The Loadable Counter Problem

In the first ACiD-WG workshop (in Leuven), we looked at the implementation of modulo-n counters. An interesting variation is the "loadable counter" C_N, which first has to input the value of n, known to be between 0 and some constant N.

This problem is actually of interest to Philips Research. Their Tangram language supports n-fold repetition of a command S by means of the construct

`for` n `do` S `od`

(A "systolic" design for the modulo-n counter is used in its implementation.) In order to improve the efficiency of Tangram programs as VLSI circuits, Philips Research might be prepared to extend the Tangram compiler, allowing n to be an expression that evaluates to a natural number, rather than simply a constant.

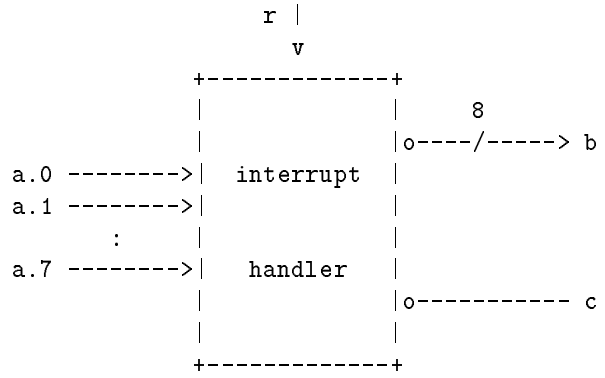
This would require a module C_N with two handshake ports, A and B, say. A is an input port of type $[0..N]$ and B is an output port of type bit. The communication behavior is

$$(A?n; (B!0)^n; B!1)*$$

where each communication involves a four-phase handshake. The problem is thus to design a small number of cells from which, for any N, module C_N can be constructed. C_N should have both bounded reponse time and bounded power dissipation.

Interrupt Handler

Our second problem is a simplified version of an interrupt handler as is found in, amongst other applications, microcontrollers.



Interface:

* r is an input (reset);
* $a.0$ .. $a.7$ are inputs;
* b is an 8-bit output, passive handshake channel:
  - the data is single-rail encoded in output wires $b.j$, $(0 \leq j < 8)$;
  - the handshake is 4-phase along wire pair (br, bk);
  - wires $b.j$ must be stable ("data valid") when bk is high;
* $c = (cr, ck)$ is an undirected ("nonput"), passive, 4-phase handshake channel.

The interrupt handler is reset into its initial state when r is high, and starts handling interrupts after r has gone low. (Inputs br and cr are low when r is high.) Then it must be prepared for two kinds of request:

1. b_r : Return an 8-bit interrupt vector: bit $b.j$ indicates that at least one transition has occurred on $a.j$ since reset or since the previous request along b. Note that during the data-valid interval of channel b the data wires of b must be stable whereas the input wires $a.j$ may change.
2. c_r : Wait until a transition has occurred on one of the input wires $a.0..a.7$ (since reset or since previous request along b); then complete the handshake along c.

For simplicity we assume that handshakes along b and c are mutually exclusive. We are interested in a low-power solution. Hence, there should be no internal activity when the handler is in one of its quiescent states.