu do relacji nierozróżnialności IND [19].

Możliwość przetwarzania danych o wartościach typu ciągłego zapewniona została poprzez zastosowanie pojęcia podobieństwa obiektów jako odległości między próbkami w przestrzeni cech w ramach relacji tolerancji uogólnionej teorii zbiorów przybliżonych [25, 27]. Modyfikacja ta wymagała ustanowienia określonej wartości progowej, która jest swoistą granicą między podobnymi przykładami a pozostałymi obserwacjami. Próg podobieństwa (*distance_th*) jest parametrem aplikacji. Jest to wartość procentowa średniej odległości HVDM między obiektami z różnych klas w przetwarzanym zbiorze danych. Po wykonaniu kroku filtrowania, dane mogą zostać poddane procesowi klasyfikacji.

Przewidzenie jak wiele z wygenerowanych próbek zostanie usuniętych nie jest możliwe. Zaistnienie pesymistycznego scenariusza mogłoby oznaczać, że nadmiernie dużo syntetycznych obiektów będzie wykluczonych z dalszego przetwarzania w etapie filtrowania i w związku z tym całkowita liczebność klasy mniejszościowej nie zostanie odpowiednio zwiększona. Mając na uwadze istotny cel wstępnego przetwarzania danych niezbalansowanych – a jest nim próba zrównania liczebności przykładów z obydwu klas – wprowadzono dodatkowe zabezpieczenie w postaci określanej jako parametr nadmiarowości w tworzeniu nowych próbek (*card_redundancy*). Parametr ten jest wartością procentową, wykorzystywaną do wyliczania całkowitej liczby wygenerowanych obiektów zgodnie z następującym wzorem:

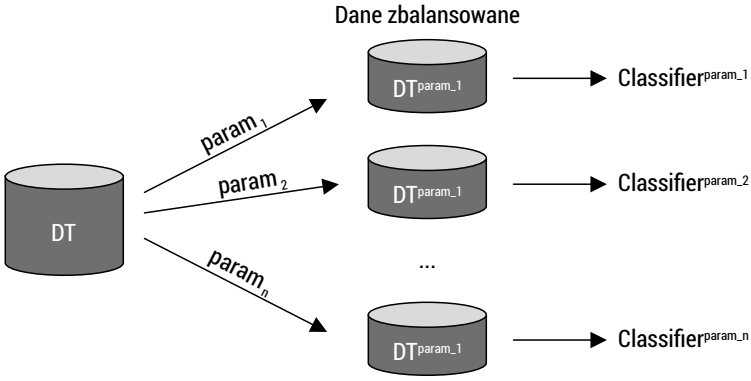
$$generated\_samples = card\_redundancy * (N^- - N^+), \tag{1.6}$$

gdzie N^- oznacza liczbę obiektów z klasy większościowej, zaś N^+ oznacza liczbę obiektów z klasy mniejszościowej.

1.3.4. Dobieranie parametrów

Opisany algorytm posiada kilka szczególnie istotnych parametrów, których modyfikacja może skutkować lepszym dopasowaniem mechanizmu działania do konkretnego rozkładu danych. Parametry te są następujące:

- *k* – liczba najbliższych sąsiadów,
- *complexity_th* – próg złożoności,
- *distance_th* – próg podobieństwa,
- *card_reundancy* – liczba dodatkowych obiektów.



RYSUNEK 1.2.: **Metoda automatycznej ewaluacji parametrów**

Aby uprościć proces doboru wymienionych parametrów, została opracowana metoda automatycznej ewaluacji (rysunek 1.2). Na początku przygotowywane są kombinacje możliwych wartości, jakie może przyjąć każdy z parametrów. W wyniku uzyskiwane są skończone zbiory potencjalnych wartości. Następnie algorytm przetwarzania danych rozpoczyna działanie wykorzystując pierwszą kombinację przygotowanych parametrów. Przetwarzanie powtarzane jest wielokrotnie, za każdym razem z użyciem innej puli wartości parametrów. W rezultacie budowane są modele klasyfikatorów korzystające z oddzielnych zbiorów danych, stanowiących oryginalny zbiór danych wzbogacony o wygenerowane według odpowiednio sparametryzowanego algorytmu RGA syntetyczne próbki z klasy mniejszościowej. Następnie uzyskane wyniki są poddawane weryfikacji tak, aby wybrać najlepszy z nich.

1.4. Eksperymenty

Przedstawiona metoda została przetestowana na piętnastu rzeczywistych zbiorach danych. Charakterystyka poszczególnych zbiorów zaprezentowana jest w tabeli 1.1. Wszystkie dane zostały pobrane z repozytorium UCI [7]. Cechują się różnymi poziomami niezbalansowania liczebności próbek z poszczególnych klas (na co wskazuje parametr IR) oraz w większości wysoką złożonością (kolumna „Obszar graniczny").

TABELA 1.1. Charakterystyka wykorzystanych zbiorów danych

Nazwa	Liczba próbek	Liczba atrybutów	IR	Obszar graniczny
ecoli-0-1 vs 5	240	6	11,00	niepusty
ecoli-0-1 vs 2-3-5	244	7	9,17	niepusty
ecoli-0-1-4-6 vs 5	280	6	13,00	niepusty
ecoli-0-1-4-7 vs 5-6	332	6	12,28	niepusty
ecoli-0-3-4-7 vs 5-6	257	7	9,28	pusty
ecoli-0-4-6 vs 5	203	6	9,15	niepusty
ecoli-0-6-7 vs 3-5	222	7	9,09	niepusty
ecoli-0-2-6-7 vs 3-5	224	7	9,18	niepusty
glass-0-1-6 vs 5	184	9	19,44	pusty
glass-0-4 vs 5	92	9	9,22	pusty
glass-0-6 vs 5	108	9	11,00	pusty
glass5	214	9	22,78	pusty
led7digit-0-2-4-5-6-7-8-9 vs 1	443	7	10,97	niepusty
page-blocks-1-3 vs 4	472	10	15,86	niepusty
yeast-0-3-5-9 vs 7-8	506	8	9,12	niepusty

Każdy ze zbiorów został podzielony na mniejsze partycje, aby umożliwić pięciokrotną walidację krzyżową. Wykorzystane w eksperymentach zbiory często występują w publikacjach o tematyce danych niezbalansowanych, co usprawnia proces porównywania wyników między różnymi metodami.

W tabeli 1.2 zaprezentowano otrzymane w trakcie eksperymentów wyniki. Algorytm RGA porównano z sześcioma podobnymi technikami przetwarzania wstępnego. Dodatkowo przedstawione zostały wyniki klasyfikacji oryginalnych zbiorów danych, bez etapu przetwarzania (kolumna noPRE). Konfiguracja algorytmu RGA pozwoliła na przetestowanie 81 kombinacji wartości parametrów. Liczba najbliższych sąsiadów *k* przyjmowała wartości 3, 5 oraz 7. Wartości *complexity_th* to 0,2, 0,3 i 0,4. Parametr *distance_th* przyjmował wartości 10, 15, 25. Z kolei *card_redundancy*

miało przypisane wartości 20, 30 oraz 40. Szczegółowe wyniki klasyfikacji w zależności od wybranych dwóch spośród wszystkich opisanych parametrów, a mianowicie parametru *k* i *complexity_th*, można znaleźć w [3].

Analiza rezultatów ujawnia, że metoda RGA osiągała największe wartości miary AUC w większości przypadków. Oznacza to, że algorytm spełnia swoje zadanie: sprawdza się w różnych okolicznościach, dostosowując swoje działanie do złożoności problemu. Opisywana metoda RGA okazała się skuteczniejsza od innych w przypadku siedmiu zbiorów danych. Eksperymenty na czterech spośród wszystkich zbiorów danych skutkowały dzieleniem najlepszego wyniku przez algorytm RGA z innymi metodami o tych samych wartościach AUC. Rezultaty osiągnięte w ramach badań na trzech zbiorach danych były nieznacznie lepsze w przypadku dwóch innych algorytmów (Safe-Level SMOTE oraz *SMOTE RSB*$_{\star}$). W tabeli 1.3 przedstawiono podsumowanie porównania. Zawiera sumaryczne liczby przypadków, w których każdy z algorytmów osiągnął najwyższy wynik. Pierwsza wartość oznacza liczbę zwycięstw w sensie absolutnym (gdy żadna inna metoda nie osiągnęła takiego samego wyniku), druga zaś wskazuje na liczbę sytuacji, w których więcej niż jedna metoda miała taki sam najlepszy wynik.

TABELA 1.2. **Wyniki klasyfikacji – porównanie metody RGA z sześcioma innymi algorytmami oraz klasyfikacją z pominięciem kroku przetwarzania wstępnego**

Zbiór danych	noPRE	SMOTE	S-ENN	Border-S	SafeL-S	S-RSB	VISROT	RGA
ecoli-0-1 vs 5	0,8159	0,7977	0,8250	0,8318	0,8568	0,7818	0,8636	**0,8886**
ecoli-0-1 vs 2-3-5	0,7136	0,8377	0,8332	0,7377	0,7550	0,7777	0,7314	**0,8673**
ecoli-0-1-4-6 vs 5	0,7885	**0,8981**	**0,8981**	0,7558	0,8519	0,8231	0,8366	0,8654
ecoli-0-1-4-7 vs 5-6	0,8318	0,8592	0,8424	0,8420	0,8197	0,8670	0,8220	**0,8853**
ecoli-0-3-4-7 vs 5-6	0,7757	0,8568	0,8546	0,8427	0,7995	0,8984	0,8471	**0,9113**
ecoli-0-4-6 vs 5	0,8168	0,8701	0,8869	0,8615	0,8923	**0,9476**	0,8060	0,8891
ecoli-0-6-7 vs 3-5	0,8250	0,8500	0,8125	0,8550	0,7950	0,8525	0,8500	**0,8700**
ecoli-0-2-6-7 vs 3-5	0,7752	0,8155	0,8179	0,8352	**0,8380**	0,8227	0,7977	0,8327
glass-0-1-6 vs 5	**0,8943**	0,8129	0,8743	0,8386	0,8429	0,8800	**0,8943**	**0,8943**
glass-0-4 vs 5	**0,9941**	0,9816	0,9754	**0,9941**	0,9261	**0,9941**	**0,9941**	**0,9941**
glass-0-6 vs 5	**0,9950**	0,9147	0,9647	**0,9950**	0,9137	0,9650	**0,9950**	**0,9950**
glass5	0,8976	0,8829	0,7756	0,8854	0,8939	0,9232	0,9951	**0,9976**
led7digit-0-2-4-5-6-7-8-9 vs 1	0,8788	0,8908	0,8379	0,8908	0,9023	0,9019	0,8918	**0,9056**
page-blocks-1-3 vs 4	**0,9978**	0,9955	0,9888	0,9978	0,9831	**0,9978**	0,9944	**0,9978**
yeast-0-3-5-9 vs 7-8	0,5868	0,7047	0,7024	0,6228	0,7296	**0,7400**	0,6868	0,7105

TABELA 1.3. Podsumowanie wyników klasyfikacji dla poszczególnych metod: liczba pełnych/współdzielonych zwycięstw

	noPRE	SMOTE	S-ENN	Border-S	SafeL-S	S-RSB	VISROT	RGA	ŁĄCZNIE
Test	0/4	0/1	0/1	0/1	1/0	2/2	0/3	7/4	10/5

W tabeli 1.4 przedstawiono wyniki testu Friedmana, wskazujące na skuteczność algorytmów we wszystkich zbiorach danych, opierając się na wartościach AUC. Metoda RGA, wykazując się bardzo stabilnym poziomem skuteczności, osiągnęła w rankingu pierwszą pozycję. Drugie miejsce uzyskała metoda *SMOTE RSB*$_*$ (również stosująca dodatkowy etap filtrowania danych). Na trzeciej pozycji pod względem całkowitej skuteczności uplasowała się metoda VISROT. Najgorsze wyniki spośród technik przetwarzania wstępnego osiągnął algorytm SMOTE-ENN.

TABELA 1.4. Wyniki testu Friedmana

Algorytm	Średnia ranga
RGA	1,93
S_RSB	3,43
VISROT	4,73
Border_S	4,83
SMOTE	4,97
SafeL_S	5,13
S_ENN	5,3
noPRE	5,67

Zgodnie z przewidywaniem oraz kolejny raz potwierdzając wcześniejsze analizy, klasyfikacja danych niezbalansowanych standardowymi klasyfikatorami z pominięciem kroku wstępnego przetwarzania skutkuje najniższą efektywnością. Dokładniejsza weryfikacja pokazuje, że brak czynności podjętych w celu zrównania liczebności próbek z poszczególnych klas może sprawdzać się jedynie w przypadku zbiorów o niedużej złożoności. Widać to szczególnie w przypadku zbiorów danych glass-0-1-6 vs 5, glass-0-4 vs 5, glass-0-6 vs 5, które posiadają pusty obszar graniczny (wskazujący na niższą złożoność rozkładu), gdy wiele algorytmów (w tym brak użycia jakiejkolwiek metody przetwarzania wstępnego) uzyskało dobre wyniki. Niemniej jednak pozostałe zbiory danych, z których większość posiada skomplikowaną charakterystykę, pozwoliły wykazać skuteczność stosowania przetwarzania wstępnego, w szczególności metodą RGA.

Podsumowanie

W niniejszej pracy opisano metodę bazującą na obliczeniach granularnych oraz teorii zbiorów przybliżonych wzbogacając klasyczne podejście znane z technik rozszerzających algorytm SMOTE o możliwość adaptacji do złożoności problemu. Główne cele powstania metody RGA można podsumować następująco:

- zwiększenie liczebności próbek z klasy mniejszościowej ukierunkowane na umożliwienie jak najwyższego poziomu rozpoznawalności przez standardowe klasyfikatory;
- uwzględnienie dodatkowych trudności, pojawiających się w trakcie eksploracji niezbalansowanych zbiorów danych poprzez zastosowanie granul informacyjnych;
- wprowadzenie kroku filtrowania bazującego na teorii zbiorów przybliżonych w celu usunięcia wszelkich niejednoznaczności, które mogłyby przyczynić się do zwiększenia złożoności rozkładu danych;
- usprawnienie procesu doboru parametrów do konkretnej charakterystyki danych.

Rezultaty przeprowadzonych eksperymentów potwierdziły poprawność przyjętych założeń. Połączenie techniki ukierunkowanego generowania przykładów z klasy mniejszościowej z dodatkowym krokiem filtrowania danych okazało się skutecznym rozwiązaniem. Badania zdecydowanie ukazały wartościowość podejścia opartego na uwzględnianiu lokalnej charakterystyki zbiorów danych w przetwarzaniu wstępnym. W szczególności obiecujące jest automatyczne dobieranie istotnych parametrów algorytmu, pozwalające na lepsze dostosowanie podejmowanych kroków przetwarzania do konkretnego problemu.

W ramach kolejnych eksperymentów może zostać przeprowadzona weryfikacja wpływu liczby atrybutów na skuteczność proponowanego rozwiązania. Równie istotna będzie analiza złożoności metody RGA w zależności od rozmiaru danych, uwzględniając także w porównaniu złożoność alternatywnych algorytmów.

Bibliografia

[1] Batista G.E.A.P.A., Prati R.C., Monard M.C., *A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data*, „Association for Computing Machinery" 2004, 6 (1)

[2] Borowska K., Stepaniuk J., A rough-granular approach to the imbalanced data classification problem, „Applied Soft Computing" 2019

[3] Borowska K., Stepaniuk J., Granular Computing and Parameters Tuning in Imbalanced Data Preprocessing [w:] K. Saeed, W. Homenda (red.), Computer Information Systems and Industrial Management. CISIM 2018, Lecture Notes in Computer Science 11127, Springer, 2018, s. 233–245

[4] Borowska K., Stepaniuk J., Rough Sets in Imbalanced Data Problem: Improving Re–sampling Process [w:] K. Saeed, W. Homenda, R. Chaki (red.), Computer Information Systems and Industrial Management. CISIM 2017, Lecture Notes in Computer Science 10244, Springer, 2017, s. 459–469

[5] Bunkhumpornpat C., Sinapiromsaran K., Lursinsap C., Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem [w:] T. Theeramunkon, B. Kijsirikul, N. Cercone, TB. Ho (red.), Advances in Knowledge Discovery and Data Mining. PAKDD 2009, Lecture Notes in Computer Science 5476, Springer, 2009, s. 475–482

[6] Chawla N., Bowyer W.K., Hall L.O., Kegelmeyer W.P., *SMOTE: Synthetic Minority Oversampling Technique*, „AI Access Foundation" 2002, 16 (1)

[7] *Data Sets – UCI Machine Learning Repository,* strona internetowa, https://archive.ics.uci.edu/ml/datasets.php [dostęp: 27.02.2021]

[8] Fernadez A., Herrera F., Chawla N.V., *SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary*, „Journal of Artificial Intelligence Research" 2018, 61

[9] Galar M., Fernandez A., Barrenechea E., Bustince H., Herrera F., *A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches*, „IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)" 2012, 42 (4)

[10] Garcia S., Luengo J., Herrera F., *Data Preprocessing in Data Mining*, „Springer International Publishing" 2015, 72

[11] Han H., Wang WY., Mao BH., *Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning* [w:] DS. Huang, XP. Zhang, GB. Huang (red.), *Advances in Intelligent Computing.* ICIC 2005. Lecture Notes in Computer Science 3644, Springer, 2005, s. 878–887

[12] He H., Garcia E.A., *Learning from Imbalanced Data*, „IEEE Transactions on Knowledge and Data Engineering" 2009, 21 (9)

[13] Hu S., Liang Y., Ma L., He Y., *MSMOTE: Improving Classification Performance When Training Data is Imbalanced*, Second International Workshop on Computer Science and Engineering, 2009, s. 13–17

[14] Jo T., Japkowicz N., *Class Imbalances Versus Small Disjuncts*, „Association for Computing Machinery" 2004, 6 (1)

[15] Krawczyk B., *Learning from imbalanced data: open challenges and future directions*, „Progress in Artificial Intelligence" 2016, 5 (4)

[16] Lopez V., Fernandez A., Garcia S., Palade V., Herrera F., An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, „Information Sciences" 2013, 250

[17] Napierala K., Stefanowski J., Types of minority class examples and their influence on learning classifiers from imbalanced data, „Journal of Intelligent Information Systems" 2016, 46 (3)

[18] Napierała K., Stefanowski J., Wilk S., *Learning from Imbalanced Data in Presence of Noisy and Borderline Examples* [w:] M. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, Q. Hu (red.), *Rough Sets and Current Trends in Computing.* RSCTC 2010. Lecture Notes in Computer Science 6086, Springer, 2010, s. 158–167

[19] Pawlak Z., *Rough sets*, „International Journal of Computer & Information Sciences" 1982, 11 (5)

[20] Pawlak Z., Skowron A., *Rudiments of Rough Sets*, „Information Sciences" 2007, 177 (1)

[21] Pedrycz W., *Granular Computing: An Introduction* [w:] N. Kasabov (red.), *Future Directions for Intelligent Systems and Information Sciences*. Studies in Fuzziness and Soft Computing 45, Physica, s. 309–328

[22] Pietruszkiewicz W., *Practical Evaluation, Issues and Enhancements of Applied Data Mining* [w:] A. Abd Manaf, A. Zeki, M. Zamani, S. Chuprat, E. El-Qawasmeh (red.), *Informatics Engineering and Information Science*. ICIEIS 2011, Communications in Computer and Information Science 252, Springer, 2011, s. 717–731

[23] Ramentol E., Caballero Y., Bello R., Herrera F., SMOTE-RSB *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory, „Knowledge and Information Systems" 2012, 33 (2)

[24] Saez J.A., Luengo J., Stefanowski J., Herrera F., SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering, „Information Sciences" 2015, 291

[25] Skowron A., Stepaniuk J., *Tolerance Approximation Spaces*, „IOS Press" 1996, 27 (2–3)

[26] Stefanowski J., *Dealing with Data Difficulty Factors While Learning from Imbalanced Data* [w:] S. Matwin, J. Mielniczuk (red.), *Challenges in Computational Statistics and Data Mining*. Studies in Computational Intelligence 605, Springer, 2015, s. 333–363

[27] Stepaniuk J., *Rough–Granular Computing in Knowledge Discovery and Data Mining*, „Studies in Computational Intelligence" 152, Springer, 2009

[28] Sun Y., Kamel M.S., Wong A.K.C., Wang Y., *Cost-sensitive boosting for classification of imbalanced data*, „Pattern Recognition" 2007, 40 (12)

[29] Weiss G.M., *Mining with Rarity: A Unifying Framework*, „Association for Computing Machinery" 2004, 6 (1)

[30] Wilson D, Martinez T.R., *Improved heterogeneous distance functions*, „Journal of Artificial Intelligence Research" 1997, 6 (1)

Granular computing in imbalanced data preprocessing

Abstract: Imbalanced data problem occurs when the numer of examples representing a class of major interest is significantly lower than the number of instances from another class. It has been one of the most interesting and important data mining problems for more than two decades. Recent studies have shown that the issue of imbalanced data should not be considered only in the context of information deficit of one of the classes. In fact, the decrease in the quality of standard classifiers when confronted with this type of datasets is mainly related to the high complexity of the data distribution. The complexity increases when additional difficulties such as minority class decomposition, class overlapping and the presence of noise occur in the data characteristics. Therefore dedicated solutions are required, in particular taking into account the local characteristics of the distribution. This paper describes an algorithm that applies information granules and rough sets theory for targeted modification of the learning set and enables to automate parameter selection.

Keywords: class imbalance, data preprocessing, granular computing, information granules, rough sets, SMOTE

DOI: 10.24427/978-83-66391-96-3_2

Rozdział 2
Tolerancyjny algorytm V-Detector

Andrzej Chmielewski
Wydział Informatyki, Politechnika Białostocka

Streszczenie: V-Detector jest przykładem algorytmu selekcji negatywnej, bazującym na idei zaczerpniętej ze sztucznych systemów immunologicznych. Przede wszystkim jest wykorzystywany do detekcji anomalii w wielowymiarowych zbiorach danych. Aby osiągnąć ten cel generowane są zbiory detektory, zwykle w sposób losowy, wyłącznie na podstawie informacji o tzw. normalnym zachowaniu i żadne przykłady negatywnych obserwacji nie są wymagane. Głównym problemem zidentyfikowanym podczas zastosowania tego algorytmu jest jego skalowalność i duża złożoność obliczeniowa. W artykule przedstawiono nowe podejście do budowy detektorów, inspirowane ideą tolerancyjnych zbiorów przybliżonych. Za pomocą tego algorytmu można wykryć nie tylko elementy odstające, ale również te, które są bardzo podobne do obydwu klas *self* oraz *nonself*, a ich poprawna klasyfikacja wymaga dodatkowej weryfikacji. Częściowo rozwiązuje również problem skalowalności, ponieważ tego rodzaju detektory mają wyższą moc dyskryminacyjną w porównaniu z wersją podstawową.

Słowa kluczowe: sztuczne systemy immunologiczne, selekcja negatywna, detekcja anomalii, zbiory przybliżone

Wprowadzenie

Algorytm selekcji negatywnej (ang. Negative Selection Algorithm – NSA) jest jednym z kilku głównych algorytmów rozwijanych w ramach dziedziny sztucznych systemów immunologicznych, inspirowanych mechanizmami układów odpornościowych (ang. Nature Immune System – NIS). Jego działanie opiera się na mechanizmach kontrolujących proces dojrzewania tymocytów (młodych limfocytów typu T), w którym przeżywają jedynie te, które nie rozpoznają żadnej własnej molekuły (komórki własnej, określanej jako *self*) [2], czego wynikiem jest zbiór detektorów, rozpoznających jedynie patogeny (komórki obce, określane jako *nonself*). Od strony klasyfikacyjnej, interesującą cechą NIS jest brak konieczności dysponowania przykładami zagrożeń w procesie ich rozróżniania od komórek własnych. Oznacza to, że tego rodzaju algorytmy mogą być stosowane do wykrywania nowych, dotąd nienapotkanych patogenów, które mogą stanowić zagrożenie, a zwykle nie są wykrywane przez tradycyjne

systemy oparte o sygnatury. Powyższa właściwość sprawia, że tego rodzaju rozwiązania są bardzo atrakcyjne w wielu zastosowaniach, szczególnie do tworzenia różnego rodzaju systemów bezpieczeństwa komputerowego, w których największe zagrożenie stanowią ataki, których sygnatury nie zostały jeszcze odnotowane.

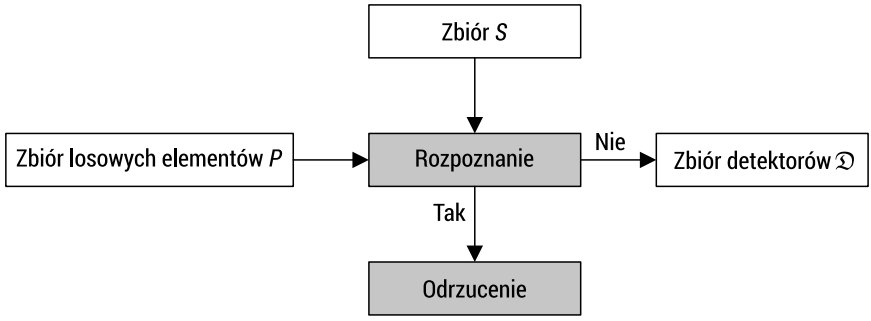
W ostatnich latach zaproponowane zostały liczne algorytmy NSA, m.in. [5, 7, 10], które mają swoje szerokie zastosowanie w wielu obszarach, takich jak choćby bezpieczeństwo komputerowe, filtrowanie spamu, wykrywanie wirusów komputerowych, rozpoznawanie pisma odręcznego (szczegóły m.in. w [6]), w których podstawą jest klasyfikacja binarna. Jednakże, w wielu przypadkach, podział na dwa rozłączne zbiory *self* i *nonself* zwykle nie jest możliwy, ze względu na możliwość występowania elementów o zbyt bliskim podobieństwie do obydwu klas, mierzonym tzw. miarą powinowactwa (odległości pomiędzy nimi). Ewentualna błędna klasyfikacja może pociągać za sobą bardzo negatywne skutki, szczególnie w odniesieniu do systemów bezpieczeństwa. W przypadkach niepewności, takie elementy powinny zostać poddane dodatkowej i bardziej szczegółowej ocenie, również przy użyciu innych technik, często bardziej złożonych czasowo i obliczeniowo od tych oferowanych przez NSA.

Jedną z bardziej efektywnych metod radzenia sobie z klasyfikacją elementów „niepewnych" są zbiory przybliżone (ang. Rough Set Theory – RST) [12]. Pewne próby zastosowania wybranych ich własności w NSA były już podejmowane. Przykładowo w [1] limfocyty, które są odpowiedzialne za wykrywanie i unicestwianie patogenów w NIS, zostały zaimplementowane jako reguły decyzyjne. Zaproponowane rozwiązanie wykorzystywało „przybliżone" limfocyty reprezentowane w postaci wektorów o wartościach rzeczywistych do emulowania niepełnego wiązania dobrze znanego w NIS. W [13] zbiory przybliżone zostały wykorzystane do redukcji liczby atrybutów obiektów w AIS, co bardzo korzystnie wpłynęło na redukcję złożoności obliczeniowej zaproponowanego algorytmu. W tym przypadku testy przeprowadzono na bardzo popularnym zbiorze KDD Cup 1999, będącym wielowymiarowym zbiorem danych, często traktowanym jako referencyjnym, z informacjami zarówno o legalnych połączeniach sieciowych jak i opisami różnego rodzaju ataków.

W tym artykule proponowany jest tolerancyjny algorytm V-Detector inspirowany ideą zaczerpniętą z RST. Wstępne tego rodzaju rozwiązanie było już prezentowane w [3], gdzie wartości dolnej aproksymacji, zarówno w przypadku reprezentacji binarnych (b-detektorów), jak i o wartościach rzeczywistych (ν-detektorów), były arbitralnie dobierane w oparciu o wyniki wstępnie przeprowadzonych eksperymentów. W tym przypadku, wartość ta jest uzależniona od liczby elementów znajdujących się w pobliżu badanego obiektu, a przez to automatycznie adoptuje się do lokalnych warunków rozpatrywanego środowiska.

2.1. Algorytm selekcji negatywnej

Algorytm selekcji negatywnej (NSA) [2] jest inspirowany procesem dojrzewania limfocytów T (detektorów patogenów) oraz mechanizmami, dzięki którym rozróżniane są antygeny „własne“ (zwane *self*) od antygenów „obcych” (zwanych *nonself*). W dużym uproszczeniu, dojrzałe limfocyty charakteryzują się umiejętnością wykrywania komórek obcych i jednoczesnym tolerowaniem komórek własnych, aby nie doszło do reakcji zagrażających życiu gospodarza. Unikalną cechą algorytmu jest wykorzystywanie w procesie generowania detektorów jedynie obserwacji *self*, stąd każdy układ immunologiczny jest unikalny, dopasowany do ochrony konkretnego systemu. Podobnie jak w przypadku układu odpornościowego, kandydaci na detektory są poddawani próbie wykrycia antygenów własnych. Jeśli co najmniej jeden antygen *self* zostanie rozpoznany, musi zostać wyeliminowany. Ogólny schemat tego procesu został przedstawiony na rysunku 2.1.



RYSUNEK 2.1. **Ogólny schemat procesu generowania detektorów $\mathfrak{D}$ za pomocą algorytmu NSA**

W procesie detekcji patogenów (*nonself*), biorą udział jedynie limfocyty dojrzałe, czyli takie, które tolerując wszystkie komórki własne są jednocześnie w stanie związać się jedynie z molekułami komórek patogennych. W idealnym przypadku organizm powinien posiadać tak znaczny repertuar strukturalnie różnych (reagujących na różne molekuły patogenów) limfocytów, aby jak najwięcej patogenów mogło być wykrytych i zneutralizowanych, co jest warunkiem skutecznej ochrony organizmu i zapewnienia mu tym samym bezpieczeństwa.

Formalnie algorytm NSA możemy przedstawić następująco. Oznaczmy zbiór $\mathcal{U}$ jako uniwersum, czyli zbiór wszystkich możliwych molekuł. W jego ramach możemy wyodrębnić dwa rozłączne podzbiory $\mathcal{S}$ oraz $\mathcal{N}$. Pierwszy z nich $\mathcal{S}$ ($\mathcal{S} \subset \mathcal{U}$) zawiera wszystkie molekuły własne (*self*), podczas gdy $\mathcal{N}$ ($\mathcal{N} \subset \mathcal{U}$), będący jego dopełnieniem, stanowi przestrzeń dla wszystkich molekuł *nonself*. Przyjmijmy, że $\mathfrak{D} \subset \mathcal{U}$ oznacza zbiór detektorów, a *match*(*d,u*) jest funkcją weryfiującą, czy detektor $d \in \mathfrak{D}$ rozpoznaje molekułę $u \in \mathcal{U}$. Zwykle *match*(*d,u*) jest modelowane jako metryka odległości lub miara podobieństwa. Wówczas możemy powiedzieć, że *match*(*d,u*) = `true`

tylko w przypadku, gdy $dist(d,u) \leq \delta$, gdzie *dist* jestodległością, a δ jest predefiniowaną wartością progową, określającą zasięg (moc dyskryminacyjną) detektora. W ostatnich latach były rozpatrywane różnego rodzaju funkcje dopasowywania m.in. w [8] oraz [11], w zależności od wymiarowości zbioru $\mathcal{U}$ oraz liczebności zbioru $\mathcal{S}$.

Od strony formalnej problem sprowadza się do konstrukcji takiego zbioru detektorów $\mathfrak{D}$, aby:

$$match(u,d) = \begin{cases} \texttt{false} \text{ jeżeli } u \in \mathrm{S} \\ \texttt{true} \text{ jeżeli } u \in \mathrm{N} \end{cases} \tag{2.1}$$

dla każdego detektora $d \in \mathfrak{D}$.

Najprostsze rozwiązanie tego problemu, wywodzące się z biologicznych mechanizmów selekcji negatywnej, składa się z następujących etapów:

(a) inicjalizacja $\mathfrak{D}$ jako zbioru pustego, $\mathfrak{D} = \emptyset$,
(b) wygenerowanie losowego detektora d,
(c) jeżeli $match(d,s) = \texttt{false}$ dla wszystkich $s \in \mathcal{S}$, dodaj d do zbioru $\mathfrak{D}$,
(d) powtarzaj kroki (b) oraz (c) aż do uzyskania wystarczającej liczby detektorów pokrywających w odpowiednim stopniu przestrzeń $\mathcal{U}$.

Jak dotąd, algorytmy NSA były opracowywane do operowania w przestrzeni ciągów binarnych (*b*-detektorów) oraz liczb rzeczywistych (*v*-detektorów). W każdym z przypadków, rozpatrywane były różne miary podobieństwa dostosowane do danej dziedziny. W następnym rozdziale został przedstawiony algorytm V-Detector operujący w przestrzeni liczb rzeczywistych, który zostanie wyposażony we właściwości RST.

2.2. Algorytm V-Detector

Algorytm V-Detector został zaproponowany w [10]. Jest on przystosowany do operowania na obiektach reprezentowanych w postaci wektorów o znormalizowanych wartościach rzeczywistych, które graficznie są przedstawiane w postaci punktu w *d*-wymiarowym jednostkowym hipersześcianie $\mathcal{U} = [0,1]^d$. Każda próbka *self* ($s_i \in \mathcal{S}$) jest w nim reprezentowana jako hipersfera $s_i = (c_i, r_s)$, $i = 1, ..., l$, gdzie l jest liczbą próbek *self*, $c_i \in \mathcal{U}$ jest jej środkiem, a r_s promieniem. Przyjmuje się, że promień r_s jest identyczny dla wszystkich próbek *self*.

Podczas klasyfikacji każdy punkt $u \in \mathcal{U}$ położony wewnątrz dowolnej hipersfery s_i jest traktowany jako obiekt *self*, w pozostałych przypadkach jako *nonself*. Przykład działania algorytmu w przestrzeni 2-wymiarowej został zaprezentowany na rysunku 2.2(a), na którym można zauważyć, że przestrzenie pokrywane przez zbiory $\mathcal{S}$ oraz $\mathfrak{D}$ są rozłączne. W idealnym przypadku, cała przestrzeń $\mathcal{U}$ powinna być pokryta w całości przez *v*-detektory, co jest zadaniem złożonym zarówno pod względem czasowym jak i obliczeniowym.

Do obliczania podobieństwa pomiędzy obiektami zwykle stosowana jest odległość euklidesowa. Konsekwencją tego wyboru jest hipersferyczny kształt detektorów d_j: $d_j = (c_j, r_j)$, $j = 1, ..., p$, gdzie p jest liczbą detektorów. W przeciwieństwie do obiektów *self*, promień r_j nie jest wartością stałą i jest wyliczany jako odległość pomiędzy losowo wygenerowanym punktem c_j (nie leżącym wewnątrz żadnej z s_i) a najbliższym obiektem *self*. Dodatkowym warunkiem jest to, aby odległość ta była większa niż r_s; w przeciwnym przypadku detektor nie jest tworzony. Zapobiega to tworzeniu zbyt małych detektorów (o niskiej mocy dyskryminacyjnej), które mogłyby negatywnie wpłynąć na proces klasyfikacji. Najbardziej pożądane są detektory o dużych promieniach r_j, gdyż wówczas mniejsza ich liczba jest potrzebna do pokrycia przestrzeni $\mathcal{U}$.

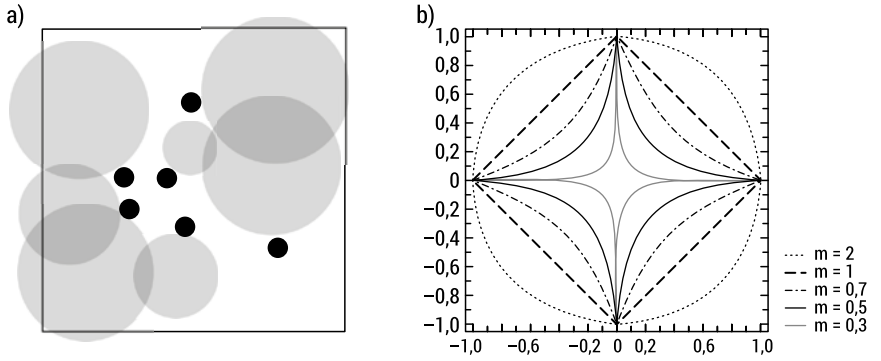
Formalnie r_j można więc zdefiniować jako:

$$r_j = \min_{1 \le i \le l} dist\left(c_j, c_i\right) - r_s . \tag{2.2}$$

W pierwotnie zaproponowanej wersji, algorytm V-Detector wykorzystywał odległość euklidesową do mierzenia podobieństwa pomiędzy obiektami, stąd zarówno obiekty *self* jaki i detektory przybierały kształt hipersfer. Należy jednak mieć na względzie, że metryka euklidesowa jest szczególnym przypadkiem metryki Minkowskiego L_m definiowanej jako:

$$L_m\left(x, y\right) = \left(\sum_{i=1}^{d} \left|x_i - y_i\right|^m\right)^{\frac{1}{m}}, \tag{2.3}$$

gdzie: $x = (x_1, x_2, \ldots x_d)$ oraz $y = (y_1, y_2, \ldots, y_d)$ są punktami w przestrzeni $\Re^d$.



RYSUNEK 2.2. (a) Przykład działania algorytmu V-Detector w przestrzeni 2D. Czarne oraz szare koła oznaczają odpowiednio próbki *self* oraz *v*-detektory, (b) Kształty jednostkowych sfer dla wybranych metryk L_m

W szczególności, metryka L_2 jest właśnie odległością euklidesową, L_1 odległością Manhattan, a L_∞ odległością Czebyszewa. Ponadto, definiowane są również tzw. metryki ułamkowe (dla $0 < m < 1$), które są lepiej przystosowane do operowania na wektorach w przestrzeniach wielowymiarowych [4].

W zależności od zastosowanej w algorytmie V-Detector metryki uzyskuje się inny kształt *v*-detektora. Na rysunku 2.2(b) zostały przedstawione kształty jednostkowych sfer dla wybranych norm L_m w przestrzeni dwuwymiarowej dla $m = 2$ (najbardziej zewnętrzna w postaci okręgu), 1, 0.7, 0.5, oraz 0.3.

2.3. Zbiory przybliżone w algorytmach selekcji negatywnej

W ostatnich latach zaproponowanych zostało wiele różnych algorytmów NSA, m.in. [5, 7, 10], które powstały w celu, ogólnie rzecz ujmując, rozwiązania problemu detekcji anomalii (próbek odstających) w wielu dziedzinach np. systemach bezpieczeństwa komputerowego, filtrowania spamu, detekcji wirusów oraz rozpoznawania pisma odręcznego. W celu szerszego spojrzenia na tą tematykę warto zapoznać się choćby z przeglądem zawartym w [6]. Duże zainteresowanie NSA świadczy o potrzebie rozwijania algorytmów, które wykorzystują inne podejście niż te powszechnie stosowane, oparte np. o sygnatury. Kluczową przewagą jest oczywiście fakt możliwości wykrywania nowych typów anomalii bez posiadania informacji o niej.

Jednakże w wielu dotąd opublikowanych artykułach sygnalizowany był problem z niezadowalającą ich skutecznością oraz dużym współczynnikiem błędnie sklasyfikowanych obiektów [14], czego przyczyną było zbyt duże podobieństwo pomiędzy obiektami *self* oraz *nonself*. Próby zastosowania innych miar powinowactwa, często skutkowało poprawą wyników [4], lecz nie było to rozwiązanie uniwersalne.

Jednym z efektywniejszych podejść do danych trudnorozróżnialnych są zbiory przybliżone [9, 12]. Pewne próby inspirowania się tą teorią w odniesieniu do NSA były już publikowane. W [1] limfocyty, które są odpowiedzialne za wyszukiwanie i unicestwianie patogenów w NIS, zostały zaimplementowane jako reguły decyzyjne dla obydwu klas *self* oraz *nonself*. Zaproponowane rozwiązanie wykorzystywało zbiory przybliżone do emulowania niepełnego wiązania znanego z NIS. W [13] zbiory przybliżone zostały użyte do redukcji liczby atrybutów w AIS. W tym przypadku testy były przeprowadzane na referencyjnym zbiorze KDD Cup 1999, wykorzystywanym do oceny detekcji anomalii w ruchu sieciowym.

W dalszej części artykułu zostaną przedstawione podstawowe definicje związane z RST, a następnie tolerancyjny algorytm V-Detector wykorzystujący pewne ich własności. Warto nadmienić, że podobne rozwiązania były już prezentowane w [3], gdzie górna aproksymacja, zarówno w przypadku *b*- oraz *v*-detektorów, była arbitralnie dobierana na podstawie wstępnych eksperymentów przeprowadzanych na podzbiorze danych. W tym przypadku, wartość ta dostosowuje się do bieżących warunków poprzez uwzględnienie pewnej liczby najbliższych obiektów *self*.

2.4. Teoria zbiorów przybliżonych

W rozdziale tym zostanie przedstawiony krótki opis, zaczerpnięty z [9], podstawowych zasad wprowadzonych przez koncepcję zbiorów przybliżonych.

Na początek wprowadźmy definicję systemu informacyjnego, na którym oparta jest koncepcja przestrzeni aproksymacyjnej.

Definicja 1. (system informacyjny)

System informacyjny jest parą $IS = (U, A)$, gdzie U jest niepustym, skończonym zbiorem obiektów zwanym uniwersum, a A jest niepustym skończonym zbiorem atrybutów. Każdy atrybut $a \in A$ jest traktowany jako funkcja a: $U \rightarrow V_a$, gdzie V_a jest wartością zbioru a.

Definicja 2. (przestrzeń aproksymacyjna)

Parametryczna przestrzeń aproksymacyjna $AS_{\#,\$}$ dla systemu informacyjnego $IS = (U, A)$ jest definiowana jako $AS_{\#,\$} = (U, I_{\#}, \nu_{\$})$, gdzie:

- U jest niepustym zbiorem obiektów,
- $I_{\#} : U \rightarrow P(U)$ jest funkcją niepewności,
- $\nu_{\$} : P(U) \times P(U) \rightarrow [0,1]$ jest funkcją inkluzji przybliżonej.

Dla każdego obiektu funkcja niepewności definiuje zbiór podobnie opisanych obiektów.

Definicja 3. (funkcja niepewności)

Funkcja niepewności I_B^{ε} jest definiowana jako:

$$I_B^{\varepsilon}(x) = \bigcap_{a \in B} I_a^{\varepsilon_a}(x), \tag{2.4}$$

gdzie $x \in U, B \subseteq A$, $\varepsilon = (\varepsilon_a : a \in B)$ jest wektorem progów takich, że $\varepsilon_a \geq 0$ dla $a \in B$, $I_B^{\varepsilon}(x) = \{ y \in U : d_a(x,y) \leq \varepsilon_a \}$, a $d_a : U \times U \rightarrow [0, \infty)$ jest miarą odległości.

Funkcja inkluzji przybliżonej określa stopień inkluzji zbioru X w zbiorze Y, gdzie $X, Y \subseteq U$.

Definicja 4. (funkcja inkluzji przybliżonej)

1. Standardowa inkluzja przybliżona $v_{SRI}(X,Y)$ jest zbiorem X w zbiorze Y jest definiowana jako:

$$V_{SRI} = \begin{cases} \dfrac{card(X \cap Y)}{card(X)} & \text{jeżeli } X \neq \varnothing \\ 1 & \text{w p.p.} \end{cases} \tag{2.5}$$

2. Przybliżona inkluzja $v_{l,u}(X,Y)$ zbioru X w zbiorze Y jest definiowana jako:

$$v_{l,u}(X,Y) = f_{l,u}(v_{\mathrm{SRI}}(X,Y)), \tag{2.6}$$

gdzie:

$$f_{l,u}(t) = \begin{cases} 0 & \text{jeżeli} \quad 0 \le t \le l \\ \dfrac{t-l}{u-l} & \text{jeżeli} \quad l < t < u \\ 1 & \text{jeżeli} \quad t \ge u \end{cases} \tag{2.7}$$

gdzie: $0 \le l < u \le 1$.

Aproksymacja podzbioru U jest definiowana następująco.

Definicja 5. (aproksymacja podzbioru uniwersum)

Dolna i górna aproksymacja będąca podzbiorem $X \subseteq U$ w $AS_{\#,\$}$ jest de*fi*niowana jako:

$$LOW(AS_{\#,\$}, X) = \{x \in U : \nu_{\$}(I_{\#}(x), X) = 1\},$$

$$UPP(AS_{\#,\$}, X) = \{x \in U : \nu_{\$}(I_{\#}(x), X) > 0\}.$$

Symbole #,$ oznaczają wektory parametrów, które mogą być dopasowywane w procesie budowania przestrzeni aproksymacji.

Definicja 6. (region graniczny)

Region graniczny, który zawiera obiekty niepewne, jest definiowany jako różnica pomiędzy górną i dolną aproksymacją, tj.

$$BND(AS_{\#,\$}, X) = UPP(AS_{\#,\$}, X) \setminus LOW(AS_{\#,\$}, \mathrm{X})$$

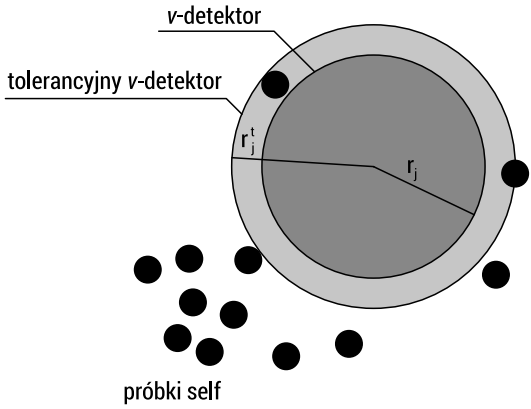
Definicja 7. (zbiór przybliżony)

Tolerancyjny zbiór przybliżony podzbioru $X \subseteq U$ jest definiowany jako para $(LOW(AS_{\#,\$}, X), UPP(AS_{\#,\$}, X))$.

2.5. Teoria zbiorów przybliżonych

Algorytm V-Detector jest skutecznym narzędziem klasyfikacyjnym szczególnie, gdy przestrzeń, w której operuje, nie jest zbyt wielowymiarowa. Niestety jest przy tym wrażliwy na odstające próbki *self*, które powodują, że znacznie większa liczba detektorów jest potrzebna do pokrycia przestrzeni $\mathcal{U}$, wydłużając w ten sposób proces zarówno uczenia się jak i klasyfikacji.

Zastosowanie zbiorów przybliżonych w odniesieniu do algorytmu V-Detector umożliwia wprowadzenie dodatkowego promienia tolerancji $r^t_j (r^t_j > = r_j)$, który jest odległością do najbliższej próbki *k-self* , zamiast do tej najbliższej. Stąd, każdy tolerancyjny *v*-detektor (v^t_k-detektor) w przestrzeni $\Re^d$ jest reprezentowany jako wektor $d^t_j = (c_j, r_j, r^t_j)$ o wartościach rzeczywistych, co graficznie zostało zobrazowane na rysunku 2.3.



RYSUNEK 2.3. **Przykład v^t_k-detektora przestrzeni 2D dla k = 2 oraz euklidesowej metryki podobieństwa**

Wprowadzenie dwóch różnych promieni dla pojedynczego detektora d_j zostało zainspirowane przez RST, gdzie r_j jest odpowiedzialny na klasyfikację wszystkich próbek *nonself* z całkowitą pewnością (dolna aproksymacja), podczas gdy r^t_j (górna aproksymacja) jest stosowana do określania obiektów, które mogą również być *nonself* (a przynajmniej są bardzo do nich podobne).

W oparciu o to założenie, podczas fazy klasyfikacji, cenzorowany obiekt jest przypisywany do jednej z 3 grup:

- *nonself* – gdy został całkowicie pokryty przez co najmniej jedną hipersferę z dolną aproksymacją r_j;
- *self* – gdy nie został pokryty przez żadną z hipersfer z górną aproksymacją r^t_j;
- *uncertain* – w pozostałym przypadku (gdy został pokryty przez co najmniej jedną hipersferę z górną aproksymacją r^t_j.

Jak już zostało to wspomniane, V-Detector jest algorytmem dosyć wymagającym pod względem czasowym jak i obliczeniowym. Ta cecha dotyczy obydwu faz tj. uczenia się oraz klasyfikacji. Proponując rozwiązanie przedstawione w tym rozdziale należy również wziąć pod uwagę ewentualny wzrost jego złożoności, warunkujący możliwość zastosowania go w konkretnych dziedzinach.

Ponieważ podczas fazy uczenia się odległość do wszystkich próbek jest zawsze obliczana, wartość r^t_j może być wyznaczona bez żadnego dodatkowego nakładu. Jest to o tyle istotne, gdyż w wielu zastosowaniach (np. w detekcji anomalii w ruchu

sieciowym, wykrywaniu wirusów) szybka adopcja do zaistniałej sytuacji (przebudowanie lub nawet wygenerowanie nowego zbioru $\mathfrak{D}$) jest często jedną z kluczowych własności wszystkich systemów klasyfikacyjnych. Stąd, ewentualny wzrost złożoności mógłby być czynnikiem zmniejszającym jego szanse na zastosowanie.

W fazie klasyfikacji, jedynie w przypadku obiektów *uncertain* jest wymagane przeprowadzenie dodatkowych operacji, które wspomogą ten proces. Ewentualny poziom wzrostu złożoności czasowej i obliczeniowej zależy tu jednak od zastosowanego rozwiązania. Należy jednak zauważyć, że w wielu przypadkach ta operacja niekoniecznie musi być przeprowadzana natychmiast i może być wykonywana np. w czasie bezczynności procesora, aby nie obciążać głównego procesu klasyfikacyjnego. Przykładem takiego rozwiązania, mogą być systemy antywirusowe, czy też systemy detekcji intruzów, które po wykryciu obiektu typu *uncertain*, mogą do czasu ostatecznej klasyfikacji wstrzymywać podejrzane połączenia lub też włączać dodatkowe sensory w celu zebrania bardziej szczegółowych informacji.

2.6. Wyniki eksperymentów

Eksperymenty zostały przeprowadzone na dwóch popularnych, wielowymiarowych i o dużej liczności zbiorach danych, czyli KDD Cup 1999 oraz Kyoto 2006+, które zwykle są stosowane do oceny skuteczności algorytmów detekcji intruzów w sieciach komputerowych. W celu zredukowania złożoności obliczeniowej testy były przeprowadzane nie na całych zbiorach, ale na ich częściach zawierających dane dla określonego protokołu transportowego: ICMP, TCP oraz UDP. Dla każdego z nich zostały wygenerowane oddzielne zbiory detektorów. W tym rozdziale zaprezentowane i przeanalizowane zostały wyniki uzyskane dla podzbiorów opisanych w tabeli 2.1, zawierających połączenia na protokole ICMP.

TABELA 2.1. Charakterystyka testowych baz danych wykorzystanych do eksperymentów

Zbiór danych	Protokół	Liczba atrybutów	Liczba próbek	Liczba próbek *self*
KDD Cup 1999	ICMP	22	11911	4428
Kyoto+	ICMP	21	4069	319

Ponadto, z obydwu zbiorów danych niektóre atrybuty zostały wyeliminowane (np. adres MAC źródłowy oraz docelowy w przypadku zbioru Kyoto+) ze względu na problemy związane z ich normalizacją (głównie ze względu na zbyt wiele unikalnych wartości symbolicznych). W przypadku zbioru KDD Cup 1999 wszystkie atrybuty niezwiązane z rozpatrywanym protokołem również nie były rozpatrywane, co w konsekwencji pozwoliło na znaczące zmniejszenie ich liczby z 42 do 22. Powyższe operacje pozwoliły na znaczącą redukcję wymiaru przestrzeni $\mathcal{U}$, co w przypadku algorytmu V-Detector ma kluczowe znaczenie w kontekście jego skuteczności i efektywności.

TABELA 2.2. Wyniki uzyskane na zbiorze Kyoto+

k	Δr [%]	*DR* [%]	*FAR* [%]	$\lvert V_t \rvert$ [%]
0	–	12,8	–	–
1	11,8	30,9	4,2	16,1
2	17,3	39,7	7,9	24,2
3	19,3	43,5	8,8	26,6
5	26,5	59,8	13,9	45,1
10	36,9	73,0	23,8	54,3
20	51,6	87,4	34,1	67,8
30	58,2	93,6	41,5	73,6

TABELA 2.3. Wyniki uzyskane na zbiorze KDD 1999 Cup

k	Δr [%]	*DR* [%]	*FAR* [%]	$\lvert V_t \rvert$ [%]
0	–	33,84	–	–
1	3,2	48,2	0,2	6,1
2	6,4	61,5	0,4	16,6
3	9,1	48,9	0,6	32,2
5	10,0	58,0	0,8	34,1
10	18,1	59,7	1,2	34,9
20	18,5	77,9	3,7	42,3
30	21,5	78,2	4,7	80,0

Tabele 2.2 oraz 2.3 zawierają wyniki uzyskane odpowiednio dla zbiorów Kyoto+ oraz KDD 1999 Cup. W obydwu przypadkach możemy zaobserwować procentowe wydłużenie długości promienia tolerancyjnych v-detektorów (Δr) wraz z rosnącą wartością k, czego rezultatem jest wzrost ich mocy dyskryminacyjnej oraz efektywniejsze pokrycie przestrzeni $\mathcal{U}$. W konsekwencji, w przypadku zbioru KDD Cup 1999, można zaobserwować znaczący wzrost współczynnika wykrycia (*DR*) z 33% dla standardowego algorytmu V-Detector ($k = 0$) do nawet 78% dla $k = 30$. Warty odnotowania jest w tym przypadku dosyć niski współczynnik fałszywego alarmu (*FAR*), co może oznaczać, że nieliczne obiekty *self* były odstające.

Podobne wyniki zostały również uzyskane w przypadku zbioru Kyoto+. O ile wartości Δr oraz *DR* rosły znacznie szybciej niż w przypadku drugiego ze zbiorów, to było to niestety powiązane ze znaczącym wzrostem wartości współczynnika FAR, co pokazuje, że wartość k należy dobierać niezwykle ostrożnie, aby zbyt duża liczba obiektów *nonself* nie została błędnie sklasyfikowana jako *self*.

Kolumna $|V_t|$ zawiera informację o procencie próbek sklasyfikowanych jako *uncertain*. Jak można było się tego spodziewać, wartość ta rośnie wraz ze wzrostem liczby k, nawet do 80% dla $k = 30$. Dlatego też k nie powinno być zbyt duże, aby w ten sposób zminimalizować liczbę obiektów, które powinny być poddane dalszej, dodatkowej analizie z wykorzystaniem innych algorytmów klasyfikacyjnych.

Wyniki zaprezentowane w tabelach 2.2 i 2.3 uzyskane zostały dla ułamkowej metryki powinowactwa $L_{0.5}$, gdyż wymiarowość testowanych baz danych była zbyt duża, aby można było zastosować odległości euklidesowe oraz Manhattan.

Podsumowanie

Zastosowanie zbiorów przybliżonych (RST) w *NSA* pozwoliło na określenie dolnej i górnej aproksymacji dla każdego *v*-detektora, zwiększając tym samym jego moc dyskryminacyjną, kosztem utworzenia dodatkowej, oprócz *self* i *nonself*, trzeciej klasy obiektów, nazwanej *uncertain*. Przypisywane są do niej wszystkie cenzorowane obiekty, które są bardzo podobne do obydwu pierwotnych klas, a ostateczna klasyfikacja, w celu uniknięcia błędnego przypisania, wymaga zastosowania innych miar podobieństwa, a nawet algorytmów. Dzięki temu możliwe jest uzyskanie znaczącego wzrostu współczynnika *DR*, przy jednoczesnej redukcji *FAR*.

Warto podkreślić, że wysokie współczynniki *DR* zostały uzyskane bez znaczącego wzrostu złożoności czasowej i obliczeniowej. W rzeczywistości, podczas fazy uczenia się, proces generowania detektorów nie wymaga dodatkowych, obciążających system operacji w porównaniu do pierwotnej wersji algorytmu V-Detector. Natomiast w fazie klasyfikacji, nadprogramowe przetwarzanie dotyczy tylko obiektów zaliczonych do grupy *uncertain*. Pozostałe obiekty są przetwarzane z tą samą złożonością, co w przypadku bazowego algorytmu.

Tolerancyjny V-Detector może znaleźć zastosowanie wszędzie tam, gdzie obiekty *self* oraz *nonself* są słabo rozróżnialne, przez co wiele algorytmów może błędnie je sklasyfikować. Przykładem mogą być różnego rodzaju systemy bezpieczeństwa np. programy antywirusowe oraz systemy detekcji intruzów, w których z pozoru mało znaczące mutacje, w odniesieniu do normalnego zachowania, są często wykorzystywane do skompromitowania celu ataku.

Przyszłe prace zostaną ukierunkowane na m.in. zastosowanie RST w modelu b-v [5] z uwzględnieniem różnych miar podobieństwa. Zasadne wydaje się również zastosowanie innych metryk w procesie pierwotnej klasyfikacji, a innych do cenzorowania obiektów ze zbioru *uncertain*.

Bibliografia

[1] Acuña, R. F., Ushio, T., *Rough lymphocytes for approximate binding in artificial immune systems*. 15th International Conference on Electronics, Communications, and Computers (CONIELECOMP), 2005, 272–277

[2] Cherukuri, R., Allen, L, Perelson, A. S., Forrest, S., *Self-nonself discrimination in a computer*. Proceedings IEEE Symposium on Security and Privacy, 1994,202–212

[3] Chmielewski, A., *Application of rough sets to negative selection algorithms*. Future Data and Security Engineering – 4th International Conference, (FDSE), 2017, LNCS-10646, 381–394

[4] Chmielewski, A., Wierzchon, S., *On the distance norms for multidimensional dataset in the case of real-valued negative selection application*. „Zeszyty Naukowe Politechniki Białostockiej", 2007, 2, 39–50

[5] Chmielewski, A., Wierzchon,′ S. T., *Hybrid Negative Selection Approach for Anomaly Detection*. 11th International Conference on Computer Information Systems and Industrial Management (CISIM), LNCS-7564, 2012, 242–253

[6] Fernandes, D. A. B. F., Inácio, P., Freire, M., Fazendeiro, P., *Applications of artificial immune systems to computer security: A survey*. „Journal of Information Security and Applications", 2017, 35, 138–159

[7] Forrest, S., Hofmeyr, S. A., Somayaji, A., Longstaff, T. A., *A sense of self for unix processes*. Proceedings IEEE Symposium on Security and Privacy, 1996,120–128

[8] Harmer, P. K., Williams, P. D., Gunsch, G. H., Lamont, G. B., *An artificial immune system architecture for computer security applications*. IEEE transactions on evolutionary computation, 2002, 6(3), 252–280

[9] Honko, P., *Compound approximation spaces for relational data*. International Journal of Approximate Reasoning, 2016, 71, 89–111

[10] Ji, Z., Dasgupta, D., *Real-valued negative selection algorithm with variable-sized detectors*. Genetic and Evolutionary Computation – GECCO 2004, LNCS-3102, 2004, 287–298

[11] Ji, Z., Dasgupta, D., *Revisiting negative selection algorithms*. Evolu-tionary Computation, 2007, 15(2), 223–251

[12] Pawlak, Z,. *Rough sets – theoretical aspects of reasoning about data* (Vol. 9). 1991, Kluwer

[13] Shen, J., Wang, J., Ai, H., *An improved artificial immune system-based network intrusion detection by using rough set*. Communications and Network, 2012, 4, 41–47

[14] Stibor, T., Mohr, P. H., Timmis, J., Eckert, C., *Is negative selection appropriate for anomaly detection?* Genetic and Evolutionary Compu-tation Conference, GECCO 2005, Proceedings, Washington DC, USA, 2005, 321–328

Tolerant V-Detector algorithm

Abstract: V-Detector algorithm is an example of negative selection algorithm, proposed to distinguish between self and nonself samples, represented as real-valued vectors. It is used mainly for detecting anomalies in high dimensional datasets. For this purpose, the set of detectors is generated, usually in random way, base only on information about normal behaviour. The main problem with this algorithm is scalability and high computational complexity. This paper presents a new

approach for building the detectors, inspired by idea from the tolerance rough sets. With this algorithm it is possible to detect an outliers as well as those samples which are very similar to both classes and their correct classification requires additional verification. Partially, it also solves the scalability problem as detectors has a higher discrimination power in comparing to its basic version.

Keywords: artificial immune systems, negative selection, rough sets

DOI: 10.24427/978-83-66391-96-3_3

Rozdział 3
Dobór optymalnego pobudzenia w zadaniu identyfikacji układu dynamicznego w czasie swobodnym

Wiktor Jakowluk
Wydział Informatyki, Politechnika Białostocka

Streszczenie: Identyfikacja systemu dynamicznego polega na doborze optymalnego sygnału wejściowego, który jest wykorzystywany do pobudzania układu w celu maksymalizacji informacji o dynamice systemu. W niniejszej pracy przeprowadzono dobór optymalnego sygnału wejściowego w czasie swobodnym, który następnie wykorzystano w zadaniu estymacji parametrów modelu układu dynamicznego. Rozwiązanie problemu doboru optymalnego sygnału wejściowego polega na minimalizacji rozszerzonego wektora stanu o współczynnik skalowania czasu swobodnego. Funkcjonał celu sformułowano w postaci funkcji Bolzy z ograniczeniem na D-efektywność oraz energię sygnału sterującego. Ograniczenia uwzględnione podczas projektowania sygnału wejściowego powinny umożliwić operatorowi estymację parametrów modelu układu w przyjazny sposób. W eksperymencie wykorzystano optymalny sygnał wejściowy, uzyskany w czasie swobodnym, do celów weryfikacji aspektów ekonomicznych procesu identyfikacji (tj. związku pomiędzy nałożonymi ograniczeniami a czasem trwania eksperymentu). Przedstawiona metoda może być stosowana dla ogólnej klasy systemów i została zweryfikowana przykładami numerycznymi.

Słowa kluczowe: D-efektywność, identyfikacja w czasie swobodnym, optymalny sygnał wejściowy, optymalne sterowanie

Wprowadzenie

Identyfikacja układu dynamicznego jest zwykle przeprowadzana przez poddanie go pobudzeniu w trakcie procesu technologicznego. Wybór sygnału wejściowego wykorzystywanego do pobudzenia układu ma fundamentalne znaczenie w przypadku estymacji parametrów modelu układu. Zasadniczym celem projektowania eksperymentu identyfikacyjnego jest jak najdokładniejsze oszacowanie parametrów, które określa się minimalną zmiennością wartości parametrów przy obecności zakłóceń losowych oddziałujących na system w rzeczywistych warunkach pracy [1]. W rzeczywistości pobudza się system wykorzystując optymalny sygnał wejściowy, a uzyskane

dane wykorzystuje się do zbudowania modelu układu z maksymalną dokładnością [2, 3]. Niedokładny model układu może powodować pogorszenie wydajności pętli sterowania i ostatecznie wpłynąć na finalny produkt. Celem prac związanych ze sterowaniem procesów przemysłowych jest stabilizacja warunków pracy, które mają wpływ na wyniki ekonomiczne. Oszacowano, że około 66%–80% rozważanych układów sterowania nie osiąga pożądanych wyników pracy [4]. Z drugiej strony procesy technologiczne są przeprowadzane w stanie ustalonym, a każda zmiana warunków pracy powoduje, że wydajność zaawansowanego systemu sterowania zmniejsza się. W konsekwencji adekwatność identyfikowanego modelu układu dynamicznego rośnie przez wykorzystanie optymalnego sygnału wejściowego [1, 5].

Dobór optymalnego sygnału wejściowego, z wykorzystaniem metody „ang. Model-based," który jest wykonywany podczas trwania procesu sterowania, przedstawiono w wielu pracach [6–8]. Wykazano, że dobór dokładnego modelu układu pochłania około trzech czwartych kosztów związanych z projektami sterowania [9]. Z tego powodu wartość ekonomiczna eksperymentu identyfikacyjnego powinna być jak najniższa, ale eksperyment powinien nadal zapewniać wysoką dokładność oszacowań parametrów modelu układu. Rozpatrując metodę „ang. Least-costly," koszt eksperymentu identyfikacyjnego określa się również jako degradację wydajności procesu, w układzie zamkniętym, podczas pobudzenia układu w normalnych warunkach pracy [10, 11].

W wielu praktycznych zastosowaniach, takich jak przemysł petrochemiczny, sygnały pobudzające przyjazne modelowi układu są stosowane do estymacji parametrów systemu w czasie rzeczywistym. Sygnały przyjazne modelowi to takie, które powodują minimalne zakłócenia warunków pracy, ograniczają czas trwania eksperymentu oraz amplitud sygnałów wejściowych i wyjściowych [12–14]. Stwierdzono, iż ograniczenia stawiane sygnałom przyjaznym modelowi są często sprzeczne z wymogiem dokładnej estymacji parametrów modelu układu [3, 15]. Okazuje się, że liniowa kombinacja harmonicznych sygnału sinusoidalnego, o dużych wartościach międzyszczytowych, może prowadzić do uszkodzenia identyfikowanego systemu. Dlatego rozważa się sygnały wymuszające „bezpieczne", które mają za zadanie zapewnić akceptowalną dokładność oszacowań parametrów modelu układu [16, 17]. W pracy [18] zaproponowano odporne sformułowanie problemu doboru przyjaznego sygnału sterującego z ograniczeniami amplitudy sygnału wejściowego oraz mocy sygnału wyjściowego. Taki rodzaj eksperymentu jest kompilacją problemu sekwencyjnego i odpornego. Metodologia rozwiązywania problemów identyfikacji w ramach ekonomicznych, której celem jest minimalizacja kosztów eksperymentu, została przedstawiona w pracy [19]. Dane dotyczące wielokryterialnych zadań optymalizacji w zastosowaniu do identyfikacji oraz procesu sterowania zamieszczono w [20, 21].

W omówionych powyżej publikacjach przedstawiono metody doboru optymalnych sygnałów wejściowych dla ustalonego czasu końcowego eksperymentu identyfikacyjnego. W tej pracy, przedstawiono problem doboru optymalnego sygnału wejściowego, w czasie swobodnym, w celu minimalizacji kosztów ekonomicznych związanych z eksperymentem identyfikacyjnym. Ograniczenia nałożone na funkcję

celu powinny pozwolić na uzyskanie niewielkiej utraty dokładności estymat parametrów modelu układu dynamicznego. Zaprojektowane w ten sposób sygnały pobudzające następnie wykorzystano w procedurze estymacji parametrów modelu układu zakłócanego szumem białym o różnej wariancji.

W artykule autora przedstawiono kontynuację prac własnych dotyczących identyfikacji układów opisanych równaniami różniczkowymi zwyczajnymi. Wyniki doboru optymalnego sygnału wejściowego z wykorzystaniem kanonicznego sformułowania Mayera funkcji celu dla układu skrętnego oraz inercyjnego zaprezentowano w pracach [22–24].

Artykuł jest zorganizowany w następujący sposób. W rozdziale 1 przeprowadzono transkrypcję doboru optymalnego sterowania dla czasu swobodnego. W rozdziale 2 opisano problem doboru optymalnego sygnału wejściowego z ograniczeniami, w celu estymacji parametrów modelu układu. Sformułowanie problemu przedstawiono w rozdziale 3. Wyniki eksperymentów symulacyjnych z wykorzystaniem modelu układu inercyjnego przedstawiono w rozdziale 4. Wnioski końcowe zamieszczono w podsumowaniu.

3.1. Sformułowanie problemu czasu swobodnego

W celu doboru optymalnego sygnału wejściowego, w czasie swobodnym, wykorzystano bibliotekę narzędziową „ang. Recursive Integral Optimal Trajectory Solver" RIOTS_95 [25]. Zestaw narzędzi Riots został opracowany do rozwiązywania problemów związanych z problemami optymalnego sterowania w czasie skończonym, które obejmują: ograniczenia trajektorii, ograniczenia na stan końcowy, zmienne warunki początkowe oraz zadania swobodnego czasu końcowego. Biblioteka narzędziowa jest przeznaczona do rozwiązywania problemów optymalnego sterowania w postaci funkcji Bolzy:

$$\min_{(u,\zeta)\in L_\infty^m[a,b]\times\mathbf{R}^n}\left\{f(u,\zeta)\doteq g_0(\zeta,x(b))+\int_a^b l_0(t,x,u)\,dt\right\}, \tag{3.1}$$

gdzie: $\dot{x}=h(t,x,u)$, $x(a)=\zeta$, $t\in[a,b]$,

przy następujących ograniczeniach:

$$u_{\min}^j(t)\le u^j(t)\le u_{\max}^j(t),\ j=1,\ldots,m,t\in[a,b], \tag{3.2}$$

$$\zeta_{\min}^{\ j}(t)\le\zeta^{\ j}(t)\le\zeta_{\max}^{\ j}(t),\ j=1,\ldots,n,t\in[a,b], \tag{3.3}$$

$$l_{ti}^{v}(t,x(t),u(t))\le 0,\ v\in\mathbf{q}_{\mathbf{ti}},t\in[a,b], \tag{3.4}$$

$$g_{ei}^{v}\left(\zeta, x\left(b\right)\right) \le 0, v \in \mathbf{q_{ei}}, \tag{3.5}$$

$$g_{ee}^{v}\left(\zeta, x\left(b\right)\right) = 0, v \in \mathbf{q_{ee}}, \tag{3.6}$$

gdzie: x jest zmienną stanu, $t \in [a, b]$ jest przedziałem czasu, $q = \{1,\ldots, q\}$ oraz l, g, h są arbitralnie dobranymi funkcjami. Indeksy *o*, *ti*, *ei* oraz *ee* funkcji $g(\cdot, \cdot)$ i $l(\cdot, \cdot, \cdot)$ oznaczają odpowiednio: funkcję celu, ograniczenie trajektorii, ograniczenie nierównościowe na stan końcowy oraz ograniczenie równościowe na stan końcowy.

Rozwiązanie problemu doboru optymalnego sterowania w czasie swobodnym odbywa się poprzez rozszerzenie wektora stanu o dodatkowe zmienne stanu, tj. jeden stan dla każdego autonomicznego problemu. Zgodnie z rozumowaniem przedstawionym w rozdziale 2 podręcznika użytkownika [26], ogólną ideą jest zdefiniowanie nominalnego przedziału czasu $[a, b]$ i zastąpienie problemu swobodnego czasu końcowego przypadkiem ustalonego czasu końcowego. Współczynnik skalowania czasu swobodnego definiowany jest jako dodatkowa zmienna stanu skalująca czas trwania eksperymentu. Z tego powodu współczynnik skalowania i czas końcowy wyrażone są dodatkowymi zmiennymi stanu. Procedura zaimplementowana w bibliotece Riots_95 pozwala na minimalizację więcej niż jednego stanu początkowego w celu optymalizacji czasu trwania eksperymentu:

$$\min_{u,T} \breve{g}\left(T, y\left(T\right)\right) + \int_{a}^{a+T} \breve{l}\left(t, y, u\right) dt, \tag{3.7}$$

gdzie: $\dot{y} = \breve{h}\left(t, y, u\right),\ y\left(a\right) = \zeta,\ t \in \left[a, a+T\right]$.

Następnie problem można łatwo przekształcić do postaci alternatywnej, problemu optymalnego sterowania w czasie swobodnym, z rozszerzonym wektorem stanu $x \doteq \left(y, x^{n-1}, x^{n}\right)$ w postaci:

$$\min_{u,\zeta^{n}} g\left(\zeta, x\left(b\right)\right) + \int_{a}^{b} l\left(t, x, u\right) dt, \tag{3.8}$$

Przyjmując, że: $\dot{x} = h\left(t, x, u\right) \doteq \begin{pmatrix} x^{n}\breve{h}\left(x^{n-1}, y, u\right) \\ x^{n} \\ 0 \end{pmatrix}, x\left(a\right) = \zeta = \begin{pmatrix} \zeta \\ a \\ \zeta^{n} \end{pmatrix}, t \in \left[a, b\right], x$ zawiera n-2 składowe y, $g\left(\zeta, x\left(b\right)\right) \doteq \breve{g}\left(a + T\zeta^{n}, y\left(b\right)\right)$, $l\left(t, x, u\right) \doteq \breve{l}\left(x^{n-1}, y, u\right)$ oraz $b = a + T$. W powyższej notacji x^{n-1} jest zmienną stanu reprezentującą czas, a ζ^{n} jest współczynnikiem skalowania czasu trwania eksperymentu. Dla każdego $t \in \left[a, b\right]$, $x^{n}\left(t\right) = \zeta^{n}$, $x^{n-1}\left(t\right) = a + \left(t - a\right)\zeta^{n}$ rozwiązanie problemu czasu swobodnego jest następujące $t_f = x^{n-1}\left(b\right) = a + \left(b - a\right)\zeta^{n} = T\zeta^{n}$. Rozważając systemy dynamiczne

autonomiczne, dodatkowa zmienna stanu x^{n-1} nie jest wymagana. Dlatego problemy autonomiczne ze swobodnym czasem końcowym można rozwiązać poprzez rozszerzenie wektora stanu tylko jedną zmienną, reprezentującą współczynnik skalowania czasu końcowego zapisaną w następujący sposób:

$$h(t,x,u) = \breve{h}\left(tx^n, y, u\right). \tag{3.9}$$

Problem doboru optymalnego sygnału wejściowego można przedstawić w formie wskaźnika jakości o postaci funkcjonału Bolzy, czyli sumy wartości końcowych zmiennych stanu i całki za okres sterowania, która jest minimalizowana. Taki rodzaj zadania optymalizacji umożliwia wyznaczenie optymalnego sygnału wejściowego w obecności wprowadzonych ograniczeń.

3.2. Sformułowanie problemu doboru optymalnego pobudzenia

Rozważaniom poddano układ dynamiczny opisany poniższą zależnością:

$$z(t) = f\left(u,t,\theta,v\right), \tag{3.10}$$

gdzie: $z(t)$, $u(t)$, $v(t)$ oraz θ są odpowiednio: wyjściem, wejściem, szumem oraz wektorem parametrów systemu. Identyfikacja systemu to proces tworzenia dokładnego modelu matematycznego układu dynamicznego na podstawie danych eksperymentalnych i wiedzy *'a priori'* o systemie. Dokładność oszacowań parametrów modelu zależy przede wszystkim od doboru odpowiedniego sygnału pobudzającego. Dobierając optymalne sygnały wejściowe w celu szacowania parametrów modelu układu, jako kryterium optymalizacji należy wybrać odpowiednią normę skalarną macierzy informacyjnej Fishera (FIM). Macierz FIM może zostać zdefiniowana w następujący sposób:

$$M = E\left[\frac{\partial}{\partial\theta}\ln p\left(z\,|\,\theta\right)\left(\frac{\partial}{\partial\theta}\ln p\left(z\,|\,\theta\right)\right)^{\mathrm{T}}\right]. \tag{3.11}$$

W praktyce często wykorzystywane jest kryterium D–optymalności, w którym wyznacznik macierzy informacyjnej Fishera *detM* jest maksymalizowany lub *det* (M^{-1}) jest minimalizowany. Wybrane kryteria jakości optymalizacji zamieszczono w [27]:

- A-optymalność: *tr* (M^{-1}), minimalizuje ślad odwrotności macierzy FIM,
- E-optymalność: $\lambda_{max}(M^{-1})$, minimalizuje maksymalną wartość własną macierzy FIM,
- D-optymalność: maksymalizuje wyznacznik macierzy FIM oraz minimalizuje objętość elipsoidy ufności estymat parametrów.

W celu wyznaczenia optymalnego sygnału pobudzającego, należy rozpatrzyć nieobciążony estymator parametrów θ. W takim przypadku kowariancja estymat parametrów jest określona przez nierówność Cramera-Rao, czyli odwrotność macierzy FIM. Kowariancję estymat parametrów $\breve{\theta}$ można zapisać jako:

$$\operatorname{cov}\left(\breve{\theta}\right) = E\left[\left(\breve{\theta} - \theta\right)\left(\breve{\theta} - \theta\right)^{\mathrm{T}}\right] \geq M.^{-1} \tag{3.12}$$

Metodę ważonej funkcji celu do zadań projektowania sygnałów pobudzających, w dziedzinie czasu, przedstawiono w pracy [28].

Chociaż określenie kryterium optymalizacji jest niezbędne, to sygnały pobudzające zaprojektowane na podstawie niektórych z nich mogą powodować małą dokładność estymat parametrów modelu układu [2]. Sygnał wejściowy wybrany do pobudzenia modelu układu powinien jednocześnie spełniać dwa warunki: akceptowalną dokładność estymat parametrów modelu oraz model układu powinien być pobudzany w najbardziej bezpieczny sposób. Te warunki można zapewnić, stosując podejście oparte na definicji D-efektywności sterowania [27]. Funkcjonał D-efektywności (3.13) jest wyrażony jako miara względnej jakości dowolnego eksperymentu frakcyjnego *e* w porównaniu z eksperymentem optymalnym *e**, który wykonano wcześniej. Wskaźnik D-efektywności często jest wyrażany w procentach i może być traktowany jako miara optymalności dowolnego sygnału wejściowego:

$$E_D\left(e\right) = 100\% \left\{ \frac{\det\left(M\left(e\right)\right)}{\det\left(M\left(e^*\right)\right)} \right\}^{1/k}, \tag{3.13}$$

gdzie: *k* jest liczbą parametrów do zidentyfikowania, a *e** oznacza eksperyment D-optymalny. Zgodnie z rozumowaniem przedstawionym w pracy [28], nakładamy ograniczenie nierównościowe na wartość funkcji D-efektywności:

$$E_D(e) \geq \mu, \tag{3.14}$$

gdzie: $\mu < 1$ i nierówność (3.14) jest równoznaczna następującemu ograniczeniu:

$$\Psi\left[M\left(e\right)\right] \leq D, \tag{3.15}$$

oraz $\Psi\left[M\left(e\right)\right] = \log\left(\det M\left(e\right)\right)$.

Ogólnie rzecz biorąc, zadanie doboru optymalnych sygnałów wejściowych jest formułowane poprzez maksymalizację wyznacznika macierzy FIM (tj. kryterium D-optymalności) i powinno uwzględniać ograniczenia fizyczne. Tak zaprojektowany eksperyment powinien zapewnić bezpieczną identyfikację układu dynamicznego. Wybrane ograniczenia zdefiniowano poniżej:

- nierównościowe ograniczenie D-efektywności zdefiniowane na podstawie (3.15),

- ograniczenia amplitudy sygnałów wejściowych, wyjściowych i zmiennych stanu w postaci:

$$u_{\min} \le u(t) \le u_{\max}. \tag{3.16}$$

- ograniczenia energii sygnałów wejściowych:

$$\int_0^T u^{\mathrm{T}}(t)u(t)dt \le E. \tag{3.17}$$

Celem tego eksperymentu jest zaprojektowanie sygnału pobudzającego o ograniczonym czasie trwania, który następnie wykorzystywany jest w eksperymencie identyfikacyjnym. W takim przypadku wskaźnik jakości formułowany jest poprzez minimalizację współczynnika skalowania czasu końcowego, z ograniczeniami na D-efektywność sterowania oraz energię sygnału wejściowego. W ten sposób (tj. biorąc pod uwagę powyższe ograniczenia) możemy uzyskać suboptymalny sygnał sterujący, który jest bezpieczny w zadaniach identyfikacji układu dynamicznego.

3.3. Dobór optymalnego sygnału wejściowego

W pracy zaprezentowano syntezę sygnału wejściowego, w czasie swobodnym, do celów estymacji parametrów modelu układu dynamicznego. Idea polega na zdefiniowaniu nominalnego czasu eksperymentu $[0, T]$ i zastąpieniu problemu swobodnego czasu końcowego problemem klasycznym, który wykorzystuje współczynnik skalujący jako zmienną rozszerzonego wektora stanu. Problem rozwiązano wykorzystując transkrypcję powyższego problemu optymalnego sterowania na zadanie optymalnego sterowania w postaci Lagrange'a z zestawem ograniczeń. W celu weryfikacji takiego podejścia do estymacji parametrów modelu układu dynamicznego, wykorzystano obiekt inercyjny pierwszego rzędu w postaci:

$$G(s) = \frac{k_p}{Ts+1}. \tag{3.18}$$

Zadanie doboru optymalnego pobudzenia dla układu inercjalnego można opisać następującym modelem w przestrzeni stanów, z jednym wejściem i jednym wyjściem:

$$\begin{aligned} \dot{x}(t) &= ax(t) + bu(t),\ x(0) = x_0, \\ z(t) &= x(t) + v(t), \end{aligned} \tag{3.19}$$

gdzie: $x(t)$ jest wektorem stanu, $u(t)$ jest wektorem wymuszeń, $z(t)$ jest wektorem wyjścia, a i b są stałymi parametrami modelu natomiast $v(t)$ jest wektorem białego szumu pomiarowego opisanego następującą zależnością:

$$E\left[v(t)\right]=0,$$
$$E\left[v(t)v^{\mathrm{T}}(\tau)\right]=R\delta(t-\tau)=\sigma_n^2\delta(t-\tau). \tag{3.20}$$

Podstawowym celem zadania identyfikacji parametrów modelu układu jest maksymalizacja wrażliwości zmiennej stanu na estymowany parametr [1]. Motywacją do takiego eksperymentu jest twierdzenie Cramera-Rao, w którym wariancja dowolnego nieobciążonego estymatora parametru jest ograniczona z dołu przez odwrotność macierzy informacyjnej Fishera. Stosując powyższą definicję do celów doboru optymalnego sterowania, otrzymujemy następującą zależność:

$$\operatorname{cov}\left(\left[a,b\right]\right)\geq M^{-1}. \tag{3.21}$$

Macierz informacyjną Fishera (FIM) dla układu inercyjnego opisanego równaniami stanu można przedstawić w następującej formie:

$$M(T)=\int_0^T X_\theta^{\mathrm{T}}R^{-1}X_\theta\,dt=\frac{1}{\sigma_n^2}\int_0^T\begin{bmatrix}x_a\\x_b\end{bmatrix}\begin{bmatrix}x_a & x_b\end{bmatrix}dt=\frac{1}{\sigma_n^2}\int_0^T\begin{bmatrix}x_a^2 & x_ax_b\\x_ax_b & x_b^2\end{bmatrix}dt, \tag{3.22}$$

gdzie: $x_a=\partial x/\partial a$, $x_b=\partial x/\partial b$ i R jest macierzą 2×2 o postaci:

$$R^{-1}=\begin{bmatrix}R^{-1} & 0\\0 & R^{-1}\end{bmatrix}=\frac{1}{\sigma_n^2}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}. \tag{3.23}$$

Podstawiając równanie (3.22) do (3.21) otrzymujemy:

$$\operatorname{cov}\left(\left[a,b\right]\right)\geq\frac{\sigma_n^2}{M(T)}. \tag{3.24}$$

W dalszych rozważaniach przyjęto $\sigma_n=1$, w celu uzyskania optymalnego sygnału pobudzającego w zadaniu estymacji parametrów modelu układu. Aby zmaksymalizować wyznacznik macierzy FIM, zdefiniowano rozszerzony wektor stanu w postaci [1]:

$$\dot{x}_a=x+ax_a,\ x_a(0)=0, \tag{3.25}$$

$$\dot{x}_b=ax_b+u,\ x_b(0)=0. \tag{3.26}$$

Dobór optymalnego sygnału wejściowego w zadaniu estymacji parametrów modelu układu inercyjnego jest formułowany poprzez minimalizację dodatkowej zmiennej stanu, reprezentującej współczynnik skalowania czasu końcowego, w obecności fizycznych ograniczeń systemu.

Współczynniki macierzy FIM uzyskują następującą postać:

$$\dot{m}_{11}(t) = x_a^2(t),\ m_{11}(0) = 0, \tag{3.27}$$

$$\dot{m}_{12}(t) = \dot{m}_{21}(t) = x_a(t)x_b(t),\ m_{12}(0) = 0, \tag{3.28}$$

$$\dot{m}_{22}(t) = x_b^2(t),\ m_{22}(0) = 0, \tag{3.29}$$

gdzie:

$$M(t) = \begin{bmatrix} m_{11}(t) & m_{12}(t) \\ m_{21}(t) & m_{22}(t) \end{bmatrix}. \tag{3.30}$$

Rozszerzony wektor stanu (3.25), (3.26) i (3.27–3.29), uwzględniający dodatkową zmienną stanu reprezentującą współczynnik skalowania czasu końcowego ζ, opisano poniższą zależnością:

$$\begin{aligned}
&x_1 = x; && \dot{x}_1 = x_7(ax_1 + bu); && x_1(0) = x_{10};\\
&x_2 = x_a; && \dot{x}_2 = x_7(x_1 + ax_2); && x_2(0) = 0;\\
&x_3 = x_b; && \dot{x}_3 = x_7(ax_3 + u); && x_3(0) = 0;\\
&x_4 = m_{11}; && \dot{x}_4 = x_7 x_2^2; && x_4(0) = 0;\\
&x_5 = m_{12} = m_{21}; && \dot{x}_5 = x_7 x_2 x_3; && x_5(0) = 0;\\
&x_6 = m_{22}; && \dot{x}_6 = x_7 x_3^2; && x_6(0) = 0;\\
&x_7 = \zeta; && \dot{x}_7 = 0; && x_7(0) = \zeta_{70}.
\end{aligned} \tag{3.31}$$

Ostatecznie, zadanie optymalnego sterowania opartego na kanonicznym sformułowaniu Bolzy, które minimalizuje funkcję celu (3.1) przy ograniczeniu wartości D-efektywności (3.6), a także ograniczeniach wejścia (3.2) i wyjścia (3.4), jest następujące:

$$J(u) = \zeta + q\int_0^T u(t)^{\mathrm{T}} u(t)\,dt, \tag{3.32}$$

biorąc pod uwagę:

$$\begin{gathered}
-1 \le u(t) \le 1,\ t \in [0, T],\\
0{,}1 \le \zeta \le 10,\\
x_1(t) \le 1,\ t \in [0, T],\\
\left(-x_4(T)x_6(T) + x_5^2(T)\right) = D,
\end{gathered} \tag{3.33}$$

gdzie: D jest wymaganą wartością D-efektywności, q jest współczynnikiem energii sygnału wejściowego, a ζ_{70} jest optymalizowanym warunkiem początkowym. Należy zauważyć, że nałożono dodatkowe ograniczenie na zmienną stanu $x_1(t)$, które ma uniemożliwić nieoczekiwane zmiany sygnału sterującego $u(t)$. Rozważając metodologię określoną równaniami (3.31–3.33), optymalnym rozwiązaniem czasu swobodnego jest $t_f = T\zeta$.

3.4. Wyniki eksperymentów numerycznych

Przedstawiony powyżej problem może zostać rozwiązany z wykorzystaniem dostępnych bibliotek narzędziowych dotyczących problemów optymalnego sterowania, takich jak: RIOTS_95 [29], DIRCOL [30] lub MISER [31]. Do rozwiązania problemu opisanego w niniejszej pracy wykorzystano zestaw narzędzi zawartych w bibliotece RIOTS_95. Oprogramowanie to zaprojektowano do implementacji w pakiecie Matlab, jako oddzielny moduł, posiadający narzędzia rozwiązywania problemów optymalnego sterowania z ograniczeniami dla ustalonych lub swobodnych warunków początkowych.

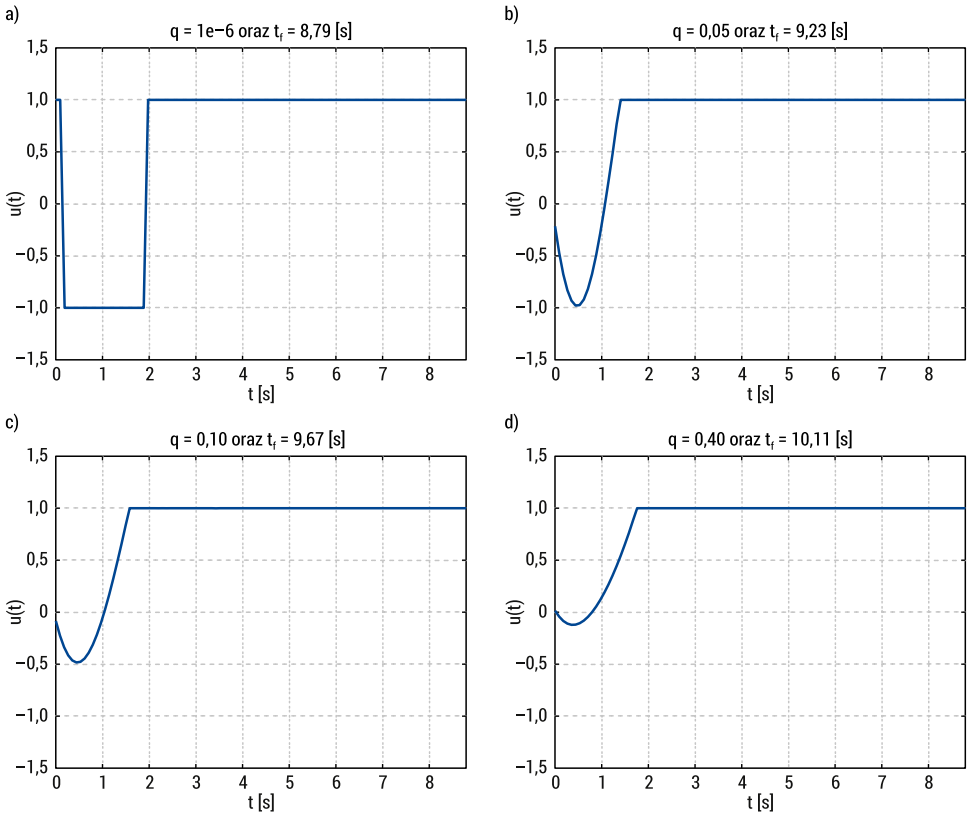
Problem doboru optymalnego sygnału wejściowego w zadaniu estymacji parametrów modelu układu inercyjnego, dla ustalonych nominalnych wartości parametrów: $a = -1$, $b = 1$ oraz arbitralnie ustalonego przedziału czasu eksperymentu $t = [0, 10]$ sekund, rozwiązano wykorzystując sekwencyjny algorytm programowania kwadratowego (SQP). Warunki początkowe modelu układu inercyjnego wybrano jako: $x_1(0) = 5$, $x_7(0) = 1$ oraz warunek początkowy sygnału wejściowego ustalono na $u(0) = 1$. Współczynnik skalowania czasu trwania eksperymentu ζ jest optymalizowany w przedziale $0{,}1 \le \zeta \le 10$, (tj. czas eksperymentu zawarty jest w przedziale od 1 do 100 sekund). Wyniki numeryczne uzyskano wykorzystując metodę Runge-Kutty, czwartego rzędu, o stałej wielkości kroku co 0,2 sekundy. Funkcję celu opisaną zależnością (3.32) przekształcono do następującej postaci:

$$J(u) = J_1 + qJ_2, \tag{3.34}$$

gdzie: J_1 oznacza współczynnik skalowania czasu końcowego ζ, a J_2 jest całką z kwadratu sygnału wejściowego.

Optymalny sygnał wejściowy otrzymany przy braku ograniczenia wartości energii sygnału sterującego (tj. $q \approx 0$ w nierówności (3.32)) przedstawiono na rysunku 3.1(a). Odpowiada on D-optymalnemu eksperymentowi e, gdzie wartość wyznacznika macierzy FIM (zgodnie z (3.13)) uzyskuje $D_{eff} = 90\%$ jej optymalnej wartości. Suboptymalne sygnały pobudzające uzyskane przy różnych wartościach współczynnika kosztu energii sterowania q i stałej wartości D-efektywności równej $D_{eff} = 90\%$ przedstawiono na rysunkach 3.1(b-d).

Optymalny sygnał wejściowy otrzymany przy braku ograniczenia na składową kosztu energii sygnału wejściowego (tj. $J_1 = 0{,}88$, $qJ_2 = 1{,}0 \times 10^{-4}$ i $t_f = 8{,}79$ [s]) przedstawiono na rysunku 3.1(a). Powiększono wartość kosztu energii sterowania (rysunek 3.1(b)) w celu uzyskania suboptymalnego sterowania, które odpowiada następującym wartościom składowych wskaźnika jakości: $J_1 = 0{,}92$, $qJ_2 = 5{,}00$ i $t_f = 9{,}23$ [s]. Dla porównania, na rysunku 3.1(c) przedstawiono sygnał wejściowy, który wygenerowano dla następujących wartości funkcji celu: $J_1 = 0{,}97$, $qJ_2 = 8{,}70$ oraz $t_f = 9{,}67$ [s]. Po zwiększeniu kosztu energii sterowania do wartości $q = 0{,}40$, otrzymano sygnał pobudzający przedstawiony na rysunku 3.1(d) uzyskując kolejne wartości wskaźnika jakości: $J_1 = 1{,}01$, $qJ_2 = 33{,}44$, $t_f = 10{,}11$ [s].



RYSUNEK 3.1. **Sygnały pobudzające w czasie swobodnym: (a) Optymalny sygnał wejściowy dla $q \approx 0$ i D_{eff} = 90%; (b) Suboptymalny sygnał wejściowy dla q = 0,05 i D_{eff} = 90%; (c) Suboptymalny sygnał wejściowy dla q = 0,10 i D_{eff} = 90%; (d) Suboptymalny sygnał wejściowy dla q = 0,40 i D_{eff} = 90%**

Można zauważyć, że gdy wartość współczynnika kosztu energetycznego wzrasta, kształt optymalnego sygnału wejściowego znacznie się zmienia. Podczas gdy dla optymalnego eksperymentu (w sensie (3.14)) występują gwałtowne zmiany sygnału wejściowego, sygnały sterujące otrzymane dla $D_{eff} < 100\%$ są bardziej przyjazne w procesie identyfikacji systemu, dopóki wyznacznik macierzy FIM nie zostanie zdominowany przez składnik odpowiadający za koszt energii sterowania we wskaźniku jakości. Porównanie składowych funkcji celu otrzymanych dla rosnących wartości współczynnika energii sygnału wejściowego i malejących wartości współczynnika D-efektywności z przedziału [100%, 80%] przedstawiono w tabelach 3.1, 3.2, 3.3.

TABELA 3.1. Porównanie komponentów wskaźnika jakości przy D_{eff} = 100%

D_{eff}/D_{opt}	FIM	J_1	q	qJ_2	t_r [s]
100%	−43,87	1,00	1,0e−6	1,0e−4	10,00
		1,06	0,05	4,79	10,57
		1,10	0,10	8,99	11,00
		1,13	0,20	17,42	11,36
		1,15	0,30	25,91	11,53
		1,16	0,40	34,44	11,62
		1,17	0,50	42,97	11,70

TABELA 3.2. Porównanie komponentów wskaźnika jakości przy D_{eff} = 90%

D_{eff}/D_{opt}	FIM	J_1	q	qJ_2	t_r [s]
90%	−35,60	0,880	1,0e−6	1,0e−4	8,79
		0,923	0,05	5,00	9,23
		0,967	0,10	8,70	9,67
		0,991	0,20	16,86	9,91
		1,041	0,30	25,14	10,04
		1,011	0,40	33,44	10,11

TABELA 3.3. Porównanie komponentów wskaźnika jakości przy D_{eff} = 80%

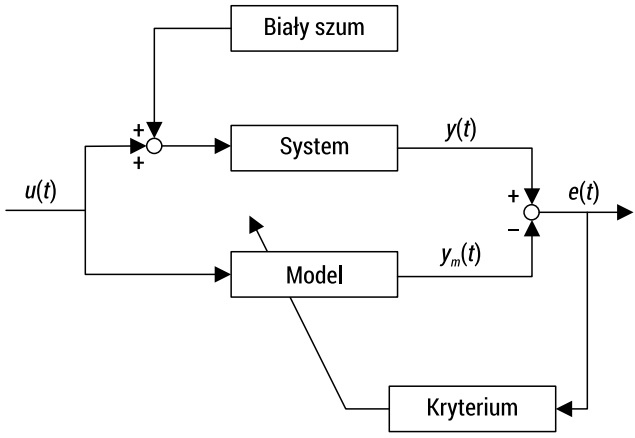
D_{eff}/D_{opt}	FIM	J_1	q	qJ_2	t_r [s]
80%	−28,10	0,763	1,0e−6	1,0e−4	7,63
		0,807	0,05	4,35	8,07
		0,837	0,10	8,27	8,37
		0,858	0,20	16,25	8,58
		0,866	0,30	24,22	8,66

Na podstawie powyższych danych można stwierdzić, że wraz ze wzrostem wartości współczynnika kosztu energii sterowania zwiększa się również czas trwania eksperymentu. Gdy założona wartość współczynnika D-efektywności z przedziału [100%, 80%] maleje, zmniejsza się czas trwania pobudzenia. Na podstawie wyników eksperymentów zamieszczonych w tabeli 3.1, odczytano optymalny czas trwania sygnału pobudzającego, który dla układu inercyjnego wynosi 10 sekund.

Sygnały wejściowe $u(t_f)$, wyznaczone jako rozwiązania problemów optymalizacji w czasie swobodnym (3.32) i (3.33), zostały następnie wykorzystane jako pobudzenia w procesie estymacji parametrów modelu układu inercyjnego. Model układu (3.19), używany w procedurze identyfikacji systemu, może zostać opisany następującym modelem w przestrzeni stanów z jednym wejściem i jednym wyjściem:

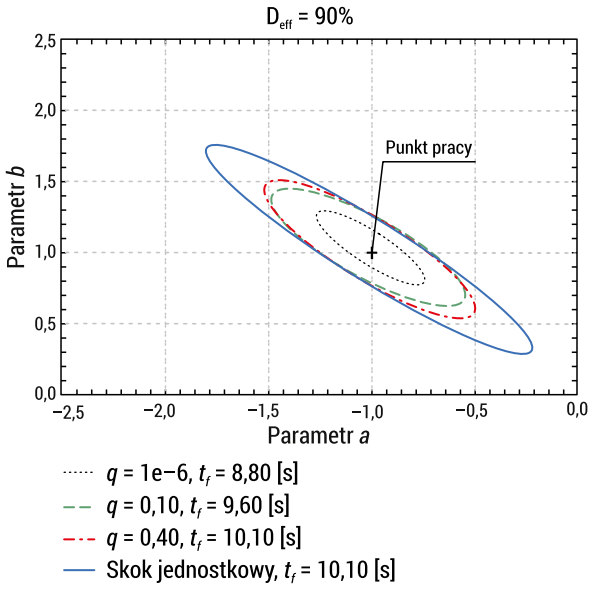
$$\begin{aligned} \dot{x}_1\left(t_f\right) &= ax_1\left(t_f\right) + bu\left(t_f\right) + v\left(t_f\right),\ x_1\left(0\right) = x_{10}, \\ z\left(t_f\right) &= x_1\left(t_f\right), \end{aligned} \tag{3.35}$$

Schemat blokowy na rysunku 3.2 przedstawia proces estymacji parametrów modelu układu inercyjnego: pobudzamy wejście układu $u(t_f)$ i zbieramy pomiary z jego wyjścia $y(t_f)$. Biały szum pomiarowy o różnych wartościach wariancji z przedziału $0{,}0 \le \sigma^2 \le 0{,}8$ zakłóca wejście sterujące systemu. Model układu (3.35) zależy od wektora nieznanych parametrów $\theta = [a, b]$. Celem takiego eksperymentu jest estymowanie nieznanych wartości parametrów modelu układu, które powinny być jak najbardziej zbliżone do rzeczywistych wartości parametrów systemu. W tym celu różnica kwadratów pomiędzy wyjściem systemu $y(t_f)$ a wyjściem modelu $y_m(t_f)$ jest minimalizowana. Warunki początkowe modelu układu inercyjnego ustalono w przedziale $5 \le x_1(0) \le 5$ a czas trwania eksperymentu uzależniono od wybranego sygnału wejściowego zgodnie z tabelą 3.2. Wyniki numeryczne otrzymano wykorzystując sympleksową metodę Neldera-Meada.



RYSUNEK 3.2. **Schemat blokowy estymacji parametrów modelu układu dynamicznego**

Rozkład estymowanych parametrów modelu układu inercyjnego a i b otrzymanych w trakcie eksperymentu identyfikacyjnego przy wymuszeniu różnymi sygnałami wejściowymi przedstawiono na rysunku 3.3. Wykonano dziewięćdziesiąt dziewięć pętli obliczeniowych, podczas gdy model układu inercyjnego startował z różnych warunków początkowych oraz szum pomiarowy zakłócający wejście systemu miał różne wariancje.



RYSUNEK 3.3. **Elipsoidalne obszary ufności estymat parametrów dla różnych sygnałów wejściowych: Optymalny sygnał pobudzający (czarna linia kropkowana, D_{eff} = 90%, q = 1,0e−6), Suboptymalny sygnał wejściowy (zielona linia przerywana, D_{eff} = 90%, q = 0,10), Suboptymalny sygnał wejściowy (czerwona linia kreska-kropka, D_{eff} = 90%, q = 0,40), Skok jednostkowy (niebieska linia ciągła)**

Na rysunku 3.3 przedstawiono obszary ufności estymat parametrów (czarna linia kropkowana) wyznaczonych przy wymuszeniu sygnałem wejściowym, który uzyskano dla minimalnej wartości współczynnika kosztu energii sterowania (tj. $q \approx 0$ i D_{eff} = 90%). Kolejne obszary ufności parametrów przedstawiają wyniki przeprowadzonych eksperymentów (dla tych samych warunków początkowych i wariancji szumu) z sygnałami pobudzającymi uzyskanymi, gdy współczynnik kosztu energetycznego zwiększa swoją wartość (tj. q = 0,10 – zielona linia przerywana oraz q = 0,40 – czerwona linia kreska-kropka). W celu porównania wyników eksperymentu, na rysunku 3.3, zamieszczono wizualizację elipsoidalnego obszaru ufności estymat parametrów otrzymanego z wykorzystaniem skoku jednostkowego (niebieska linia ciągła). Porównanie wyznaczonych obszarów ufności estymat parametrów modelu układu inercyjnego wykazuje pewne podobieństwa. Wymuszenie optymalnym sygnałem wejściowym (tj. dla $q \approx 0$) powoduje minimalny czas trwania eksperymentu identyfikacyjnego oraz minimalną objętość elipsoidalnego obszaru ufności estymat parametrów. Gdy wartość współczynnika kosztu energetycznego wzrasta, obszar ufności zwiększa swoją objętość dla tych samych warunków początkowych oraz wariancji szumu. Zwiększanie współczynnika kosztu energetycznego powoduje również wzrost czasu trwania eksperymentu identyfikacyjnego ale zakłócenie warunków pracy jest łagodniejsze. W ten sposób można uniknąć gwałtownych zmian sygnału sterującego w empirycznych zadaniach identyfikacji.

Minimalizacja czasu trwania eksperymentu pozwala obniżyć koszt ekonomiczny identyfikacji podczas eksploatacji systemu w czasie rzeczywistym. Z drugiej strony, koszt eksperymentu identyfikacyjnego może być również wyrażany poprzez utratę dokładności estymat parametrów w związku z zastosowanym pobudzeniem. Na podstawie danych zamieszczonych w tabelach 3.1, 3.2, 3.3, można stwierdzić, iż wymagania dotyczące minimalnego czasu trwania eksperymentu identyfikacyjnego i przyjazności sygnału sterującego są w pewnym sensie sprzeczne.

Podsumowanie

Celem tej pracy było zaprojektowanie sygnałów pobudzających w czasie swobodnym, z ograniczeniami dotyczącymi kosztu energii sterowania i wartości D-efektywności, które następnie wykorzystano w eksperymentach estymacji parametrów modelu układu inercyjnego. Eksperyment doboru sygnału sterującego przeprowadzono wykonując minimalizację dodatkowej zmiennej stanu, reprezentującej współczynnik skalowania czasu, w obecności zestawu ograniczeń. Wybrane sygnały sterujące otrzymane jako rozwiązanie problemu optymalizacji czasu swobodnego wykorzystano jako pobudzenie w procedurze estymacji parametrów modelu układu. Eksperymenty identyfikacyjne przeprowadzono w obecności białego szumu pomiarowego zakłócającego wejście sterujące systemu. Nałożone ograniczenia umożliwiają uzyskanie przyjaznych sygnałów wejściowych, jednocześnie zapewniając akceptowalne obszary ufności estymat parametrów modelu układu.

Kolejnym założeniem tej pracy było zbadanie zależności pomiędzy nałożonymi ograniczeniami na dobór optymalnego sygnału sterującego, a czasem trwania eksperymentu identyfikacyjnego. Wyniki przeprowadzonych eksperymentów potwierdzają założenie, iż sygnał sterujący uzyskany dla wartości współczynnika kosztu energii sterowania $q \approx 0$, gwarantuje minimalny czas trwania eksperymentu identyfikacyjnego oraz minimalną objętość obszaru ufności estymat parametrów. Gdy procentowa wartość współczynnika D-efektywności zmniejsza się, to zmniejsza się również czas trwania eksperymentu oraz koszt ekonomiczny procesu identyfikacji. Należy jednak stwierdzić, że suboptymalne sygnały pobudzające otrzymane dla różnych wartości współczynnika kosztu energii sterowania skutkują degradacją dokładności oszacowań estymowanych parametrów modelu układu (obserwowaną jako zwiększona objętość obszarów ufności parametrów).

Uzyskane podczas eksperymentów identyfikacyjnych wyniki pozwalają stwierdzić, iż istnieje kompromis pomiędzy czasem trwania eksperymentu identyfikacyjnego (związanego z czasem trwania pobudzenia) a dokładnością estymowanych parametrów, która zależy od przyjazności sygnału wykorzystanego do wzbudzenia modelu układu. Zatem ekonomiczny koszt identyfikacji można traktować jako czas trwania eksperymentu albo jako miarę dokładności uzyskiwanych estymat parametrów modelu układu.

Bibliografia

[1] Kalaba, R. and Spingarn, K., *Control, identification, and input optimization*. Plenum Press, New York, USA, 1982

[2] Ljung, L.: *System identification: Theory for the user.* Prentice Hall, Inc., Upper Saddle River, New Jersey, USA, 1999

[3] Pintelon, R. and Schoukens, J., *System identification: A frequency domain approach*. John Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2012

[4] Hugo, A. J., *Process controller performance monitoring and assessment. Control.* Arts Inc., 2001. Available online: http://www.controlarts.com/

[5] Hildebrand, R. and Gevers, M., *Identification for control: Optimal input design with respect to worst–case ν–gap cost function*. „SIAM Journal on Control Optimization" 2003, 41, 1586–1608

[6] Antoulas, A. and Anderson, B., *On the choice of inputs in identification for robust control*. „Automatica" 1999, 35, 1009–103

[7] Gevers, M. and Ljung, L., *Optimal experiments designs with respect to the intended model application*. „Automatica" 1986, 22, 543–554

[8] Pronzato, L., *Optimal experimental design and some related control problems*. „Automatica" 2008, 44, 303–325

[9] Hussain, M., *Review of the applications of neural networks in chemical process control–simulation and on–line implementation*. „Artificial Intelligence in Engineering" 1999, 13, 55–68

[10] Bombois, X., Scorletti, G., Van den Hof, P.M.J. and Gevers, M., *Least costly identification experiment for control: a solution based on a high-order model approximation*. [w:] CD–ROM Proc. American Control Conference, Boston, Massachusetts, USA, 2004, 2818–2823

[11] Bombois, X., Scorletti, G., Gevers, M., Van den Hof, P.M.J. and Hildebrand, R., *Least costly identification experiment for control*. „Automatica" 2006, 42, 1651–1662

[12] Narasimhan, S. and Rengaswamy, R., *Multi–objective input signal design for plant friendly identification of process systems*. Proceeding of the American Control Conference, Boston, Massachusetts, USA, 2004, 4891–4896

[13] Narasimhan, S. and Rengaswamy, R., *Multi–objective optimal input design for plant friendly identification*. Proceeding of the American Control Conference, Seattle, Washington, USA, 2008, 1304–1309

[14] Rivera, D., Braun, M. and Mittelmann, H., *Constrained multisine inputs for plant friendly identification of chemical processes*. [w:] 15th IFAC World Congress, Barcelona, Spain, vol. 15, part 1, 2002, 480–485

[15] Rivera, D., Lee, H., Braun, M. and Mittelmann, H., *Plant friendly system identification: A challenge for the process industries*. [w:] 13th IFAC Symposium on System Identification (SYSID 2003), Rotterdam, Netherlands, 2003

[16] Rafajłowicz, E. and Rafajłowicz, W., *A variational approach to optimal input signals for parameter estimation in systems with spatio–temporal dynamics*. [w:] 10th Int. Workshop in Model-Oriented Design and Analysis, Springer, Heidelberg, 2013, 219–227

[17] Rafajłowicz, E. and Rafajłowicz, W., *More safe optimal input signals for parameter estimation of linear systems described by ODE*. System modelling and optimization, IFIP AICT, vol. 443, Springer, Heidelberg, 2014, 267–277

[18] Kumar, A. and Narasimhan, S., *Robust plant friendly optimal input design*. [w:] 10th IFAC Symposium on Dynamics and Control of Process Systems, Mubai, India, 2013, 553–558

[19] Kumar, A., Nabil, M. and Narasimhan, S., *Economical and plant friendly input design for system identification*. European Control Conference, Strasbourg, France, 2014, 732–737

[20] Steenis, R. and Rivera, D., *Plant-Friendly Signal Generation for System Identification Using a Modified Simultaneous Perturbation Stochastic Approximation (SPSA) Methodology*. „IEEE Transactions on Control Systems Technology" 2011, 19, 1604–1612

[21] Rivera, D., Lee, H., Mittelmann, H. and Braun, M., *High-Purity Distillation-Using plant-friendly multisine signals to identify a strongly interactive process*. „IEEE Control Systems Magazine" 2007, 27 (5), 72–89

[22] Jakowluk, W., *Design of an optimal input signal for plant-friendly identification of inertial systems*. „Przegląd Elektrotechniczny" 2009, 85 (6), 125–129

[23] Jakowluk, W., *Design of an optimal actuation signal for identification of a torsional spring system*. „Przegląd Elektrotechniczny" 2011, 87 (6), 154–160

[24] Jakowluk, W., *Optimal input signal design for a second order dynamic system identification subject to D–efficiency constraints*. [w:] 14th IFIP TC8 Int. Conference in Computer Information Systems and Inustrial Management, Springer, Heidelberg, 2015, 351–362

[25] Schwartz, A., Polak, E. and Chen, Y., *Riots – a Matlab toolbox for solving optimal control problems*. Version 1.0 for Windows, May 1997, available at: http://www.schwartz-home.com/RIOTS/

[26] Tricaud, C. and Chen, Y., *Solving fractional order optimal control problems in riots_95 a general–purpose optimal control problem solver*. [w:] 3rd IFAC Workshop on Fractional Differentiation and its Applications, Ankara, Turkey, 2008

[27] Atkinson, A., Donev, A. and Tobias, R., *Optimum experimental design with SAS*. Oxford University Press, Oxford 2007

[28] Jakowluk, W., *Plant friendly input design for parameter estimation in an inertial system with respect to D–efficiency constraints*. „Entropy" 2014, 16(11), 5822–5837

[29] Uciński, D. and Chen, Y., *Sensor Motion Planning in Distributed Parameter Systems Using Turing's Measure of Conditioning*. Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, USA, 2006, 759–764

[30] Stryk, O., *User's guide for DIRCOL, a direct collocation method for the numerical solution of optimal control problems*. Version 2.1, Technische Universität Darmstadt, November 1999, available at: http://www.sim.informatik.tu-darmstadt.de/index/leftnav.html.

[31] Jennings, L., Fisher, M., Teo, K. and Goh, C., MISER 3: *Optimal control software. Theory and user manual*. Version 2.0, Dept. of Mathematics, University of Western Australia, Nedlands, 2002, available at: school.maths.uwa.edu.au/~les/miser/Misermanual.pdf.

Free final time optimal input design for process system identification

Abstract: Input design is the process of designing an optimal input signal that is used for the actuation of the system to maximize the information about system dynamics. The novelty of this work is to design the free final time constraint input signals which are then used in the system identification experiments. The solution of constrained optimal input design problem is based on the minimization of the extra state variable representing the free final time scaling factor, formulated in the Bolza functional form, subject to the D-efficiency constraints as well as input energy constraints. The constraints imposed on input signal design should allow the operator for model parameter estimation in the safest way. An in-novative contribution of this work is to use obtained free terminal time inputs to examine the economic aspects between the imposed constraints on the input signal

design and the experiment duration while undertaking an identification experiment. The presented methodology could be utilized for a general class of systems and is illustrated using numerical examples.

Keywords: D–efficiency, free final time problem, input signal design, optimal control

DOI: 10.24427/978-83-66391-96-3_4

Rozdział 4
Dwuprzebiegowe testy krokowe z regularnym indeksem inkrementacji

Ireneusz Mrozek
Wydział Informatyki, Politechnika Białostocka

Streszczenie: Konwencjonalne testy krokowe (ang. *march* test) pamięci RAM wyróżniają się szczególnie wysoką efektywnością w przypadku uszkodzeń prostych, takich jak: uszkodzenie sklejeniowe (ang. *Stuck-at fault* – SAF), uszkodzenie przejścia (ang. *Transition fault* – TF) czy uszkodzenie sprzężeniowe (ang. *Coupling fault* – CF). Należy jednak zaznaczyć, iż sesje testowe złożone z pojedynczego wykonania testu krokowego, są niewystarczające w przypadku uszkodzeń złożonych takich, jak uszkodzenia uwarunkowane zawartością sąsiadów (ang. *Pattern Sensitive Faults* – PSF). Jedną z technik zwiększających pokrycie uszkodzeń PSF jest technika sesji wieloprzebiegowych, której ideą jest wielokrotne wykonanie transparentnego testu krokowego przy różnych warunkach początkowych. Rozwiązanie to jest szczególnie interesujące w przypadku wbudowanych testów automatycznych (ang. *Built-in self-test* – BIST). Podejście oparte na testach transparentnych posiada niepodważalną zaletę, w stosunku do testowania tradycyjnego. Po zakończeniu procesu testowania, zawartość pamięci RAM jest taka sama jak przed rozpoczęciem testu. Nie istnieje w tym przypadku potrzeba zapisywania zawartości testowanej pamięci przed rozpoczęciem sesji testowej, a następnie jej odtwarzania po zakończeniu procesu testowania. Jest to jedna z przyczyn, dla których technika testowania pamięci RAM, oparta na wieloprzebiegowych transparentnych testach krokowych, jest szeroko stosowana do cyklicznych testów pamięci systemów o znaczeniu krytycznym (elektronika medyczna, sterowanie kolejami, awionika, telekomunikacja i innych). W wielu przypadkach dostępny czas na realizację takiej sesji testowej jest limitowany. Biorąc pod uwagę powyższe ograniczenie w pracy uwaga skupiona zostanie nad krótkimi, dwuprzebiegowymi sesjami testowymi wykorzystującymi zmienne sekwencje adresowe generowane na podstawie regularnego indeksu inkrementacji. W pracy dokonano dogłębnej analizy warunków, które muszą spełniać sekwencje adresowe, aby uzyskać jak największe pokrycie uszkodzeń. Określono optymalną wartość indeksu inkrementacji prowadzącą do uzyskania maksymalnego pokrycia uszkodzeń. Wyniki prac wskazują, iż sesje testowe wykorzystujące tak generowane sekwencje adresowe dają wyższe pokrycie uszkodzeń niż ma to miejsce w przypadku sekwencji pseudolosowych. Jako uzupełnienie zaproponowano rozwiązanie o niskim narzucie sprzętowym, umożliwiające implementację generatora sekwencji adresowych opartych na optymalnym indeksie inkrementacji.

Słowa kluczowe: RAM, BIST, PSF, testowanie pamięci, testy transparentne

Afiliacja: Publikacja zrealizowana w ramach pracy statutowej WZ/WI-IIT/2/2020.

Wprowadzenie

Pamięć półprzewodnikowa jest kluczowym elementem szeroko rozumianych współczesnych systemów i układów elektronicznych. Jej bezawaryjne działanie ma fundamentalny wpływ na zachowanie całego systemu. Procentowy udział podłoża krzemowego przeznaczonego na różne kategorie pamięci w systemach typu SoC (ang. *systems-on-chip*) jest obecnie na poziomie 90% [2]. Produkowana pamięć musi spełniać bardzo wysokie wymogi związane z miarą jakości (od 50 ppm w przypadku standardowych komputerów do poniżej10 ppm w przypadku układów stosowanych w systemach o wysokiej wiarygodności działania), [2]. Obecna technologia umożliwia produkcję układów pamięci wysokiej gęstości o niespotykanych dotychczas pojemnościach. Jednocześnie wzrost upakowania układów i tym samym zmniejszenie fizycznych odległości pomiędzy poszczególnymi jego elementami sprzyja niezamierzonym interakcjom pomiędzy sąsiadującymi komórkami pamięci. Fizyczną przyczyną tych interakcji są z reguły prądy upływu [8]. W konsekwencji, wzajemna interakcja międzykomórkowa zależy w znacznej mierze od wartości logicznych występujących w tych komórkach, czego skutkiem są błędy występujące tylko w kontekście specyficznych wartości (wzorców) tam zapisanych. Wpływa to znacząco na wzrost stopnia złożoności procesów testowania pamięci. Należy podkreślić, że w obecnych pamięciach DRAM błędy będące wynikiem wzajemnego oddziaływania komórek na siebie są jednymi z najczęściej zgłaszanych podczas procesów testowania [5]. Mimo, że waga i mechanizm powstawania tych uszkodzeń znane są od dawna, to wciąż nie zostały opracowane metody umożliwiające dokładną i efektywną analizę pamięci pod tym kątem [5]. Fakt ten sprawia, iż nadal istnieje zapotrzebowanie rozwijania technik i algorytmów w tym obszarze.

Na potrzeby testowania pamięci zdefiniowanych zostało wiele modeli uszkodzeń (ang. *fault models*). Definicje podstawowych z nich można znaleźć w pracach Thatte i Abrahama w [6]. Liczna grupa modeli została zaproponowana przez Van de Goora w [8]: uszkodzenia dekodera adresu (ang. *address decoder faults* – AF), uszkodzenia sklejeniowe (ang. *stuck-at faults* – SAF), uszkodzenia sprzężeniowe (ang. *coupling faults* – CF), uszkodzenia przejścia (ang. *transition faults* – TF) czy uszkodzenia uwarunkowane zawartością (ang. *pattern sensitive faults* – PSF), w tym uszkodzenia uwarunkowane zawartością sąsiadów (ang. *neighborhood pattern sensitive faults* – NPSF). Rozwinięcia powyższych modeli dla różnych typów pamięci można znaleźć m.in w [1, 3, 4, 5, 7]. Przyjęło się również dzielić uszkodzenia pamięci ze względu na liczbę komórek, które są powiązane danym uszkodzeniem. Dotyczy to uszkodzeń prostych związanych z pojedynczymi komórkami pamięci (ang. *one cell faults*) i uszkodzeń złożonych, wiążących zależnością wiele komórek pamięci (ang. *multi-cell faults*).

Ze względu na szybko rosnącą pojemność i gęstość upakowania układów pamięci, szczególnego znaczenia nabierają modele prowadzące do opracowania testów umożliwiających efektywne wykrywanie uszkodzeń, których źródłem jest wzajemna interakcja pomiędzy poszczególnymi komórkami pamięci. Defekty, których rezultatem

są interakcje między komórkami pamięci najczęściej są analizowane z wykorzystaniem dwóch dobrze udokumentowanych w literaturze modeli uszkodzeń: uszkodzenia sprzężeniowego CF i uszkodzenia uwarunkowanego zawartością PSF [8]. Uszkodzenie CF opisuje interakcję dwóch dowolnych komórek pamięci [8]. O uszkodzeniu tym mówi się wówczas, gdy zmiana wartości (lub określona wartość) w komórce określanej mianem agresora wpływa na wartość w komórce określanej mianem ofiary.

Uogólnionym modelem umożliwiającym analizę oddziaływania na siebie większej liczby komórek, niż ma to miejsce w przypadku uszkodzenia CF, jest model uszkodzenia uwarunkowanego zawartością – PSF. Uszkodzenie PSF jest najbardziej ogólnym modelem opisującym niezamierzone interakcje pomiędzy komórkami pamięci. Może ono być rozpatrywane jako uogólnienie uszkodzenia sprzężeniowego [8]. Model ten składa się z grupy komórek będących agresorami (komórkami wiążącymi) i jednej komórki będącej ofiarą (komórką bazową). W uszkodzeniu PSF stan bazowej komórki pamięci (wartość, lub zmiana wartości) zależny jest od stanów wszystkich pozostałych komórek (komórek wiążących). Testowanie pamięci pod kątem powyższego uszkodzenia jest jednak niepraktyczne czy wręcz niemożliwe z uwagi na czas niezbędny do realizacji takiego testu. Bardziej praktyczny jest model uszkodzenia uwarunkowanego zawartością sąsiadów NPSF. W odróżnieniu od uszkodzenia PSF w uszkodzeniu NPSF przyjmuje się, iż na komórkę bazową oddziałują tylko komórki z jej najbliższego fizycznego sąsiedztwa. Opierając się na powyższym modelu zostało zaproponowanych wiele klasycznych rozwiązań, np. w [8, 10]. Jednakże w przypadku współczesnych pamięci, algorytmy te charakteryzują się zbyt długim czasem wykonania. W rezultacie opracowano wiele nowych propozycji odnoszących się do uszkodzeń NPSF, np. [5, 8, 11, 17]. Niestety, rozwiązania oparte na modelu NPSF nie zawsze mogą być wykorzystane. Źródłem problemu jest optymalizacja stosowana podczas produkcji układów cyfrowych, której wynikiem jest różnica pomiędzy logicznym (widzianym przez użytkownika) i fizycznym ułożeniem komórek pamięci względem siebie (ang. *scrambling*), [18]. Dlatego do efektywnego wykorzystania powyższych rozwiązań niezbędny staje się schemat pozwalający określić fizyczne rozmieszczenie komórek pamięci (ang. *scrambling information*), [1, 8]. Schemat ten nie zawsze jest upubliczniany przez producentów, ponadto mechanizmy redundancji (ang. *memory row and column redundancy*) sprawiają, że konfiguracja pamięci może ulec zmianie w sposób niezauważalny dla użytkownika podczas jej normalnej pracy [18]. Jednocześnie, w przypadku rozpatrywanych uszkodzeń złożonych, koniecznym warunkiem (w większości wypadków wystarczającym) wykrycia (lub aktywacji) wszystkich potencjalnych k-komórkowych uszkodzeń w wybranych k komórkach pamięci, jest wygenerowanie wszystkich 2^k możliwych binarnych kombinacji w tych komórkach. Dlatego w sytuacji, gdy nie jest znana topologia testowanej pamięci jako model uszkodzenia można przyjąć model PSFk, będący uogólnieniem modelu NPSF. W modelu PSFk rozpatrywanych jest dowolne k z N komórek pamięci (N – rozmiar pamięci), wśród których jedna jest komórką bazową, a pozostałe $k - 1$ – komórkami wiążącymi. Należy

zaznaczyć, iż rezultaty otrzymane dla tak zdefiniowanego modelu, można łatwo odnieść do innych modeli, gdyż jest on najbardziej złożony w procesie wykrywania uszkodzeń pamięci [19].

Wiele różnych algorytmów zostało zaproponowanych w obszarze testowania pamięci RAM. Kluczowe wydają się podejścia oparte na testach krokowych nazywanych również maszerującymi (ang. *march tests*). Ich największymi zaletami są: złożoność liniowa w stosunku do rozmiaru pamięci, wysoki poziom pokrycia uszkodzeń i regularność struktury umożliwiająca stosunkowo prostą ich implementację wbudowaną (ang. *built-in self-test* – BIST), [8]. Jednak tradycyjne testy krokowe nie generują wszystkich możliwych wzorców niezbędnych do detekcji uszkodzeń PSF*k*. Dlatego zaproponowano techniki, wykorzystujące w swej podstawie testy krokowe, umożliwiające zwiększenie pokrycia uszkodzeń PSF*k*. Jednym z takich podejść jest testowanie wieloprzebiegowe. Technika ta polega na wielokrotnej realizacji zadanego testu przy jednoczesnej zmianie warunków początkowych (np. zawartości pamięci i/lub sekwencji adresowej).

W przypadku pojedynczego przebiegu testu krokowego warunki początkowe nie mają wpływu na finalne pokrycie uszkodzeń [8,12]. Dla dowolnej sekwencji adresowej oraz dowolnej zawartości pamięci liczba wykrywanych uszkodzeń jest zawsze taka sama i może zostać określona na podstawie przyjętych modeli uszkodzeń. W przypadku testów wieloprzebiegowych w literaturze dziedziny jasno wskazano, iż zmiana warunków początkowych w sposób znaczący wpływa na finalne pokrycie uszkodzeń testu. Dlatego jednym z kluczowych czynników wpływających na pokrycie uszkodzeń w przypadku wieloprzebiegowych testów krokowych są sekwencje adresowe kolejnych iteracji testów oraz wzajemne relacje między tymi sekwencjami. [13]. Różne sekwencje adresowe w kolejnych iteracjach testu pozwalają wygenerować nowe wzorce w komórkach pamięci co umożliwia aktywację i wykrycie dodatkowych uszkodzeń w porównaniu z iteracjami wcześniejszymi. Na przykład w przypadku dwóch przebiegów testu pamięci, musimy wybrać dwie sekwencje adresowe, które muszą się różnić pod względem zdolności wykrywania różnych konfiguracji uszkodzeń PSF. Jak zostało wykazane, różne pary sekwencji adresowych skutkują w różnym pokryciu uszkodzeń [16]. Niezmiernie ważnym problemem jest zatem wybranie odpowiednich sekwencji adresowych, uwzględniając przy tym nie tylko pokrycie uszkodzeń, ale również koszt wygenerowania tych sekwencji.

W dalszej analizie będzie rozważany model uszkodzenia PSF*k* i rozpatrywana pamięć RAM o pojemności $N = 2^m$ bitów, gdzie m jest dodatnią liczbą całkowitą oznaczającą liczbę linii adresowych pamięci. Jak pokazują liczne prace van de Goor'a, Nicolaidis'a i innych, w przypadku złożonych uszkodzeń pamięci, takich jak PSF*k*, pokrycie uszkodzeń na poziomie 100% można osiągnąć w bardzo ograniczonych przypadkach. W ogólnym przypadku 100% pokrycie złożonych uszkodzeń pamięci można osiągnąć poprzez realizację nieskończonej liczby testów lub wykorzystując test o złożoności rzędu 2^N, gdzie N to rozmiar pamięci. Dlatego we współczesnych badaniach związanych z detekcją złożonych uszkodzeń pamięci RAM dąży się do uzyskania

wysokiego pokrycia uszkodzeń przy umiarkowanej złożoności testów. Stąd badania autora zostały skupione na bardzo krótkich, dwuprzebiegowych testach krokowych. Głównymi wynikami są:

- wyznaczenie parametrów umożliwiających wygenerowanie optymalnych (w kontekście pokrycia uszkodzeń) sekwencji licznikowych jako sekwencji adresowych w dwuprzebiegowych testach krokowych,
- zaproponowanie rozwiązania umożlwiającego implementację, o niskiej złożoności sprzętowej, generatora optymalnych sekwencji adresowych,
- dokładna analityczna analiza proponowanego rozwiązania w kontekście detekcji uszkodzeń PSF.

Niski narzut sprzętowy w proponowanym rozwiązaniu osiągnięto przez implementację generatora adresów w oparciu o licznik sekwencyjny. Generator adresów posiada dwa tryby pracy. W pierwszej iteracji testu działa on jak standardowy licznik realizując działania (+1) i (–1) (pierwszy tryb pracy). W przypadku drugiej iteracji sekwencja adresowa generowana jest w oparciu o licznik modyfikujący swój stan zgodnie z indeksem inkrementacji q: ($+q$) i ($-q$) (drugi tryb pracy). Przeprowadzona analiza pozwoliła określić optymalną wartość indeksu inkrementacji q jak również adres początkowy drugiej sekwencji adresowej umożliwiające uzyskanie wysokiego pokrycia uszkodzeń PSFk przez dwuprzebiegowy test krokowy.

4.1. Efektywność testów krokowych

Kluczową rolę w testowaniu pamięci odgrywają testy krokowe (maszerujące), [8]. Test krokowy składa się ze skończonej liczby sekwencji elementów typu March. Element typu March składa się ze skończonej liczby sekwencji operacji czytania (r) i pisania (w), z których wszystkie oddziałują na określoną komórkę przed przejściem do następnej komórki pamięci. Komórka następna określona jest zgodnie z sekwencję adresową oznaczaną przez ⇑ (nazywana często sekwencją rosnąca), lub ⇓ (nazywaną często sekwencją malejącą). Uszeregowanie adresów w sekwencji adresowej może być dowolne, przy założeniu jednak, iż w sekwencji adresowej oznaczonej symbolem ⇑ kolejność adresów jest odwrotna do sekwencji oznaczonej symbolem ⇓ [8, 15, 24]. Symbol ⇕ wykorzystywany jest w przypadku, gdy możemy przyjąć dowolnie albo sekwencję rosnącą albo sekwencję malejącą. Możliwe działania odnoszące się do komórek pamięci dostępne w testach krokowych to: wx – zapisz do komórki pamięci wartość x i rx – odczytaj wartość z komórki pamięci (wartość spodziewana jest równa x). Cały test krokowy ograniczony jest przez parę nawiasów klamrowych { ... }, podczas gdy element typu March ograniczony jest parą nawiasów okrągłych (...). Jako przykład rozpatrzymy test MATS+ [23]:

$$\{ \Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1,w0) \}.$$
$$M0 \qquad M1 \qquad M2$$

Test ten składa się z trzech elementów typu March: M0, M1, M2. Element M0 zeruje pamięć. Element M1, w porządku sekwencji rosnącej odpowiednio czyta wartość 0 z komórki, następnie zapisuje tam wartość 1. Element M2 testu MATS+, w porządku adresów sekwencji malejącej odczytuje wartość 1 z komórki, po czym do tej samej komórki wpisuje wartość 0. Jednymi z najbardziej znanych testów krokowych są testy MATS++ {⇕ (w0); ⇑(r0,w1); ⇓(r1,w0,r0)} i March C– {⇕ (w0); ⇑(r0,w1); ⇑(r1,w0); ⇓(r0,w1); ⇓(r1,w0); ⇕ (r0)}. W przypadku tradycyjnych testów korkowych pierwsza faza testu, ustala początkową zawartość pamięci. Dlatego testy krokowe w postaci klasycznej nie są predysponowane do stosowania w procedurach testowych uruchamianych w systemach w trakcie ich normalnej pracy (ang. *online testing*). W tym celu należy wykorzystać testy transparentne (ang. *transparent tests*).

Technika transparentnego testowania pamięci jest dobrze znanym podejściem umożliwiającym przeprowadzenie procesu testowania pamięci w sposób przezroczysty dla jej zawartości. Zaproponowana została w 1986 roku przez B. Koenemana [20]. Technikę transformacji klasycznych testów krokowych w testy transparentne zaproponował Nicolaidis [22]. Testy transparentne umożliwiają zachowanie niezmienionej zawartości pamięci po zakończeniu testu w stosunku do zawartości z momentu rozpoczęcia testu. Dzięki temu są one szczególnie predysponowane do realizacji periodycznych testów wykonywanych w czasie normalnej pracy urządzenia (ang. *periodic field testing*). Wybrane rozwiązania oparte na testach transparentnych można znaleźć m.in. w [14,17, 21]. Transparentne odpowiedniki testów MATS++ i March C– mają odpowiednio postać {⇑(ra,w$\bar{a}$); ⇓(r$\bar{a}$,wa,ra)} i {⇑(ra,w$\bar{a}$); ⇑(r$\bar{a}$,wa); ⇓(ra,w$\bar{a}$); ⇓(r$\bar{a}$,wa); ⇕ (ra)}, gdzie $a \in \{0, 1\}$ i $\bar{a}$ jest inwersją a. Inne testy krokowe, ich złożoności i efektywność w odniesieniu do uszkodzeń prostych przedstawia tabela 4.1 [8].

TABELA 4.1. Pokrycie uszkodzeń prostych przez wybrane testy krokowe

Test	Pokrycie uszkodzeń							
	SAF	AF	TF	CFin	CFid	CFdyn	SCF	Złożoność
MATS	+							4*N*
MATS+	+	+						5*N*
MATS++	+	+	+					6*N*
March X	+	+	+	+				6*N*
March C–	+	+	+	+	+	+	+	10*N*
March A	+	+	+	+				15*N*
March Y	+	+	+	+				8*N*
March B	+	+	+	+				17*N*

„+" oznacza pełne pokrycie danego typu uszkodzeń przez test.

Należy podkreślić, iż czas realizacji testu zależy nie tylko od samego testu, ale również od technologii, w jakiej została zrealizowana pamięć. Dlatego do opisu czasu trwania testu powszechnie używana jest złożoność testu, przez którą rozumie się liczbę poleceń *r*/w występujących w teście [8,22,25,31]. Wartość ta jest bezpośrednio skorelowana z czasem niezbędnym do realizacji testu i jest jednocześnie niezależna od technologii. Dane, które pokazuje tabela 4.1 wskazują, iż testy krokowe posiadają wysoką efektywność w odniesieniu do uszkodzeń prostych. Jednocześnie w przypadku uszkodzeń złożonych efektywność ta jest niewystarczająca. Dla testu złożonego z jednej iteracji testu MATS++ pokrycie uszkodzeń PSF*k* można określić jako $FC_{MATS++}(PSFk) = (1/2^{k-1})100\%$, a w przypadku testu March C– $FC_{MATS++}(PSFk) = (1/2^{k-2})100\%$ [8,27]. Przykładowo, $FC_{MATS++}(PSF3) = (1/2^2)100\% = 25\%$, $FC_{MATS++}(PSF5) = (1/2^4)100\% = 6{,}25\%$, $FC_{MarchC-}(PNPSF3) = (1/2^1)100\% = 50\%$, i $FC_{MarchC-}(PNPSF5) = (1/2^3)100\% = 12{,}5\%$.

4.2. Dwuprzebiegowe testy krokowe

Jak wskazano we wprowadzeniu, do uzyskania wysokiego pokrycia uszkodzeń PSF*k* niezbędne jest wykorzystanie optymalnych sekwencji adresowych [27, 30]. W przypadku dwuprzebiegowej sesji testowej wykorzystywane są dwie sekwencje adresowe A_j i A_k, gdzie $A_l = A_l(0)A_l(1)A_l(2) \ldots A_l(N-2)A_l(N-1)$; $(A_l(i) \in \{0, 1, 2, \ldots, N-1\}, i \in \{0, 1, 2, \ldots, N-1\})$. Należy zauważyć, iż liczba możliwych różnych sekwencji adresowych A_l, $l \in \{0, 1, 2, \ldots, N!-1\}$ jest bardzo duża i równa $N! = 2^m!$ [27].

W celu uzyskania wysokiego pokrycia uszkodzeń PSF*k* dwuprzebiegowego testu krokowego niezbędne jest, aby dwie przyjęte sekwencja adresowe A_j i A_k, w jak największym stopniu różniły się między sobą [25]. W najprostszym przypadku oznacza to, iż na tych samych pozycjach *i* w sekwencjach adresowych A_j i A_k, powinny być różne wartości. Wymóg ten spełniony jest również w standardowych testach krokowych [8]. W kolejnych fazach testu krokowego generowane są dwie różne sekwencje adresowe: rosnąca 0, 1, 2, …, 2^m-2, 2^m-1 (⇑) i malejąca 2^m-1, 2^m-2, 2^m-3, …, 1, 0 (⇓). Jako miara zróżnicowania sekwencji adresowych może zostać wykorzystana metryka Manhattan:

$$D_{Manh}(A_j, A_k) = \sum_{i=0}^{N-1} \left| A_j(i) - A_k(i) \right|, \tag{4.1}$$

która została przeanalizowana i eksperymentalnie sprawdzona w [27].

Odległość miejska $D_{Manh}(A_j,A_k)$ pomiędzy dwoma sekwencjami adresowymi A_j i A_k, może zostać wykorzystana jako charakterystyka numeryczna umożliwiająca określenie poziomu zróżnicowania dwóch sekwencji adresowych. Przykładowo, dla dwóch sekwencji $A_j = 0, 1, 2, \ldots, 2^m-1, 2^m-1$ i $A_k = 2^m-1, 2^m-2, 2^m-3, \ldots, 1, 0$ powyższa odległość jest równa $D_{Manh}(A_j,A_k) = 2^{2m-1}$. Jak zostało udowodnione w [29] wartość 2^{2m-1} jest

maksymalną możliwą wartością $maxD_{Manh}(A_j,A_k)$ powyższej metryki dla dwóch różnych sekwencji adresowych A_j, A_k. Jednocześnie wartością minimalną $minD_{Manh}(A_j,A_k)$ jest wartość 2.

Istnieje wiele rozwiązań umożliwiających generowanie sekwencji adresowych. Wszystkie opierają się na prostych algorytmach, takich jak generowanie sekwencji pseudolosowych, negacja bitów adresu, przesunięcie i/lub permutacje bitów adresu oraz wiele różnych sekwencji liczbowych opartych na kodzie Gray'a, odwrotnym kodzie Gray'a, maksymalizacji średniej odległości Hamminga i innych [27, 29, 30, 32]. Wszystkie te rozwiązania były proponowane w kontekście zapewnienia niskiej złożoności sprzętowej implementacji mechanizm*ów testowania wbudowanego BIST*, jak również osiągnięcia jak najwyższego pokrycia uszkodzeń w tym PSF*k*.

Przykładowe pokrycie uszkodzeń PSF*k* przez dwuprzebiegowy test wykorzystujący pseudolosowe sekwencje adresowe przedstawia tabela 4.2. Kolumna „Iteracja pierwsza" zawiera pokrycie uszkodzeń po pierwszej iteracji testu, zaś kolumna „Iteracja druga" przedstawia sumaryczne pokrycie po wykonaniu drugiej iteracji testu. W obu przypadkach (zarówno dla testu MATS++ jak i March C–) sekwencje adresowe generowane były oparte na rejestrze przesuwającym LFSR stowarzyszonym z wielomianem pierwotnym $\varphi(x)$ [26,28,33]. W przypadku tego eksperymentu wartość miary zróżnicowania dwóch losowych sekwencji adresowych A_j i A_k może być aproksymowana przez $N^2/4$ [33].

TABELA 4.2. **Pokrycie uszkodzeń PSF*k* (%) dwuprzebiegowego testu krokowego z pseudolosowymi sekwencjami adresowymi**

Test	PSF3		PSF5	
	Iteracja pierwsza	Iteracja druga	Iteracja pierwsza	Iteracja druga
MATS++	24,90	42,74	6,31	11,87
March C–	49,87	72,74	12,46	22,15

W kolejnym eksperymencie (tabela 4.3) jako sekwencję adresową w pierwszej iteracji testu wykorzystano standardową sekwencję licznikową A_j, zaś w drugiej iteracji wykorzystaną sekwencję A_k będącą modyfikacją sekwencji A_j. Modyfikacja polegała na negacji najmniej znaczącego bitu w każdym adresie sekwencji adresowej A_j. W tym przypadku wartość zróżnicowania sekwencji adresowych jest równa $D_{Manh}(A_j,A_k) = N$, co jest wartością znacząco mniejszą w porównaniu z wynikiem poprzednim.

TABELA 4.3. Pokrycie uszkodzeń PSF*k* (%) dwuprzebiegowego testu krokowego z drugą sekwencją adresową wygenerowaną przez modyfikację pierwszej sekwencji (negacja bitu LSB pierwszej sekwencji)

Test	PSF3		PSF5	
	Iteracja pierwsza	Iteracja druga	Iteracja pierwsza	Iteracja druga
MATS++	25,08	25,84	6,20	11,87
March C–	50,10	51,72	12,57	13,36

Wyniki, które zawierają tabela 4.2 i tabela 4.3 wskazują na dużą zależność pokrycia uszkodzeń od wartości zróżnicowania sekwencji adresowych. W przypadku dwóch sekwencji adresowych A_j i A_k, dla których wartość zróżnicowania $D_{Manh}(A_j,A_k)$ jest duża pokrycie uszkodzeń jest znacznie większe w porównaniu z sytuacją, gdy wartość zróżnicowania sekwencji adresowych jest stosunkowo niska.

4.2.1. Dwuprzebiegowe testy krokowe z indeksem inkrementacji $q = 2$

W punkcie tym zostanie przeanalizowany przypadek, gdzie druga sekwencja adresowa testu dwuprzebiegowego zostanie wygenerowana na podstawie pierwszej sekwencji z wykorzystaniem indeksu inkrementacji o wartości $q = 2$. Jako pierwszą sekwencję adresową przyjmijmy dowolną sekwencje liczbową $A_j(i)$, gdzie $A_j(i) \in \{0,1,2,\ldots,2^m-1\}$, $i \in \{0,1,2,\ldots,2^m-1\}$. Wtedy druga sekwencja adresowa będzie miała postać $A_k = A_k(0)$ $A_k(1)A_k(2)\ldots A_k(2^m-2)A_k(2^m-1)$ zdefiniowaną zgodnie z poniższą definicją.

Definicja 4.1: Sekwencja adresowa A_k będąca wynikiem inkrementacji sekwencji adresowej $A_j = A_j(0)A_j(1)A_j(2)\ldots A_j(2^m-2)A_j(2^m-1)$ zgodnie z indeksem $q = 2$ będzie miała następująca postać:

$$\begin{aligned} A_k(i) &= A_j(2i), && i \in \{0, 1, 2, \ldots, 2^{m-1}-1\}; \\ A_k(i) &= A_j(2(i-2^{m-1})+1), && i \in \{2^{m-1}, 2^{m-1}+1, 2^{m-1}+2, \ldots, 2^m-1\}. \end{aligned} \tag{4.2}$$

Należy zauważyć, iż Definicja 4.1 może zostać wykorzystana w odniesieniu do dowolnej sekwencji adresowej A_j, w tym standardowej sekwencji licznikowej $A_j = A_j(0)A_j(1)$ $A_j(2) \ldots A_j(2^m-2)A_j(2^m-1) = 0, 1, 2, \ldots, 2^m-2, 2^m-1$. Przykładowo, dla standardowej sekwencji licznikowej dla $m = 4$ $A_j = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$ sekwencja wynikowa A_k przyjmie wartość 0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13, 15.

W zależności od adresu początkowego s, istnieje $N = 2^m$ wersji sekwencji A_k otrzymanych w oparciu o standardową sekwencję licznikową $A_j = A_j(0)A_j(1)A_j(2) \ldots A_j(2^m - 2)A_j(2^m - 1) = 0, 1, 2, \ldots, 2^m - 2, 2^m - 1$. Wszystkie sekwencje adresowe w zależności od adresu początkowego s dla $m = 3$ i indeksu inkrementacji $q = 2$, wygenerowane na podstawie standardowej sekwencji licznikowej zawiera tabela 4.4.

TABELA 4.4. Zbiór sekwencji adresowych wygenerowanych na podstawie standardowej sekwencji licznikowej dla $m = 3$ i indeksu inkrementacji $q = 2$

I	$A_j(i)$	A_k							
		$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
0	0	0	2	4	6	1	3	5	7
1	1	2	4	6	1	3	5	7	0
2	2	4	6	1	3	5	7	0	2
3	3	6	1	3	5	7	0	2	4
4	4	1	3	5	7	0	2	4	6
5	5	3	5	7	0	2	4	6	1
6	6	5	7	0	2	4	6	1	3
7	7	7	0	2	4	6	1	3	5

Jak zostało już wskazane, w przypadku testu dwuprzebiegowego muszą zostać wybrane dwie optymalne sekwencje adresowe [27, 29, 30]. Jako miara dopasowania może zostać wykorzystana metryka miejska [27]. Wartości metryk $D_{Manh}(A_j, A_k(s))$ dla $m = 3$ i $m = 4$ zawiera tabela 4.5 i tabela 4.6.

TABELA 4.5. Wartości metryki zróżnicowania dla $q = 2$ i $m = 3$

s	0	1	2	3	4	5	6	7
$D_{Manh}(A_j, A_k(s))$	12	20	20	24	24	24	24	20

TABELA 4.6. Wartości metryki zróżnicowania dla $q = 2$ i $m = 4$

s	0	1	2	3	4	5	6	7
$D_{Manh}(A_j, A_k(s))$	12	20	20	24	24	24	24	20
s	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
$D_{Manh}(A_j, A_k(s))$	96	96	96	92	92	84	84	72

Dane zawarte w powyższych tabelach pozwalają stwierdzić, iż w przypadku $m = 3$ zbiór optymalnych par sekwencji adresowych A_j i $A_k(s)$ to $\{(A_j,A_k(3)), (A_j,A_k(4)), (A_j,A_k(5)), (A_j,A_k(6))\}$, zaś w przypadku $m = 4$ zbiór par sekwencji optymalnych to $\{(A_j,A_k(7)), (A_j,A_k(8)), (A_j,A_k(9)), (A_j,A_k(10))\}$.

Z uwagi na regularną strukturę sekwencji adresowej $A_k(s)$ nie jest trudno pokazać, iż wartość powyższej metryki zróżnicowania, dla dowolnego rozmiaru pamięci $N = 2^m$ oraz dowolnego adresu początkowego s, można wyznaczyć na podstawie (4.3).

$$D_{Manh}(A_j, A_k(s)) = \begin{cases} \dfrac{s(2^m - s + 2)}{2} + 2^{2m-2} - 2^{m-1}, & \textit{dla parzystych } s; \\ \dfrac{s(2^m - s) + 1}{2} + 2^{2m-2}, & \textit{dla nieparzystych } s. \end{cases} \tag{4.3}$$

Optymalna wartość adresu początkowego s w kontekście otrzymania największej wartości metryki zróżnicowania $D_{Manh}(A_j,A_k(s))$ dla parzystego s można otrzymać na podstawie (4.4).

$$\frac{dD_{Manh}(A_j, A_k(s))}{ds} = 2^{m-1} - s = 0. \tag{4.4}$$

Najbliższe parzyste wartości będące rozwiązaniem powyższego równania to $s = 2^{m-1}$ i $s = 2^{m-1} + 2$. W przypadku $m = 3$ i $m = 4$ są to wartości odpowiednio 4,6 oraz 8,10 (porównaj tabela 4.5 i tabela 4.6).

Optymalny nieparzysty adres początkowy s może zostać wyznaczony na podstawie (4.5).

$$\frac{dD_{Manh}(A_j, A_k(s))}{ds} = 2^{m-1} - s = 0. \tag{4.5}$$

Najbliższe nieparzyste wartości będące rozwiązaniem powyższego równania to $s = 2^{m-1} - 1$ oraz $s = 2^{m-1} + 1$. W przypadku $m = 3$ i $m = 4$ są to wartości odpowiednio 3,5 oraz 7,9 (porównaj tabela 4.5 i 4.6).

4.2.2. Dwuprzebiegowe testy krokowe z indeksem inkrementacji $q = 3$

Dla pamięci o rozmiarze $N = 2^m$ bitów, gdzie m dowolna całkowita liczba dodatnia, zbiór adresów dostępnych komórek pamięci to $A_l(i) \in \{0,1,2,\ldots, 2^m-1\}$, $i \in \{0,1,2,\ldots,2^m-1\}$. Przed zdefiniowaniem sekwencji adresowej będącej rezultatem inkrementacji sekwencji bazowej o indeks $q = 3$, udowodnione zostanie stwierdzenie 4.1.

Stwierdzenie 4.1: Relacja (2^m-1) ***mod*** $3 \neq 2$ jest prawdziwa dla dowolnej całkowitej dodanej wartości m.
Dowód:
Załóżmy, że (2^m-1) ***mod*** $3 = 2$, wówczas $(2^m-1) = 3p+2$,gdzie p jest dodatnią liczbą całkowitą. Po przekształceniu otrzymamy $3p = 2^m-3$. Z ostatniego równania wynika, iż 2^m jest podzielne przez 3 co nie jest prawdą, dlatego (2^m-1) ***mod*** $3 \neq 2$.

□

W następstwie powyższego stwierdzenia można przedstawić Własność 4.1 i Własność 4.2.

Własność 4.1: (2^m-1) ***mod*** $3 \in \{0, 1\}$.
Uwzględniając Własność 4.1 oraz równanie 4.6

$$(a \otimes b) \textbf{ mod } d = ((a \textbf{ mod } d) \otimes (b \textbf{ mod } d)) \textbf{ mod } d, \tag{4.6}$$

dla liczb całkowitych a, b i d, gdzie $\otimes \in \{+, \times\}$, prawdziwa jest własność 4.2.

Własność 4.2: $((2^m-1)+2)$ ***mod*** $3 = (2^m+1)$ ***mod*** $3 \in \{0,2\}$.

Stwierdzenie 4.2: Dla parzystych m (2^m-1) ***mod*** $3 = 0$, zaś dla nieparzystych m $(2^m - 1)$ ***mod*** $3 = 1$
Dowód:
Załóżmy, iż $m = 2n$ jest parzystą liczbą całkowitą, wówczas $(2^m-1) = (2^{2n}-1) = (2^n+1)(2^n-1)$. Biorąc pod uwagę Własność 4.1 otrzymujemy (2^n-1) ***mod*** $3 = 0$ lub 1 i jednocześnie (2^n+1) ***mod*** 3 odpowiednio równe jest 2 lub 0. Biorąc po uwagę założenie początkowe oraz równanie 4.6 otrzymujemy (2^m-1) ***mod*** $3 = ((2^n+1) \times (2^n-1))$ ***mod*** $3 = (((2^n+1)$ ***mod*** $3) \times ((2^n-1)$ ***mod*** $3))$ ***mod*** 3.

Rozważmy nieparzystą wartość m. Jest łatwo zauważyć, iż $(2 \times (2^m-1))$ ***mod*** $3 = ((2^{m+1}-1) -1)$ ***mod*** $3 = 2$. Można to stwierdzić biorąc pod uwagę, że $(m + 1)$ jest parzyste, a (-1) mod 3 = 2. Dlatego w rozpatrywanym przypadku (2^m-1) ***mod*** 3 jest równe 1.

□

Biorąc pod uwagę Stwierdzenie 4.2, zdefiniowana zostanie postać sekwencji adresowej będacej wynikiem inkrementacji sekwencji bazowaj o indeks $q = 3$.

Definicja 4.2: Sekwencja adresowa A_k będąca wynikiem inkrementacji sekwencji adresowej $A_j = A_j(0)A_j(1)A_j(2)\ldots A_j(2^m-2)A_j(2^m-1)$ zgodnie z indeksem $q = 3$ dla dowolnej sekwencji A_j i $A_j = 0, 1, 2, \ldots, 2^m-2, 2^m-1$ będzie miała następująca postać:

$$A_k(i) = A_j(3i \textbf{ mod } (2^m-1)) = 0, 3, 6, \ldots, 2^m-4, 2^m-1, 2, 5, 8, \ldots,$$
$$2^m-5, 2^m-2, 1, 4, 7, \ldots, 2^m-6, 2^m-3; \qquad \textit{dla parzystych } m, \tag{4.7}$$

$$A_k(i) = A_j(3i \ \mathbf{mod}\ (2^m-1)) = 0, 3, 6, \ldots, 2^m-5, 2^m-2, 1, 4, 7, \ldots,$$
$$2^m-4, 2^m-1, 2, 5, 8, \ldots, 2^m-6, 2^m-3; \qquad \textit{dla nieparzystych } m. \tag{4.8}$$

Wartości otrzymanych metryk $D_{Manh}(A_j,A_k(s))$ przedstawiają tabela 4.7 i tabela 4.8. Krótka analiza wyników zawartych w tych tabelach pozwala stwierdzić, że wartość metryki zróżnicowania silnie zależy od adresu początkowego *s*.

TABELA 4.7. **Wartości metryki zróżnicowania dla $q = 3$ i $m = 3$**

s	0	1	2	3	4	5	6	7
$D_{Manh}(A_j,A_k(s))$	16	20	16	24	24	20	24	24

TABELA 4.8. **Wartości metryki zróżnicowania dla $q = 3$ i $m = 4$**

s	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
$D_{Manh}(A_j,A_k(s))$	72	68	72	88	80	84	96	88
s	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
$D_{Manh}(A_j,A_k(s))$	88	100	88	88	96	84	80	88

Zdefiniowana zostanie teraz metryka $D_{Manh}(A_j,A_k(s))$ jako funkcja rozmiaru pamięci i adresu początkowego sekwencji adresowej będącej wynikiem inkrementacji sekwencji bazowej zgodnie z indeksem $q = 3$.

W przypadku parzystych wartości *m* zastosujmy następujące podstawienie: $a = (N - 1)/3$. Wówczas kolejne adresy sekwencji A_j i $A_k(s = 0)$ można zapisać w postaci:

$$A_j = 0, 1, \ldots, a, a+1, a+2, (3a+1)/2, (3a+1)/2+1, (3a+1)/2+2, \ldots, 2a, 2a+1, 2a+2, \ldots, 3a. \tag{4.9}$$

$$A_k(s = 0) = 0, 3, \ldots, 3a, 2, 5, \ldots, (3a+1)/2, (3a+1)/2+3, (3a+1)/2+6, \ldots, 3a-1, 1, 4, ..., 3a-2$$

Następnie, dla parzystej wartości *m* i $N = 2^m$ wartość miary zróżnicowania można wyznaczyć jako:

$$D_{Manh}(A_j, A_k(s=0)) = \sum_{i=0}^{N-1} \left|A_j(i) - A_k(i)\right| = 2+4+\ldots+2a+2+4+\ldots+a-1+2+4+\ldots$$
$$\ldots+a-1+2+4+\ldots+2a = \frac{5a^2+4a-1}{2} = \frac{5N^2+2N-16}{18}. \tag{4.10}$$

Podobnie w przypadku nieparzystej wartości *m* można pokazać, iż wartość miary zróżnicowania sekwencji adresowych może być wyznaczona poprzez:

$$D_{Manh}(A_j, A_k(s=0)) = \sum_{i=0}^{N-1} \left|A_j(i) - A_k(i)\right| = \frac{5N^2-2N-16}{18}. \tag{4.11}$$

Bardziej złożona sytuacja jest w przypadku $s \neq 0$. W przypadku parzystych wartości m wartość miary zróżnicowania można wyznaczyć z:

$$D_{Manh}(A_j, A_k(s)) = \begin{cases} \dfrac{5\times 2^{2m} + 2^{m+1} + 3\times 2^{m+1}\times s - 16 + 12\times s - 6\times s^2}{18}, & s \in \{0,6,12,...,2^m - 4\}; \\ \dfrac{5\times 2^{2m} + 2^{m+1} + 3\times 2^{m+1}\times s + 2 + 12\times s - 6\times s^2}{18}, & s \in \{3,9,15,...,2^m - 1\}; \\ \dfrac{5\times 2^{2m} - 5\times 2^{m+1} + 3\times 2^{m+1}\times s - 16 + 12\times s - 6\times s^2}{18}, & s \in \{2,4,8,...,2^m - 2\}; \\ \dfrac{5\times 2^{2m} - 5\times 2^{m+1} + 3\times 2^{m+1}\times s + 2 + 12\times s - 6\times s^2}{18}, & s \in \{1,5,7,...,2^m - 3\}. \end{cases} \quad (4.12)$$

Optymalne adresy początkowe s są wartościami, które maksymalizują wartość miary zróżnicowania sekwencji adresowych $D_{Manh}(A_j,A_k(s))$ i mogą być otrzymane na podstawie wyrażenia (4.13), które jest identyczne dla wszystkich czterech powyższych funkcji.

$$\frac{dD_{Manh}(A_j, A_k(s))}{ds} = \frac{3\times 2^{m+1} + 12 - 12\times s}{18} = 0, \qquad s \in \{0, 1, 2, ..., 2^m - 1\}. \quad (4.13)$$

Z wyrażenia (4.13) otrzymujemy, iż w zależności od wartości s dla wszystkich funkcji (4.12) maksymalna wartość zostanie otrzymana dla wartości s najbliższej do $s = 2^{m-1}+1$. Dlatego maksymalną wartość dla pierwszej funkcji otrzymamy na podstawie $s = 2^{m-1}-2$ ponieważ w zbiorze wartość $s \in \{0, 6, 12, \ldots, 2^m - 4\}$ (4.12) jest to najbliższa wartość do $s = 2^{m-1}+1$. Odpowiednio w przypadku drugiej funkcji optymalny adres startowy będzie równy $s = 2^{m-1} + 1$, w przypadku trzeciej funkcji $s = 2^{m-1}$ i w przypadku czwartej funkcji $s = 2^{m-1} - 1$. Po podstawieniu optymalnych wartości s do (4.12) łatwo jest pokazać, iż maksymalną wartość miary zróżnicowania sekwencji adresowych $D_{Manh}(A_j,A_k(s))$ otrzymujemy dla adresu startowego $s = 2^{m-1} + 1$. Przykładowo, dla $m = 4$ optymalna wartość $D_{Manh}(A_j, A_k(9))$ jest równa 100 dla adresu początkowego $s = 24-1 + 1 = 9$, co jest zgodne z wynikami, które przedstawia tabela 4.8.

W przypadku nieparzystej wartości m otrzymujemy:

$$D_{Manh}(A_j, A_k(s)) = \begin{cases} \dfrac{5\times 2^{2m} - 2^{m+1} + 3\times 2^{m+1}\times s - 16 + 12\times s - 6\times s^2}{18}, s \in \{0,4,6,10,...,2^m - 2\}; \\ \dfrac{5\times 2^{2m} - 2^{m+1} + 3\times 2^{m+1}\times s + 2 + 12\times s - 6\times s^2}{18}, s \in \{1,3,7,9,...,2^m - 1\}; \\ \dfrac{5\times 2^{2m} - 7\times 2^{m+1} + 3\times 2^{m+1}\times s - 16 + 12\times s - 6\times s^2}{18}, s \in \{2,8,14,...,2^m - 6\}; \\ \dfrac{5\times 2^{2m} - 7\times 2^{m+1} + 3\times 2^{m+1}\times s + 2 + 12\times s - 6\times s^2}{18}, s \in \{5,11,17,...,2^m - 3\}; \end{cases} \quad (4.14)$$

Tak jak w poprzednim przypadku, optymalną wartością adresu początkowego s dla wszystkich czterech funkcji (4.14) jest adres najbliższy $s = 2^{m-1} + 1$. W przypadku pierwszej funkcji są to adresy $s = 2^{m-1}+2$ i $s = 2^{m-1}$, ponieważ są one najbliżej wartości $s = 2^{m-1} + 1$ pośród 0, 4, 6, …, $2^m - 2$. W przypadku drugiej funkcji maksymalną wartość otrzymujemy dla $s = 2^{m-1} - 1$ i $s = 2^{m-1} + 3$, w przypadku trzeciej dla – $s = 2^{m-1} - 2$ i $s = 2^{m-1} + 4$ i w przypadku czwartej dla $s = 2^{m-1} + 1$. Po podstawieniu optymalnych wartości s do (4.14) jest łatwo pokazać, iż maksymalną wartość zróżnicowania sekwencji adresowych otrzymamy dla adresów startowych $s = 2^{m-1} + 2$, $s = 2^{m-1} - 1$, $s = 2^{m-1}$ i $s = 2^{m-1} + 3$. Dla $m = 3$ $D_{Manh}(A_j,A_k(6)) = D_{Manh}(A_j,A_k(3)) = D_{Manh}(A_j,A_k(4)) = D_{Manh}(A_j,A_k(7)) = 24$ (patrz tabela 4.7).

4.3. Wyniki eksperymentalne

W celu potwierdzenia otrzymanych wyników teoretycznych przeprowadzono odpowiednie badania symulacyjne. Jako pierwsze zostały wykonane badania odnośnie wskaźnika inkrementacji $q = 2$. Symulacje przeprowadzono dla pamięci o rozmiarze $N = 16$ bitów w celu porównania z danymi przedstawionymi w tabeli 4.6. Doświadczenia wykonano dla testów złożonych z dwóch iteracji opartych odpowiednio o test MATS++ i March C–. W obu przypadkach przeprowadzono symulacje dla wszystkich możliwych adresów startowych sekwencji adresowej drugiej iteracji testu. W ramach przeprowadzonych eksperymentów wyznaczono średnie pokrycie uszkodzeń PSF*k*. Otrzymane wartości odpowiednio dla $k = 3$ i $k = 4$ zawierają odpowiednio tabela 4.9 i tabela 4.10.

TABELA 4.9. **Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu MATS++ i indeksu inkrementacji $q = 2$**

s	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
FC_{MATS++}(PSF3)	34,87	37,37	38,37	40,62	41,50	42,83	43,37	44,03
FC_{MATS++}(PSF5)	10,11	10,84	11,05	11,63	11,72	11,93	11,99	12,15
s	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
FC_{MATS++}(PSF3)	44,20	44,18	44,00	43,36	42,77	41,52	40,58	38,66
FC_{MATS++}(PSF5)	12,13	12,14	12,05	12,03	11,95	11,75	11,53	11,12

TABELA 4.10. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu March C– i indeksu inkrementacji $q = 2$

s	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
$FC_{MarchC-}$(PSF3)	66,46	69,77	69,29	71,94	71,57	73,24	73,02	73,98
$FC_{MarchC-}$(PSF5)	20,00	21,51	21,17	22,41	22,14	22,91	22,71	23,15
s	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
$FC_{MarchC-}$(PSF3)	73,80	73,83	73,82	73,12	73,17	71,55	71,81	69,40
$FC_{MarchC-}$(PSF5)	22,99	23,05	23,01	22,70	22,83	22,09	22,27	21,14

Na podstawie otrzymanych wyników można stwierdzić, iż miara zróżnicowania sekwencji adresowych $D_{Manh}(A_j,A_k(s))$ pozwala określić optymalne sekwencje adresowe dla dwuprzebiegowego testu krokowego. W przypadku indeksu inkrementacji $q = 2$ w pierwszej iteracji jako sekwencję adresową wykorzystujemy standardową sekwencję licznikową zaś w drugiej iteracji sekwencję adresową rozpoczynającą się od adresu startowego $s \in \{2^{m-1}-1, 2^{m-1}, 2^{m-1}+1, 2^{m-1}+2\}$.

Celem potwierdzenia rozważań analitycznych dla indeksu inkrementacji $q = 3$, przeprowadzono te same badania co w przypadku $q = 2$, przy czym sekwencję adresową dla drugiej iteracji testu generowano zgodnie z Definicją 2.2. Otrzymane wartości odpowiednio dla $k = 3$ i $k = 4$ zawiera odpowiednio tabela 4.11 i tabela 4.12.

TABELA 4.11. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu March C– i indeksu inkrementacji $q = 3$

s	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
FC_{MATS++}(PPSF3)	39,61	38,49	39,59	42,93	41,52	42,15	44,93	43,20
FC_{MATS++}(PPSF5)	11,21	10,98	11,18	11,90	11,68	11,71	12,17	11,88
S	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
FC_{MATS++}(PPSF3)	43,32	45,64	43,37	43,13	44,92	42,20	41,47	42,93
FC_{MATS++}(PPSF5)	11,94	12,25	11,86	11,91	12,24	11,67	11,63	11,92

Porównanie wyników otrzymanych dla indeksów inkrementacji $q = 2$ i $q = 3$ nie wskazują jednoznacznie, czy zwiększenie indeksu inkrementacji wpływa na zwiększenie pokrycia uszkodzeń. Porównując wyniki z tabel 4.9 i 4.10 oraz 4.11 i 4.12 można zauważyć, iż w niektórych wypadkach pokrycie uszkodzeń nieznacznie wzrosło zaś w innych wręcz spadło. Dlatego przeprowadzono dalsze badania symulacyjne dla indeksów inkrementacji $q = 4$ i $q = 5$. Otrzymane wyniki zawierają tabele 4.13–4.16.

TABELA 4.12. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu March C– i indeksu inkrementacji $q = 3$

s	s = 0	s = 1	s = 2	s = 3	s = 4	s = 5	s = 6	s = 7
$FC_{MarchC-}$(PSF3)	71,49	71,20	71,43	72,33	72,48	72,58	72,80	73,13
$FC_{MarchC-}$(PSF5)	21,96	21,80	21,99	22,33	22,35	22,44	22,58	22,61
s	s = 8	s = 9	s = 10	s = 11	s = 12	s = 13	s = 14	s = 15
$FC_{MarchC-}$(PSF3)	73,16	73,04	73,11	73,09	72,84	72,62	72,40	72,33
$FC_{MarchC-}$(PSF5)	22,66	22,65	22,63	22,70	22,57	22,33	22,37	22,35

TABELA 4.13. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu MATS++ i indeksu inkrementacji $q = 4$

s	*s* = 0	*s* = 1	*s* = 2	*s* = 3	*s* = 4	*s* = 5	*s* = 6	*s* = 7
FC_{MATS++}(PSF3)	37,32	40,51	41,34	40,16	40,76	43,44	43,70	42,24
FC_{MATS++}(PSF5)	10,71	11,42	11,49	11,51	11,50	12,03	11,98	11,84
S	*s* = 8	*s* = 9	*s* = 10	*s* = 11	*s* = 12	*s* = 13	*s* = 14	*s* = 15
FC_{MATS++}(PSF3)	42,77	44,67	44,57	42,75	42,56	43,96	43,48	41,10
FC_{MATS++}(PSF5)	11,84	12,16	12,13	11,87	11,82	12,00	12,06	11,50

TABELA 4.14. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu March C– i indeksu inkrementacji $q = 4$

S	*s* = 0	*s* = 1	*s* = 2	*s* = 3	*s* = 4	*s* = 5	*s* = 6	*s* = 7
$FC_{MarchC-}$(PSF3)	69,42	72,01	73,07	72,03	71,28	73,35	73,59	72,73
$FC_{MarchC-}$(PSF5)	21,02	22,25	22,50	22,51	21,86	22,61	22,79	22,46
S	*s* = 8	*s* = 9	*s* = 10	*s* = 11	*s* = 12	*s* = 13	*s* = 14	*s* = 15
$FC_{MarchC-}$(PSF3)	72,55	73,80	73,79	72,52	72,71	73,62	73,23	71,42
$FC_{MarchC-}$(PSF5)	22,38	22,97	22,89	22,35	22,44	22,90	22,74	21,88

TABELA 4.15. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu MATS++– i indeksu inkrementacji $q = 5$

s	$s = 0$	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$
FC_{MATS++}(PSF3)	42,90	41,09	40,46	41,10	42,83	45,82	43,57	42,83
FC_{MATS++}(PSF5)	11,69	11,64	11,49	11,51	11,72	12,31	11,99	11,89
s	$s = 8$	$s = 9$	$s = 10$	$s = 11$	$s = 12$	$s = 13$	$s = 14$	$s = 15$
FC_{MATS++}(PSF3)	43,27	44,57	47,00	44,62	43,27	43,13	43,97	45,86
FC_{MATS++}(PSF5)	11,84	11,98	12,37	11,94	11,93	11,88	11,96	12,20

TABELA 4.16. Pokrycie uszkodzeń PSF*k* dla dwóch iteracji testu March C– i indeksu inkrementacji $q = 5$

S	s = 0	s = 1	s = 2	s = 3	s = 4	s = 5	s = 6	s = 7
$FC_{MarchC-}$(PSF3)	72,43	72,99	72,98	72,81	72,37	71,48	72,56	73,22
$FC_{MarchC-}$(PSF5)	22,37	22,74	22,79	22,66	22,41	22,09	22,50	22,86
s	s = 8	s = 9	s = 10	s = 11	s = 12	s = 13	s = 14	s = 15
$FC_{MarchC-}$(PSF3)	73,19	72,56	71,34	72,38	73,10	73,14	72,51	71,63
$FC_{MarchC-}$(PSF5)	22,65	22,51	21,92	22,43	22,85	22,79	22,48	21,89

Otrzymane wyniki jednoznacznie wskazują, iż zwiększanie indeksu inkrementacji nie wpływa na poziom pokrycia uszkodzeń PSF*k*. Na rysunku 4.1 przedstawiono maksymalne wartości pokrycia uszkodzeń otrzymane w dwóch iteracjach testu March C– dla różnych q. Z tego rysunku wynika, iż największe pokrycie uszkodzeń udało się uzyskać w przypadku $q = 2$. Jednocześnie należy podkreślić, iż we wszystkich wypadkach otrzymane wyniki są wyższe niż w przypadku wykorzystania sekwencji pseudolosowych (porównaj rysunek 4.1, tabela 4.2).

Analiza danych z tabel 4.2, 4.9 i 4.10 pozwala stwierdzić, iż w przypadku uszkodzenia PSF3 prezentowane podejście pozwala uzyskać pokrycie uszkodzeń na poziomie 44,20% i 73,80% odpowiednio przez testy MATS++ i March C–. Jednocześnie należy zauważyć, iż wynik pokrycia powyższych uszkodzeń przy adresacji pseudolosowej to odpowiednio 42,74% i 72,74%. Ważne jest aby zaznaczyć, iż w obu przypadkach złożoność testu jest identyczna.

RYSUNEK 4.1. Maksymalne pokrycie uszkodzeń dwuprzebiegowego testu March C– przy różnych wartościach q

Podsumowując otrzymane wyniki można stwierdzić, iż indeks inkrementacji $q = 2$ wydaje się być najbardziej optymalnym dla omawianej techniki testowania opartej na dwuprzebiegowych testach krokowych. Wskazuje na to wysokie pokrycie uszkodzeń zweryfikowane badaniami symulacyjnymi. Kolejnym argumentem przemawiającym na korzyść *indeksu inkrementacji* $q = 2$, jest możliwość stosunkowo łatwej implementacji sprzętowej takiego generatora adresów. Standardowo przy $q > 1$ implementacja sprzętowa generatora adresów może składać się z binarnego sumatora o rozmiarze $r = log_s q$ bitów oraz $(m - r)$ bitowego standardowego licznika $q = 1$. Wraz ze wzrostem q, z uwagi na większą złożoność sumatora, rośnie również narzut sprzętowy związany z implementacją generatora adresów. W przypadku $q = 2$ implementację całego generatora adresów można oprzeć wyłącznie o standardowy licznik. Przesunięcie najstarszego bitu (MSB) w standardowej sekwencji adresowej ($q = 1$) na pozycję najmłodszego bitu (LSB) pozwala osiągnąć sekwencję adresową zgodną z $q = 2$. Mechanizm ten ilustruje tabela 4.17.

TABELA 4.17. Generowanie sekwencji adresowej dla $q = 2$ na podstawie $q = 1$

Sekwencje adresowe					
q = **1**			q = **2**		
Q_2	Q_1	Q_0	Q_1	Q_0	Q_2
0	0	0	0	0	**0**
0	0	1	0	1	**0**
0	1	0	1	1	**0**
0	1	1	1	1	**0**
1	0	0	0	0	**1**
1	0	1	0	1	**1**
1	1	0	1	0	**1**
1	1	1	1	1	**1**

W takiej sytuacji implementacja sprzętowa generatora adresów testu dwuprzebiegowego ograniczy się do standardowego licznika generującego sekwencję adresową dla $q = 1$ oraz dodatkowo na każdej linii adresowej jednego dwuwejściowego multipleksera, co jest znacznie mniejszym narzutem sprzętowym niż przy rozwiązaniu opartym na sumatorze.

Podsumowanie

W pracy przeanalizowano dwuprzebiegowe testy krokowe pamięci RAM z inkrementacją sekwencji adresowej. Jako docelowy model uszkodzenia, na podstawie którego dokonano oceny proponowanej techniki testowania, wybrano PSF*k*. Podczas pierwszej iteracji testu jako bazową sekwencję adresową wykorzystano standardową sekwencję licznikową, a w przypadku drugiej iteracji były analizowane zróżnicowane sekwencje adresowe otrzymywane na podstawie inkrementacji o indeks q sekwencji bazowej oraz przez ustalenie optymalnego adresu startowego. Największe pokrycie uszkodzeń uzyskano w przypadku indeksu inkrementacji $q = 2$. W tym przypadku pokrycie uszkodzeń jest większe lub porównywalne niż dla innych wartości q. Dodatkową zaletą przemawiającą za indeksem $q = 2$ jest stosunkowo mały narzut związany z implementacją sprzętową generatora adresów. Jak zostało również dowiedzione analitycznie i sprawdzone eksperymentalnie, ważnym czynnikiem wpływającym na sumaryczne pokrycie uszkodzeń testu dwuprzebiegowego jest adres początkowy sekwencji adresowej drugiej iteracji testu. W analizowanych przykładach, różnica w pokryciu uszkodzeń przy optymalnym i nieoptymalnym doborze adresu startowego wynosiła ponad 26%.

Bibliografia

[1] Adams R. D., *High performance memory testing: design principles, fault modeling and self-test*. luwer Academic Publishers, USA 2003

[2] Di Carlo S., and Prinetto P., *Models in Memory Testing*. Springer Netherlands, Dordrecht, 2010, s. 157–185

[3] Hamdioui S., Taouil M., and Haron N. Z., Testing open defects in memristor-based memories. IEEE Transactions on Computers 64, 1 (2015), 247–259

[4] Hayrapetyan D., Manukvan A., and Tshagharyan G., *Implementation of memory static, coupling and dynamic fault models at the register transfer level*. [W:] 2018 IEEE East-West Design Test Symposium (EWDTS) (9 2018), 1–4

[5] Sfikas Y., and Tsiatouhas Y., *Testing neighbouring cell leakage and transition induced faults in DRAMs*. "Transactions on Computers" 2016, 65, 7, 2339–2345

[6] Thatte S., and Abraham J., *Testing of semiconductor random access memories*. Proceedings of the Annual International Conference on Fault-Tolerant Computing 1977, 81–87

[7] Tshagharyan G., Harutyunyan G., Zorian Y., Gebregiorgis A., Golanbari M. S., Bishnoi R., and Tahoori M. B., *Modeling and testing of aging faults in finfet memories for automotive applications*. [W:] 2018 IEEE International Test Conference (ITC) 2018, 10, 1–10

[8] Van de Goor A. J., *Testing Semiconductor Memories: Theory and Practice*. John Wiley & Sons, Chichester, England 1991

[9] Yarmolik V. N., Klimets Y., and Demidenko S., *March PS(23N) test for DRAM pattern--sensitive faults*. [W:] Proceedings of the 7th Asian Test Symposium 1998, ATS'98, IEEE Computer Society, 354–357

[10] Hayes J. P., *Detection of Pattern-Sensitive Faults in Random-Access Memories*. "IEEE Transactions on Computers" 1975, 24.2,150–157

[11] Cockburn B. F., *Deterministic tests for detecting scrambled pattern-sensitive faults in RAMs*. Proceedings of the IEEE International Workshop on Memory Technology, Design and Testing. MTDT'95. Washington, DC, USA: IEEE Computer Society 1995, 117–122

[12] Nicolaidis M., *Theory of Transparent BIST for RAMs*. "IEEE Transaction of Computers" 1996, 45, 10

[13] Yarmolik S.V., *Address Sequences and Backgrounds with different Hamming Distance for Multiple run March tests*. "International Journal of Applied Mathematics and Computer Science" 2008, 1 8, 329–339

[14] B. Cockburn F. i Sat Y. F. N., *A transparent built-in self-test scheme for detecting single V-coupling faults in RAMs*. Proceedings of the IEEE International Workshop on Memory Technology, Design, and Test 1994, 119–124. doi: 10.1109/MTDT.1994.397187

[15] Yarmolik V. N. and Yarmolik S.V., *Address sequences for multiple run march tests*. "Autom. Control Comput. Sci." 2006, 5, 59–68

[16] Yarmolik S. V. and Yarmolik V. N., *Memory address generation for multiple run march tests with different average Hamming distance*. Proc. IEEE East-West Design and Test Workshop, Sochi, Russia 2006, 212–216

[17] Yarmolik V.N., Klimets Y. i Demidenko S., *March PS(23N) Test for DRAM Pattern-Sensitive Faults Proceedings of the 7th Asian Test Symposium. ATS'98*. "IEEE Computer Society" 1998, 354–357

[18] Van de Goor A. J. i Schanstra I., *Address and data scrambling: causes and impact on memory tests*, Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications. 2002, 128–136, doi: 10.1109/DELTA.2002.994601

[19] Van de Goor A. J. i in., *March LR: a test for realistic linked faults,* Proceedings of the 14th VLSI Test Symposium. Kw. 1996, 272–280. doi: 10.1109/VTEST.1996.510868

[20] Koeneman B., *oral presentation*, Design For Testability Workshop 1986

[21] Nicolaidis M., *Theory of Transparent BIST for RAM*, IEEE Transactions on Computers 1996, 45.10, 1141–1156. ISSN: 0018-9340

[22] Nicolaidis M., *Transparent BIST for RAMs*. Proceedings of the IEEE International Test Conference, Discover the New World of Test and Design. ITS'92. Baltimore, Maryland, USA: IEEE Computer Society 1992, 598–607

[23] Hamdioui S., *Testing static random access memories: defects, fault models and test patterns*, Frontiers in Electronic Testing, Kluwer Academic 2004

[24] Niggemeyer D., Redeker M. i Otterstedt J., *Integration of Non-classical Faults in Standard March Tests*. Proceedings of the 1998 IEEE International Workshop on Memory Technology, Design and Testing, 91, IEEE Computer Society, San Jose 1998

[25] Cheng K., Tsai M. i Wu C., *Effcient neighborhood pattern-sensitive fault test algorithms for semiconductor memories*, Proc. 19th IEEE VLSI Test Symp. (VTS '01), IEEE Computer Society, Washington, DC, 2001, 225–230

[26] Hellebrand S., Wunderlich H.-J., Ivaniuk A. A., Klimets Y. V. i Yarmolik V. N., *Efficient online and online testing of embedded DRAMs*, "IEEE Trans. Comput." 2002, 51, 801–809

[27] Yarmolik S., *Address sequences and backgrounds with different hamming distances for multiple run march tests* "Int. J. Appl. Math. Comput. Sci." 2008, 18, 329–339

[28] Hellebrand S., Wunderlich H.-J. i Yarmolik V. N., *Symmetric transparent BIST for RAMs*, Proc. Conf. Design, Automation and Test in Europe ACM, New York 1999, 702–707

[29] Yarmolik V. and Yarmolik S., Address sequences for multiple run march tests. "Autom. Control Comput. Sci." 2006, 5, 59–68

[30] Yarmolik S. V. and Yarmolik V. N., *Memory address generation for multiple run march tests with different average Hamming distance*, Proc. IEEE East-West Design and Test Workshop, Sochi, Russia 2006, 212–216

[31] Wunderlich H.-J., *Multiple distributions for biased random test patterns*, Proc. 1988 Int. Conf. Test: New Frontiers in Testing (ITC'88), IEEE Computer Society, Washington, DC, 1988, 236–244

[32] Sokol B. i Yarmolik S. V., *Address sequences for march tests to detect pattern sensitive faults*, Proc. Third IEEE Int. Workshop Electronic Design, Test and Applications (DELTA '06), IEEE Computer Society, Kuala Lumpur, 2006, 354–360

[33] Karpovsky M. G. i Yarmolik V. N., *Transparent memory testing for pattern-sensitive faults*, Proc. IEEE Int. Test Conf. TEST: The Next 25 Years, IEEE Computer Society, Washington, DC, 1994, 860–869

Two runs ram march testing with address decimation

Abstract: Conventional march memory tests have high fault coverage, especially for simple faults like stack-at (SAF) , transitions (TF) or coupling faults (CF). The same-time standard march tests, which are based on only one run, are becoming insufficient for complex faults like pattern-sensitive faults (PSF). To increase fault coverage, the multi-run transparent march test algorithms have been used. This solution is especially suitable for built-in self-test (BIST) implementation. The transparent BIST approach presents the incomparable advantage of preserving the content of the random access memory (RAM) after testing. We do not need to save the memory content before the test session or to restore it at the end of the session. Therefore, these techniques are widely used in critical applications (medical electronics, railway control, avionics, telecommunications, etc.) for periodic testing in the field. Unfortunately, in many cases, there is very limited time for such test sessions. Taking into account the above limitations, we focus on short, two--run march test procedures based on counter address sequences. The advantage of this paper is that it defines requirements that must be taken into account in the address sequence selection process and presents a deeply analytical investigation of the optimal address decimation parameter. From the experiments we can conclude that the fault coverage of the test sessions generated according to the described method are higher than in the case of pseudorandom address sequences. Moreover, the benefit of this solution seems to be low hardware overhead in implementation of an address generator.

Keywords: RAM, BIST, PSF faults, march tests, multiple tests

DOI: 10.24427/978-83-66391-96-3_5

Rozdział 5
Metodyka ASMD-FSMD projektowania układów cyfrowych na FPGA z wykorzystaniem języka Verilog

Valery Salauyou
Wydział Informatyki, Politechnika Białostocka

Streszczenie: W ostatnim czasie obserwuje się wzrost złożoności projektów układów cyfrowych przy jednoczesnym zaostrzeniu wymagań dotyczących czasu opracowywania i zwiększenia niezawodności projektów. Jednym z kierunków rozwiązania tego problemu jest wprowadzanie nowych metod projektowania układów cyfrowych.

W artykule omówiono metodykę projektowania układów cyfrowych oparta na *automatcie skończonym ze ścieżką przetwarzania danych* (finite state machine with datapath – FSMD), w którym działanie układu opisano w postaci diagramu blokowego automatu ze ścieżką przetwarzania danych (algorithmic state machine with datapath – ASMD). Metodyka ta jest nazywana *metodyką ASMD-FSMD*.

Porównanie różnych metod projektowania układów cyfrowych zostanie przedstawione na przykładzie projektowania mnożnika synchronicznego na bazie *programowalnych układów logicznych* (FPGA – field programmable gate array).

Wykazano, że metodyka ASMD-FSMD, w porównaniu z tradycyjną, pozwala na obniżenie kosztów realizacji z 28,6% do 39,7% oraz zwiększenie wydajności poszczególnych projektów do 17,6%. Ponadto, zastosowanie metodyki ASMD-FSMD może znacznie skrócić czas projektowania i zwiększyć niezawodność projektów. W podsumowaniu przedstawiono zalecenia dotyczącymi stosowania metodyki ASMD-FSMD.

Słowa kluczowe: układ wykonawczy, układ sterujący, automat skończony, automat skończony ze ścieżką przetwarzania danych, diagram blokowy automatu, diagram blokowy automatu ze ścieżką przetwarzania danych, projektowanie układów cyfrowych, programowalne układy logiczne, język Verilog.

Wprowadzenie

Urządzenie cyfrowe projektowane w sposób tradycyjny jest zwykle przedstawiane w postaci *układu wykonawczego* (datapath) i *układu sterującego* (controller lub control unit), które są zwykle projektowane oddzielnie: układ wykonawczy (operacyjny)

ma postać zbioru standardowych jednostek funkcjonalnych (rejestrów, magistrali, multiplekserów itp.), a układ sterujący ma postać *automatu skończonego* (finite state machine – FSM).

W [1] zaproponowano połączenie układów wykonawczych i sterujących oraz przedstawienie ich jako *automatu skończonego ze ścieżką przetwarzania danych* (FSMD). Model FSMD szybko stał się popularny i w [2] zastosowano FSMD do projektów układów synchronicznych i asynchronicznych. Model FSMD okazał się bardzo wygodny do sprawdzania równoważności dwóch układów uzyskanych w wyniku syntezy lub różnych przekształceń projektowych [3,4]. W [5] zaproponowano przedstawienie systemu cyfrowego w postaci sieci FSMD, co prowadzi do realizacji sprzętu pozbawionego zjawiska wyścigów.

Ogólny model FSMD nie jest zawsze wygodny podczas projektowania układów do pewnych określonych zastosowań. Dlatego w szeregu prac proponuje się rozszerzenia FSMD: w [6] – reprezentujące architekturę procesora i ASIC (application-specific integrated circuit); w [7] – dla układu synchronicznego dostępu do pamięci; w [8] – dla programów do przetwarzania tablic.

Dekompozycja FSMD mająca na celu zmniejszenie poboru mocy jest rozważana w [9], a dekompozycja z wykorzystaniem bramkowania w [10].

Porównanie skuteczności FSM i FSMD omówiono w [11]. Wykazano tam, że model FSMD automatu typu Moore'a jest bardziej efektywny pod względem kosztów realizacji w porównaniu ze strukturą kanoniczną FSM.

Ze względu na swoją przejrzystość, *diagramy blokowe maszyn stanów* (algorithmic state machine – ASM) są szeroko stosowane do reprezentacji algorytmu działania FSM. ASM zostały po raz pierwszy zaproponowane w [12] jako alternatywa dla grafów automatów skończonych. W [13] rozważono implementację ASM na PROM, FPLA i multiplekserach. W [14] zaproponowano metody minimalizacji liczby wierzchołków ASM. Praca [15] opisuje narzędzie ABELITE do syntezy sterowników opartych na ASM.

Tradycyjnie, ASM reprezentuje algorytm działania układu sterującego, tj. FSM. W [16] proponuje się wykorzystanie ASM zarówno do opisu działania układu sterującego, jak i do opisu operacji rejestrowych wykonywanych w układzie wykonawczym. W tym celu, do reprezentacji stanów układu sterującego służy ASM automatu Moore'a, na którego przejściach wstawiane są owale (elementy ASM automatu Mealy'ego). W owalach zapisywane są operacje wykonywane w układzie wykonawczym w danym przejściu. Taki ASM jest nazywany *diagram blokowy automatu ze ścieżką przetwarzania danych* (algorithmic state machine with datapath – ASMD).

Diagramy ASMD są ostatnio coraz częściej wykorzystywane w projektach FPGA: przy realizacji przemysłowych systemów sterowania [17]; do implementacji funkcji asin za pomocą algorytmu CORDIC [18]; do sprzętowej implementacji algorytmu kryptograficznego AES [19]; przy projektowaniu uniwersalnego asynchronicznego odbiornika-nadajnika UART [20] i innych.

W tym artykule zaproponowano metodykę zwaną ASMD-FSMD do projektowania układów cyfrowych na układach FPGA z wykorzystaniem języka Verilog. Opisowi metodyki zostanie zilustrowany przykładem projektowania mnożnika synchronicznego.

Artykuł jest zorganizowany w następujący sposób. W rozdziale 1 opisano zaimplementowany algorytm mnożenia. W rozdziale 2 omawiono tradycyjne podejście do projektowania mnożnika w postaci układów wykonawczego i sterującego. W rozdziale 3 opisano ASM. W rozdziale 4 zdefiniowano ASMD i FSMD. W rozdziale 5 omawiono możliwość konwersji ASMD w celu poprawy wydajności projektu. W rozdziale 6 przedstawiono opis FSMD w języku Verilog. W rozdziale 7 zawierto formalną reprezentację metodyki ASMD-FSMD jako algorytmu. W rozdziale 8 opisano badania eksperymentalne porównujące podejście tradycyjne i metodyki ASMD-FSMD. Wyniki badań eksperymentalnych omówiono w rozdziale 9. Podsumowanie zawiera zalecenia dotyczące efektywnego stosowania metodyki ASMD-FSMD.

5.1. Algorytm mnożenia

Rozważmy najprostszy szkolny algorytm mnożenia, który wykonuje operację mnożenia arytmetycznego $P = A \cdot B$ dwóch liczb binarnych bez znaku, gdzie A i B są mnożnikami (czynnikami), a P jest iloczynem (produktem). Niech szerokość słów binarnych A i B będzie taka sama i równa N bitów, wtedy iloczyn P będzie miał szerokość $2N$ bitów. Początkowo P jest wyzerowane. W każdym cyklu mnożenia sprawdzana jest wartość najmniej znaczącego bitu mnożnika $B[0]$. Jeśli $B[0] = 1$, to mnożnik A jest dodawany do iloczynu P. Jeśli $B[0] = 0$, to do iloczynu P jest dodawane zero lub nic nie jest wykonywane. Następnie zawartość czynnika B zostaje przesunięta o jeden bit w prawo, a wartość iloczynu P – w lewo. Algorytm kończy się po uwzględnieniu wszystkich bitów czynnika B.

Rozważany algorytm mnożenia w języku Verilog można opisać w następujący sposób.

```
module algorithm_mult #(parameter N = 4)
        (input [N-1:0] a,b,                    // słowa wejściowe
        output [2*N-1:0] p);                   // słowo wyjściowe
        reg [N-1:0] rb;                        // zmienne pośrednie
        reg [2*N-1:0] ra,rp;
        integer i;                             // iterator
        always @(*)                            // blok proceduralny
        begin
                ra = a;   rb = b;   rp = 0;    // inicjalizacja zmiennych
                for (i = 0; i<N; i = i + 1)    // cykl mnożenia
                begin
```

```
                    // dodawanie wyników cząstkowych
                    if (rb[0])          rp = rp + ra;
                    else                rp = rp + {2*N{1'b0}};
                    rb = rb >> 1;               // przesunięcie zawartości zmiennych
                    ra = ra << 1;
                end
        end
        assign p = rp;                          // formowanie wyniku
endmodule
```

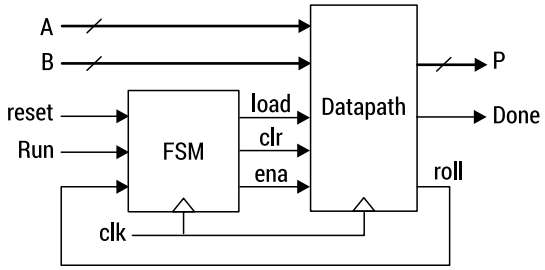
W rzeczywistości powyższy kod opisuje układ kombinacyjny, a nie mnożnik synchroniczny. Aby uzyskać mnożnik synchroniczny, na szynach wejściowych a i b, a także na szynie wyjściowej p, należy ustawić rejestry taktowane sygnałem zegarowym clk. Operacja mnożenia dla takiego układu zawsze będzie wykonywana w jednym cyklu zegara.

Projekt *algorytm_mult* ma dość wysoką wydajność. Jednak koszt realizacji projektu *algorytm_mult* gwałtownie rośnie wraz ze wzrostem liczby bitów *N* słów wejściowych *A* i *B*. Dlatego algorytmiczny opis układów cyfrowych jest rzadko stosowany do implementacji sprzętowej. W tym artykule jest rozważany mnożnik synchroniczny, w którym każdy cykl mnożenia jest wykonywany w jednym lub większej liczbie cykli zegara.

5.2. Tradycyjne podejście przy implementacji mnożnika synchronicznego

W przypadku tradycyjnego podejścia, sprzętową implementacją mnożnika synchronicznego jest obwód synchroniczny, do którego wejścia trafiają wartości mnożonych słów: mnożnika *A* i mnożnika *B*, a na wyjściu formuje się wartość iloczynu *P*. Obwód jest sterowany sygnałem zegarowym clk i sygnałem resetowania reset. Po ustawieniu wartości mnożonych słów na wejściach A i B rozpoczyna się proces mnożenia przez ustawienie sygnału Run na wartość „1". Koniec procesu mnożenia jest sygnalizowany przez wartość „1" na wyjściu Done, po czym wartość iloczynu można odczytać z wyjścia P.

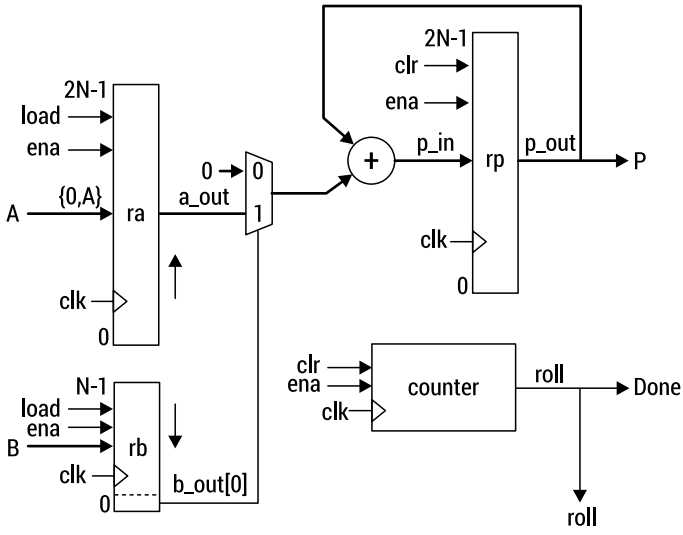
Na rysunku 5.1 przedstawiono schemat blokowy szeregowego mnożnika w postaci *układu wykonawczego* (datapath) i *układu sterującego* (controler), który jest realizowany w postaci automatu skończonego FSM. Wartości mnożonych słów A i B są podawane na wejście układu sterującego. Produkt P i sygnał Done są tworzone na wyjściach układu wykonawczego. Ponadto układ wykonawczy generuje sygnał roll, który jest wyjściem *licznika modulo* i sygnalizuje koniec procesu mnożenia.



RYSUNEK 5.1. Schemat blokowy mnożnika synchronicznego w postaci układu wykonawczego i sterującego

Wejścia FSM to zewnętrzne sygnały reset i Run oraz wewnętrzny sygnał roll. Automat skończony generuje następujące sygnały sterujące: clr – resetowanie rejestrów układu wykonawczego; load – ładowanie wartości mnożonych słów A i B do rejestrów układu wykonawczego; ena – w celu umożliwienia przesunięcia zawartości rejestrów przesuwających. Układ wykonawczy i układ sterujący są synchronizowane tym samym zegarem clk.

Schemat układu realizującego algorytm mnożenia przedstawiono na rysunku 5.2.

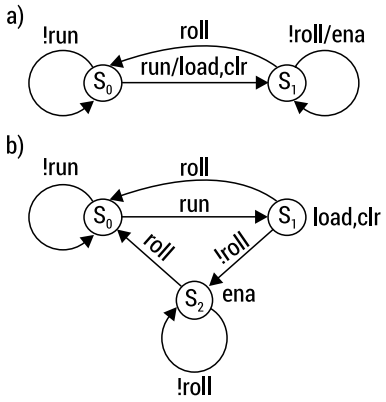


RYSUNEK 5.2. Schemat układu realizującego algorytm mnożenia

Układ wykonawczy zawiera 2*N*-bitowy rejestr przesuwający ra (w lewo) do przechowywania mnożnika *A*, *N*-bitowy rejestr przesuwający rb (w prawo) do przechowywania mnożnika *B*, 2*N*-bitowy multiplekser (2-1) magistrali, 2*N*-bitowy sumator i licznik modulo, który generuje sygnał roll, sygnalizujący koniec procesu mnożenia. Sygnał roll jest taki sam, jak zewnętrzny sygnał Done.

Należy zwrócić uwagę, że w układzie na rysunku 5.2, sygnał b_out[0], który steruje wyborem wejść multipleksera, jest generowany nie przez układ sterujący, ale jest wartością bitu zerowego rejestru rb. W takim przypadku układ wykonawczy działa jako urządzenie sterujące. Zasadniczo nie ma ścisłej granicy między funkcjami układu wykonawczego i układu sterującego. Niektóre funkcje układu sterującego mogą być realizowane przez układ wykonawczy (jak w pokazanym przykładzie) i odwrotnie.

Działanie synchronicznego układu sterującego mnożnikiem można przedstawić jako automat skończony. W tym przypadku można użyć zarówno automatu Mealy'ego (rysunek 5.3, a), jak i automatu typu Moore'a (rysunek 5.3, b).



RYSUNEK 5.3. **Reprezentacja synchronicznego układu sterującego mnożnikiem w postaci grafu automatu skończonego: a – typu Mealy'ego, b – typu Moore'a**

Można zauważyć, że automat Mealy'ego zawiera dwa stany: S_0 i S_1, podczas gdy automat typu Moore'a zawiera trzy stany: S_0, S_1 i S_2.

Aby zaimplementować układ mnożący na FPGA, konieczne jest przedstawienie wszystkich części układu w jednym z języków opisu sprzętu. W omawianym przypadku używany jest do tego język Verilog. Szczegóły opisu elementów składowych układu wykonawczego (rysunek 5.2) oraz automatów skończonych Mealy'ego i Moore'a (rysunek 5.3) do realizacji mnożnika synchronicznego w przypadku podejścia tradycyjnego, podano w [21].

5.3. Diagramy blokowe automatów ASM

Diagram blokowy automatu (algorithmic state machine – ASM) służy do wizualnego opisu algorytmu działania automatu skończonego i jest skierowanym grafem sprzężonym [12] zawierającym trzy typy wierzchołków:

- prostokąty – *wierzchołki stanów* (state box);
- romby – *wierzchołki warunkowe* (decision box);
- owale – *wierzchołki wyjść warunkowych* (conditional output box).

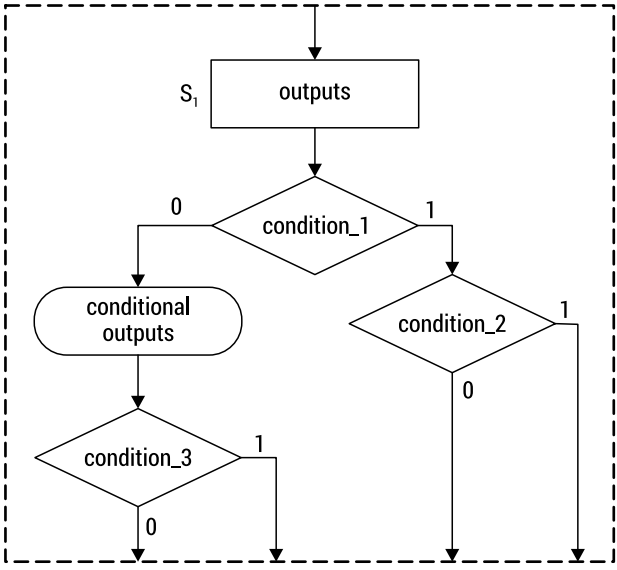
Wierzchołek stanu ASM (prostokąt) określa stan automatu. W górnej części wierzchołka można zapisać nazwę stanu (na przykład S0, START, INITIAL itp.), a także kod binarny stanu. W przypadku automatu Moore'a sygnały wyjściowe zapisywane są wewnątrz wierzchołka stanu, który w tym stanie przyjmuje wartość „1". Domyślnie przyjmuje się, że wszystkie inne wyjścia w tym stanie mają wartość równą zero. Należy zauważyć, że ASM nie pozwala na opisywanie automatów o skończonych stanach z wyjściami nieokreślonymi.

Warunki do sprawdzenia zapisane są w wierzchołkach warunkowych ASM (rombach), czyli wierzchołek warunkowy ASM jest punktem rozgałęzienia algorytmu. Wyjścia wierzchołka warunkowego są oznaczone wartościami 0 i 1, które odpowiadają przejściom w przypadku zerowej (fałszywej) lub jedynkowej (prawdziwej) wartości wyniku sprawdzenia warunku. Warunkiem może być zmienna wejściowa automatu skończonego, wyrażenie logiczne, pojedynczy bit wektora bitowego itp.

W wierzchołkach wyjść warunkowych (owali) zapisywane są sygnały wyjściowe automatu Mealy'ego, które przyjmują wartość jeden przy pewnym przejściu. Sygnały wyjściowe zapisane w owalach nazywane są *wyjściami warunkowymi* (conditional output).

W ASM dla automatów Moore'a nie ma wierzchołków wyjść warunkowych (owale), a w ASM dla automatów Mealy'ego nie jest nic zapisane w wierzchołkach stanów.

Głównym blokiem konstrukcyjnym diagramu blokowego ASM jest *blok ASM* (ASM block), którego przykład pokazano na rysunku 5.4.



RYSUNEK 5.4. **Blok ASM**

Blok ASM zawiera tylko jeden wierzchołek stanu (prostokąt) i może mieć kilka wierzchołków warunkowych (rombów) i wierzchołków wyjść warunkowych (owale), a romby mogą poprzedzać owale lub następować po nich. Wejścia i wyjścia wierzchołków są połączone za pomocą krawędzi. Blok ASM ma tylko jedno wejście, które jest wejściem do początku stanu i może mieć jedno lub więcej wyjść. Blok ASM opisuje zachowanie automatu w jednym stanie podczas jednego cyklu synchronizacji.

Diagram blokowy ASM (zwany dalej po prostu *diagramem ASM*) jest kompozycją połączonych ze sobą bloków ASM. W takim przypadku konieczne jest przestrzeganie następujących zasad konstruowania ASM: każde wyjście dowolnego wierzchołka ASM można podłączyć tylko do jednego wejścia innego wierzchołka, tj. rozgałęzienie algorytmu jest możliwe tylko w wierzchołkach warunkowych; sprzężenia zwrotne są zabronione wewnątrz bloku ASM.

Należy zwrócić uwagę na niektóre charakterystyczne cechy ASM. Za pomocą bloków ASM w ASM jawnie wyznaczają się wewnętrzne stany automatu skończonego. Przejście z jednego bloku ASM do drugiego odbywa się zawsze w jednym cyklu zegarowym, tj. algorytm działania automatu jest bezpośrednio powiązany z czasem automatu. Dzięki temu, zawsze można prześledzić, ile cykli zegara zajmie wykonanie danego fragmentu algorytmu. Jednocześnie ASM zachowuje przejrzystość diagramów blokowych. Ponadto ASM wstępnie określa typ automatu: Mealy'ego lub Moore'a.

Przedstawmy sposób konstruowania ASM automatu skończonego w postaci następującego algorytmu.

Algorytm 1. Metodologia konstruowania ASM.

1. Określa się stany automatu skończonego.
2. Dla każdego stanu budowany jest blok ASM. W tym przypadku wszystkie przejścia od tego stanu są określane dla wszystkich możliwych wartości zmiennych wejściowych. Jeśli jakaś zmienna wejściowa nie wpływa na przejścia z danego stanu, to nie występuje wewnątrz bloku ASM.
3. W przypadku automatu Moore'a zmienne wyjściowe są zapisywane w wierzchołkach stanów (prostokątach), które w tym stanie przyjmują wartość jeden.
4. W przypadku automatu Mealy'ego zmienne wyjściowe są zapisywane w wierzchołkach wyjść warunkowych (owale), które przyjmują wartości „1" w danym przejściu.
5. Bloki ASM są ze sobą połączone zgodnie z algorytmem pracy układu. W takim przypadku każde wyjście bloku ASM można podłączyć tylko do jednego wejścia tego samego lub innego bloku ASM.
6. Koniec.

Można zauważyć, że konstrukcja ASM zaczyna się od określenia stanów automatu skończonego, tj. od samego początku projektant jednoznacznie określa stany automatu. Następnie określane są przejścia z każdego stanu, tj. definiuje się przejścia między stanami. Następnie określane są wartości zmiennych wyjściowych: osobno dla automatu Mealy'ego i osobno dla automatu Moore'a. Ostatni etap, czyli połączenie ze sobą bloków ASM, ma raczej charakter formalny, co pozwala jeszcze raz sprawdzić poprawność konstrukcji ASM.

5.4. Diagramy blokowe ASMD i automaty skończone ze ścieżką przetwarzania danych FSMD

Diagram blokowy ASM ze ścieżką przetwarzania danych (ASM with datapath – ASMD) to ASM, w którym, w prostokątach i owalach, można zapisywać dowolne operacje na rejestrach, które są dozwolone w języku Verilog, a wszelkie wyrażenia logiczne języka Verilog można sprawdzać w wierzchołkach warunkowych. Diagram ASMD, podobnie jak ASM, składa się z bloków. Działania opisane w bloku ASMD wykonywane są w jednym cyklu zegarowym.

Automat skończony realizowany zgodnie z ASMD będzie nazywany *automatem skończonym ze ścieżką przetwarzania danych* (FSM with datapath – FSMD), a metodyka projektowania FSMD według ASMD nazywana jest *metodyką ASMD-FSMD*.

Należy zauważyć, że w naszym przypadku, w przeciwieństwie do [1], FSMD nie jest podzielony na *układ sterujący* (FSM) i *układ wykonawczy* (datapath), ale raczej przypomina behawioralny (algorytmiczny) opis działania urządzenia. Jednak w przeciwieństwie do opisu algorytmicznego (projekt algorithm_mult), ASMD jawnie definiuje stany FSMD.

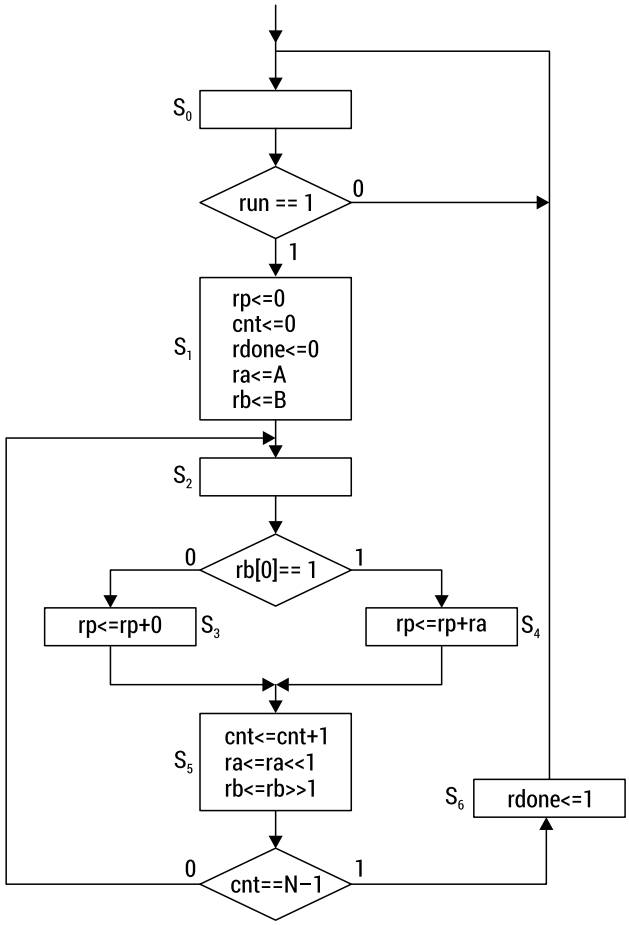
Przedstawione tu podejście do konstruowania ASMD różni się również od podejścia, które jest zdefiniowane w [16]: pokazany tu ASMD może opisywać dowolny typ automatu (Mealy'ego lub Moore'a), a operacje wykonywane w układzie wykonawczym mogą być zapisywane zarówno w owalach, jak i w prostokątach ASMD.

Na rysunku 5.5 pokazano ASMD, który odpowiada FSMD Moore'a w celu zaimplementowania rozważanego algorytmu mnożenia.

Stan S_0 jest stanem oczekiwania na sygnał uruchomienia run, który rozpoczyna proces mnożenia. W stanie S_1, rejestr iloczynu rp, licznik cnt i przerzutnik przechowujący wartość sygnału Done są zerowane. Dodatkowo w stanie S_1 ładowane są początkowe wartości rejestrów ra i rb.

Następny stan S_2 jest punktem wejścia w cykl procesu mnożenia. Jeden cykl mnożenia odpowiada utworzeniu iloczynu cząstkowego, dodaniu go do wyniku i wykonaniu niezbędnych przesunięć zawartości rejestrów, a także zwiększeniu wartości licznika. W stanie S_2, w zależności od wartości rb[0], konieczne jest dodanie albo wartości zerowej, albo wartości rejestru ra do rejestru rp. Ponieważ jednak ASMD opisuje automat typu Moore'a, czynności te mogą być wykonywane tylko w oddzielnych stanach S_3 i S_4.

Po stanie S_3 lub S_4 następuje stan S_5, w którym licznik cnt jest zwiększany, a rejestry ra i rb są przesuwane. Dodatkowo w stanie S_5 sprawdzana jest wartość licznika cnt. Jeśli wartość cnt jest równa *N*-1, następuje przejście do stanu S_6, w przeciwnym razie do stanu S_2 w celu wykonania nowej iteracji procesu mnożenia. W stanie S_6, przerzutnik rdone jest ustawiany na jeden i następuje przejście do stanu początkowego S_0.



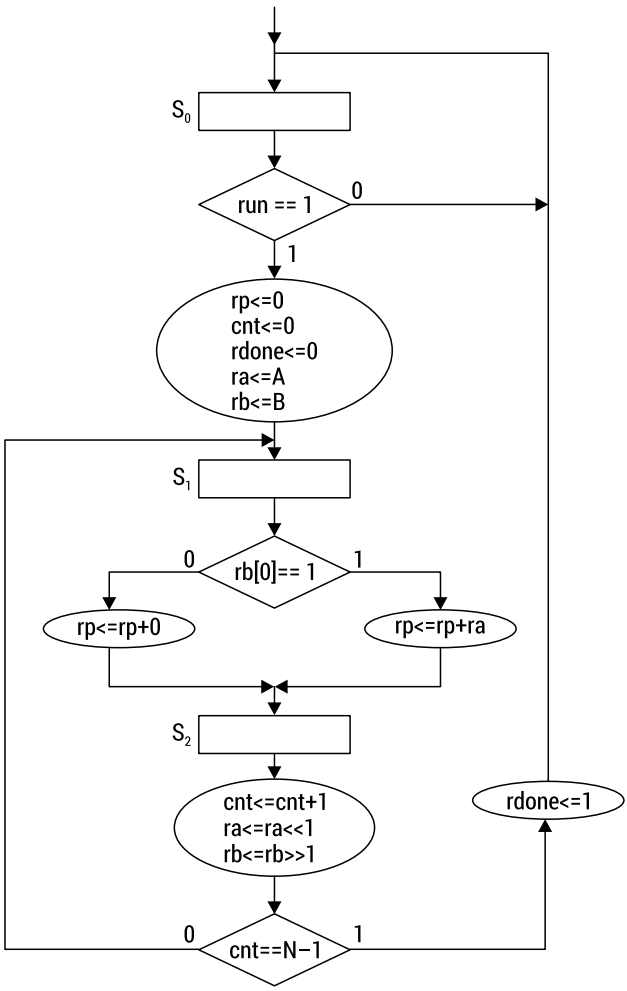
RYSUNEK 5.5. **Diagram ASMD opisujący algorytm mnożenia a w postaci FSMD Moore'a**

Bezpośrednio z ASMD pokazanego na rysunku 5.5, można określić liczbę taktów zegara n_{Moore} wymaganych do pomnożenia *N*-bitowych liczb binarnych. Stan początkowy S_0 dla automatów typu Moore'a zawsze wymaga jednego taktu zegara. W stanie S_1 rejestry są inicjalizowane i ładowane, co wymaga jeszcze jednego cyklu taktowania. Wykonanie jednej iteracji mnożenia (pętli) związanej z przetwarzaniem jednego bitu mnożnika B wymaga trzech cykli zegarowych (stany S_2, S_3 lub S_4 i S_5). Utworzenie wyjścia rdone wymaga również jednego cyklu zegara (stan S_6). Zatem FSMD dla automatu Moore'a wykonuje mnożenie *N*-bitowych liczb binarnych w n_{Moore} cyklach zegara, gdzie:

$$n_{Moore} = 3N + 3 \tag{5.1}$$

Poprawność zależności (1) potwierdzają również wyniki symulacji czasowej.

Na rysunku 5.6 przedstawiono ASMD, który odpowiada FSMD typu Mealy'ego służącego do implementacji rozważanego algorytmu mnożenia.



RYSUNEK 5.6. **Diagram ASMD służący do realizacji algorytmu mnożenia a przy użyciu automatu FSMD typu Mealy'ego**

W stanie początkowym S_0 automat oczekuje na nadejście wartości sygnału run równej „1". Po ustawieniu sygnału run na „1", wszystkie rejestry mnożnika są inicjalizowane zgodnie z wymaganiami.

Następny stan S_1 jest punktem wejścia w cykl procesu mnożenia. W stanie S_1 sprawdzana jest wartość bitu rb[0] i zależnie od jego wartości, do zawartości rejestru rp dodawana jest albo wartość zerowa, albo zawartość rejestru ra. Można zauważyć, że dwa oddzielne stany nie są tutaj wymagane, jak w FSMD Moore'a (S_3 i S_4 na rysunku 5.5).

W stanie S_2 wykonywane są niezbędne przesunięcia rejestrów ra i rb oraz inkrementacja licznika. Następnie w stanie S2 sprawdzana jest poprzednia wartość licznika, jeśli jest równa *N*-1, to kończy się algorytm mnożenia i automat przechodzi do stanu S_0 z ustawieniem przyrzutnika rdone. W przeciwnym razie cykl procesu mnożenia jest powtarzany.

Zgodnie z rysunkiem 5.6 można zauważyć, że jeden cykl mnożenia jest wykonywany w dwóch cyklach zegara, dlatego FSMD dla automatu Mealy'ego wykonuje mnożenie *N*-bitowych liczb binarnych w n_{Mealy} taktach zegara, gdzie:

$$n_{Mealy} = 2N + 1 \tag{5.2}$$

Porównanie FSMD Moore'a i FSMD Mealy'ego z rozważanego przykładu pokazuje, że FSMD Mealy'ego ma znacznie mniej stanów (odpowiednio 3 i 7), a także działa znacznie szybciej, ponieważ każdy cykl procesu mnożenia jest wykonywany w dwóch taktach zegara (w FSMD Moore'a w trzech taktach zegara).

5.5. Zwiększenie prędkości mnożnika synchronicznego

Fakt, że jeden cykl procesu mnożenia jest wykonywany w kilku cyklach zegarowych, sprawia, że praktyczne zastosowanie omówionych powyżej konstrukcji mnożników jest niezwykle nieefektywne, ponieważ znacznie wydłuża czas mnożenia.

Pytanie brzmi: jak zaprojektować FSMD, który wykonuje każdy cykl mnożenia w jednym takcie zegara? Aby to zrobić, konieczne jest zbudowanie diagramu ASMD, w którym jeden blok ASMD odpowiada cyklowi mnożenia. Diagram ASMD dla automatu Mealy'ego spełniającego postawione wymagania pokazano na rysunku 5.7.

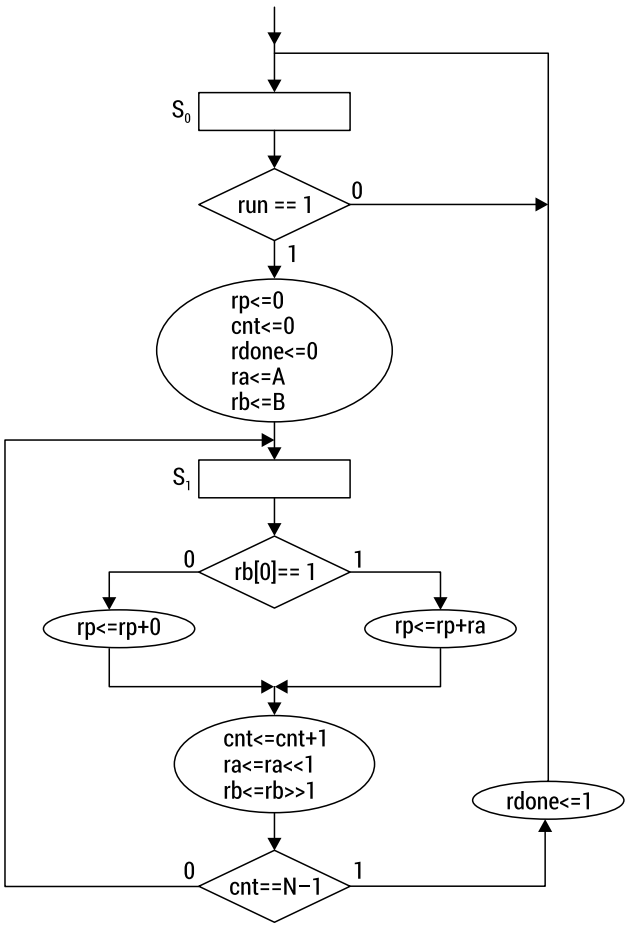
Cechą szczególną diagramu ASMD z rysunkiem 5.7 jest to, że istnieją tu tylko dwa bloki ASMD: w stanie S_0 wykonywana jest inicjalizacja rejestrów, a stan S_1 odpowiada pełnemu cyklowi mnożenia.

W ASMD z rysunku 5.7 stany S_1 i S_2 ASMD z rysunku 5.6 zostały połączone w jeden stan. Stało się to możliwe dzięki temu, że operacje wykonywane w stanach S_1 i S_2 zmieniają zawartość różnych rejestrów (w stanie S_1 zmienia się zawartość rejestru rp, a w stanie S_2 zawartość rejestrów ra i rb).

Należy zauważyć, że wszystkie metody optymalizacji syntezy behawioralnej mogą być zastosowane w ASMD [22], np. *eliminacja martwego kodu* (dead code elimination), *propagacja stałych* (constant propagation), *ruch kodu* (code motion), *ekspansja w linii* (in-line expansion), *rozwijanie pętli* (loop unrolling) i *redukcja wysokości drzewa* (tree height reduction). Ponadto do ASMD można zastosować inne techniki optymalizacji specyficzne dla ASMD, na przykład w celu zmniejszenia liczby stanów w pętli.

Na rysunku 5.7 można zauważyć, że jeden cykl mnożenia jest wykonywany w jednym cyklu zegara, dlatego mnożenie *N*-bitowych liczb binarnych zgodnie z ASMD z rysunku 5.7 odbywa się w n_{Mealy_1} cyklach zegarowych, gdzie:

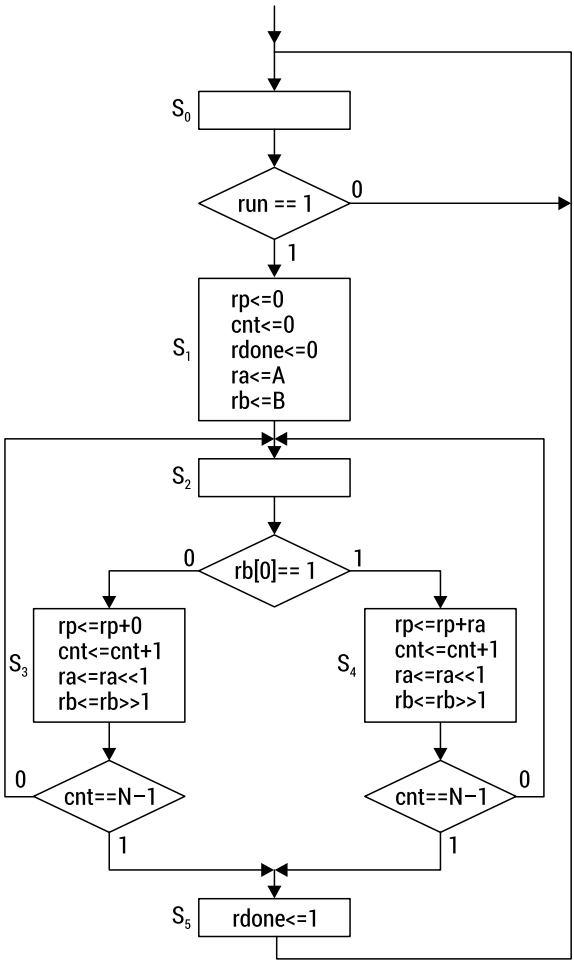
$$n_{Mealy\_1} = N + 1 \tag{5.3}$$



RYSUNEK 5.7. **Diagram ASMD automatu Mealy'ego, który zapewnia największą szybkość mnożnika synchronicznego**

Aby zwiększyć wydajność FSMD typu Moore'a, należy przekształcić ASMD z rysunku 5.5 w celu zmniejszenia liczby stanów w cyklu. Przekształcony diagram ASMD pokazano na rysunku 5.8.

W tym przypadku operacje zwiększania licznika i przesuwania rejestrów są wykonywane w stanach S_3 i S_4. Pozwala to na usunięcie stanu S_5 z ASMD z rysunek 5.5. Ta metoda transformacji ASMD odnosi się do metod *redukcji wysokości drzew* (tree height reduction) [22].



RYSUNEK 5.8. **Diagram ASMD automatu Moore'a, zapewniający wzrost szybkości działania mnożnika synchronicznego**

Niestety, ASMD Moore'a z rysunku 5.8 nie opisuje cyklu mnożenia za pomocą jednego stanu, jak ASMD Mealy'ego. Stan S_2 jest wymagany do wejścia w cyklu obliczania iloczynu cząstkowego. Następnie sprawdzany jest najmniej znaczący bit mnożnika (rb[0]). Wykonywanie działań zależnych od wartości rb[0] wymaga również co najmniej jednego dodatkowego stanu. Tak więc cykl operacji mnożenia przechodzi przez dwa stany FSMD Moore'a, czyli jest wykonywany w dwóch cyklach zegarowych. W rezultacie pomnożenie *N*-bitowych liczb binarnych za pomocą ASMD z rysunku 5.8 odbywa się w n_{Moore_1} cyklach zegara, gdzie:

$$n_{Moore\_1} = 2N + 3 \tag{5.4}$$

5.6. Opis FSMD w języku Verilog

Aby zaimplementować synchroniczny mnożnik na FPGA w postaci FSM Mealy'ego, należy wybrać diagram ASMD z rysunku 5.7. Poniżej znajduje się kod projektu FSMD Mealy w języku Verilog.

```
module mult_FSMD_Mealy #(parameter N = 4)
        (input clk, reset, run,                 // sygnały sterujące
        input [N-1:0] a,b,                      // słowa wejściowe
        output [2*N-1:0] p,                     // wynik
        output done);                           // sygnał zakończenia mnożenia

        reg [2*N-1:0] ra,rp;                    // deklaracja rejestrów
        reg [N-1:0] rb;
        reg rdone;
        reg [N_cnt-1:0] cnt;                    // licznik

        localparam N_cnt=clogb2(N-1); // określenie rozmiaru licznika

        function integer clogb2(input [N-1:0] v); // funkcja stała
                for (clogb2 = 0; v > 0; clogb2 = clogb2 + 1)
                        v = v >> 1;
        endfunction

        localparam [0:0] s0 = 0, s1 = 1;        // dklaracja stanów
        reg [0:0] state;                        // zmienna stanu

        always @(posedge clk)           // rozpoczęcie procesu mnożenia
                if(reset)       state <= s0;
                else
                        case (state)
                        s0:     if(run)
                                        begin
                                                rp <= 0; cnt <= 0; rdone <= 0;
                                                ra <= {{N{1'b0}},a};
                                                rb <= b;
                                                state <= s1;
                                        end
                                else    state <= s0;
                        s1:             begin
```

```
                                        if(rb[0])       rp <= rp + ra;
                                        else            rp<=rp+{2*N{1'b0}};
                                        cnt <= cnt + 1'b1;
                                        rb <= rb >> 1;
                                        ra <= ra << 1;
                                        if (cnt == N-1)
                                        begin
                                                rdone <= 1'b1;
                                                state <= s0;
                                        end
                                        else    state <= s1;
                                end
                        default: state <= s0;
                        endcase
        assign p = rp;                  // określenie wartości wyjściowych
        assign done = rdone;
endmodule
```

Moduł mult_FSMD_Mealy jest opisany za pomocą stylu opisu automatów skończonych z jednym procesem [21]. W naszym przypadku jest to dopuszczalne, ponieważ na wyjściu rejestrów generowane są iloczyn p i sygnał done. Powyższy kod używa funkcji stałej clogb2 do określenia rozmiaru licznika. Funkcja ta oblicza najmniejszą liczbę całkowitą większą lub równą logarytmowi przy podstawie 2 z liczby v.

Kod projektu FSMD Moore'a w języku Verilog, który odpowiada ASMD z rysunku 5.8 przedstawia się następująco.

```
module mult_FSMD_Moore #(parameter N = 4)
        (input clk, reset, run,                 // sygnały sterujące
        input [N-1:0] a,b,                      // słowa wejściowe
        output [2*N-1:0] p,                     // wynik
        output done);                           // sygnał zakończenia mnożenia

        reg [2*N-1:0] ra,rp;                    // deklaracja rejestrów
        reg [N-1:0] rb;
        reg rdone;
        reg [N_cnt-1:0] cnt;                    // licznik

        localparam N_cnt=clogb2(N-1); // określenie rozmiaru licznika

        function integer clogb2(input [N-1:0] v); // funkcja stała clogb2 –
                for (clogb2 = 0; v > 0; clogb2 = clogb2 + 1)
                        v = v >> 1;
        endfunction
```

```
        localparam [2:0] s0 = 0,s1 = 1,s2 = 2,s3 = 3,s4 = 4,s5 = 5;  // dklaracja stanów
        reg [2:0] state;                                // zmienna stanu

        always @(posedge clk)               // rozpoczęcie procesu mnożenia
                if(reset)       state <= s0;
                else
                        case (state)
                        s0:     if(run)         state <= s1;
                                else            state <= s0;

                        s1:     begin
                                                rp <= 0; cnt <= 0; rdone <= 0;
                                                ra <= {{N{1'b0}},a};
                                                rb <= b;
                                                state <= s2;
                                end
                        s2:     if(rb[0]) state <= s4;
                                else            state <= s3;
                        s3:     begin
                                                rp <= rp + {2*N{1'b0}};
                                                cnt <= cnt + 1'b1;
                                                rb <= rb >> 1;
                                                ra <= ra << 1;
                                                if (cnt == N-1)         state <= s5;
                                                else                    state <= s2;
                                end
                        s4:     begin
                                                rp <= rp + ra;
                                                cnt <= cnt + 1'b1;
                                                rb <= rb >> 1;
                                                ra <= ra << 1;
                                                if (cnt == N-1)         state <= s5;
                                                else                    state <= s2;
                                end
                        s5:     begin
                                                rdone <= 1'b1;
                                                state <= s0;
                                end
                        endcase
        assign p = rp;                          // określenie wartości wyjściowych
        assign done = rdone;
endmodule
```

Do opisu automatu Moore'a został również użyty styl opisu z jednym procesem [21]. W tym przypadku stan następny jest określany w każdym stanie obecnym, a wszystkie czynności wykonywane na rejestrach opisane są zgodnie z diagramem ASMD z rysunku 5.8.

5.7. Metodologia ASMD-FSMD projektowania układów cyfrowych

W tej części zostanie pokazany formalny opis rozważanej metodyki ASMD-FSMD w postaci następującego algorytmu.

Algorytm 2. Metodyka ASMD-FSMD projektowania układów cyfrowych.

1. Określane są stany FSMD.
2. Dla każdego stanu budowany jest blok ASMD.
   2.1. W wierzchołkach warunkowych ASMD zapisywane są funkcje logiczne, których wartość jest sprawdzana w danym stanie.
   2.2. W przypadku FSMD Moore'a wierzchołki stanów (prostokąty) zapisuje się operacje wykonywane na zawartości rejestrów w danym stanie.
   2.3. W przypadku FSMD Mealy'ego operacje wykonywane na zawartości rejestrów w danym przejściu są zapisywane w wierzchołkach wyjść warunkowych (owale).
3. Bloki ASMD są ze sobą połączone zgodnie z algorytmem pracy układu. W takim przypadku każde wyjście bloku ASMD można podłączyć tylko do jednego wejścia tego samego lub innego bloku ASMD.
4. W razie potrzeby diagram ASMD jest modyfikowany w celu zwiększenia prędkości projektowanego urządzenia. W tym celu analizowane są pętle algorytmu, a ASMD przekształca się w taki sposób, aby zminimalizować liczbę stanów w obrębie pętli.
5. Kod FSMD w Verilog jest budowany bezpośrednio z ASMD. Zmiennym algorytmu w kodzie odpowiadają rejestry. Funkcje logiczne testowane w wierzchołkach warunkowych ASMD odpowiadają wyrażeniom logicznym w instrukcjach if. Akcje wykonywane w blokach ASMD są opisywane jako bloki proceduralne begin… end. Każdy blok dla FSMD Moore'a opisuje operacje wykonywane w danym stanie diagramu ASMD i definiuje stany następne (stany przejść) zgodnie z przejściami diagramu ASMD. W każdym bloku dla FSMD Mealy'ego tworzony jest opis algorytmiczny wszystkich czynności wykonywanych w odpowiednim bloku ASMD.
6. Koniec.

Należy zwrócić uwagę na niektóre cechy metody ASMD-FSMD. Głównym etapem projektowania jest budowa diagramu ASMD w oparciu o algorytm działania (funkcjonowania) układu. Nie ma ścisłego oddzielenia układu wykonawczego od układu sterującego. Jednocześnie ASMD definiuje stany automatu skończonego, które odpowiadają

stanom układu sterującego. Pozwala to powiązać algorytm działania projektowanego urządzenia z taktami zegarowymi. Dlatego, korzystając z ASMD, można prześledzić, ile cykli zegara zajmuje każda gałąź algorytmu. ASMD nie definiuje również jawnie struktury układu wykonawczego.

Różnica między techniką ASMD-FSMD a innymi znanymi metodami polega na tym, że w ASMD można używać zarówno automatu typu Moore'a, jak i automatu typu Mealy'ego (w [16] tylko automatu Moore'a); czynności wykonywane w układzie wykonawczym na zawartości rejestrów mogą być zapisywane zarówno w prostokątach ASMD, jak i w owalach (w [16] tylko w owalach).

Należy również zauważyć, że jeden i ten sam algorytm działania układu można opisać różnymi diagramami ASMD, co wpływa na szybkość i koszt realizacji projektu. Aby zwiększyć wydajność, pętle algorytmu należy opisać jak najmniejszą liczbą stanów w celu zminimalizowania liczby stanów w obrębie pętli. Do tego celu lepiej nadaje się automat typu Mealy'ego.

5.8. Badania eksperymentalne

Na przykładzie prostego mnożnika synchronicznego rozważono trzy techniki projektowania układów cyfrowych: implementację algorytmiczną w języku Verilog, technikę tradycyjną (w postaci złożenia układu wykonawczego i sterującego) oraz technikę wykorzystującą automaty FSMD ze ścieżką przetwarzania danych (metodyka ASMD-FSMD). Aby przetestować skuteczność tych technik, zbadano następujące projekty mnożników synchronicznych:

- algorytm_mult – algorytmiczna implementacja mnożnika w języku Verilog;
- mult_FSM_Mealy – mnożnik zbudowany w postaci układów wykonawczego i sterującego, gdzie rolę układu sterującego pełni automat Mealy'ego;
- mult_FSM_Moore – mnożnik zbudowany w postaci układów wykonawczego i sterującego, gdzie rolę układu sterującego pełni automat Moore'a;
- mult_FSMD_Mealy – mnożnik zbudowany zgodnie z diagramem ASMD automatu Mealy'ego z rysunek 5.7;
- mult_FSMD_Moore – mnożnik zbudowany zgodnie z diagramem ASMD automatu Moore'a z rysunek 5.8.

Wyniki przprowadzonych badan w systemie Quartus Prime w wersji 18.1 z domyślnymi parametrami syntezy układów FPGA z rodziny Cyclone IV E przedstawiono w tabelach 5.1–5.5, gdzie N jest liczbą bitów słów wejściowych mnożnika; L to liczba użytych elementów logicznych FPGA (koszt realizacji); F – maksymalna częstotliwość pracy układu w MHz (szybkość); T – czas trwania jednego taktu zegara w nanosekundach, $T = 1/F$; n – liczba taktów zegara potrzebnych do obliczenia wyniku; t – czas (w nanosekundach) wymagany do obliczenia wyniku, $t = T \cdot n$.

Należy zauważyć, że przy realizacji tych projektów na układach FPGA nie zostały wykorzystane wbudowane bloki DSP i wbudowane bloki mnożnika.

Dla projektu algorytm_mult $n = 1$, dla projektu mult_FSM_Mealy wartość n wynosi $(N + 1)$, dla projektu mult_FSM_Moore wartość n wynosi $(N + 2)$, dla projektu mult_FSMD_Mealy wartość n jest określana zgodnie z (3), a dla projektu mult_FSMD_Moore – zgodnie z (4).

TABELA 5.1. **Wyniki badań projektu algorytm_mult**

N	L	F	T	n	t
4	34	130,23	7,679	1	7,679
8	129	75,00	13,333	1	13,333
16	513	30,39	32,916	1	32,916
32	2043	11,28	88,652	1	88,652
64	8187	4,47	223,714	1	223,714
128	32854	0,99	1010,101	1	1010,101

TABELA 5.2. **Wyniki badań projektu mult_FSM_Mealy**

N	L	F	T	n	t
4	36	245,28	4,077	5	20,39
8	66	212,95	4,696	9	42,26
16	123	224,77	4,449	17	75,63
32	236	155,23	6,442	33	212,59
64	464	110,29	9,067	65	589,36
128	919	31,43	31,817	129	4104,39

TABELA 5.3. **Wyniki badań projektu mult_FSM_Moore**

N	L	F	T	n	t
4	35	282,84	3,536	6	21,22
8	64	159,26	6,279	10	62,79
16	124	177,65	5,659	18	101,86
32	237	157,01	6,369	34	216,55
64	462	110,19	9,075	66	598,95
128	916	31,57	31,676	130	4117,88

TABELA 5.4. Wyniki badań projektu mult_FSMD_Mealy

N	L	F	T	n	t
4	28	285,88	3,498	5	17,49
8	49	228,89	4,369	9	39,32
16	90	202,18	4,946	17	84,08
32	172	172,06	5,812	33	191,80
64	335	111,64	8,957	65	582,21
128	657	32,50	30,769	129	3969,20

TABELA 5.5. Wyniki badań projektu mult_FSMD_Moore

N	L	F	T	n	t
4	48	192,98	5,182	11	57,00
8	102	174,19	5,741	19	109,08
16	192	155,13	6,446	35	225,61
32	304	152,65	6,551	67	438,92
64	578	104,98	9,526	131	1247,91
128	1145	29,64	33,738	259	8738,14

W celu lepszego porównania rozpatrywanych projektów, w tabeli 5.6 i tabeli 5.7 przedstawiono wartości kosztu realizacji L i czasu t wykonania operacji mnożenia w tych projektach dla różnych wartości N.

TABELA 5.6. Koszt L realizacji mnożników (liczba elementów logicznych)

Project	N = 4	N = 8	N = 16	N = 32	N = 64	N = 128
algorithm_mult	34	129	513	2043	8187	32854
mult_FSM_Moore	35	64	124	237	462	916
mult_FSM_Mealy	36	66	123	236	464	919
mult_FSMD_Moore	47	102	192	309	583	1139
mult_FSMD_Mealy	28	49	90	172	335	657

TABELA 5.7. Czas t operacji mnożenia (w nanosekundach)

Project	N = 4	N = 8	N = 16	N = 32	N = 64	N = 128
algorithm_mult	8	13	33	89	224	1010
mult_FSM_Moore	21	63	102	217	599	4118
mult_FSM_Mealy	20	42	76	213	589	4104
mult_FSMD_Moore	76	156	345	663	1831	12232
mult_FSMD_Mealy	17	39	84	192	582	3969

5.9. Dyskusja wyników

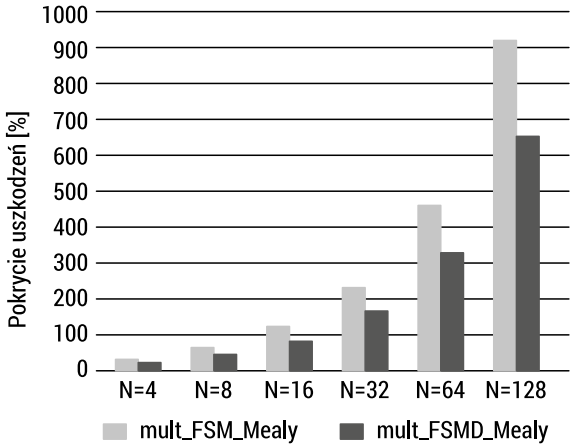
Analizując tabele 5.6 i 5.7 można zauważyć, że przy zastosowaniu tradycyjnej metodologii projektowania układów cyfrowych (projekty mult_FSM_Moore i mult_FSM_Mealy) typ automatu (Mealy lub Moore) nie odgrywa szczególnej roli, ponieważ koszt realizacji i szybkość działania projektów mult_FSM_Moore i mult_FSM_Mealy są w przybliżeniu takie same.

W przypadku zastosowania metodyki ASMD-FSMD projekt mult_FSMD_Mealy ma wyraźną przewagę nad projektem mult_FSMD_Moore, zarówno pod względem kosztów realizacji, jak i wydajności. Przewagę projektu mult_FSMD_Mealy pod względem kosztów realizacji dowodzi fakt, że w cyklu mnożenia ASMD dla automatu Moore'a z rysunku 5.8, wiele operacji jest dublowanych (inkrementacja licznika, przesuwanie rejestrów), a przewaga szybkości wynika z faktu, że jeden cykl mnożenia w ASMD z rysunku 5.8 jest wykonywany w dwóch taktach zegarowych. Dlatego przy stosowaniu metodyki ASMD-FSMD należy preferować automaty typu Mealy'ego.

Porównanie metodyki ASMD-FSMD (projekt mult_FSMD_Mealy) z metodologią tradycyjną (projekty mult_FSM_Moore i mult_FSM_Mealy) pokazuje, że metodyka ASMD-FSMD przewyższa tradycyjną metodologię zarówno pod względem kosztów realizacji (od 28,6% do 39,7%), jak i szybkości działania układów (maksymalnie o 17,6% dla projektu mult_FSM_Moore przy N = 4).

Przewaga metodyki ASMD-FSMD nad metodą tradycyjną pod względem kosztów implementacji, pokazana na rysunku 5.9.

Należy zwrócić szczególną uwagę, że projekt mnożnika mult_FSMD_Mealy został opisany bezpośrednio w języku Verilog zgodnie z ASMD z rysunku 5.7, bez opracowywania układów wykonawczego i sterującego, w porównaniu do projektów mult_FSM_Moore i mult_FSM_Mealy. Z tego powodu metodyka ASMD-FSMD znacznie skraca czas projektowania, w porównaniu z podejściem tradycyjnym, ponieważ nie ma potrzeby projektowania układu wykonawczego i wszystkich jego elementów składowych, a także układu sterującego i modułu nadrzędnego.



RYSUNEK 5.9. Porównanie kosztów realizacji mnożników synchronicznych w przypadku zastosowania tradycyjnej metodyki projektowania (projekt mult_FSM_Mealy) oraz metodyki ASMD-FSMD (projekt mult_FSMD_Mealy)

Należy również zauważyć, że metodyka projektowania ASMD-FSMD poprawia niezawodność projektów. Faktem jest, że wiele etapów tradycyjnego projektowania jest wykonywanych ręcznie przez projektanta i jest źródłem trudnych do znalezienia błędów. W metodyki ASMD-FSMD po zbudowaniu diagramu ASMD, opis projektu w języku Verilog wykonuje się bezpośrednio na bazie diagramu ASMD, bez żadnych opisów pośrednich.

Poniżej podjęto próbę oszacowania czasu projektowania i niezawodności projektów mnożników synchronicznych realizowanych przy użyciu tradycyjnego podejścia i metodyki ASMD-FSMD. Aby to zrobić, zostanie porównana długość kodu (liczba wierszy) projektów mult_FSM_Mealy (podejście tradycyjne) i mult_FSMD_Mealy (metodyka ASMD-FSMD). Projekt mult_FSM_Mealy składa się z modułu najwyższego poziomu (mult_a_Mealy), układu sterującego (fsm_Mealy) i układu wykonawczego (datapath_mult). Z kolei układ wykonawczy zawiera następujące komponenty: sumator (adder), licznik modulo (counter_modulo_M), multiplekser magistrali (mux_2), rejestr przesuwający w lewo (shl_load) i rejestr przesuwający w prawo (shr_load), a także zwykły rejestr (rejestr). Liczbę wierszy dla każdego modułu podano w tabela 5.8.

Analizując tabelę 5.8 widać, że w przypadku zastosowania metodyki ASMD-FSMD, w porównaniu z podejściem tradycyjnym, długość kodu skraca się 144/58 = 2,483 razy, tj. około 2,5 razy. Dlatego możemy założyć, że w rozważanym przykładzie mnożnika synchronicznego, użycie metodyki ASMD-FSMD może skrócić czas projektowania około 2,5 krotnie.

TABELA 5.8. Liczba linii kodu dla elementów projektu mult_FSM_Mealy i projektu mult_FSMD_Mealy

Nazwa modułu	Liczba linii kodu
mult_a_Mealy	17
fsm_Mealy	28
datapath_mult	26
adder	8
counter_modulo_M	23
mux_2	9
shl_load	11
shr_load	11
register	11
∑ (mult_FSM_Mealy)	144
mult_FSMD_Mealy	58

Trudniej jest określić ilościowo wzrost niezawodności projektu w wyniku zastosowania metodyki ASMD-FSMD. Można założyć, że niezawodność projektu jest również proporcjonalna do rozmiaru kodu projektu. Jednak ocena niezawodności powinna uwzględniać liczbę powiązań między modułami oraz szereg innych czynników. Jeżeli niezawodność oceniana jest tylko na podstawie liczby wierszy kodu, to zastosowanie w naszym przypadku metodyki ASMD-FSMD pozwala co najmniej 2,5-krotnie zwiększyć niezawodność projektu.

W metodyki ASMD-FSMD, na konstrukcję diagramu ASMD nakładane są specjalne wymogi. Doświadczenie ze stosowania metodyki ASMD-FSMD pokazało, że do jej efektywnego wykorzystania konieczne jest przestrzeganie pewnych zasad. Przede wszystkim należy dążyć do zbudowania diagramów blokowych ASMD dla automatów typu Mealy'ego. Jeżeli algorytm pracy układu zawiera cykle, to cały cykl w ASMD należy opisać jednym stanem. Jeśli nie jest to możliwe, należy użyć minimalnej liczby stanów.

Algorytmiczna implementacja mnożnika (projekt algorytm_mult) znacznie przewyższa wszystkie rozważane projekty pod względem szybkości (3–5 razy), jednak ma gorszy koszt realizacji, który szybko rośnie wraz ze wzrostem długości słowa mnożnika wejściowego N. Metoda projektowania układów cyfrowych oparta na implementacji algorytmicznej może być zalecana do budowy niewielkich układów, a także do celów symulacyjnych w celu sprawdzenia poprawności algorytmu działania układu.

Podsumowanie

Na przykładzie projektowania mnożnika synchronicznego rozważono trzy metody projektowania układów cyfrowych: implementację algorytmiczną w języku Verilog, tradycyjną (gdy projekt przedstawiany jest jako kompozycja układów wykonawczego i sterującego) oraz opartą na automatach skończonych ze ścieżką przetwarzania danych (metodyka ASMD-FSMD).

Porównanie proponowanej metodyki ASMD-FSMD z podejściem tradycyjnym pokazuje, że zastosowanie nowej metodyki może obniżyć koszt realizacji od 28,6% do 39,7%, a także w niektórych przypadkach zwiększyć szybkość działania projektów o 17,6%.

Jednak główną zaletą metodyki ASMD-FSMD jest to, że pozwala ona na skrócenie czasu projektowania i zwiększenie niezawodności projektów co najmniej 2,5-krotnie.

Metodyka ASMD-FSMD może być wykorzystana przy projektowaniu układów cyfrowych opartej na dowolnej bazie technologicznej (niekoniecznie FPGA), na przykład z wykorzystaniem układów ASIC. Jako język opisu dla automatów skończonych ze ścieżką przetwarzania danych FSMD może być użyty dowolny język opisu sprzętu, na przykład VHDL lub SystemVerilog. Obiecującym kierunkiem rozwoju jest również wykorzystanie metodyki ASMD-FSMD do wysokopoziomowego projektowania złożonych układów, a także opracowanie specjalnych algorytmów do optymalizacji ASMD, na przykład w celu zmniejszenia liczby stanów w cyklach.

Praca została wykonana przy częściowym wsparciu finansowym Politechniki Białostockiej (Polska, grant WZ / WI-IIT / 4/2020).

Bibliografia

[1] Gajski D.D., Dutt N.D., Wu A.C., Lin S.Y., *High—Level Synthesis: Introduction to Chip and System Design*, Kluwer, Boston, USA 1992

[2] Auletta R., Reese B., Traver, C., *A comparison of synchronous and asynchronous FSMD designs*, Proc. of the IEEE International Conference on Computer Design (ICCD'93), Cambridge, MA, USA, October 3–6, 1993, IEEE: Cambridge, USA 1993,178–182

[3] Karfa C., Sarkar D., Mandal C., *Verification of datapath and controller generation phase in high-level synthesis of digital circuits*. "IEEE Transactions on CAD" 2010, 29 (3), 479–492

[4] Hu J., Wang G., Chen G., Wei X., *Equivalence Checking of Scheduling in High-Level Synthesis Using Deep State Sequences*, "IEEE Access" 2019, nr 7, s. 183435–183443

[5] Schaumont P., Shukla S., Verbauwhede I, *Design with race-free hardware semantics*. Proceedings of the Design Automation & Test in Europe Conference, Munich, Germany, 6–10 March 2006, IEEE: 2007, 1, 6

[6] Zhu J., Gajski D.D., *A unified formal model of ISA and FSMD*. Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES'99), Rome, Italy, 3 March 1999, IEEE, 1999, (IEEE Cat. No. 99TH8450), 121–125

[7] Kavvadias N., Masselos K., *Automated synthesis of FSMD-based accelerators for hardware compilation*. Proceedings of the 23rd International Conference on Application-Specific Systems, Architectures and Processors, Delft, Netherlands, 9–11 July 2012, IEEE, 2012, 157–160

[8] Banerjee K., Sarkar D., Mandal C., *Extending the FSMD framework for validating code motions of array-handling programs*, "IEEE Transactions on CAD" 2014, 33(12), 2015–2019

[9] Hwang E., Vahid F., Hsu Y.C., *FSMD functional partitioning for low power*. Proceedings of the Conference on Design, automation and test in Europe, Munich, Germany, 9–12 March 1999, IEEE, 1999, 7

[10] Abdullah A.C., Ooi C.Y., Ismail N.B., Mohammad N.B., *Power-aware through-silicon-via minimization by partitioning finite state machine with datapath*. Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 22–25 May 2016, IEEE, 2016, 1942–1945

[11] Babakov R., Barkalov A., Titarenko L., *Research of efficiency of microprogram final-state machine with datapath of transitions*. Proceedings of the 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, Ukraine, 21–25 Feb. 2017, IEEE, 2017, 203–206

[12] Clare C.R., *Designing logic systems using state machines*, McGraw-Hill Book Company, New York USA 1973

[13] Green D.H., Chughtai M.A., *Use of multiplexers in direct synthesis of ASM-based designs*, "IEE Proceedings E-Computers and Digital Techniques" 1986, 133(4), 194–200

[14] Baranov S., *Minimization of algorithmic state machines*. Proceedings of the 24th EUROMICRO Conference (Cat. No. 98EX204), Vasteras, Sweden, 27–27 Aug. 1998, IEEE, 1998, 1, 176–179

[15] Jenihhin M., Baranov S., Raik J., Tihhomirov V., *PSL assertion checkers synthesis with ASM based HLS tool ABELITE*. Proceedings of the 13th Latin American Test Workshop (LATW), Quito, Ecuador, 10–13 April 2012, IEEE, 2012, 1–6

[16] Ciletti M.D., *Advanced digital design with the Verilog HDL*, Prentice Hall of India, New Delhi, India 2005

[17] Martin P., Bueno E., Rodriguez F.J., Saez V., *A methodology for optimizing the FPGA implementation of industrial control systems*. Proceedings of the 35th Annual Conference of IEEE Industrial Electronics, Porto, Portugal, 3–5 Nov. 2009, IEEE, 2009, 2811–2816

[18] Saha A., Ghosh A., Kumar K.G., *FPGA Implementation of Arcsine Function Using CORDIC Algorithm*, "AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances A" 2017, 54(2), 197–202

[19] Burciu P., *An Efficient (Low Resources) Modular Hardware Implementation of the AES Algorithm*, "Journal of Electrical Engineering, Electronics, Control and Computer Science" 2019, 5(3), 1–10

[20] Sowmya K.B., Shreyans G., Vishnusai R.T., *Design of UART Module using ASMD Technique*. Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020), Coimbatore, India, 10–12 June 2020, IEEE, 2020, 176–181

[21] Salauyou V.V., *Verilog language in embedded systems design on FPGA*, Hotline – Telecom, Moscow, Russia 2020

[22] Bergamaschi R.A., *Behavioral Synthesis: an Overview* [w:]. R. Reis, M. Lubaszewski, J.A. Jess (red.), *Design of Systems on a Chip: Design and Test*. Springer, Boston, USA 2006, 103–131

ASMD-FSMD for designing digital devices on FPGA, based on the Verilog hardware description language

Abstract: Recently, there has been, on the one hand, an increase in the complexity of digital device designs and, on the other hand, an increase in the requirements for the development time and the reliability of the designs. One of the directions of solving this problem is developing new techniques for designing digital devices.This paper proposes a new technique for designing digital devices based on finite state machines with datapath (FSMD), when the functioning of the device is described in the form of an algorithm state machine with datapath (ASMD) charts. The new technique is called ASMD-FSMD.

In the technique ASMD-FSMD, the device is not separated into the control unit (FSM) and the datapath, but more closely resembles a behavioral (algorithmic) description of device functioning. However, in contrast to the algorithmic description, the FSMD states are explicitly defined in ASMD. This paper provides a formal description of ASMD-FSMD technology. The main design stage is building an ASMD chart based on the behavior (the operating algorithm) of a digital device. There is no strict separation of the digital device into the datapath and the control unit (FSM). At the same time, the ASMD chart determines the FSM states that correspond to the states of the control unit. This allows you to bind the algorithm of the functioning of the designed device to synchronization clocks. Therefore, according to the ASMD chart, the developer can track how many clock cycles each branch of the algorithm performs. In addition, the ASMD chart does not explicitly define the structure of the datapath. Our ASMD chart can describe any type of machine, both Mealy and Moore. The same operating algorithm of the device can be described by different ASMD charts, this affects the performance and the implementation cost (area) of the design. To increase performance, the algorithm loops should be described with a minimum number of states to minimize the number of states in the loop path. Mealy FSMs are better suited for this purpose.

Different digital device design techniques are compared to each other using design examples of a synchronous multiplier on field programmable gate array (FPGA). The efficiency of the ASMD-FSMD technique compared to the traditional approach in terms of area and performance was investigated. The ASMD-FSMD technique, compared to the traditional one, reduces the area from 28.6% to 39.7% and increases the speed for some designs to 17.6%. In addition, using the ASMD-FSMD technique significantly reduces design time and increases design reliability. In conclusion, recommendations for using the ASMD-FSMD technique are made.

Keywords: digital device, finite state machines with datapath, algorithm state machine with datapath, field programmable gate array, design technique, development time, reliability, area, performance.

Politechnika Białostocka