Poufność i bezpieczeństwo

Poufność (security) – zapewnienie tajemnicy informacji – kryptografia

- cel: zapewnienie poufności i bezpieczeństwa

Bezpieczeństwo (safety) – zapewnienie odporności na zagrożenia

Spolegliwość (dependability) – zapewnienie wykonania zadania
Gotowość (availability) – zapewnienie dostępności środków wykonawczych

- wykrywanie niesprawności sprzętu
- sygnalizowanie błędów transmisji

Niezawodność (reliability) – zapewnienie satysfakcjonującej obsługi błędów

- wykrywanie niesprawności sprzętu
- sygnalizowanie błędów wykonania
- sygnalizowanie błędów transmisji:
  - błędy adresowania – naruszenie reguł dostępu lub adres „donikąd"
  - błędy danych – kody korekcyjne
  - błędy arbitrażu – limitowanie czasu potwierdzenia
- minimalizacja ryzyka błędów

Aspekty niezawodności

Poprawienie niezawodności wymaga wprowadzenia do systemu/urządzenia zapasu (nadmiaru) (ang. *redundancy*):

- informacji – kodowanie umożliwiające sprawdzenie poprawności danych
- czasu – zapas czasu potrzebny do wykonania czynności kontrolnych
- sprzętu – dodatkowe urządzenia niezbędne do przeprowadzenia kontroli

Przykład: bit parzystości:

- zapas informacji – dołączenie bitu parzystości
- zapas czasu – czas potrzebny do weryfikacji parzystości
- zapas sprzętu – wielowejściowa bramka XOR

Oczekiwane efekty działań pro-niezawodnościowych:

- zapobieganie błędom
- przetwarzanie pomimo uszkodzeń (ang. *Fault-Tolerant Computing, FTC*)
  - wykrywanie błędów (ang. *error check*) – kody detekcyjne
  - poprawianie błędów (ang. *error correction*) – kody korekcyjne
  - rozbudowa struktur wykonawczych – wprowadzenie redundancji
  - implementacja metod i algorytmów zapewniających poprawne obliczenia

Niezawodność transmisji i przechowywania danych

Niezawodność transmisji

- wykrywanie błędów
  - bity parzystości
  - kody CRC
- identyfikacja błędów permanentnych i chwilowych
  - powtórzenie transmisji
- korekcja błędów
  - kody Hamminga
  - kody BCH
  - kody Reeda-Solomona

Niezawodność danych

- kody korekcyjne i detekcyjne w modułach pamięci (ang. *SEC/DED*, *Single Error Correction, Double Error Detection*) – rozszerzone kody Hamminga
- powielanie operacji na danych (ang. *mirroring*)
- paskowanie danych i powielanie nośników (RAID)

Kody detekcyjne i korekcyjne

Odległość Hamminga – minimalny odstęp dowolnych słów kodu zapewniający wykrycie błędów określonej krotności: najmniejsza liczba pozycji (bitów lub znaków) słowa kodu, które są różne dla dowolnie wybranych słów kodu

Kody detekcyjne

Aby kod miał zdolność wykrycia p błędów, odległość Hamminga musi być większa od p (równa co najmniej p+1)

Kody korekcyjne

Aby kod miał zdolność korekcji p błędów, odległość Hamminga musi być większa od 2p (równa co najmniej 2p+1)

Kody blokowe – każdy blok m znaków (bitów) informacyjnych jest rozszerzony do m+k znaków, niezależnie od innych bloków

kody systematyczne: znaki kontrolne są dołączane do znaków informacyjnych

Kody splotowe – każdy ciąg m bieżących i $t \cdot m$ poprzednich znaków informacyjnych jest zamieniany na m+k znaków

Syndrom błędu

Wykrycie błędu można zrealizować czasochłonną metodą przeglądu zupełnego albo na podstawie objawów (syndromów) błędu. Syndromy mogą być tworzone przez przekształcenia kodu.

By możliwa była korekcja dowolnego błędu, każdy błąd powinien mieć unikatowy syndrom. Nie jest to możliwe, więc najczęściej korekcja błędów jest ograniczona do błędów najbardziej prawdopodobnych. Jest to tzw. d*ekodowanie o największej wiarygodności* (ang. *Maximum Likelihood Decoding, MLD*)

Liczba różnych stanów słowa kodu o długości n znaków, zawierających co najwyżej t błędów wynosi $1+C(1,n)+\ldots+C(t,n)$, gdzie $C(i,n)$ jest liczbą rozmieszczeń i wśród n.

Ograniczenie Hamminga (ang. *Hamming bound*)
Jeśli k spośród n znaków słowa pochodzących z alfabetu q-znakowego są znakami informacyjnymi ($k<n$), to koniecznym warunkiem możliwości poprawienia każdego z tych błędów jest (w kodach binarnych $q=2$):

$$1+C(1,n)+\ldots+C(t,n)\leq q^{n-k}.$$

Kody liniowe i kody cykliczne

Jeśli na każdej pozycji słowa informacyjnego **m** mogą występować dowolne znaki z ograniczonego zbioru znaków (np. {0,1}) i jest ustalona reguła przekształcenia sumy dowolnych znaków na inny znak (np. XOR bitów), to suma dowolnych słów kodu jest słowem kodu, a taki kod nazywa się kodem liniowym

Jeśli znaki kontrolne dołączane do słowa kodu liniowego **m** są tworzone przez przekształcenie liniowe $f(\mathbf{m})$, to uzyskany kod $\mathbf{m}|f(\mathbf{m})$ jest też kodem liniowym.

Kod cykliczny $\mathbb{C} \subset 2^n$

$$\mathbf{a} = \mathbf{a}_{(0)} = [a_{n-1}, a_{n-2}, \ldots, a_1, a_0] \in \mathbb{C} \Rightarrow \mathbf{a}_{(i)} = [a_{n-1-i}, a_{n-2-i}, \ldots, a_1, a_0, a_{n-1}, \ldots, a_{n-i}] \in \mathbb{C}$$

(*przesunięcie cykliczne* znaków słowa kodu daje wyniku słowo kodu)

Kod cykliczny jest kodem liniowym.

Przekształcenie słowa informacyjnego **m** na słowo $\mathbf{a} = \mathbf{m}|f(\mathbf{m})$ kodu cyklicznego polega na dołączeniu znaków (bitów) kontrolnych według ustalonej reguły.

Regułę można opisać za pomocą macierzy kodowania lub wykorzystując opis wielomianowy kodu cyklicznego

Kody cykliczne – reprezentacja wielomianowa

Słowo kodu $\mathbf{a}=\mathbf{a}_{(0)}=[a_{n-1},a_{n-2},\ldots,a_1,a_0]\in\mathbb{C}$ *można reprezentować jako wielomian nad ciałem skończonym, którego współczynnikami są znaki ciągu kodowego*

$$a(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}\ldots+a_2x^2+a_1x+a_0.$$

Jeśli $\mathbf{a}\approx a(x)$ i $\mathbf{b}\approx b(x)$, to $\mathbf{a}\oplus\mathbf{b}\approx a(x)+b(x)$ $(a_i+b_i \bmod 2\approx a_i \text{ XOR } b_i)$

TWIERDZENIE: Jeśli $a(x)\approx\mathbf{a}$, to reszta $a_{(i)}(x)=x^i a(x) \bmod (x^n-1)\approx\mathbf{a}_{(i)}$

DOWÓD:

$$xa(x) \bmod (x^n-1)=a_{n-1}x^n+a_{n-2}x^{n-1}\ldots+a_2x^3+a_1x^2+a_0x \bmod (x^n-1)=$$

$$=a_{n-2}x^{n-1}\ldots+a_2x^3+a_1x^2+a_0x+a_{n-1}(x^n-1+1) \bmod (x^n-1)=$$

$$=a_{n-2}x^{n-1}\ldots+a_2x^3+a_1x^2+a_0x+a_{n-1}\approx \mathbf{a}_{(1)}$$

Kod k-bitowy można jednoznacznie przekształcić w n-bitowy kod cykliczny. Zdolność detekcji i korekcji błędów zależy od liczby $n-k$ bitów kontrolnych.

Wielomian kodu cyklicznego (n,k) jest wielokrotnością generatora kodu $g(x)$

Generator kodu cyklicznego (n,k) musi być podzielnikiem x^n-1 i ma postać:

$$g(x)=x^{n-k}+g_{n-k-1}x^{n-k-1}\ldots+g_2x^2+g_1x+1.$$

Generowanie bitów kontrolnych kodu cyklicznego

Algorytm kodowania

Dany jest k-bitowy ciąg informacyjny $\mathbf{m}=[m_{k-1}, m_{k-2}, \ldots, m_1, m_0]$, któremu $\mathbf{m} \approx m(x) = m_{k-1}x^{k-1} + m_{k-1}x^{k-2} + \ldots + m_1 x + m_0$, i generator $g(x)$ stopnia $n-k$.

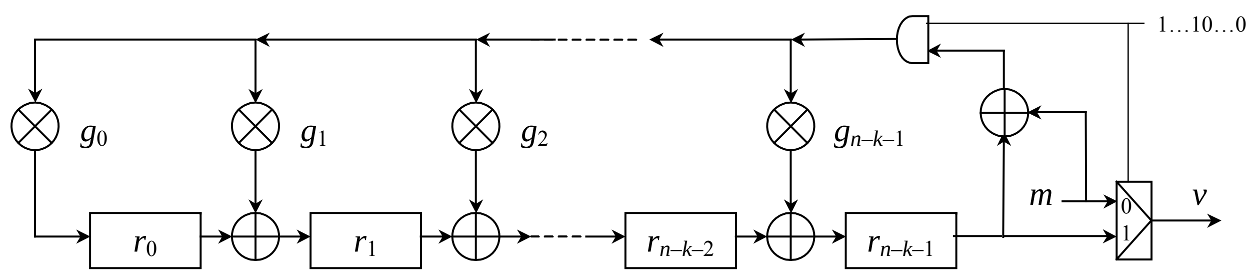
Z twierdzenia o rozkładzie wielomianu wynika, że ($r(x) = x^{n-k}m(x) \bmod g(x)$):

$$x^{n-k}m(x) = q(x)\,g(x) + r(x),$$

więc $x^{n-k}m(x) + r(x) = q(x)\,g(x)$ (jest wielokrotnością $g(x)$ bo $r(x)+r(x)=0$) oraz

$$\mathbf{m} \mid \mathbf{r} = \mathbf{a} \approx a(x) = x^{n-k}m(x) + r(x) = q(x)\,g(x)\,.$$

Dla danego $\mathbf{m}$ ciąg $\mathbf{a}=[m_{k-1}, \ldots, m_1, m_0, r_{n-k-1}, \ldots, r_1, r_0]$ jest słowem kodu (n,k).



Schemat generowania kodu cyklicznego (n,k) ($\oplus$ – XOR, $\otimes$ – iloczyn przez g_i)

Dekodowanie kodu cyklicznego

Zniekształcenie bitów a_i, a_j,… słowa kodu **a** można opisać jako sumę typu XOR słowa błędu $\mathbf{e}=[0,\ldots,1_i,0,\ldots,1_j,0,0]$ (zniekształcenie bitów i, j,…) ze słowem kodu **a**.

$$\mathbf{a}\oplus\mathbf{e}\approx a(x)+e(x)$$

Ale $e(x)=q_e(x)\,g(x)+s(x)$, gdzie $s(x)=e(x)\bmod g(x)$ jest wielomianem *syndromu* (objawu) błędu. Taka sama jest reszta z dzielenia $a(x)+e(x)$ przez $g(x)$, bo

$$a(x)+e(x)=[\,q(x)+q_e(x)\,]\,g(x)+s(x).$$

Dekodowanie o największej wiarygodności (ang. *Maximum Likelihood Decoding, MLD*)

- różne błędy $e(x)$ mają takie same syndromy $s(x)$
- konieczne założenie, wystąpił błąd najbardziej prawdopodobny

Kod (n,k) pozwala odróżnić 2^{n-k} *objawów*, w tym *syndrom zerowy* – brak błędu.

Dekodowanie kodu cyklicznego

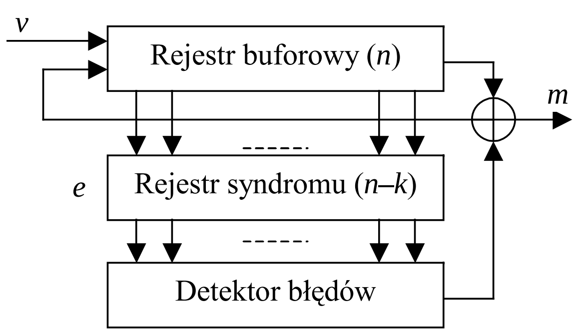
Jeśli w kodzie cyklicznym (n,k) jest nie więcej niż $t \cong (n-k)(\log_2 n)^{-1} < n-k$ błędów, to:

- błędy korygowalne skumulowane na $n-k$ pozycjach kontrolnych kodu (n,k) dają syndrom o wadze (liczbie bitów 1) równej liczbie błędnych bitów
- błędy korygowalne na *dowolnych* pozycjach kodu (n,k) dają syndrom wagi większej od liczby błędnych bitów ciągu kodowego.

Jeśli liczba błędów t nie jest większa od $n-k$ możliwe jest takie przesunięcie cykliczne kodu, aby błędy zostały skumulowane na $n-k$ pozycjach. Zatem:

- jeśli *błędy są skumulowane na pozycjach kontrolnych*, to $e(x)=s(x)$, zatem $q_e(x)=0$, co oznacza, że część informacyjna nie zawiera błędu
- jeśli syndrom błędu miał wagę większą niż t, to cykliczne przesunięcie $x^p r(x)$ odebranego ciągu $r(x)=a(x)+e(x)$ może dać syndrom błędu o wadze mniejszej niż t, co umożliwi korekcję błędu.
- jeśli błędy są skumulowane na $n-k$ kolejnych bitach, to cykliczne przesunięcie wektora odebranego „przesunie błąd" na pozycje kontrolne: $e^{(\rightarrow p)}(x)=s^{(\rightarrow p)}(x)$ i wtedy po korekcji przesuniętego wektora $x^p r(x)$, należy dokonać przesunięcia zwrotnego (cyklicznego w przeciwną stronę).

Dekodowanie uproszczone – dekoder Meggitta



Powtarzaj nie więcej niż n razy lub dopóki powstanie syndrom zerowy:

1. Oblicz syndrom.
2. Jeśli syndrom wskazuje błąd na pozycji skrajnej (x^n), skoryguj ten błąd.
3. Przesuń skorygowany wektor i oblicz nowy syndrom.

Jeśli najpóźniej po n krokach syndrom jest różny od 0, błąd jest nieusuwalny.

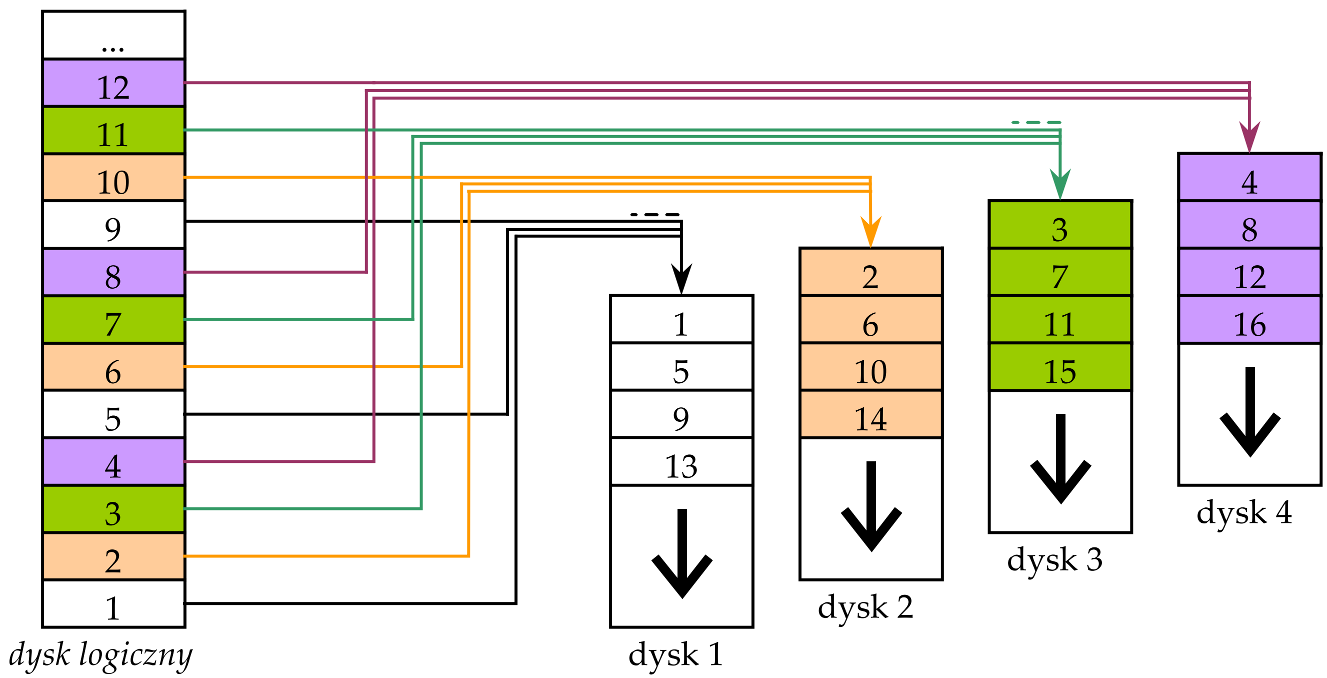
Kod cykliczny $(2^m-1, 2^m-1-m)$ – kod Hamminga. Generatorem kodu (7,3) jest $g(x)=x^3+x+1$, dla kodu (15,4) mamy $g(x)=x^4+x+1$.

W kanałach transmisyjnych i pamięciach masowych powstają błędy grupowe (ang. *burst errors*) – skutek impulsowych zakłóceń *EMG* lub uszkodzenia strefowego. Binarne kody cykliczne nie pozwalają korygować takich błędów. Umożliwiają to kody cykliczne BCH, generowane przez wielomiany o współczynnikach z ciała rozszerzonego (wektory), tzw. *kody Reeda-Solomona*.

Paskowanie danych (ang. striping)

Paskowanie danych (ang. *striping*) – technika rozpraszania danych na wielu dyskach

- umożliwia przyśpieszenie odzyskiwania danych z pamięci dyskowej
- zwiększa przepustowość (ang. *performance*) systemu lub szybkość odczytu
  - o duże jednostki danych – równoległa transmisja z różnych dysków
  - o małe jednostki danych – jednoczesne odczyty przez różne procesy



Macierze dyskowe RAID

Ryzyko błędu w transmisji dużej liczby informacji jest znaczące, np. jeśli ryzyko błędnej transmisji bitu jest równe 10^{-8}, to ryzyko błędu w transmisji serii 2MB wynosi około $2 \cdot 10^{6} \cdot 10^{-8} = 0{,}02$. Uzasadnione jest zatem stosowanie redundancji. Na ryzyko błędu ma też wpływ struktura informacji.

Popularną realizacją redundantnych pamięci masowych są macierze dyskowe RAID (ang. *Redundant Arrays of Inexpensive/Independent Disks*)

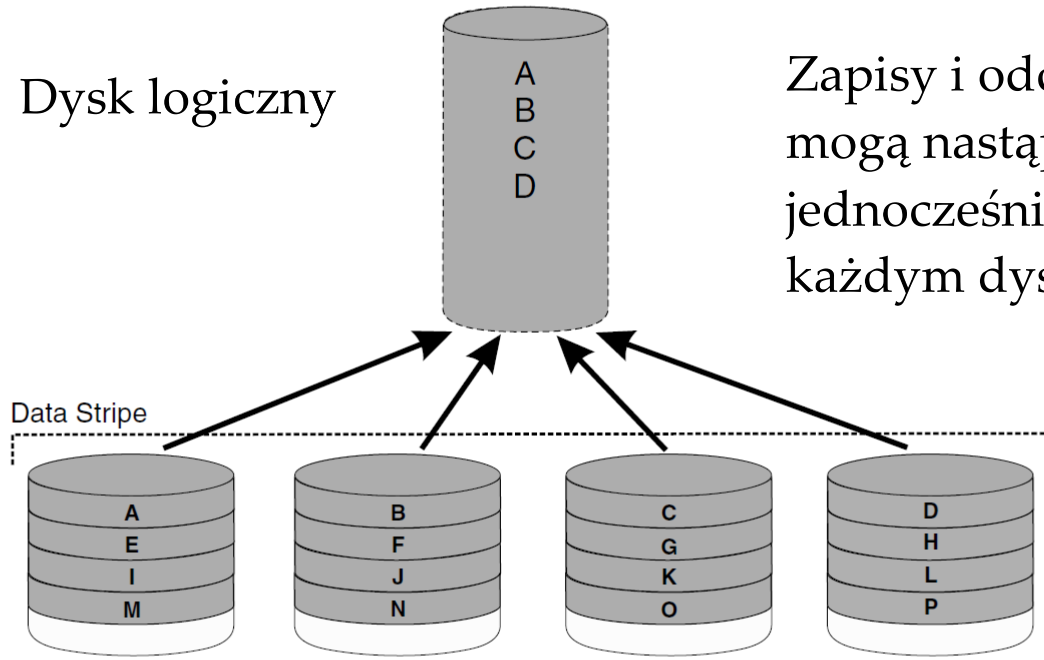
Jednostką informacji w macierzach RAID jest „pasek" (ang. *stripe*) – blok danych stanowiący wielokrotność jednostki danych na dysku (sektor na ścieżce)

Podstawowe konfiguracje to:

RAID-0 – paski pliku na różnych dyskach (ang. *striping*)
RAID-1 – dublowanie z ewentualną inwersją (ang. *duplexing with possible mirroring*)
RAID-2 – paskowanie z kodem korekcyjnym (ang. *error-correcting coding*)
RAID-3 – przeplot pasków z parzystością (ang. *bit-interleaved parity*)
RAID-4 – przeplot pasków z dyskiem parzystości (ang. *dedicated parity drive*)
RAID-5 – przeplot z rozproszeniem parzystości (ang. *block-interleaved distributed parity*)
RAID-6 – podwójna parzystość z rozproszeniem (ang. *independent disks with double parity*)

RAID 0

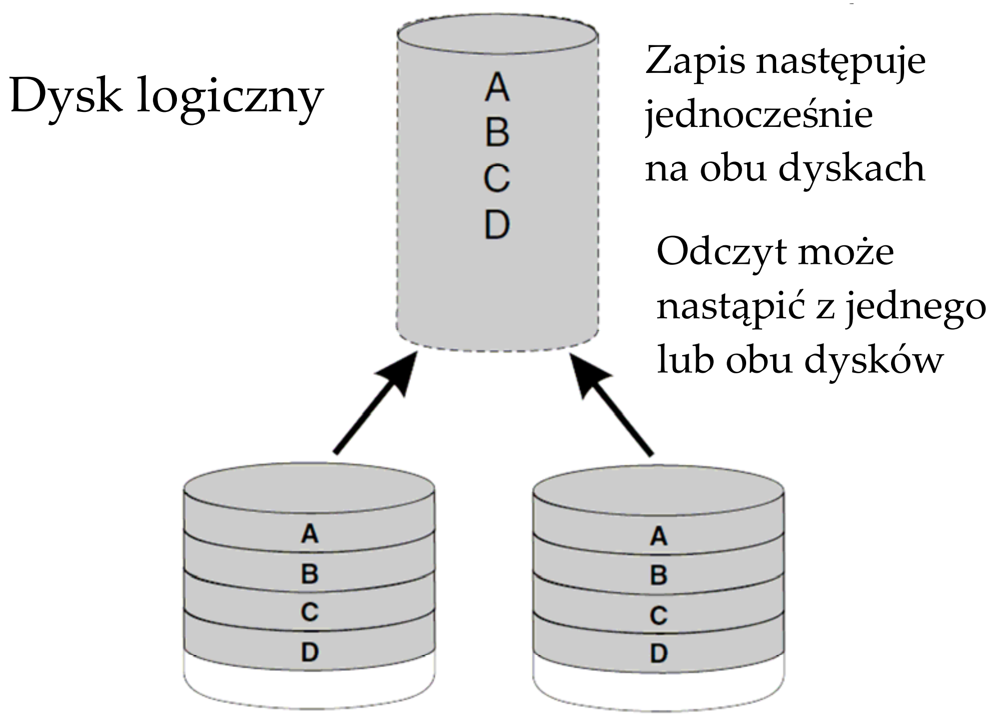
Dysk logiczny



Zapisy i odczyty mogą nastąpić jednocześnie na każdym dysku

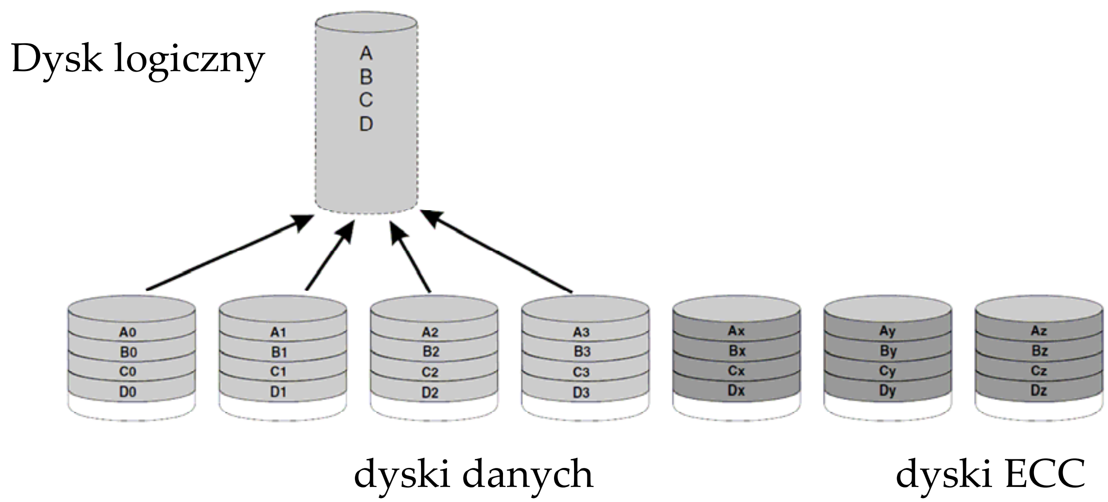
Macierz dyskowa nieodporna na uszkodzenia (ang. *without Fault Tolerance*)
Uszkodzenie dowolnego dysku powoduje utratę wszystkich danych.

Macierz dyskowa RAID 1



Podwajanie (ang. *duplexing*) i powielanie (ang. *mirroring*)
Możliwa podwójna szybkość odczytu ale szybkość zapisu pojedyncza

Macierz dyskowa RAID 2

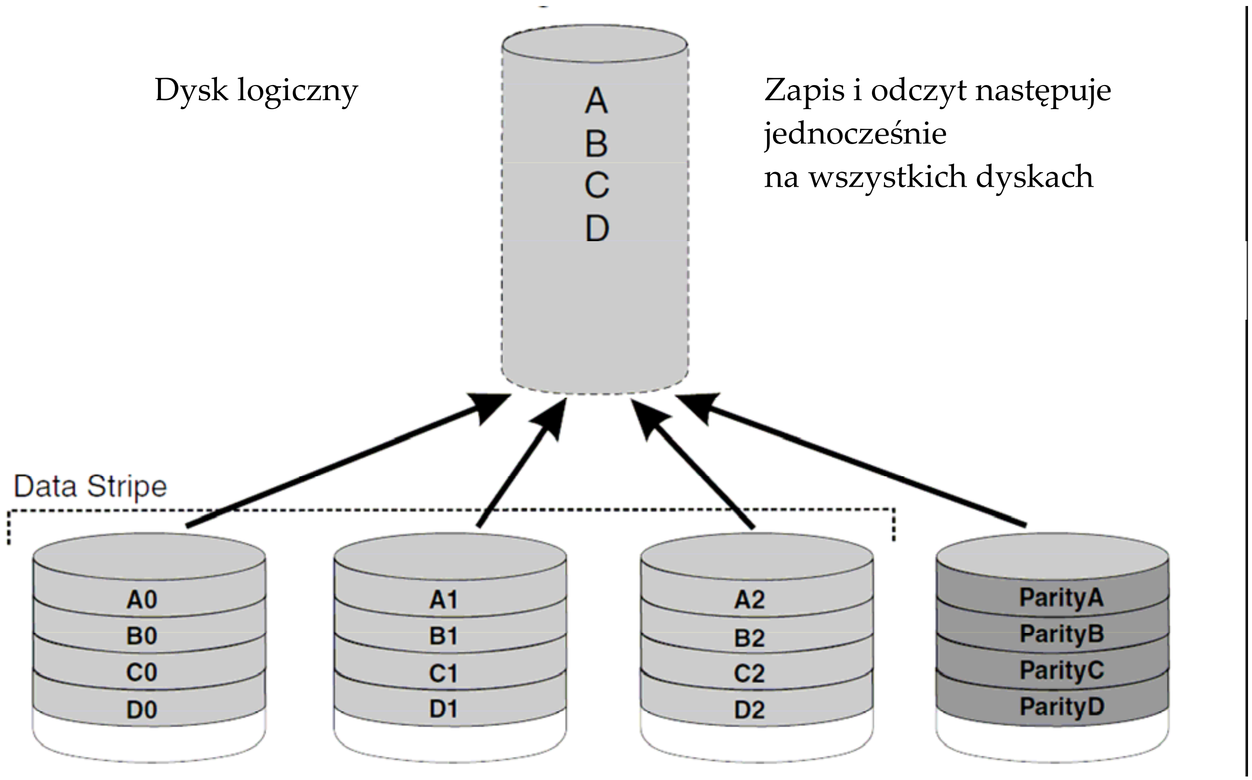


Zastosowanie kodu korekcyjnego (ang. *Error-Correcting Coding, ECC*)

- bardzo rzadkie implementacje
- paskowanie danych na poziomie bitów a nie bloków
- jedno słowo: wszystkie paski A# / B# / ... et

Przykład: kod Hamminga (7,4) jednocześnie wszystkie paski .

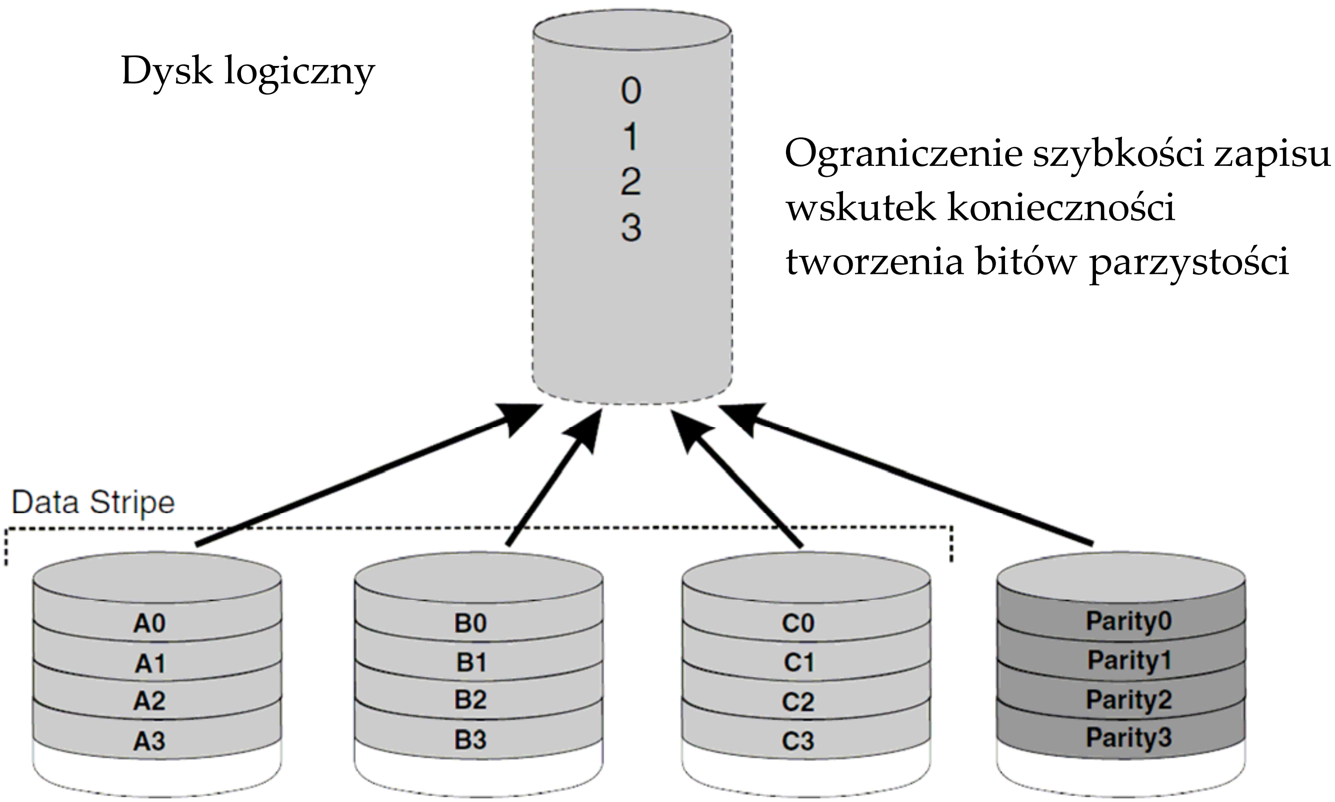
Macierz dyskowa RAID 3



- paskowanie na poziomie bajtów z osobnym dyskiem parzystości
- parzystość z przeplotem bitów (ang. *Bit-Interleaved Parity*)
- rzadko używane – nie obsługuje jednoczesnych żądań dostępu

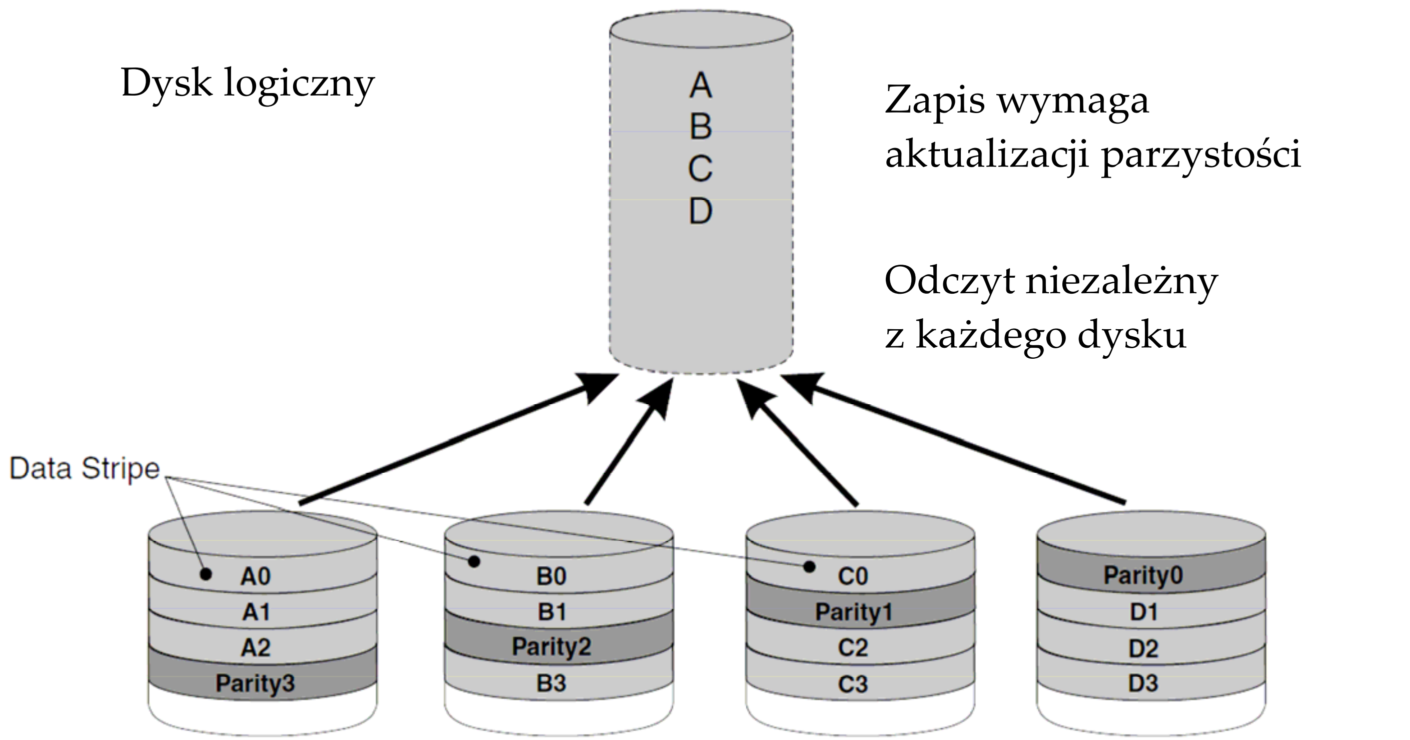
(RAID 7 – RAID 3/4 z pamięcią podręczną dysków)

Macierz dyskowa RAID 4



- osobny dysk parzystości, paskowanie bloków (jak RAID 0) – wada
- dysk parzystości może być użyty do odtworzenia danych z uszkodzonego
- powszechnie używane – nie obsługuje jednoczesnych żądań dostępu

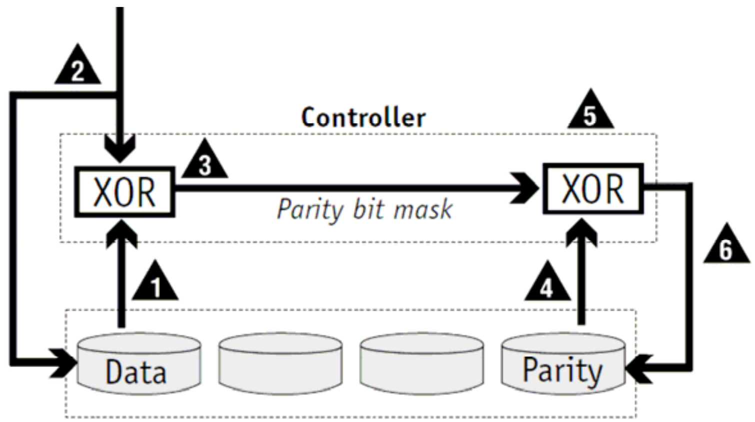
Macierz dyskowa RAID 5



- rozproszona parzystość z przeplotem bloków (ang. *Block Interleaved Distributed Parity*)
- paskowanie danych na poziomie bajtów
- duża przepustowość i odporność na błędy
- jedna z najbardziej popularnych implementacji

RAID 5 – aktualizacja parzystości

Host I/O

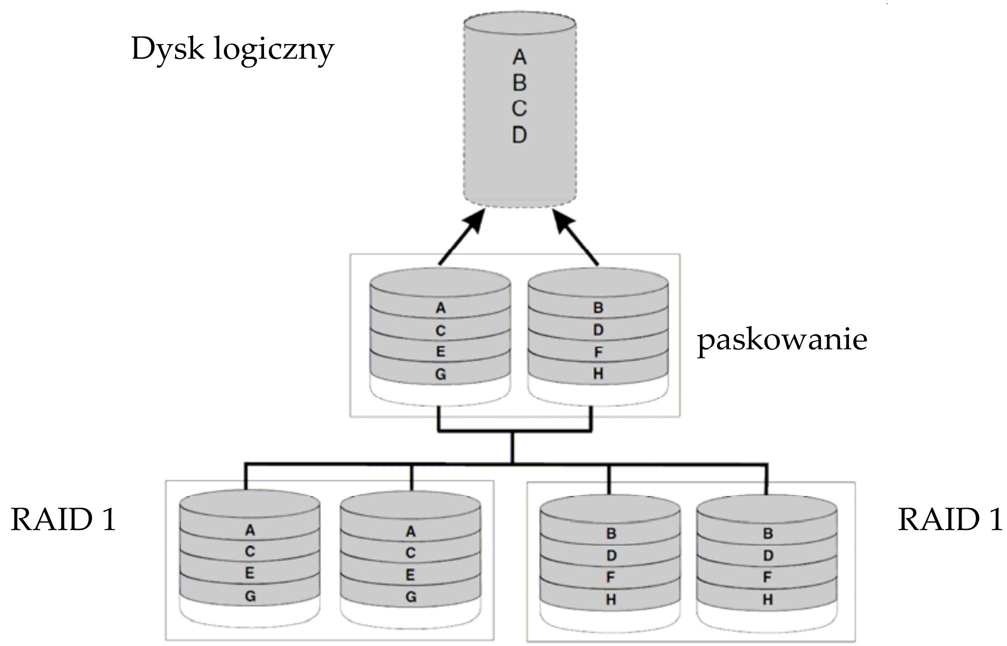


1. Odczytaj poprzednie dane
2. Zapisz nowe dane
3. Maska: XOR nowe i poprzednie dane
4. Odczytaj poprzednią parzystość
5 XOR maskę i poprzednią parzystość
6. Zapisz nową parzystość

RAID 6

Dane są paskowane z podwójną rozłożoną parzystością, co zapewnia odporność danych na uszkodzenia w razie gdy uszkodzi się drugi dysk przed wymianą pierwszego uszkodzonego. Obniżona jest znacznie przepustowość z uwagi na konieczność podwójnego wytwarzania parzystości podczas każdego zapisu.

Macierz dyskowa RAID 10



(1/0) powielanie pasków (ang. *A Stripe of Mirrors*) – paski tworzone jak w RAID 0 są powielane jak w RAID 1

(0+1) paskowanie kopii (ang. *A Mirror of Stripes*) – powielane jak w RAID 1, dwa paski RAID 0 (używane do powielania i współdzielenia danych

Macierz dyskowa RAID 50