Architektury systemów komputerowych

Lista 6
$x_6 = 6$ (minimum na bdb)

1. W tym zadaniu rozważamy liczby dodatnie reprezentowane w naturalnym kodzie binarnym. Do dyspozycji masz układy dwóch rodzajów:

   - układ przyjmujący na wejściu dwie liczby dwubitowe i zwracający ich czterobitowy iloczyn
   - układ sumatora otrzymujący dwie dwubitowe liczby i dodatkowo wstępne przeniesienie, zwracający dwubitową sumę oraz bit przeniesienia.

   Używając pewnej liczby układów powyższego typu skonstruuj układ przyjmujący na wejściu dwie liczby czterobitowe i zwracający ich ośmiobitowy iloczyn. Nie wolno używać żadnych dodatkowych bramek logicznych.

2. Zasymuluj działanie algorytmu Booth'a na następujących przykładach:

   (a) $0101 \cdot 1001$
   (b) $01110 \cdot 01000$
   (c) $10100 \cdot 11111$
   (d) $11100011 \cdot 00111100$

3. Znajdź opis jakiegoś algorytmu maszynowego dzielenia całkowitego liczb n-bitowych w naturalnym kodzie binarnym, dla którego liczba wykonywanych operacji jest rzędu n^2. Omów jego implenentację sprzętową i przedstaw działanie na przykładach.

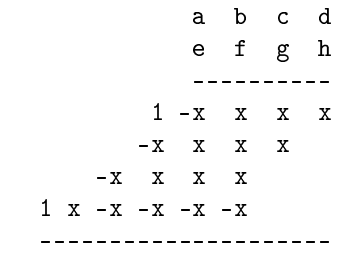
4. Na wykładzie przedstawiłem dwa algorytmy mnożenia maszynowego: dla liczb bez znaku w naturalnym kodzie binarnym (notatki do wykładu 6, rys. 1) i dla liczb w reprezentacji uzupełnień do 2 – alg. Bootha (notatki do wykładu 6, rys. 3). Przypomnij ich działanie i wytłumacz dlaczego w drugim z nich nie potrzebujemy rejestru C, pamiętającego przeniesienie z ostatniego sumatora.

5. Ile operacji dodawań/odejmowań wykona pierwszy z algorytmów z poprzedniego zadania, a ile drugi, w przypadku, gdy mnożymy przez następującą liczbę o długości n bitów:

   (a) 00000000...00000000
   (b) 11111111...11111111
   (c) 01010101...01010101
   (d) 00110011...00110011
   (e) 10000000...00000000
   (f) 01111111...11111111

6. Przedstaw na poziomie bramek logicznych i przerzutników realizację układu mnożącego liczby binarne z wykładu 6, rysunek 1. Przyjmij, że dane wejściowe są 3-bitowe. Możesz użyć gotowego sumatora 3-bitowego. Układ dostaje dodatkowo niewidoczny na rysunku sygnał zegarowy, pozwalający synchronizować wykonywanie dodawań i przesuwanie.

7. * (2 pkt.) Poniżej przedstawiony jest schemat array multipliera dla reprezentacji uzupełnień do 2: Wyliczamy iloczyny odpowiednich bitów wejściowych (znaki `x`), nietkóre z nich negujemy (znaki `(-x)`, dopisujemy dwie jedynki, a następnie wykonujemy dodawania (tak jak w rozwiązaniu dla liczb bez znaku). Uzasadnij, że ten schemat jest poprawny.



8. * (2 pkt.) Układ array multipliera można przystosować do wykonywania serii mnożeń *potokowo*, tzn. w taki sposób, że wylicznanie kolejnych iloczynów nakłada się na siebie. W tym zadaniu przyjrzymy się bliżej tej technice (zbudujemy *pipelined array multiplier*). Nasz układ sterowany będzie sygnałem zegarowym. W $i-tym$ ($i = 1, \ldots, l$) cyklu zegara na dwa n-bitowe wejścia układu podawana jest para liczb x_i, y_i. W $(k+i)$-tym cyklu na $2n$-bitowym wyjściu układu powinien się pojawić wynik mnożenia $x_i \cdot y_i$ (k powinno być równe około $2n$).

   Aby zrealizować opisane zadanie musisz podzielić układ array multipliera na „warstwy" (warstw jest około $2n$). Obliczenia kolejnych iloczynów powinny „spływać" od warstwy najwyższej do najniższej, przy czym w momencie, gdy obliczanie pierwszego iloczynu opuszcza pierwszą warstwę, to wchodzi do niej obliczanie drugiego iloczynu; w kolejnym takcie zegara obliczanie pierwszego iloczynu przechodzi do warstwy trzeciej, drugiego do drugiej, a trzeciego wchodzi do pierwszej, itd. Pomiędzy warstwami należy wstawić przerzutniki, które będą zapamiętywały wyniki pośrednie. Od pewnego momentu obliczeń układ będzie jednocześnie pracował nad około $2n$ iloczynami. Zatem obliczenie l iloczynów będzie trwało mniej więcej $l + 2n$, a nie $l \cdot 2n$ jak w przypadku układu niepotokowanego (w tym stwierdzeniu pomijam fakt, że dodanie przerzutników sterowanych sygnałem zegarowym spowolni nieco pracę układu w przypadku pojedynczego obliczenia).

   Przedstaw szczegóły zarysowanego rozwiazania dla liczb 4-bitowych ($n = 4$). Przedstaw konstrukcje układu na poziomie bramek logicznych i przerzutników.

9. Przypomnij pomysł „pełnego podglądu przeniesienia" (carry-lookahead) budując według tej zasady sumator 4 bitowy. Używając go jako „bloku" zbuduj sumator 128 bitowy. Zakładając, że każda bramka logiczna powoduje opóźnienie 1 (z wyjątkiem bramki NOT, której opóźnienie można pominąć) wylicz całkowite opóźnienie przedstawionego układu. Podobnie jak na wykładzie, rozważ teraz funkcje propagowania i generowania wyższego poziomu i zastosuj je ponownie do konstrukcji sumatora 128-bitowego. Wylicz całkowite opóźnienie uzyskanego układu.

Emanuel Kieroński