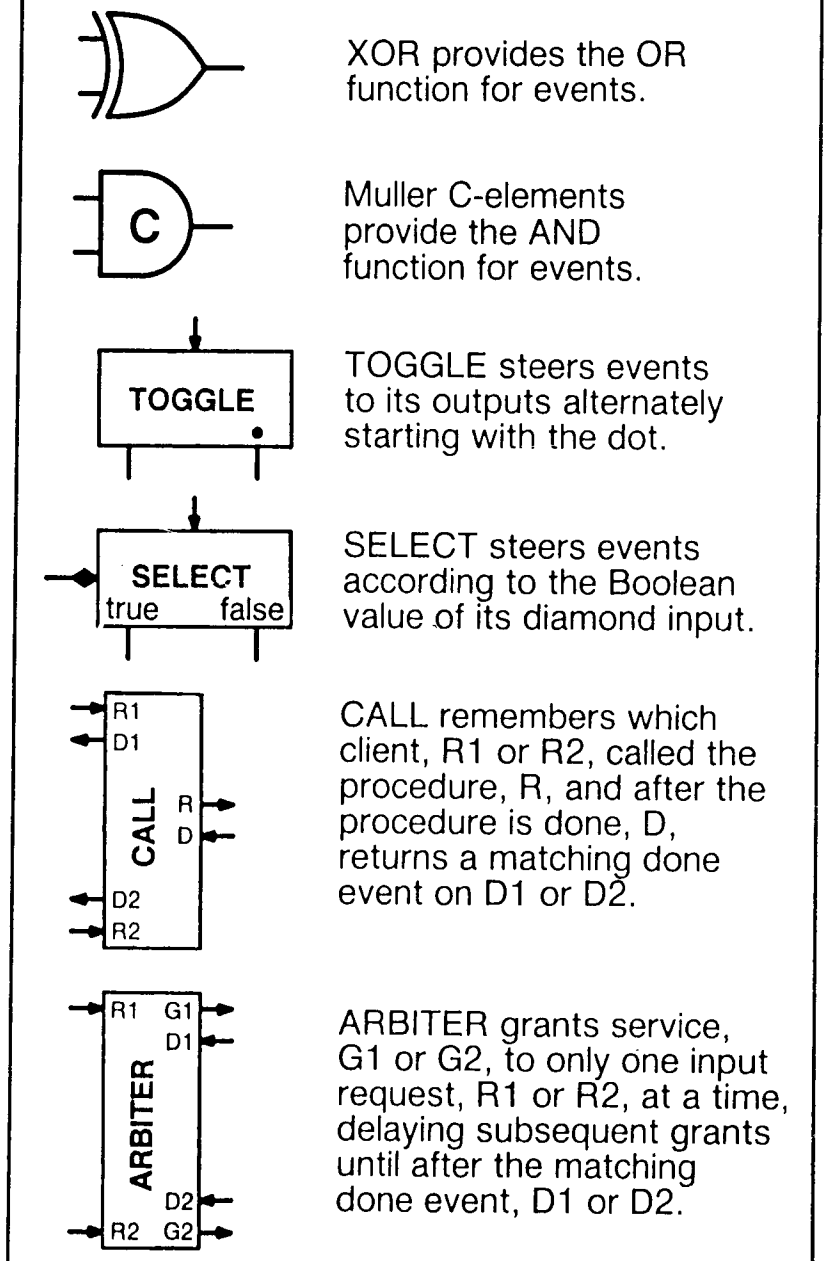

FIGURE 9. Logic Modules for Events

Modules of 10 to 100 transistors can perform useful logical functions on events. The modules whose symbols are shown provide the functions indicated. Note the similarity of these functions to the basic structures used in programming. One might think that the arbiter would require 8 terminals, since the request signals at the left seem to lack corresponding acknowledge signals. Either the grant or the done signals are used to acknowledge incoming requests, depending on the application.

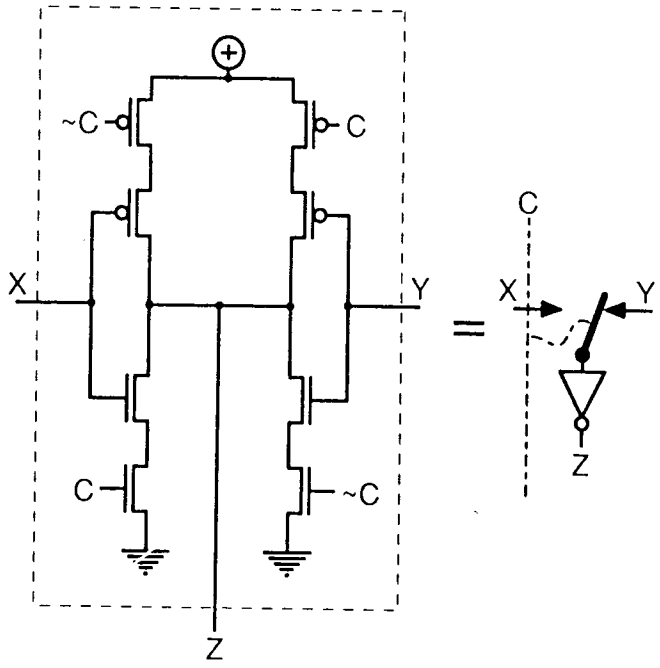

FIGURE 11. Circuit for the Switch Symbol

The double-throw switch symbol at the right of this drawing represents the transistor circuit shown inside the dotted line. When the control wire, *C*, is low, the output terminal, Z, is controlled by the *Y* input, as shown. When the control wire is high, the switch flips to the *X* input. The output of this form of switch is controlled by its selected input, but inverted in value. Other implementations of such a switch using pass transistors are also possible.

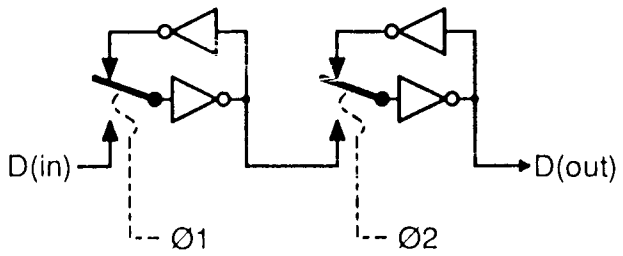

FIGURE 13. Conventional D Flip-Flop

A conventional D flip flop is controlled by non-overlapping clock signals $\phi 1$ and $\phi 2$ illustrated in Figure 5. Compare this circuit to the event-controlled storage element of Figure 12.

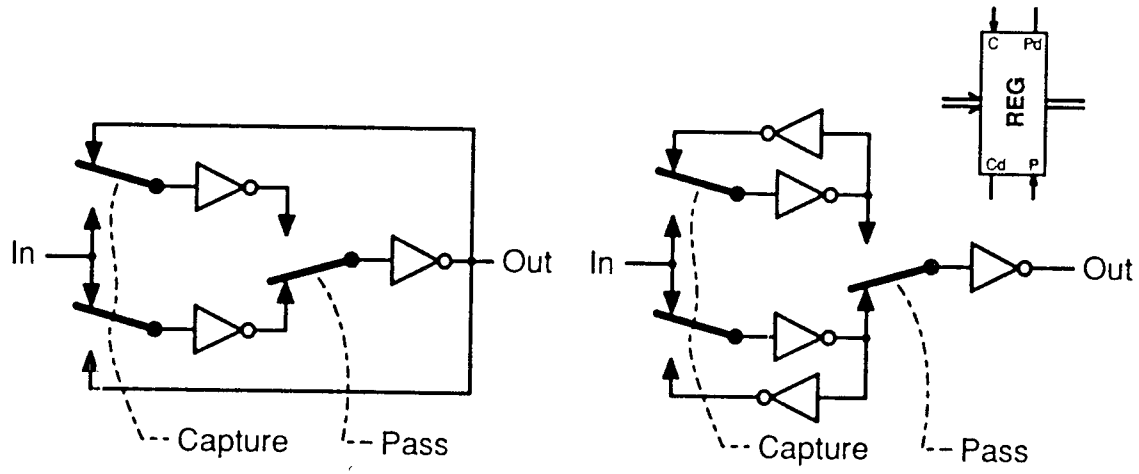

FIGURE 12. Event-Controlled Storage Elements

An event-controlled storage element responds to events on its two control wires, called "capture" and "pass" in this drawing. Two different configurations are shown. The form on the right, with five inverters, is slightly faster than the form on the left, with only three inverters, because its feedback paths contain only one switch rather than two. After master clear the switches will be in the position shown, making a direct connection without loops between input and output, a state in which the storage element is said to be transparent. Storage elements of either type are formed into registers just as are flip flops by connecting their capture and pass control wires in parallel. The register symbol includes control outputs, Cd and Pd, which are amplified, and thus necessarily delayed, versions of the control input signals, C and P. Cd and Pd, named for "capture done" and "pass done," deliver output events after the register has done its action.

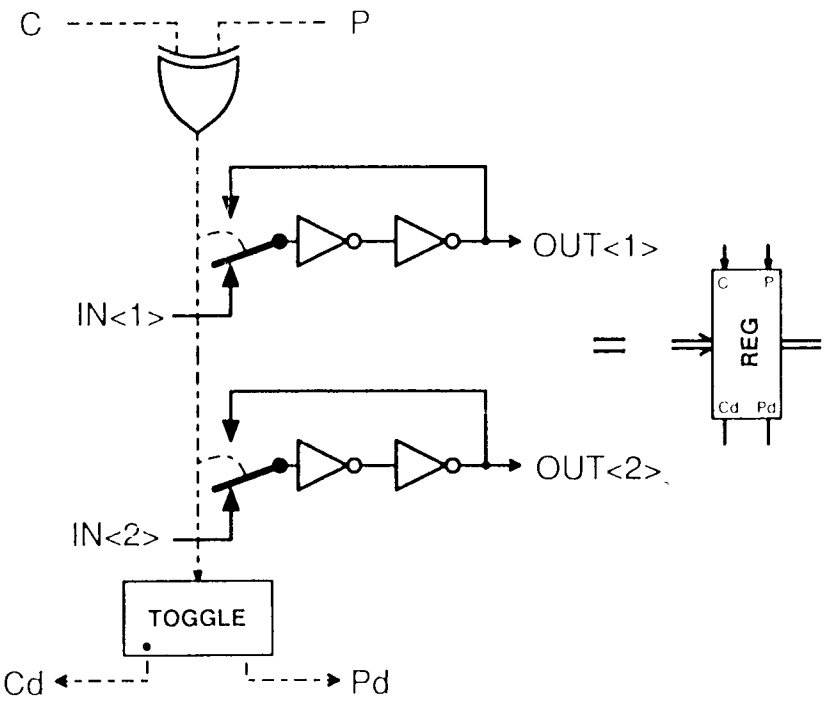

FIGURE 14. Latches Used as an Event-Controlled Storage Register

An event-controlled register made from ordinary latches requires an XOR module and a TOGGLE module for control. A 2-bit register is shown; dashed wires carry events. Capture and pass events arrive alternately at the separate control inputs, C and P, but the XOR merges them onto one wire. At the XOR output, each capture event becomes a rising transition in the latch control wire and flips the switches, causing the latches to capture data. Each pass event becomes a falling transition in the latch control wire and flips the switches back to the position shown, making the latches transparent again. The TOGGLE module separates the capture and pass events back into two separate output paths, Cd and Pd, after the register has done its action. This circuit is slower than the event-controlled register of Figure 12 and delays its output events, Cd and Pd, accordingly, but except for delay provides exactly the same function.

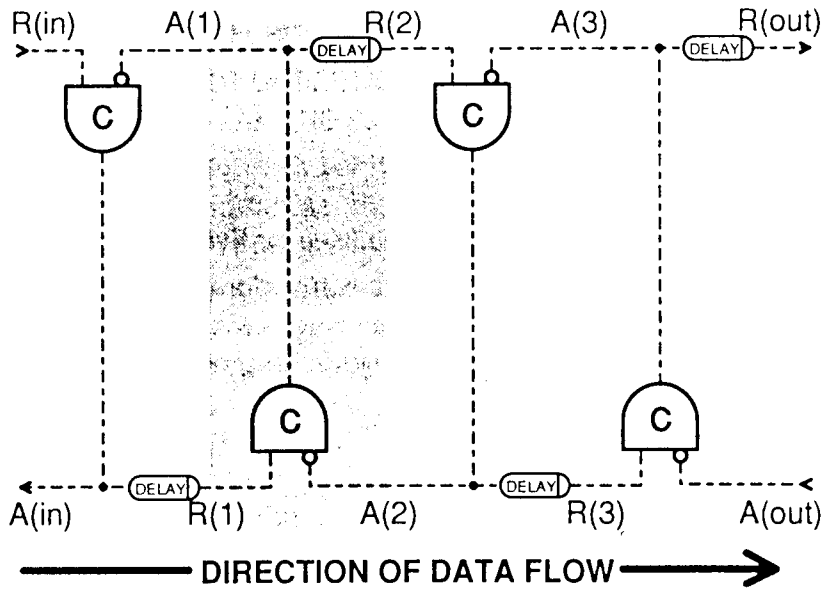

FIGURE 10. Control Circuit for a Micropipeline

With data paths omitted, the control circuit for a micropipeline is a string of Muller C-elements. In this figure one of four identical stages is shaded and alternate stages have been drawn upside down. At the input and output to each stage there are request, R(n), and acknowledge, A(n), signals. Inverters in the acknowledge paths are represented by "bubbles" at one input of each Muller C-element. The delays shown explicitly here may not be required for simple data paths. Notice that each loop in this circuit contains exactly one inversion, the bubble, and is therefore an oscillator. The Muller C-elements retard the oscillation in each loop to coordinate it with the actions of adjacent loops. In this and other figures, dotted wires carry event signals.

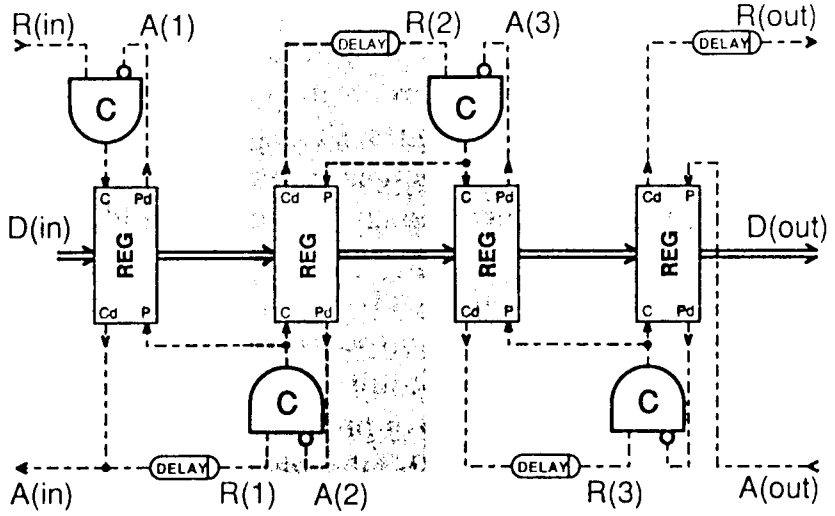

FIGURE 15. Micropipeline without Processing

A micropipeline without processing has event-controlled registers for data path and Muller C-elements for control. Four stages are shown; one of them is shaded. Each interface between stages conforms to the two-phase bundled data convention of Figure 4. This drawing suggests the form of an integrated circuit layout; the control signals pass back and forth across the data path to accommodate transmission delays.

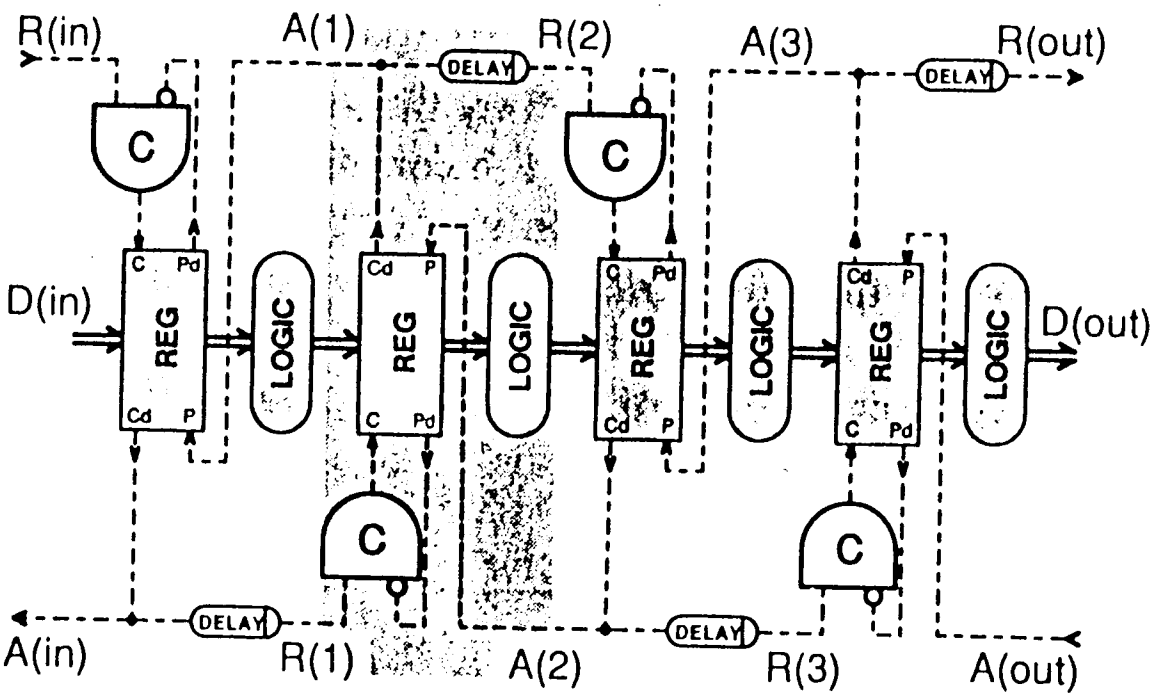

FIGURE 17. Micropipeline with Processing

A micropipeline with processing uses combinatorial logic between the event-controlled registers of Figure 15. Four stages are shown; one of them is shaded. The delay elements in the request event path model the processing logic delay to preserve the bundling convention. All interfaces between stages, taken either before or after the logic circuits, conform to the two-phase bundled data convention of Figure 4. The capture done, Cd, output of each register is shown connected to the pass, P, input of its predecessor, a more conservative connection than was used in Figure 15; either connection works.

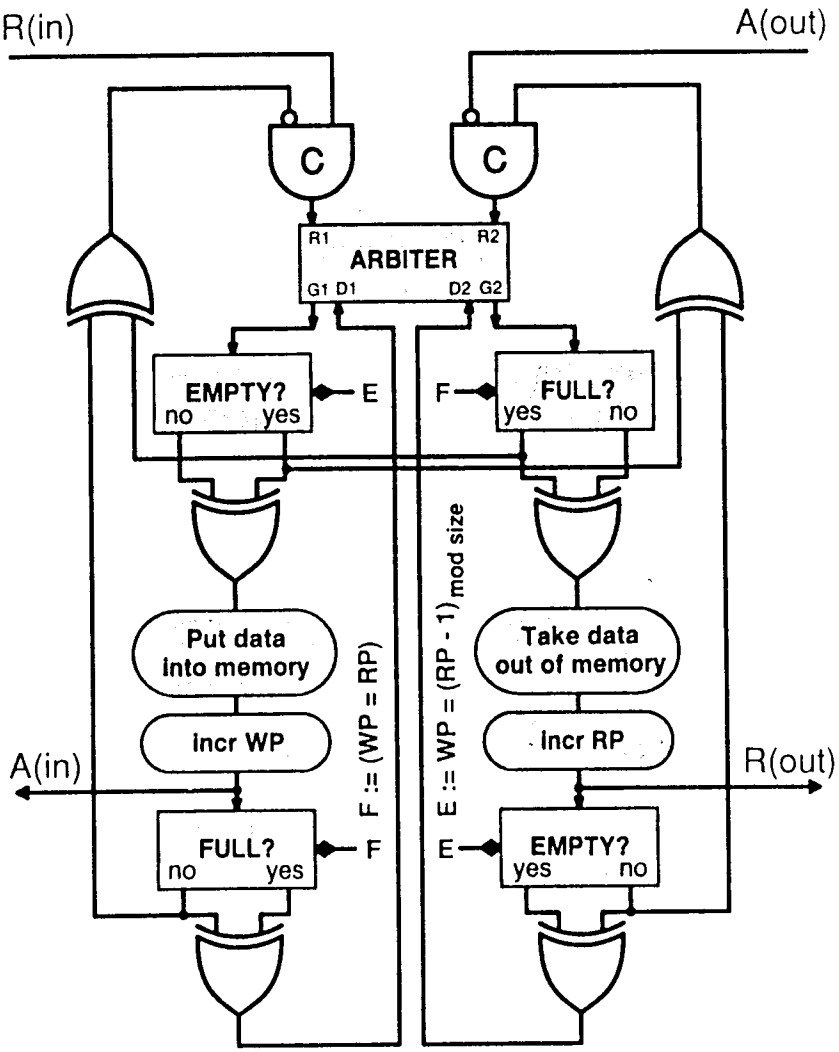

FIGURE 21. Ring-Buffer FIFO Control Logic

The control logic for a ring-buffer FIFO can be composed from the event logic modules shown in Figure 9. Except for the test values, all wires shown here carry event signals; the data path, the address pointers and the memory are not shown. In each of the four SELECT modules I have written the name of its test; the wires labeled "E" and "F" carry the required Boolean values. The functions described in the four lozenges include memory access and incrementing the read and write pointers, RP and WP. Although this figure looks like a block diagram, it is actually a circuit ready for direct implementation. It has been proven [5] than an external observer cannot distinguish this ring-buffer FIFO control circuit from the micropipeline control circuit of Figure 10.