DataFlow

page

This same machine can actually compute any sequence of the form s(n)=s(n-a)+s(n-b) where a and b are >1 and <5 and given any arbitrary starting conditions. This is accomplished by injecting the desired input conditions to the two inputs of the addition object. In the case of the fibonacci sequence, 0 was injected on the left and 0 and 1 were injected on the right. Unfortunately, the current implementation has no way of displaying the contents of the input queues so that there is hidden information which is not displayed directly on the screen which can lead to serious errors in larger programs.

The next example is a program to count from 1 to 10 and then go back to 1 without using the MOD function. The first attempt at the program is shown below

This version, however gives incorrect results after the first time around and eventually deadlocks because the queue on the right hand side of the select object fills up. This is because the addition symbol was adding 1 to 10 and firing an 11 token before it received the return-to-1 token. The solution was to add a distribute to throw away the extra token as shown below. This new version works correctly when a single 0 is injected into the right-hand input of the addition object to get the machine started.

Conclusions

Data flow is an interesting language in that the underlying model of computation is about as far away from the Von Neuman model as you can get. While it holds promise in the areas of parallel programming. My simulation was too primitive to exploit the more powerful of these abilities due to the lack of subroutines, recursion, and arrays. Dataflow suffers also from a large physical area on paper (or on a CRT screen) in which to express a relatively simple program. This can perhaps be improved upon using programs to tidy and compact the graph as well as using icons to represent subroutines containing subgraphs. These are further research issues. Other questions brought up by this work are as to whether the strict definition of data flow (as defined by Ackerman in his article in the February issue of COMPUTER) is appropriate for a simulation on a von-neuman machine. Immediate extensions include a merge object which will simply pipe two outputs to a single input with a non-deterministic (but fair) method of choosing which token will be output first should two arrive simultaneously from either input; also, a memory element which would output its input token as many times as needed until a new input is received at which time it would switch to the new value. This would allow a value to be discarded or the same value to be repeated. Both of these are timing dependent and introduce non-determinism in the machine. However, in a sequential simulation, rules can be set up as to the order in which things happen, and determinism is re-introduced.

All of these possibilities are topics for further study.