

Architektury systemów komputerowych

Lista 2

14.X.2009

$x_1 = 10$ (minimum na bdb)

1. Napisz tabelkę funkcji o czterech wejściach stawiając w kolumnie wyjścia dokładnie 13 jedynek i trzy 0 (na „chybił-trafił”). Naturalne wyrażenie w dysjunkcyjnej postaci normalnej reprezentujące tę funkcję ma 13 składników. Przedstaw wyrażenie równoważne w koniunkcyjnej postaci normalnej.
2. Rozważmy funkcje o czterech wejściach, których tabelki mają w kolumnie wyjścia dokładnie 12 jedynek. Wskaż wśród nich taką funkcję, która ma reprezentację w koniunkcyjnej postaci normalnej składającą się z jednego makstermu. Czy wśród tych funkcji jest taka, która ma reprezentację w dysjunkcyjnej postaci normalnej składającą się z jednego mintermu?
3. Pokaż w jaki sposób bramkę XOR można zastąpić układem zbudowanym tylko z bramek NAND.
4. Używając poznanych na wykładzie bramek o dwóch wejściach zbuduj obwód logiczny o trzech wejściach i jednym wyjściu, taki że na wyjściu pojawia się 1 wtedy i tylko wtedy gdy na wejściach podanych jest nieparzysta liczba (czyli w naszym przypadku jedna lub trzy) jedynek. Uogólnij następnie swoją konstrukcję na przypadek n wejść. Układy tego typu nazywamy *generatorami bitu parzystości*. W pewnym sensie są one uogólnieniami dwuwejściowej bramki XOR. Postaraj się skonstruować ten obwód używając tylko bramek XOR.
5. Dane jest wyrażenie:
$$\bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c.$$
Zbuduj układ logiczny równoważny temu wyrażeniu, używając tylko bramek NAND. Postaraj się użyć jak najmniejszej liczby bramek (mogą mieć więcej niż dwa wejścia).
6. Udowodnij, że dla dowolnej funkcji logicznej istnieje reprezentujący ją obwód logiczny zbudowany tylko z bramek NOR.
7. (2 pkt.) Udowodnij, że żadna bramka z dwoma wejściami, oprócz bramek NAND i NOR, nie wystarcza do zbudowania obwodów odpowiadających dowolnym funkcjom logicznym (czyli, że żaden zbiór złożony z jednej, dwuargumentowej funkcji logicznej, z wyjątkiem funkcji NAND i NOR, nie jest funkcjonalnie pełny).
8. Zbuduj układ demultipleksa: układ ma 1 wejście z danymi i pewną liczbę wejść adresowych; wejście powinno zostać skierowane na to wyjście, którego numer jest wskazywany przez linie adresowe. Pozostałe wyjścia mają być równe 0.
9. Zbuduj następującą wersję multipleksa: układ ma 12 wejść, na które patrzymy jak na 6 wejść dwubitowych oraz dwa wyjścia (czyli jedno wyjście dwubitowe). Na wyjście powinna zostać przekazane wejście (para wejść jednobitowych) wskazywana przez wejścia adresowe. Pomyśl, ile powinno być wejść adresowych.
10. Narysuj układ reprezentujący (w najprostszy sposób) wyrażenie $\bar{a}b + ac$. Zaobserwuj zjawisko *hazardu*: wybierz pewien stan wejść (zastanów się jaki), przy którym funkcja przyjmuje wartość 1. Następnie zmień wartość jednego z wejść (zastanów się którego) tak aby funkcja wciąż przyjmowała wartość 1, ale po zmianie wejścia na wyjściu układu na moment pojawiła się błędna wartość 0.
Wskazówka: bramka NOT działa szybko, ale jednak chwilę działa.

11. Zmodyfikuj (tzn. zbuduj układ reprezentujący wyrażenie równoważne) w możliwie prosty sposób układ z poprzedniego zadania, tak aby nie było możliwości wystąpienia hazardu.
12. Zbuduj układ o dwóch wejściach a, b i trzech wyjściach, porównujący swoje wejścia. Na pierwszym wyjściu powinna pojawić się 1 dokładnie wtedy, gdy $a > b$, na drugim, gdy $a < b$, a na trzecim, gdy $a = b$.
13. Wykorzystaj układ zbudowany w poprzednim zadaniu do konstrukcji układu porównującego liczby 4 bitowe, czyli zbuduj układ z ośmioma wejściami $a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0$ i trzema wyjściami. Wyjścia powinny się zachowywać analogicznie jak w przypadku układu porównującego pojedyncze bity. Układy tego typu nazywają się *komparatorami*. Zakładamy, że wejścia są liczbami dodatnimi, reprezentowanymi w naturalnym kodzie binarnym.

Emanuel Kieroński