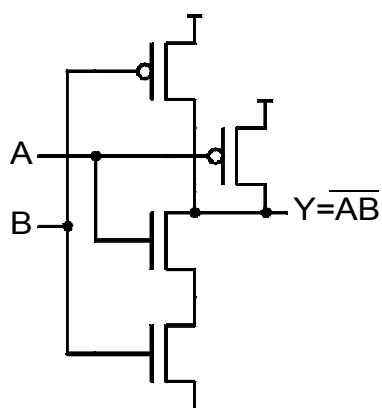


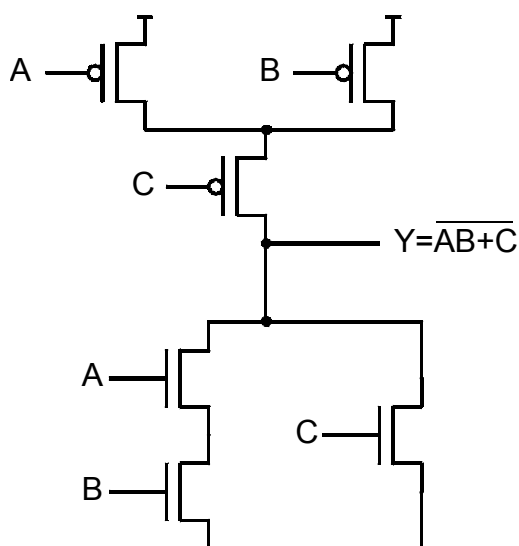
„EUROELEKTRA”
Ogólnopolska Olimpiada Wiedzy Elektrycznej i Elektronicznej
Rok szkolny 2010/2011
Rozwiązania zadań dla grupy teleinformatycznej na zawody II. stopnia

ZADANIE 1

Bramka logiczna w technologii CMOS składa się z sieci tranzystorów PMOS łączącej wyjście z napięciem zasilania i sieci tranzystorów NMOS łączącej wyjście z masą w taki sposób, że dla każdej kombinacji stanów logicznych na wejściach A i B, prąd przewodzi dokładnie jedna z tych sieci. Rysunek przedstawia schemat połączeń dla bramki NAND. Narysować schemat bramki realizującej w podobny sposób funkcję logiczną $Y = \overline{AB + C}$.



Rozwiązanie:



ZADANIE 2

Zaprojektuj sześciobitowy licznik liczący w górę od 33 do 47 z krokiem 2, taktowany zegarem CLK składający się z bramek logicznych oraz synchronicznych przerzutników D. Po dojściu do wartości 47 licznik powinien przyjąć jako następną wartość stan początkowy czyli 33. Sekwencja wartości licznika powinna mieć zatem następującą postać:

33, 35, 37, ..., 45, 47, 33, 35, ...

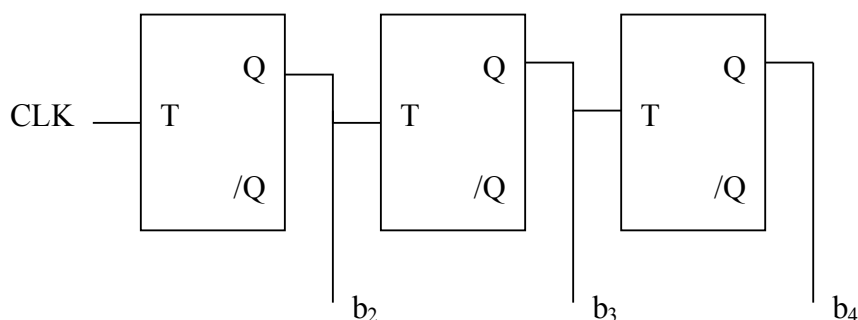
Rozwiązanie:

Zgodnie z założeniem licznik powinien mieć sześć wyjść reprezentujących bity od b_1 do b_6 . Wyjścia powinny przyjmować stany 0 lub 1. Licznik powinien zliczać wartości od 33 do 47 z krokiem 2 czyli przyjmować stany 33, 35, 37, ..., 45, 47, 33, 35, W kodzie binarnym poszczególne stany można zapisać jako:

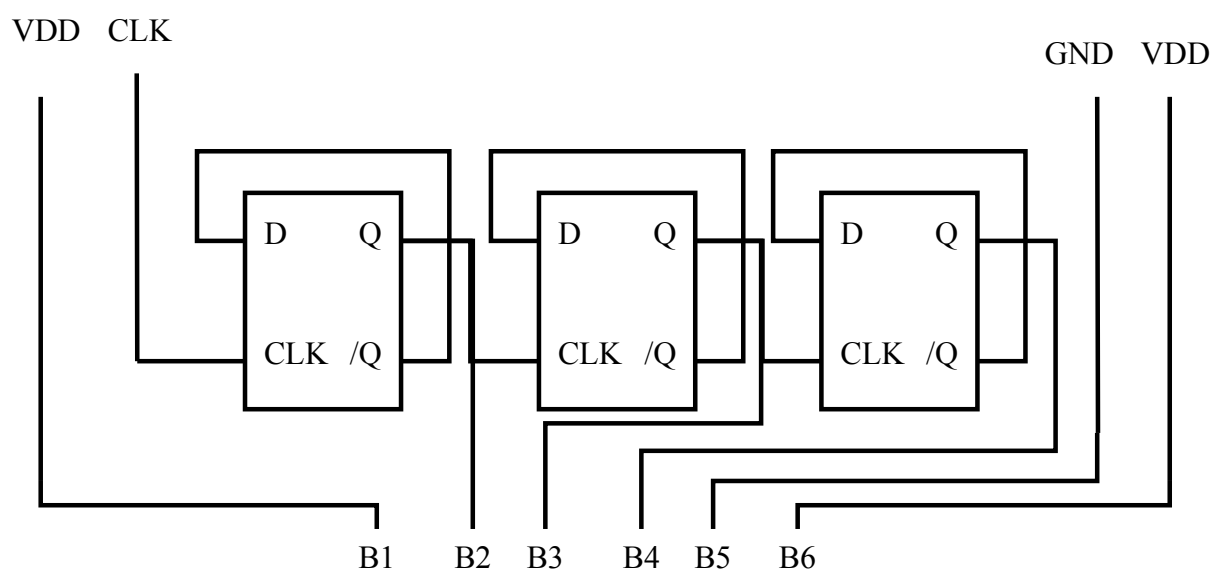
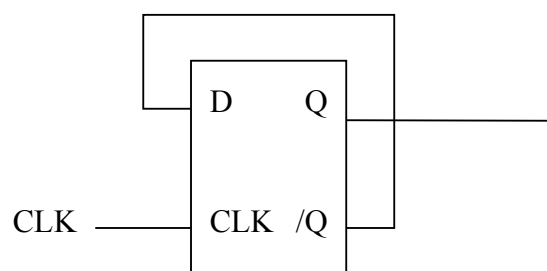
Wartość dziesiętna	Wartość binarna $b_6 b_5 b_4 b_3 b_2 b_1$
33	1 0 0 0 0 1
35	1 0 0 0 1 1
37	1 0 0 1 0 1
39	1 0 0 1 1 1
41	1 0 1 0 0 1
43	1 0 1 0 1 1
45	1 0 1 1 0 1
47	1 0 1 1 1 1

Zauważyć należy że bity 1, 5 i 6 pozostają bez zmian i sygnały je reprezentujące można na stałe zewrzeć z masą (bit 5) lub z zasilaniem (bit 1 i 6).

Zmiana stanów dokonuje się jedynie na pozycjach od b_2 do b_4 . Pomijając sygnały reprezentujące bity b_1 , b_5 , b_6 zadanie upraszcza się do projektu licznika od 0 do 7. Licznik taki można zrealizować za pomocą trzech przerzutników T (toggle) jako:



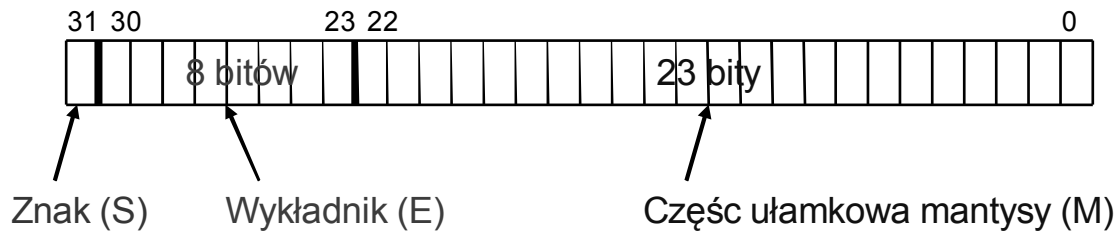
Zgodnie z poleceniem dostępny jest jedynie przerzutnik D. Przerzutnik T można zbudować z przerzutnika D łącząc zanegowane wyjście z wejściem D, tak jak przedstawiono na rysunku poniżej:



W tym rozwiązaniu nie trzeba korzystać z bramek logicznych.

ZADANIE 3

Format liczb zmiennoprzecinkowych pojedynczej precyzji zgodnie ze standardem IEEE jest przedstawiony na rysunku. Zakodować liczbę 27,75 w tym standardzie.



0: Dodatni

1: Ujemny

$$\text{Liczba} = -1^S * (1 + M) \times 2^{E-127}$$

Rozwiązanie:

Liczbę 27,75 należy rozpisać na część całkowitą i ułamkową tj. $27 + 0,75$. Postać binarna każdej z części wynosi odpowiednio 11011_2 i $0,11_2$. Cała liczba zapisana w systemie dwójkowym wynosi $11011,11_2$. Liczba jest dodatnia więc bit znaku S trzeba ustawić na 0. Następnie należy znormalizować zapis liczby binarnej. W tym celu należy przesunąć przecinek w lewo o 4 pozycje tak, aby binarna część całkowita była równa 1. Przy każdym przesunięciu należy odpowiednio zwiększyć o jeden wykładnik liczby 2 przez którą, mnożymy przesuniętą wartość. W znormalizowanym zapisie liczba będzie miała zatem postać $1,10111_2 \cdot 2^4$ a przy założeniu, że część ułamkowa jest zapisana na 23 bitach postać ta będzie następująca:
 $1,10111100000000000000000_2 \cdot 2^4$.

Stąd odczytujemy wartość M jako 10111100000000000000000 .

Wykładnik 4 liczby 2 musi spełniać równanie $4 = E - 127$, zatem wartość pola E będzie wynosić $127 + 4 = 131$ (binarnie 10000011_2). Ostatecznie liczba 27,75 zakodowana jako liczba zmiennoprzecinkowa pojedynczej precyzji będzie miała postać:

01000001110111100000000000000000₂

ZADANIE 4

Mając do dyspozycji ciąg 30 symboli:

DBDB CADA ABDA DABA BAAC FABC ABAC DA

zaproponuj jednoznacznie dekodowalny kod binarny przypisujący ciągi zer i jedynek poszczególnym symbolom (A, B, C, D, F) tak, aby po zakodowaniu długość ciągu nie przekraczała 65 bitów.

Rozwiązanie:

Kod jednoznacznie dekodowalny to kod, w którym każdy ciąg kodowy ma tylko jedną odpowiadającą mu wiadomość. Innymi słowy nie da się odczytać innej wiadomości niż ta, która została zapisana.

W kodowaniu stosuje się różne techniki. W najprostszej zakłada się, że każdy znak koduje się na takiej samej ilości bitów. Stosuje się też inne techniki, w których dopuszcza się, że różne znaki mogą być zakodowane na różnej ilości bitów. Przykładowo, kodowanie ASCII jest kodowaniem ze stałą ilością bitów.

Gdybyśmy chcieli zapisać informację

DBDB CADA ABDA DABA BAAC FABC ABAC DA

w kodzie ASCII to na każdą literę potrzebowalibyśmy aż 8 bitów. W sumie liter jest 30 zatem zapisując powyższą informację w postaci binarnej potrzebowalibyśmy aż 240 bitów.

Gdyby założyć, że wykorzystujemy tylko 5 liter i każdą z nich zakodować za pomocą trzech bitów (5 symboli odpowiada 5 stanom, które w klasycznym rozumieniu można zakodować na przynajmniej 3 bitach) to cały ciąg potrzebowałby już tylko 90 bitów.

Wymagane jest znalezienie rozwiązania, tak aby długość ciągu po zakodowaniu nie przekraczała 65 bitów. Jedynym rozwiązaniem zatem jest zróżnicowanie długość danego słowa kodowego w zależności od częstości wystąpienia danego znaku. Można to zrobić za pomocą jednego z dwóch powszechnie stosowanych w kompresji danych algorytmów – Huffmana lub Shannona-Fano. Algorytmy te są stosowane w prawie wszystkich standardach kompresji.

W przypadku powyższych algorytmów najpierw należy policzyć liczbę poszczególnych symboli, tj:

A – 12

B – 7

C – 4

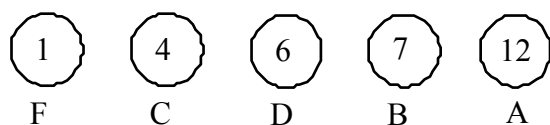
D – 6

F – 1 .

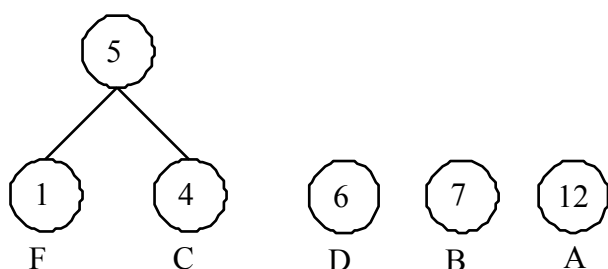
Algorytm Huffmana polega na przyporządkowaniu symbolom najczęściej występującym najkrótszego słowa kodowego i odwrotnie symbolom najrzadziej występującym najdłuższego słowa kodowego. Wtedy długość bitowa całej informacji skróci się. Chcąc zatem zakodować literę A na jednym bicie uzyskamy najlepszy współczynnik kompresji. Oczywiście nie można zakodować wszystkich symboli na jednym bicie, dlatego pozostałe symbole wymagają większej liczby bitów.

Algorytm Huffmana zakłada, że ułożymy wszystkie symbole w kolejności częstości ich wystąpień od najmniejszej do największej (symbol najrzadziej występujący będzie po lewej stronie, zaś najczęściej po prawej).

W przypadku naszego przykładu będzie to wyglądać tak jak pokazano poniżej (wewnątrz okręgu pokazano liczbę wystąpień danego symbolu w ciągu):

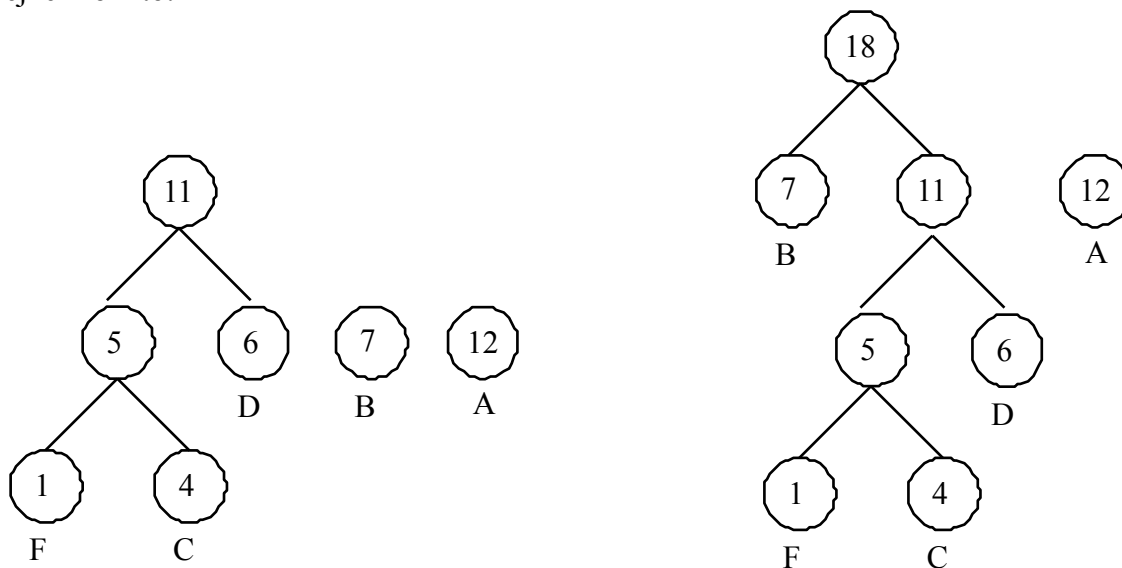


Następnie dwa skrajnie lewe symbole zastępujemy jednym wspólnym węzłem, którego częstość wystąpienia będzie sumą wystąpień tworzących go symboli. Na liście symboli pozostanie zatem $n - 1$ symboli, gdzie n to liczba symboli przed dokonaniem połączenia.

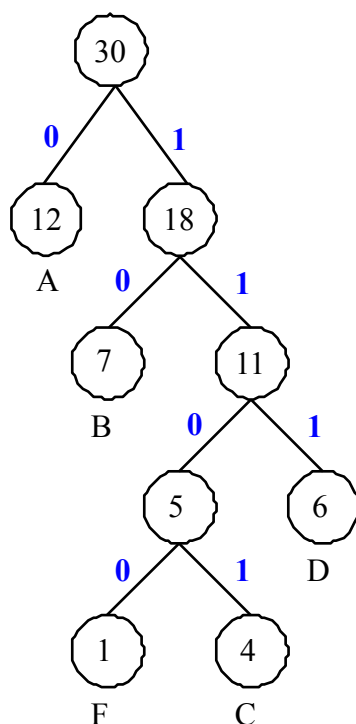


Nowy węzeł należy traktować jako zwykły symbol. Nową listę należy znów przesortować w kolejności niemalejącej i znów dwa skrajnie lewe symbole zastąpić nowym węzłem. Algorytm należy powtarzać do wyczerpania wszystkich symboli (pozostanie tylko jeden nowy węzeł).

Kolejne kroki to:



W wyniku zastosowania algorytmu Huffmana otrzymujemy poniższe drzewo kodowe pozwalające przypisać kod binarny każdemu z 5 symboli występującemu w ciągu. Należy przyjąć dowolną nomenklaturę, np. gałąź skierowana w lewo to „0” zaś w prawo to „1”. Czytając z góry na dół poszczególne jedyńki i zera można dotrzeć do każdego symbolu.



Ostatecznie można zapisać:

Symbol	Kod dla danego symbolu	Długość kodu dla danego symbolu	Liczba wystąpień symbolu	Iloczyn długości kodu i liczby wystąpień symbolu
A	0	1	12	12
B	10	2	7	14
C	1101	4	4	16
D	111	3	6	18
F	1100	4	1	4
			SUMA	64

Sumując iloczyny liczby bitów kodujących dany symbol i krotności wystąpienia danego znaku otrzymujemy długość sekwencji wyjściowej równą 64 bity, czyli o 1 mniej niż było wymagane w zadaniu.

Zakodowany ciąg DBDB CADA ABDA DABA BAAC FABC ABAC DA będzie wyglądał następująco:

1111011110 1101011110 01011110 1110100 10001101 11000101101 01001101 1110

Algorytm Huffmana gwarantuje otrzymanie kodu jednoznacznie dekodowalnego. Podobne rozwiązanie otrzymuje się korzystając z algorytmu Shannona-Fano.

ZADANIE 5

Mamy do dyspozycji przestrzeń adresową 211.57.212.87/xx. Posiadamy trzy podsieci, z których pierwsza zawiera 25 komputerów, druga 55 komputerów, a trzecia 42 komputery. Podać maksymalną wartość xx dla której zadanie ma rozwiązanie oraz adresy i maski każdej z podsieci dla tego przypadku.

Rozwiązanie

Liczba xx występująca przy przestrzeni adresowej 211.57.212.87/xx określa liczbę początkowych jedynek w masce sieci. Im większe xx, tym mniejszy rozmiar przestrzeni adresowej. Oznacza to, że w danej przestrzeni adresowej będzie mniej dostępnych adresów IP dla komputerów oraz dla ewentualnych podsieci. Obecnie pula dostępnych adresów IPv4 jest na wyczerpaniu, w związku z czym należy nią gospodarować oszczędnie i nie projektować sieci, które będą alokowały więcej adresów niż jest to niezbędne.

W celu określenia masek poszczególnych podsieci w pierwszym etapie należy określić liczbę adresów wymaganych w każdej podsieci. Możemy to policzyć, dodając do liczby komputerów w każdej podsieci liczbę trzy, odpowiadającą adresowi podsieci, adresowi rozgłoszeniowemu (ang. broadcast) i adresowi routera (są to adresy, które muszą wystąpić w każdej podsieci). Uzyskamy w ten sposób odpowiednio liczby 28, 58 oraz 45. Maski podsieci można utworzyć jedynie dla rozmiarów podsieci równych całkowitej potędze liczby 2, a więc w tym celu liczby te należy zaokrąglić w górę do odpowiednio 32, 64 oraz 64. Odpowiada to przestrzeni adresowej opisanej za pomocą odpowiednio 5, 6 oraz 6 bitów. Suma rozmiarów podsieci to $32+64+64=160$, co po zaokrągleniu w górę do całkowitej potęgi liczby 2 daje liczbę 256, która określa minimalny rozmiar zadanej przestrzeni adresowej 211.57.212.87. Zatem maska sieci dla tej przestrzeni na ostatnich 8 bitach musi mieć zera czyli w całości jest ona równa 11111111.11111111.11111111.00000000. Liczba jedynek tej maski wynosi 24 więc xx wynosi 24.

Kolejnym etapem jest podział przestrzeni adresowej 211.57.212.0/24 na trzy podsieci o rozmiarach 32, 64 oraz 64 adresów. Należy zauważyć, że opisy 211.57.212.0/24 i 211.57.212.87/24 odnoszą się do tej samej przestrzeni adresowej – 8 ostatnich bitów jest maskowane.

Przykładowe adresy i maski takich podsieci to odpowiednio 211.57.212.0/27, 211.57.212.64/26 oraz 211.57.212.128/26. Podsieci te mieszczą się w założonej przestrzeni adresowej i nie zachodzą na siebie (są rozłączne).

Przy takim podziale na podsieci większość przestrzeni adresowej pozostaje niewykorzystana. W alokowanej przestrzeni adresowej można byliby umieścić w sumie 253 komputery, pamiętając o trzech zarezerwowanych adresach. Bez podziału na podsieci można byliby umieścić wszystkie zadane $25 + 55 + 42 = 122$ komputery w przestrzeni adresowej 211.57.212.0/25.