MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Computation Structures Group Memo 82

A Speed-Independent Implementation of Data Flow Schemas

by

Becky Clark

This work was submitted for credit in Subject 6.534, "Semantic Theory for Computer Systems", Spring 1973.

June 1973

A SPEED-INDEPENDENT IMPLEMENTATION OF DATA FLOW SCHEMAS

In this paper we present a set of speed-independent modules useful for the implementation of data flow schemas. The intention was to design a set of modules which would function properly without regard to delays in the gates while using the assumption that there is no delay in the wires.

Two types of links, control and data, are required for signals between the different modules. The same signalling conventions are used for both. A data link consists of three lines, two of which carry the data signal from one module to another and a third which returns an acknowledge that the data has been received and may be taken away. The two data lines may be in any of three states. A 1 on either line indicates that a 0 or 1 is present according to the line upon which it appears. A 00 state for the two lines indicates that no data is present, and an exclusive-or of the two data lines can therefore be used as a request. This request and the acknowledge must follow a four-phase arrangement in which the acknowledge becomes 1 after the request becomes 1. The request may then be taken away, permitting the acknowledge to go to 0. An example of the labelling convention for a link k is shown in Figure 1 with d_{k0} indicating the presence of a 0 and d_{k1} the presence of a 1. The exclusive-or to find r_k will appear in the module receiving the data and is shown here to clarify the relationship between the data and acknowledge lines. An example of waveforms on a link is shown in Figure 2. A control link

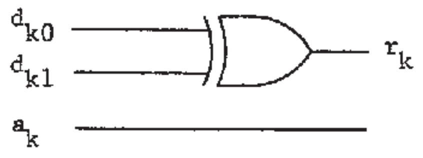


Figure 1.

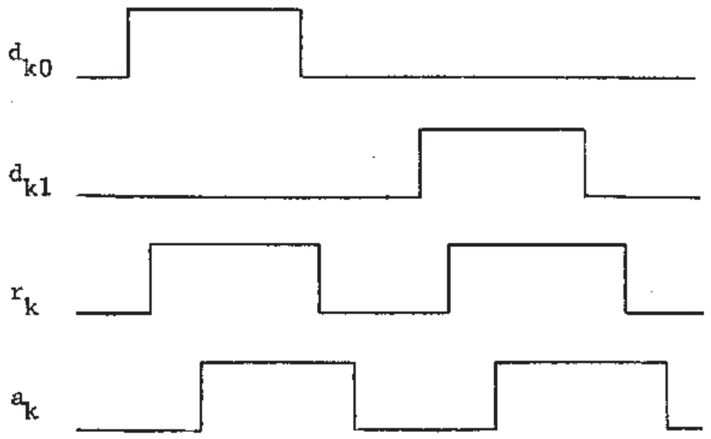


Figure 2.

has the same characteristics as a data link except that a signal of true or false is associated with each of the two data lines instead of a 0 or 1.

The set of modules required can be divided into five types: gates, merges, links, deciders, and operators. Since the links are identical for data and control, the gate, merge, and link modules will be identical for the similar data and control functions. A description of each of these follows.

The gate module is shown in Figure 3. One link is required as input, and a control link determines whether the input will be put onto the output link. The example shown here is a true data gate. A false gate may be obtained by interchanging d_{2T} and d_{2F}.

The merge module takes two links as input and a control link to determine the required output. A data merge module is shown in Figure 4.

Link modules must be used at all data link nodes and control link nodes. The function of a link module is to store the input and acknowledge it while transmitting it on the output links until they acknowledge. Any new input data waits until the output links have cleared. A two-way link module is shown in Figure 5. This may be extended to an n-way module by branching the outputs to the n output links and using an n-way C-element to detect acknowledges from all output links. Alternatively, the two-way modules could be cascaded to obtain any degree of branching required.

The remaining types of modules, deciders and functions, are dependent upon the specific interpretation of a data flow schema, and thus there are an unlimited number of these modules. An example

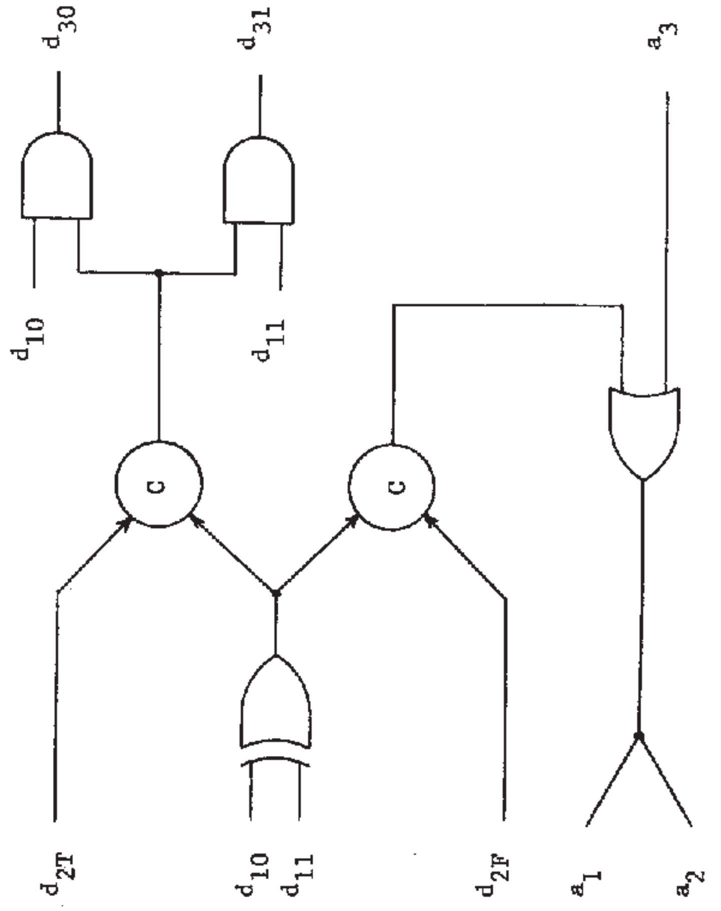
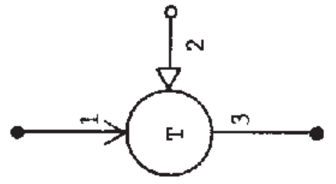


Figure 3: Gate Module.



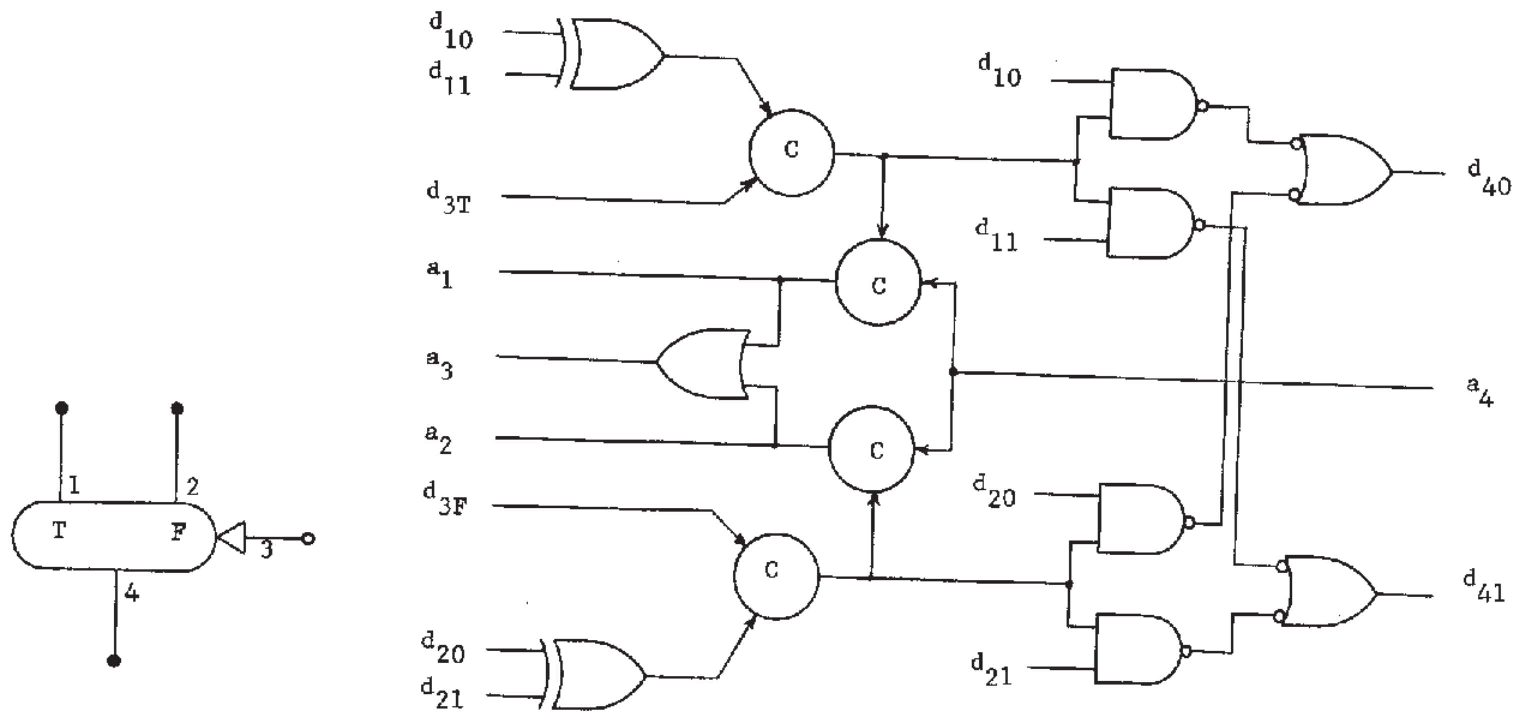


Figure 4: Merge Module.

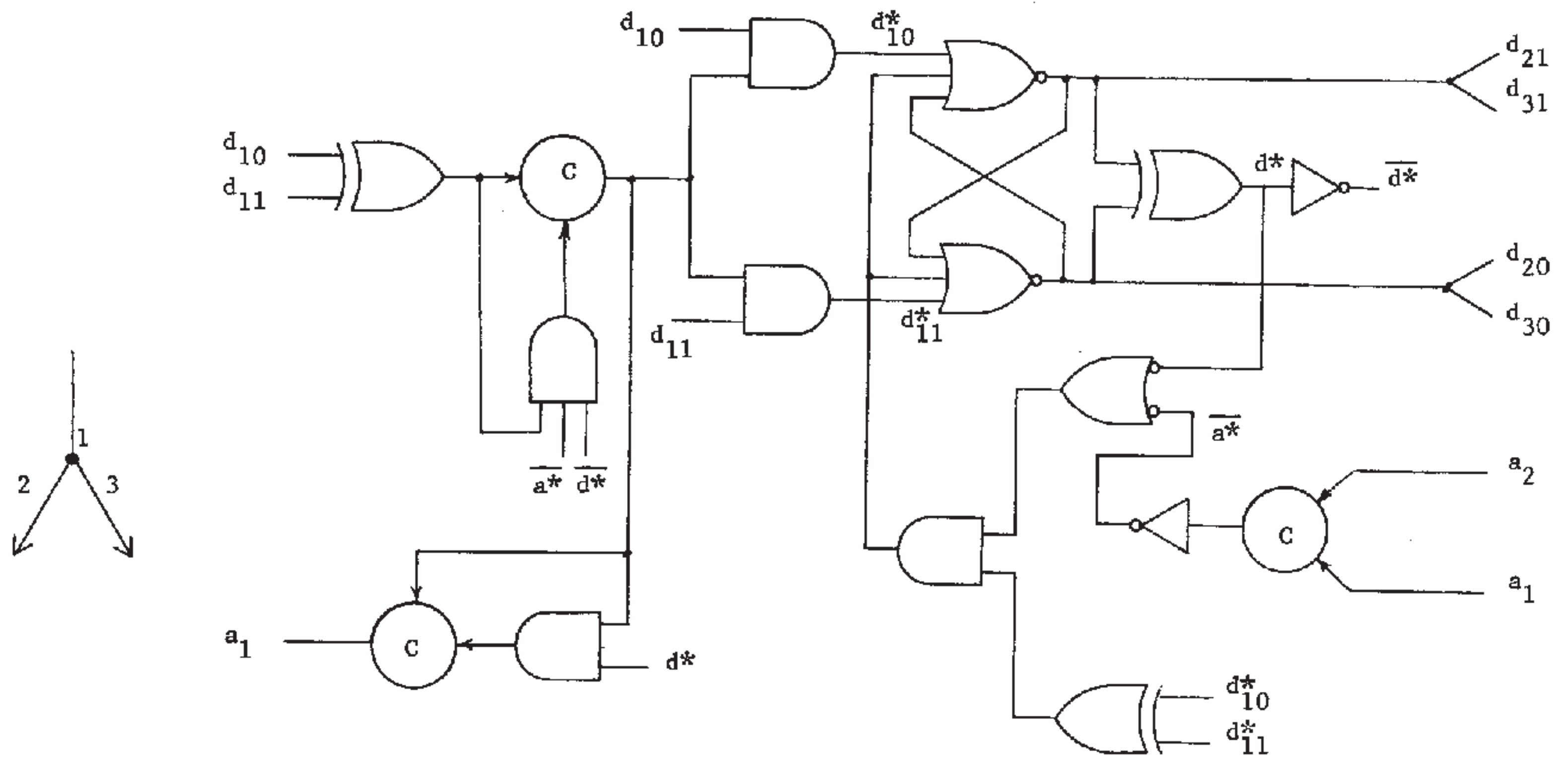


Figure 5: Link Module.

of a decider module to check for equality of two inputs is shown in Figure 6, and an AND function module for two inputs is shown in Figure 7. The structure of the two modules is quite similar, and any two-input Boolean function will have the same type of structure.

Initialization in a data flow schema requires the capability of presetting the control arc to a merge module. This can be accomplished by inserting a link module on the control arc before it reaches the module and presetting the latch within it to indicate an output of false.

The system described here deviates slightly from the model of data flow schemas in that the firing of an actor does not remove the tokens from the input arcs. This should not affect the operation of the schemas if link modules are used on every path between actors. If link modules were added to all the outputs of each actor, this condition would be satisfied.

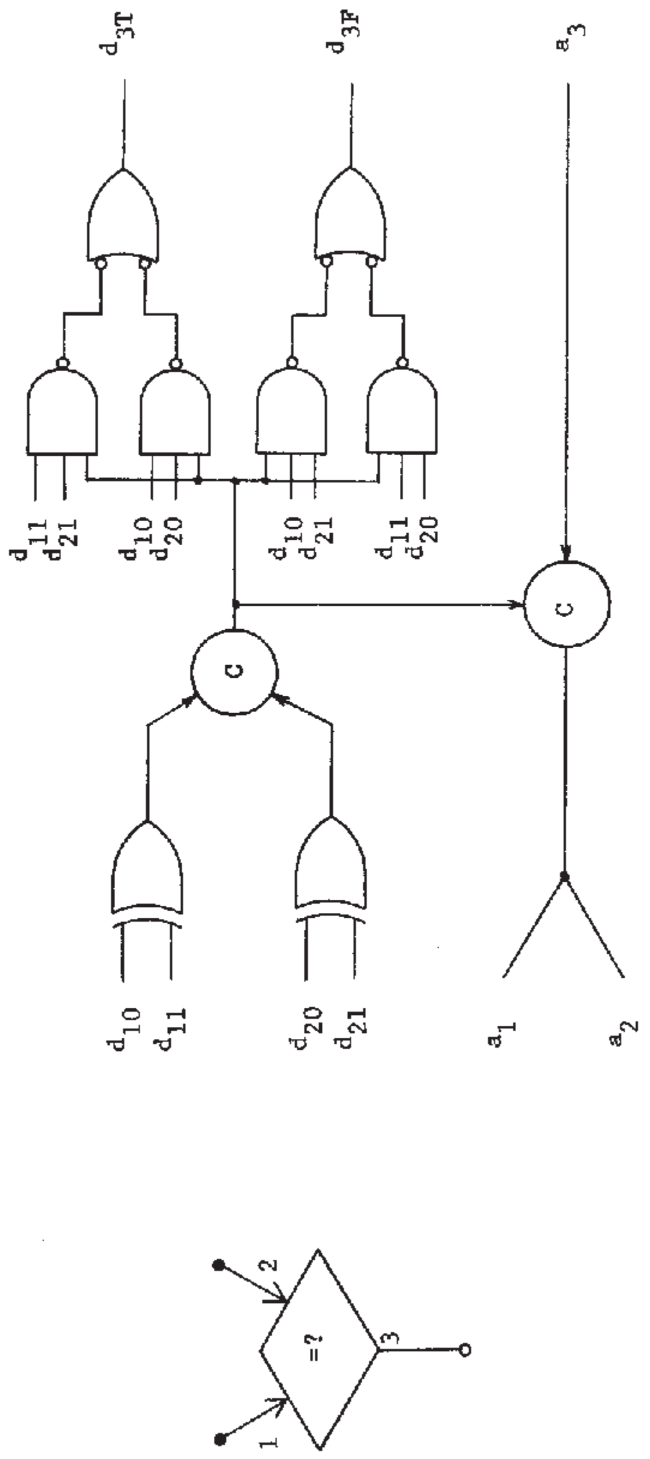


Figure 6: Equal? Module.

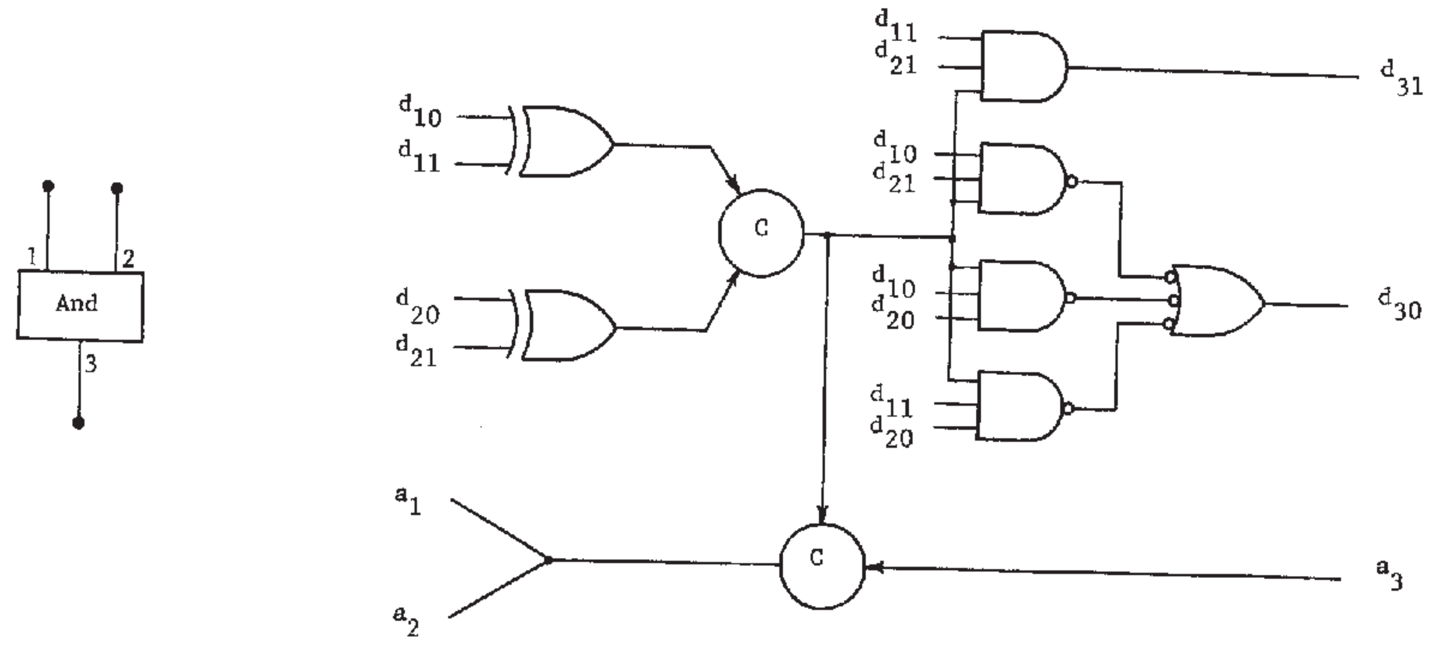


Figure 7: And Module.

REFERENCES

Dennis, J.B., J.B. Fosseen, and J.P. Linderman, "Data Flow Schemas," Computation Structures Group Note, July 1972.

Misunas, David, "Petri Nets and Speed Independent Design," Preprint, March, 1973.