Correspondence

On Asynchronous Machines with Flip-Flops

Abstract—The use of transition sensitive flip-flops for state variables in an asynchronous machine eliminates the need for race considerations. The requisite clocking is provided by the input variables changing.

Index Terms—Assignment, asynchronous, critical race, flip-flop.

The use of capacitor-diode coupled or transition sensitive flip-flops in asynchronous machines rather than, say, NAND-NOR implementation eliminates critical race considerations from the assignment problem. For example, consider the asynchronous machine as assigned in Fig. 1. Clearly, with the given assignment, critical races occur in unstable states 2 and 6. In fact, this machine requires three state variables for a race-free assignment.

For a flip-flop implementation, set and reset equations may be obtained in terms of the input variables and their change. Essential terms are obtained from input transitions out of stable states, while don't care terms are obtained from input transitions out of unstable states. It must be assumed that only adjacent input transitions are permitted, and that they occur at a rate slower than the flip-flop switching time.

The machine of Fig. 1, using don't-cares, yields

$$A_{set} = A(\bar{Q}P + PQ + \bar{Q}P)$$

$$A_{re} = A(Q' + P)$$

$$B_{set} = B(Q')$$

$$B_{re} = B(Q' + \bar{Q}P),$$

where $X'$ is the variable $X$ changing from 0 to 1, and $X''$ is the 1 to 0 change.

If the transition inputs of the flip-flops are sensitive to the 0 to 1 change of the input variable, then $X'$ may be replaced by $X$ and $X''$ by $X$. Conversely, for 1 to 0 sensitive units, $X'$ is equivalent to $X$ and $X''$ to $X$.

Corroboratively, for JK-type flip-flops with both leading edge and level sensitive inputs, the set and reset equations for the example become

$$A_{set} = \bar{Q}[P] + P[\bar{Q}] + \bar{Q}[P]$$

$$A_{re} = [Q] + P[Q]$$

$$B_{set} = [Q]$$

$$B_{re} = P[Q] + \bar{Q}[P],$$

while the literal in brackets indicates the transition sensitive input.

Essentially this procedure converts an asynchronous machine to a synchronous one with changes in the input serving as the clock.

This method has also been pedagogically useful in the introduction of asynchronous machines from synchronous machines.

M. D. McIntosh
Dept. of Elec. Engrg.
Michigan State University
East Lansing, Mich. 48823

B. L. Weinberg
Dept. of Computer Science
Michigan State University
East Lansing, Mich. 48823.

Presently with the Landis Tool Company, Waynesboro, Pa. 17268.

A Theorem on the State Reduction of Synthesized Stochastic Machines

Abstract—Stochastic sequential machines produced by the insertion of noise in a deterministic machine are considered; the question of the relationship between the state reduction of the machine so formed and that of the underlying deterministic machine is raised. The theorem states that, for the case of observational noise only, if the stochastic machine has stochastically distinguishable outputs, then it is reduced iff the underlying machine is reduced. Machines with additive noise are used as an example of application.

Index Terms—Additive noise machines, state reduction, stochastic sequential machines.

Different aspects of the problem of synthesis of stochastic machines have been studied by several authors [1], [2]. The idea is simple: given the matrix $M$ of a stochastic sequential machine (SSM) [3], produce another machine with the same stochastic properties of (stochastically equivalent to, [3]) the given machine, using a deterministic sequential machine and noise. The problem is an important one, and appears, for instance, in attempting to simulate actual physical systems for which the entries of $M$ have been determined experimentally.

One of the first practical considerations in such cases is the number of states of the synthetic machine; hence, the importance of techniques for state reduction and state minimization of stochastic machines [4]. In our context, very natural questions may be raised. How does the state reduction or minimization of the SSM affect the underlying deterministic machine, and vice versa? Does a reduced deterministic machine always yield a reduced SSM? Can one synthesize first and then reduce the underlying machine, or should one minimize the SSM first and then synthesize it?

These questions are meaningless unless more precisely stated. Since the underlying deterministic machine is, in most cases, taken as a sequential circuit, we will consider the SSM to be assigned, that is, to be a 4-tuple $M = (X, Y, S, M)$, where $X$, $Y$, and $S$ are finite dimensional vector spaces over finite fields, and $M$ is, as usual, the combined transition probability matrix. $M$ is to be isomorphic to a