|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J.J. Śniadeckich w Bydgoszczy  Wydział Telekomunikacji, Informatyki i  Elektrotechniki  Zakład Informatyki Stosowanej i Inżynierii Systemów | |  | |
| Przedmiot | Układy cyfrowe | | Kierunek/ Tryb | IS /  ST |
| Temat | Zapis liczb ujemnych w systemach ZM, ZU1, ZU2 | |  | |
| Imię i nazwisko: | Nikodem Gębicki | |  | |
| Numer lab. | 3 | Data oddania sprawozdania: | 02.04.2023 | |

# Cel ćwiczenia

# Przebieg

## Zoptymalizuj następujące wyrażenia

1. Podać postać zoptymalizowaną funkcji y dla ∏ (𝟎, 𝟑,𝟓, 𝟗,𝟏𝟐, 𝟏𝟒) oraz dla ∑ (𝟏, 𝟐, 𝟒, 𝟔, 𝟕, 𝟖, 𝟏𝟎, 𝟏𝟏, 𝟏𝟑, 𝟏𝟓)
   * Tabela prawdy

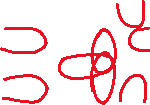
Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

* + Tabela Karnaugh

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie



* + Postać zoptymalizowana

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

* + Układ
  + Testy

1. Podać postać zoptymalizowaną funkcji y dla ∑ (𝟏,𝟐, 𝟒,𝟕, 𝟏𝟏,𝟏𝟓) oraz dla ∏ (𝟎, 𝟑, 𝟓, 𝟔, 𝟖, 𝟗, 𝟏𝟎, 𝟏𝟐, 𝟏𝟑, 𝟏𝟒)
   * Tabela prawdy

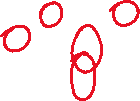
Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

* + Tabela Karnaugh

Obraz zawierający tekst, zrzut ekranu, numer, kwadrat

Opis wygenerowany automatycznie



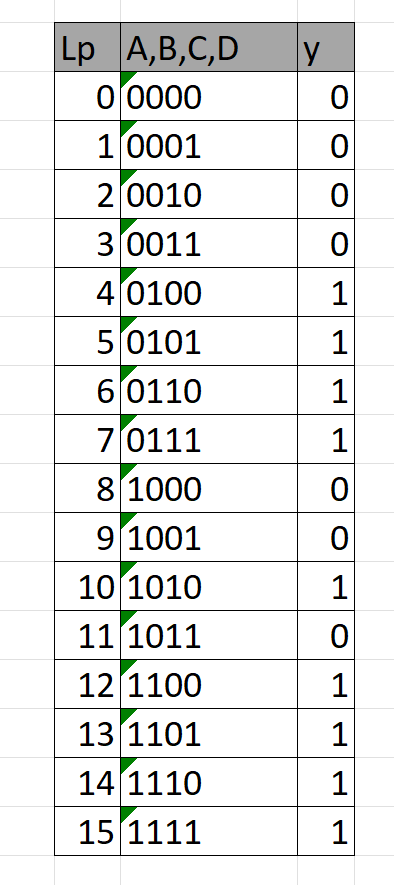
* + Postać zoptymalizowana

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

* + Układ
  + Testy

1. Określić kanoniczną postać ∏ dla zoptymalizowanej funkcji 𝒚 = 𝑨 !𝑩 𝑪 !𝑫 + 𝑩
   * Tabela prawdy



* + Tabela Karnaugh

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie



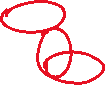
* + Postać zoptymalizowana
  + Układ
  + Testy

1. Określić kanoniczną postać ∑ dla zoptymalizowanej funkcji 𝒚 = (𝑨 + 𝑩)(!𝑨 + 𝑪)(!𝑩 + 𝑫)
   * Tabela prawdy
   * Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

     Opis wygenerowany automatycznie
   * Tabela Karnaugh

Obraz zawierający tekst, zrzut ekranu, numer, kwadrat

Opis wygenerowany automatycznie



* + Postać zoptymalizowana
  + Układ
  + Testy

1. Określić kanoniczną postać ∑ dla zoptymalizowanej funkcji 𝒚 = 𝑨(𝑩 + !𝑪) + !𝑩 𝑫
   * Tabela prawdy

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

* + Tabela Karnaugh

Obraz zawierający tekst, zrzut ekranu, numer, kwadrat

Opis wygenerowany automatycznie

* + Postać zoptymalizowana
  + Układ
  + Testy

# Wnioski

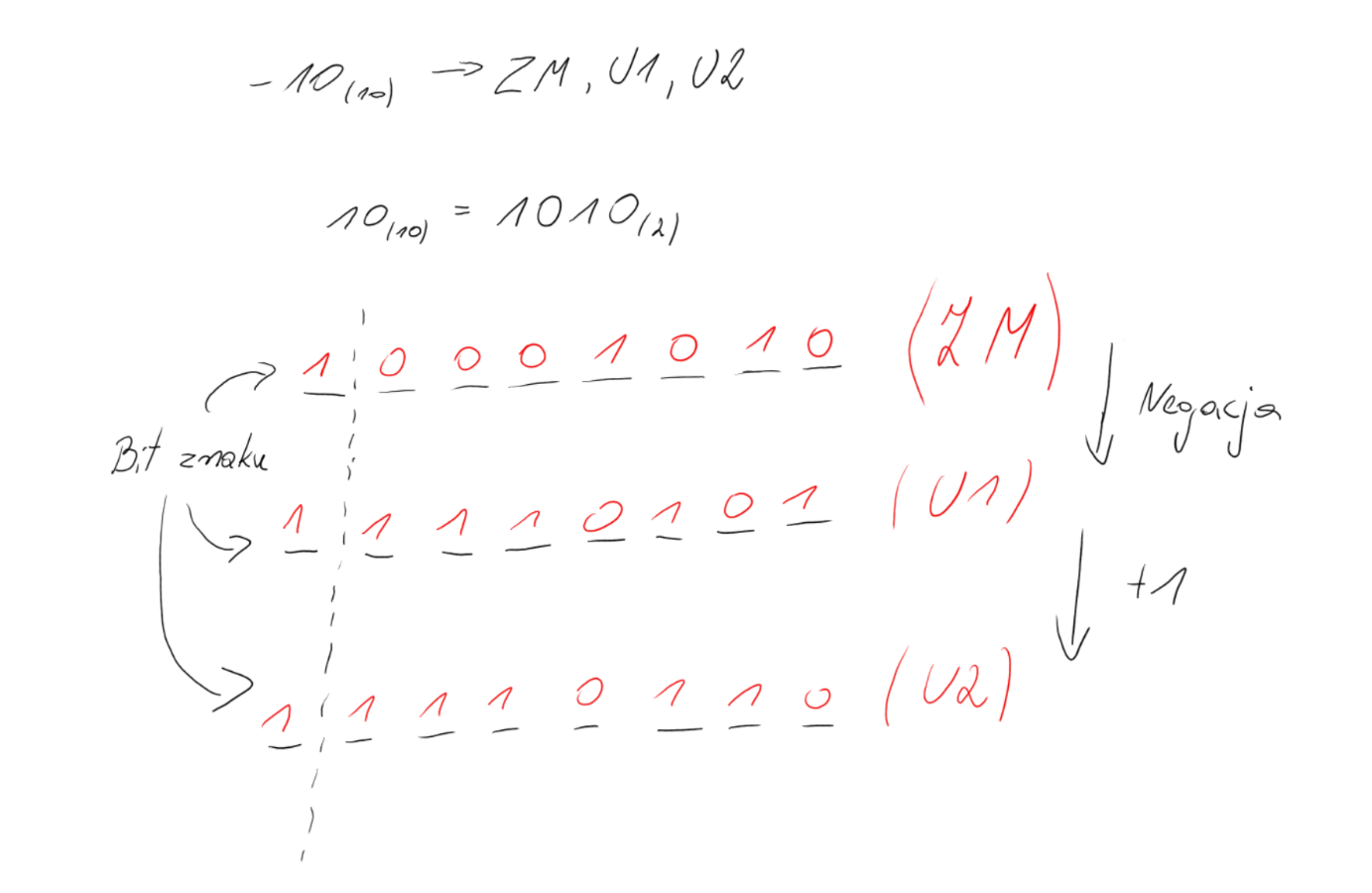
# Podaj zakresy liczbowe dla liczb ujemnych zapisanych w kodach ZM, ZU1, ZU2. Z czego wynikają między nimi różnice?

Zakresy liczb ujemnych dla kodów ZM, ZU1 i ZU2 zależą od długości słowa binarnego używanego w danym kodzie. Dla przykładu, przyjmując długość słowa binarnego na 8 bitów (1 bajt), zakresy liczb ujemnych dla poszczególnych kodów to:

* ZM: od -127 do -1 (11111111 w kodzie ZM oznacza -1, a 10000001 oznacza -127)
* ZU1: od -127 do -1 (11111111 w kodzie ZU1 oznacza -0, a 10000001 oznacza -127)
* ZU2: od -128 do -1 (11111111 w kodzie ZU2 oznacza -1, a 10000000 oznacza -128)

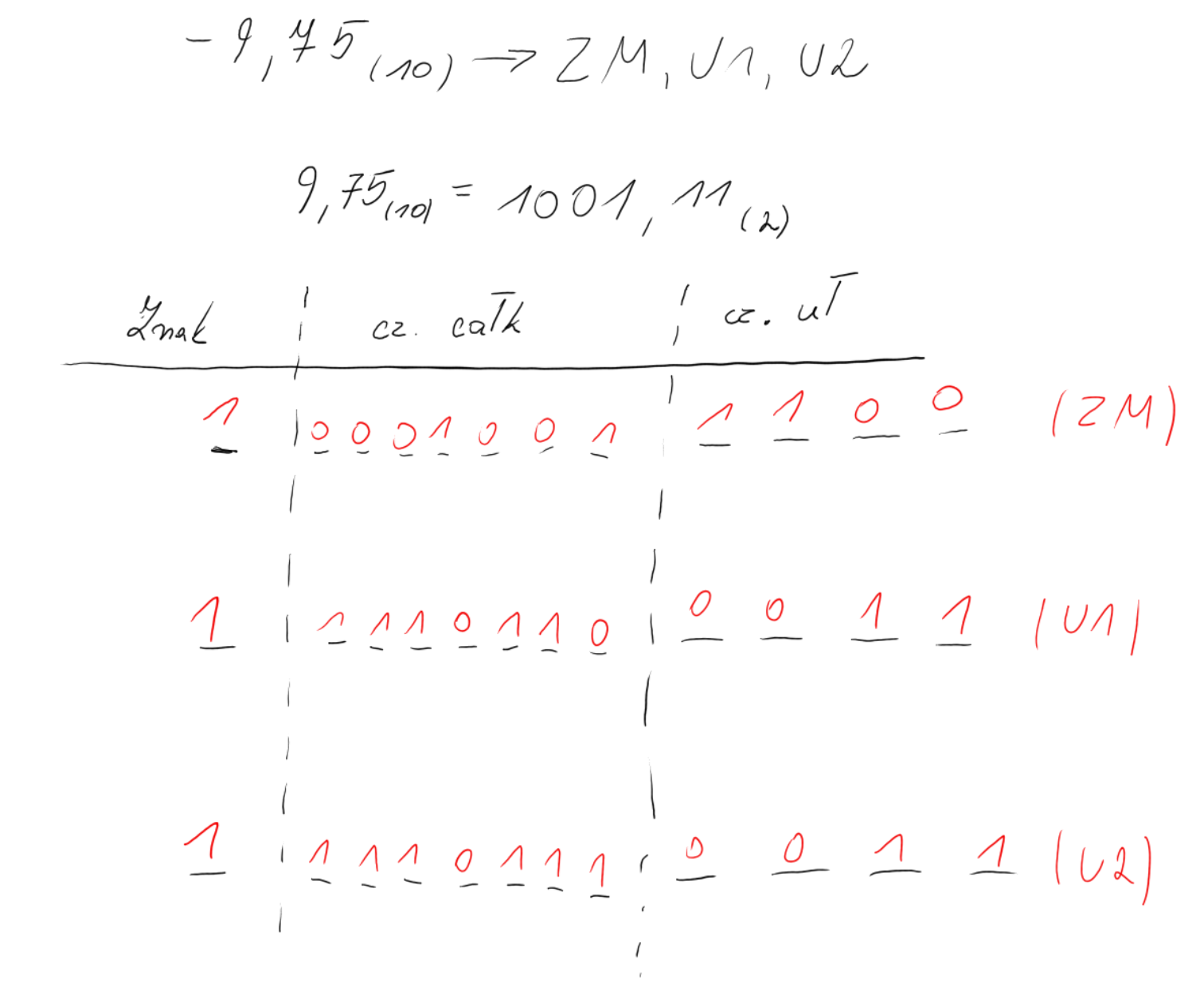
Różnice między zakresami wynikają przede wszystkim z faktu, że w kodzie ZM używany jest jeden bit na oznaczenie znaku, co powoduje mniejszy zakres liczb ujemnych niż w kodach ZU1 i ZU2, gdzie na oznaczenie znaku używa się jednego z bitów wartości liczby. W kodzie ZU1 wartość -0 zajmuje jeden z przedziałów, co powoduje, że liczby ujemne mają zakres od -127 do -1, podobnie jak w kodzie ZM. W kodzie ZU2 wartość -128 zajmuje miejsce w zakresie liczb ujemnych, co powoduje, że jego zakres rozpoczyna się od tej wartości.

# Zapisz liczbę -10 w systemach ZM, ZU1, ZU2 na 8-bitach

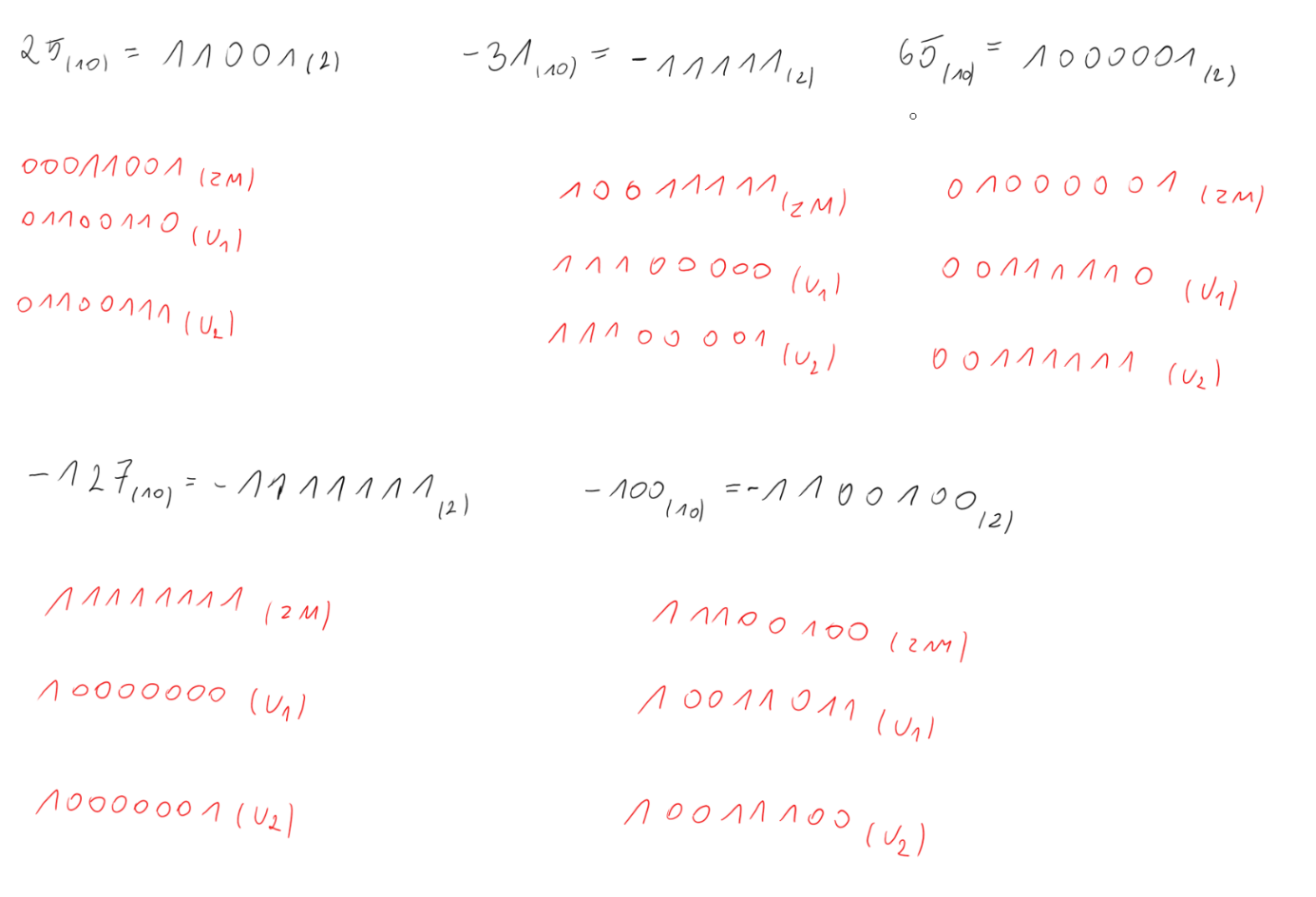


# Wyznacz zapis liczby dziesiętnej -9,75 w systemach ZM, ZU1, ZU2. Docelowy format powinien być 8-mio bitowy. Moduł posiada cztery cyfry ułamkowe.

Liczba jest niemożliwa do zapisania na 8 bitach. Liczba dla zapisu 8 bitów + 4 bity ułamkowe wygląda następująco:



# Wyraź podane liczby dziesiętne w 8-mio bitowym zapisie ZM, ZU1, ZU2 a. 25(10), -31(10), 65(10), -127(10), -100(10)



# Jak wykrywa się i co oznacza „przepełnienie” podczas realizacji obliczeń w poszczególnych systemach zapisu binarnych liczb ujemnych?

Przepełnienie występuje podczas wykonywania operacji arytmetycznych na liczbach binarnych, gdy wynik przekracza zakres reprezentacji liczby w danym systemie liczbowym.

Zakresy dla 1 bajtu:

* W systemie ZM, przepełnienie występuje, gdy wartość bezwzględna wyniku przekracza 127
* W systemie ZU1, przepełnienie występuje, gdy wynik operacji jest równy -128 lub mniejszy
* W systemie ZU2, przepełnienie występuje, gdy wynik operacji jest równy -129 lub mniejszy

Aby wykryć przepełnienie podczas realizacji obliczeń w poszczególnych systemach zapisu binarnych liczb ujemnych, można porównać znak wyniku z znakiem składników i w ten sposób stwierdzić, czy wynik powinien być ujemny czy dodatni. Następnie, jeśli wynik jest ujemny i jego wartość bezwzględna przekracza zakres reprezentacji liczby ujemnej w danym systemie, oznacza to, że wystąpiło przepełnienie.

# Dlaczego to właśnie system zapisu ZU2 przyjął się w systemach komputerowych?

Po pierwsze, system ZU2 umożliwia reprezentację liczb ujemnych i dodatnich za pomocą tej samej reprezentacji binarnej, co jest bardzo wygodne w przypadku wykonywania operacji arytmetycznych. Dzięki temu, operacje na liczbach z różnych znaków mogą być przeprowadzane w taki sam sposób.

Po drugie, system ZU2 umożliwia łatwe wykrywanie przepełnienia, ponieważ wystarczy porównać najbardziej znaczący bit wyniku z najbardziej znaczącym bitem składników. W przypadku, gdy wynik ma inny znak niż składniki, należy jeszcze dokonać korekty wyniku, ale nie ma potrzeby stosowania skomplikowanych algorytmów.

Po trzecie, system ZU2 jest stosunkowo prosty i tani w implementacji. ZU2 może być implementowany przy użyciu układów logicznych, a do jego realizacji nie jest potrzebny specjalny sprzęt ani oprogramowanie.

# Oblicz wartość dziesiętną liczb w kodzie U2: 11110000(U2), 10101010(U2), 01111110(U2), 10000001(U2), 11110111(U2)

