Y0885-0611 TPD
R. K. Brayton and G. DeMicheli

IBM Technical Disclosure Bulletin Vol. 30 No. 2 July 1987

METHOD FOR MINIMIZING CRITICAL TIMING OF A DIGITAL SYSTEM

A technique is described whereby the critical delay timings of logic circuits, as used in digital computer systems, are minimized by synthesizing the logic in a hierarchical manner. Since input signals arrive at different times and the outputs are required at different times, logic blocks (macros) are re-synthesized along the systems critical timing paths. This method, when used in conjunction with other techniques, decreases the overall delay of the logic flow.

Since each logic block of a digital system contains a number of inputs and outputs, the system can be viewed as a direct graph where each node is a logic block. A branch is drawn from node i toward node j if one of the outputs of node i is a primary input for node j. It is then assumed that the graph is acyclic. In synthesizing each logic sub-block, each input signal arrival time is assumed to arrive at a given time. The outputs are also assumed to be required at a required time. This would be the situation if the logic macro received its inputs from latches and its output drove other latches. However, in digital systems, inputs arrive at different times and the outputs are required at different times. The technique described herein synthesizes the logic in a hierarchical manner, taking into account system timing considerations for minimizing the critical path lengths.

Each logic macro is also represented by another graph defining a Boolean network. Each node of that network has a logic function associated with it. This is represented by a gate implementing that function or simply that logic in a disjunctive form. Assuming that each gate or logic function has an associated delay equation outlining the delay from its input signals to its output, the system timing can be computed by tracing the delays through each gate in a block of logic, and in a hierarchical manner through each logic block. The systems critical path is computed by starting at the primary inputs for the system, and assigning them appropriate arrival times. Since the logic macros are interconnected by an acyclic directed graph, the logic macro delays can be processed as an ordered set.

Inside each logic macro, each signal delay is computed as follows:

$$\text{delay}(i) = \text{gate delay}(i) + \max_{j \text{ in } \text{FEEDS}(i)} (\text{delay}(j))$$

FEEDS (i) is a set of nodes which feed into node i. By tracing forward through the logic macros, the delay of each signal or wire of the system can be computed.

The system slacks on each signal can be computed by a similar computation by tracing backward through the graph and computed as follows:

Initialize all slacks to be a large number

Let the outputs of the digital system be denoted by OUT.

slack (i) = desired arrival time (i) - delay (i) for i in OUT and for each node j, compute for all k in FEEDS(j)

$$\text{slack (k)} = \text{minimum}\langle \text{slack (k)} , \text{slack (j)} + \max_{\substack{1 \text{ in} \\ \text{FEEDS(j)}}} (\text{delay (l)}) - \text{delay (k)} \rangle$$

The system slacks are computed by providing the slacks of the outputs of each logic macro and performing the slack computation inside the macro to determine the slacks of the inputs to the logic macro. This is performed in a backward order on the partially-ordered set of logic blocks.

The signals and logic macros that are on a critical path are those which involve the minimum slacks or in general those that are within some percent of the minimum slack. To begin the minimization of the critical path delays, the first logic block on the critical path is chosen. Re-synthesizing the logic is performed in order to reduce the critical path length and the procedure is iterated until no further improvements can be made. While the iteration is proceeding, the objective is to minimize the delay while keeping the area of the implementation as small as possible. The delay iteration is controlled by parameters which give an indication of the amount of trade-off between delay and the area affected.

The determination of the trade-off (TRADES) can be obtained in the following procedure:

Let PIDS = the delay (arriving time) on the inputs of block 1
Let PODS = the slack on the outputs of block 1

Re-synthesize block 1 using PIDS and PODS, the system delay information.

1) Determine the delays and slack on all branches of the Boolean network of logic block 1. This is done with PODS serving as the initial slack of the outputs of block 1, and PIDS serving as the arrival times for the inputs of block 1.

2) Determine sub-graph of the Boolean network of logic block 1 formed by critical paths. The graph of critical paths are all nodes and branches which have minimum slack or are within some percentage of the minimum slack.

3) Determine a set of nodes (gates) in the Boolean network of logic block 1 which form a minimum weighted separation set between the inputs and outputs on the critical path graph of block 1. Each node is given a weight appropriate to the cost of absorbing all functions which feed into the node. The minimum weighted separation set can be determined by a standard algorithm.

4) Absorb all critical edges leading into each of the nodes in the cut set chosen in item 3.

5) Recompute the delays and slacks of block 1, repeat items 1 thru 4 until the total accumulation absorption costs exceed some pre-specified amount.

6) Find common expressions and substitute these into the appropriate nodes. The substitution is allowed only if the critical delay of block 1 is not increased. This can be determined using the signal delays and slacks. This recovers some of the area spent in decreasing the delay in the previous steps.

7) Decompose any node which cannot be implemented in a single gate of the target. This is performed by decomposition, using the signal delay information discussed later.

8) Substitute any existing functions into another function if this does not increase the critical delay. Eliminate any nodes that have a value less than a preset specified value and which will not change the fact that another node can be done in as single gate. Repeat until no change occurs, keeping the delay constant to recover some area.

TRADES is used as an inner procedure inside the iteration which selects which logic block to re-synthesize. The decrease in the delay inside a logic block is controlled by a parallel giving the total amount of absorption cost allowed in one pass. The entire procedure of TRADES may also be repeated on a given logic macro. By iterating TRADES, a trade-off is effectively made between delay and area. The more TRADES is used, the less delay,but the more area is required to implement the circuit chip.

The method of decomposing a single logic function into a parallel decomposition of gates uses the delays of the inputs to that function to find a sub-expression which uses only early-arriving inputs. The decomposition is done as usual but each sub-expression is screened so that only those which do not increase the delay are used. Although this may not be possible, the expressions which have the earliest arrival times for their inputs are used.

An example of the arrival time logic function is as follows:

$$W = ABC + ABD + BCD$$

with arrival times of A=B=2, and C=D=1. Then

$$X = C + D$$
$$Y = CD$$
$$W = ABX + BY$$

This produces a good parallel decomposition, but

$$X = A + C$$
$$Y = AC$$
$$W = DBX + BY$$

does not produce a good parallel decomposition.

The method can be applied to reduce the number of logic stages in a system critical path by using the delay equation which attaches a delay of one to each node. The minimum separation set operation described will then reduce the number of stages of logic through the macro by one each time it is applied. The slack information of the system is used to be sure that this logic stage reduction applies to the reduction in the number of logic stages of the system critical path.