Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka techniczna (ITE)**
Przedmiot: **Niezawodność i Diagnostyka układów cyfrowych**

SPRAWOZDANIE Z PROJEKTU

Jednostka arytmetyczna z detekcją błędów

Krzysztof Czerniachowicz

Prowadzący
Dr hab inż. Stanisław Piestrak prof. uczelni

WROCŁAW, 2025

Spis treści

1. Wstęp **5**
1.1. Cel i zakres pracy 5
1.2. Tło i znaczenie problemu 5
1.3. Założenia eksperymentu 6
2. Podstawy teoretyczne **7**
2.1. Model błędów i uszkodzeń 7
2.2. Metody wykrywania błędów w układach arytmetycznych 7
2.2.1. Kod parzystości 7
2.2.2. Metoda duplikacji z porównywaniem (DWC) 8
2.2.3. Kody resztowe (Residue Codes) 8
2.3. Arytmetyka resztowa 8
2.3.1. Przystawanie liczb 8
3. Plan eksperymentu/Symulacja **10**
3.1. Struktura układu 10
3.2. Model uszkodzeń i symulacja 10
3.3. Procedura testowa 11
4. Analiza wyników **12**
5. Podsumowanie **13**
Bibliografia **14**

Spis rysunków

3.1. Schematyczny model arytmetyczego układu samosprawdzającego modulo 3. Źródło: Design of the coarse-grained reconfigurable architecture DART with on-line error detection.[1] 10

Spis tabel

Rozdział 1

Wstęp

1.1. Cel i zakres pracy

Celem niniejszej pracy jest analiza niezawodności układów arytmetycznych narażonych na występowanie błędów przemijających, na przykładzie czterobitowego sumatora wyposażonego w układ samosprawdzający oparty na arytmetyce resztowej modulo 3. W szczególności badaniu poddano skuteczność detekcji błędów wprowadzanych do struktury układu poprzez symulację pojedynczych uszkodzeń logicznych typu *stuck-at-0* oraz *stuck-at-1*.

Praca obejmuje zarówno część teoretyczną - przedstawiającą podstawy działania kodów resztowych i ich właściwości w kontekście niezawodności układów cyfrowych - jak i część eksperymentalną, obejmującą symulację działania układu oraz analizę uzyskanych wyników.

1.2. Tło i znaczenie problemu

Wraz z miniaturyzacją technologii półprzewodnikowych rośnie podatność układów cyfrowych na zakłócenia promieniowania kosmicznego i jonizującego. Cząstki wysokoenergetyczne mogą powodować chwilowe zmiany stanów logicznych w tranzystorach, prowadząc do wystąpienia błędów przemijających (*soft errors*). Tego typu zjawiska nie powodują trwałego uszkodzenia sprzętu, lecz mogą skutkować błędnymi wynikami obliczeń, co w systemach krytycznych - takich jak sterowanie lotem, przetwarzanie danych naukowych czy systemy kosmiczne - jest niedopuszczalne.

W celu ograniczenia wpływu takich błędów opracowano szereg metod detekcji i korekcji, w tym techniki oparte na kodach resztowych, które pozwalają na bieżące sprawdzanie poprawności obliczeń bez znaczącego wzrostu złożoności układu. Kody resztowe, a w szczególności kod modulo 3, umożliwiają realizację tzw. układów samosprawdzających, w których każda operacja arytmetyczna ma równoległy tor kontrolny pracujący w arytmetyce resztowej.

1.3. Założenia eksperymentu

W eksperymencie zbudowano model układu kombinacyjnego składającego się z:

- głównego sumatora czterobitowego,
- modułu generującego resztę z dzielenia przez 3 dla pięciobitowej liczby reprezentującej sumę wejść,
- sumatora kontrolnego realizującego operacje w arytmetyce modulo 3,
- modułów generujących resztę mod 3 dla operandów wejściowych,
- komparatora porównującego reszty wyniku z obu torów.

Kluczowym elementem jest możliwość symulowania pojedynczych uszkodzeń w strukturze układu w celu odwzorowania wpływu błędów przemijających spowodowanych promieniowaniem kosmicznym. Uszkodzenia te modelowane są jako sklejenie sygnału do wartości logicznej 0 lub 1 (stuck-at-0 / stuck-at-1) na wybranych połączeniach pomiędzy bramkami logicznymi. Dla każdego wprowadzonego uszkodzenia przeprowadzana jest pełna analiza działania układu, obejmująca porównanie wyniku obliczeń głównego toru arytmetycznego z wynikiem toru kontrolnego modulo 3.

Otrzymane wyniki klasyfikowane są w trzech kategoriach:

- **błąd wykryty** – wynik błędny został poprawnie zidentyfikowany przez komparator,
- **błąd niewykryty (false negative)** – wynik błędny został błędnie uznany za poprawny,
- **błąd fałszywy (false positive)** – wynik poprawny został błędnie zidentyfikowany jako błędny.

Takie podejście pozwala na ilościową ocenę skuteczności detekcji błędów przez układ samosprawdzający oraz identyfikację elementów najbardziej wrażliwych na uszkodzenia logiczne.

Rozdział 2

Podstawy teoretyczne

2.1. Model błędów i uszkodzeń

W układach cyfrowych występują dwa główne typy nieprawidłowości: **uszkodzenia** (*faults*) i **błędy** (*errors*). Uszkodzenie jest zjawiskiem fizycznym lub logicznym powodującym trwałe lub chwilowe odstępstwo działania układu od jego specyfikacji, natomiast błąd jest skutkiem tego uszkodzenia widocznym w danych lub wynikach obliczeń. Jeśli błąd nie zostanie wykryty, może prowadzić do **awarii** systemu (*failure*), objawiającej się nieprawidłowym działaniem całego układu.[1]

Szczególne znaczenie w kontekście współczesnych technologii mają **błędy przemijające** (*transient faults*). Powstają one w wyniku zjawisk losowych, takich jak oddziaływanie cząstek promieniowania kosmicznego lub promieniowania jonizującego, które chwilowo zaburzają stan logiczny tranzystorów lub węzłów sygnałowych. Takie błędy nie powodują trwałego uszkodzenia struktury układu, ale mogą prowadzić do błędnych wyników obliczeń — szczególnie w urządzeniach reprogramowalnych, takich jak FPGA, oraz w procesorach o wysokiej gęstości integracji.[1]

Model błędu przemijającego w układach kombinacyjnych często opisuje się jako **uszkodzenie typu stuck-at**, polegające na trwałym wymuszeniu wartości logicznej „0” lub „1” na danym sygnale, niezależnie od rzeczywistej funkcji logicznej.

2.2. Metody wykrywania błędów w układach arytmetycznych

Wykrywanie błędów w czasie rzeczywistym (*on-line error detection*) jest kluczowe dla systemów o podwyższonej niezawodności. W praktyce stosuje się kilka podejść, różniących się skutecznością i złożonością sprzętową:

2.2.1. Kod parzystości

Najprostszym mechanizmem wykrywania błędów jest **kod parzystości**, w którym do każdego słowa binarnego dodaje się dodatkowy bit informujący o parzystości liczby jedynek. Kod ten umożliwia wykrycie wszystkich pojedynczych błędów bitowych oraz błędów o nieparzystej krotności. Jego główną wadą jest jednak niska skuteczność w przypadku układów arytmetycznych, gdzie występuje propagacja przeniesień — nawet pojedyncze uszkodzenie może spowodować zmianę kilku bitów, a wynikowy błąd może mieć parzystą krotność i pozostać niewykryty.

Ponadto, aby układ był samosprawdzający, cała struktura arytmetyczna musiałaby być przeprojektowana w celu przewidywania i propagacji bitów parzystości.

2.2.2. Metoda duplikacji z porównywaniem (DWC)

Drugim klasycznym podejściem jest **duplikacja z porównywaniem** (*Dual Modular Redundancy, DWC*), w której funkcjonalny blok obliczeniowy jest zdublowany, a wyniki obu torów są porównywane przez komparator. Rozwiązanie to pozwala na wykrycie dowolnego pojedynczego błędu w jednym z torów, jednak kosztem dwukrotnego zwiększenia liczby zasobów sprzętowych i poboru mocy. Z tego względu DWC stosuje się głównie w systemach krytycznych, gdzie koszt jest drugorzędny wobec niezawodności.

2.2.3. Kody resztowe (Residue Codes)

Kody resztowe (*residue codes*) stanowią efektywną metodę wykrywania błędów w układach arytmetycznych. Wykorzystują one własności arytmetyki modularnej, przypisując każdej liczbie dodatkową część kontrolną — jej resztę z dzielenia przez ustaloną podstawę A:

$$|X|_A = X \bmod A.$$

W układzie chronionym kodem resztowym obliczenie zasadnicze i jego odpowiednik modularny są wykonywane równolegle, a następnie porównywane. Jeśli wyniki nie są przystające modulo A, oznacza to wystąpienie błędu.

Kody resztowe są kodami **separowalnymi**, co pozwala dodać tor kontrolny bez modyfikacji głównego układu arytmetycznego. Wyróżniają się niskim kosztem implementacji i wysoką skutecznością detekcji — wykrywają wszystkie błędy pojedyncze oraz znaczną część błędów wielokrotnych.

Szczególnie korzystny w praktyce jest kod **modulo 3**, który wymaga jedynie dwóch bitów kontrolnych i prostych bloków logicznych, dzięki czemu znajduje zastosowanie w samosprawdzających sumatorach i jednostkach arytmetycznych.

2.3. Arytmetyka resztowa

Arytmetyka resztowa (zwana także modularną) jest działem matematyki zajmującym się operacjami na liczbach całkowitych z uwzględnieniem ich reszty z dzielenia przez pewną ustaloną liczbę całkowitą m, nazywaną **modułem**. W arytmetyce tej interesują nas nie same wartości liczb, lecz ich reszty z dzielenia przez moduł, co pozwala znacząco uprościć obliczenia i ograniczyć zakres wartości liczbowych.[2]

Działania w arytmetyce modularnej znajdują szerokie zastosowanie w informatyce i elektronice, m.in. w algorytmach kryptograficznych, kodach korekcyjnych oraz układach samosprawdzających, takich jak stosowane w niniejszym projekcie.

2.3.1. Przystawanie liczb

Podstawowym pojęciem w arytmetyce modularnej jest **przystawanie liczb**. Dwie liczby całkowite a oraz b nazywamy **przystającymi modulo** m, jeśli ich różnica jest całkowitą wielokrotnością liczby m. Formalnie zapisuje się to jako:

$$a \equiv b \pmod{m} \quad \Leftrightarrow \quad m \mid (a - b)$$

co oznacza, że liczby a i b dają tę samą resztę z dzielenia przez m. Na przykład:

$$14 \equiv 2 \pmod{12}$$

ponieważ $14 - 2 = 12$, a więc różnica jest podzielna przez 12.

Przystawanie jest relacją równoważności i zachowuje zgodność wobec działań arytmetycznych. Jeśli zachodzą przystawania:

$$a \equiv b \pmod{m} \quad i \quad c \equiv d \pmod{m},$$

to również:

$$a + c \equiv b + d \pmod{m}$$

oraz

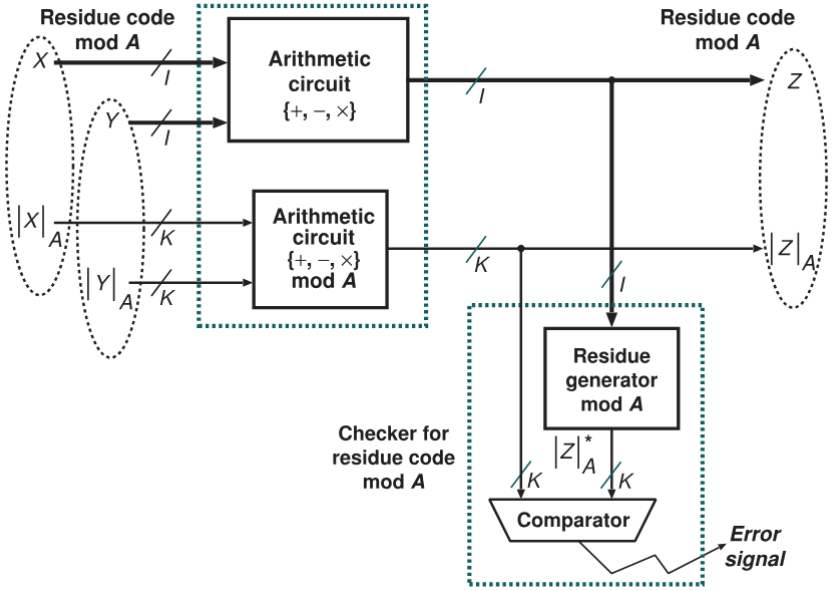
$$a \cdot c \equiv b \cdot d \pmod{m}.$$

Oznacza to, że dodawanie i mnożenie liczb przystających zachowuje przystawanie – co stanowi podstawę działania układów arytmetycznych w kodach resztowych.

Rozdział 3

Plan eksperymentu/Symulacja

3.1. Struktura układu



Rysunek 3.1: Schematyczny model arytmetyczego układu samosprawdzającego modulo 3. Źródło: Design of the coarse-grained reconfigurable architecture DART with on-line error detection.[1]

3.2. Model uszkodzeń i symulacja

Symulacja została zaprojektowana tak, aby odwzorować pojedyncze błędy przemijające (ang. *transient faults*), których źródłem mogą być cząstki promieniowania kosmicznego powodujące chwilową zmianę stanu logicznego. Każdy taki błąd modelowany jest w postaci **uszkodzenia typu stuck-at**, polegającego na wymuszeniu stałej wartości logicznej 0 lub 1 na wyjściu wybranej bramki logicznej.

W modelu symulacyjnym każda bramka oraz kluczowy sygnał pośredni zostały wyposażone w dodatkowy moduł `fault_mux`, który umożliwia wymuszenie błędnej wartości przy pomocy magistrali sterującej `fault_en_bus`. Dzięki temu możliwe jest systematyczne testowanie wszystkich pojedynczych punktów potencjalnej awarii.

3.3. Procedura testowa

Symulacja obejmuje pełne przeszukanie przestrzeni wejść i uszkodzeń:

$$\textit{Wejścia} \times \textit{Punkty iniekcji} \times \{\textit{stuck-at-0, stuck-at-1}\}.$$

Dla każdej kombinacji przeprowadzane są dwa przebiegi:

1. **Referencyjny** - bez aktywnego uszkodzenia, w celu uzyskania poprawnego wyniku.
2. **Testowy** - z aktywnym pojedynczym uszkodzeniem (`fault_en`=1), w celu porównania wyników toru głównego i kontrolnego.

Na podstawie porównania wyników klasyfikuje się przypadki jako:

- **Błąd wykryty** - wynik błędny poprawnie zgłoszony przez komparator,
- **Błąd niewykryty (false negative)** - wynik błędny uznany za poprawny,
- **Błąd fałszywy (false positive)** - wynik poprawny błędnie zidentyfikowany jako błędny.

Rozdział 4

Analiza wyników

Rozdział 5

Podsumowanie

Bibliografia

[1] S. Jafri, S. Piestrak, O. Sentieys, S. Pillement. Design of the coarse-grained reconfigurable architecture dart with on-line error detection. *Microprocessors and Microsystems*, 38(2):124–136, 2014.

[2] Wikipedia contributors. Arytmetyka modularna. Accessed: 2025-11-04.