Introduction to Digital Computers - Spring 1999

Assignment No. 3

Course homepage: http://www.eng.tau.ac.il/~guy/Digital_Computers99/dc_home.html

Messages:

1. Firm Deadline: May 5th - before the beginning of the lecture.
2. A group of less than 3 students may submit answers to five questions to get a full grade.
3. Course's homepage provides instructions on how to join the course's mailing list. Please join before April 28th because I need to confirm such additions manually.

Questions:

1. Consider the recurrence equation:

$$f(n) = \begin{cases} b & \text{if } n \leq 1 \\ a \cdot f(n/c) + bn & \text{if } n > 1 \end{cases}$$

Prove that if $n = c^k$, for a positive integer k, then

$$f(n) = \begin{cases} O(n) & \text{if } a < c \\ O(n \log n) & \text{if } a = c \\ O(n^{\log_c a}) & \text{if } a > c \end{cases}$$

2. Define:

$$\begin{aligned} T_n &= \{-2^{n-1}, -2^{n-1}+1, \ldots, 2^{n-1}-1\} \\ [x[n-1:0]] &= -2^{n-1} \cdot x[n-1] + \langle x[n-2:0] \rangle \end{aligned}$$

and let two_n denote the function $two_n : T_n \to \{0,1\}^n$ which satisfies:

$$\forall j \in T_n : [two_n(j)] = j$$

(a) Let $two_n(j) = x[n-1:0]$. Prove that:

$$x[n-1] = 0 \text{ if and only if } j \geq 0$$

(b) Prove that for every $j \in T_n$:

$$\langle two_n(j) \rangle = \begin{cases} j & \text{if } j \geq 0 \\ 2^n - |j| & \text{if } j < 0 \end{cases}$$

3. The *fanout* of a net is the number of gate inputs that are fed by the net. In many technologies the fanout is limited. In such cases, a gate-output that needs to be fed to many gate-inputs must pass through a "buffer" that deals with amplifying and restoring the voltage. A d-buffer is a gate that has 1 input and d outputs, and all the output values equal the input value.

   Suppose that the fanout is limited by 2, and that one may use 1-buffers the cost of which equals 1 and the delay of which equals 1.

   Suppose that a gate-output has to be fed to n gate-inputs. Suggest and optimal way to distribute the signal using 1-buffers? Prove that your suggestion is optimal.

4. Suppose that the fanout is limited by 2, and that one may use 1-buffers the cost of which equals 1 and the delay of which equals 1.

   (a) Rewrite the recurrence equations for the delay and cost of the two types of decoders described in class. (Assume that in the second decoder $k = \lfloor n \rfloor$).

   (b) Compute the cost and delays according to the Motorola technology of the two types of decoders for $2 \leq 2^n \leq 128$ with and without the fanout constraint.

5. Draw the $PPC(16)$ circuit using only $*$-gates (that is, unfold the recursion).

6. The description of $CLA(n)$ given in class did not address the issue of how the alphabet $\{0, 1, 2\}$ is encoded. Three encodings are given in the Table below:

symbol	binary encoding	"hot one" encoding	"g & p" encoding
0	00	001	00
1	01	010	01
2	10	100	10 or 11

   (a) Design a $*$-gate for every encoding. Compute the cost and delay of each gate according to the Motorola technology.

   (b) Compute the cost and delay of $CLA(n)$ with respect to each encoding according to the Motorola technology for $n = 8, 16, 32, 64, 128$.

   (c) (5 points bonus) Use 1-buffers to limit the fanout to 2. Modify the design so that the fanout is 2. Re-compute the cost and delay of $CLA(n)$ for the same values of n.

7. Consider the problem of "move to front" defined as follows:

   Input: An alphabet Σ and symbols $z, x_0, x_1, \ldots, x_n \in \Sigma$. The alphabet Σ contains a special symbol λ (which denotes "end of list"). The input satisfies:

   - $x_n = \lambda$;
   - the symbols $x_0, x_1, \ldots, x_{n-1}$ are distinct and do not equal λ.
   - $z \in \{x_0, x_1, \ldots, x_{n-1}\}$

   ($x_0, \ldots, x_n$ denotes the "list", z is the list element which is supposed to be moved to the front of the list).

   Output: The symbols $y_0, y_1, \ldots, y_n$ defined by

$$
\begin{aligned}
y_0 &= z \\
y_{i+1} &= \begin{cases} x_i & \text{if } z \notin \{x_0, \ldots, x_i\} \\ x_{i+1} & \text{otherwise} \end{cases}
\end{aligned}
$$

   (a) Suggest a design that solves this problem by reducing it to a prefix computation problem. What is the alphabet that you use (how large is it)? What is the associative operator that you use? Prove the correctness of your reduction.

   (b) Analyze the cost and delay of your move-to-front design.