State Assignment for Initializability of Synchronous Finite State Machines*

Montek Singh and Steven M. Nowick
Department of Computer Science
Columbia University
New York, NY 10027

Abstract:
It is well known that state encoding can influence the logic initializability of an FSM implementation. If the sole objective of an “optimal” state assignment is to minimize the amount of logic, one may end up with implementations that are logically uninitializable. That is, a 3-valued (0,1,X) logic simulator may not be able to initialize the circuit even when its FSM has a synchronizing sequence. Initializability is required for (most) non-scan ATPG’s such as CONTEST [1] and STG [2] to work effectively.

Traditionally, several different approaches to initializability have been used. Each of these assumes different models of initializability (such as single- or multi-vector) and of simulation (such as true-value or 3-value). Further, while the Weihbeh and Saab [3] algorithm analyzes the gate-level circuit to determine if it is initializable, there are other methods that enable one to synthesize for initializability. We follow the approach of Cheng and Agrawal [4,5] who attempt to produce a state assignment that makes the circuit initializable.

Contribution:
Cheng and Agrawal impose certain constraints on state assignment to translate functional initializability into logic initializability [4]. We demonstrate that these constraints are neither necessary nor sufficient. We propose an alternative method which is safe and not as conservative.

Cheng and Agrawal also mention that logic minimization seems to influence 3-valued initializability [5]. However, the technique of logic realizability proposed by them is a heuristic one, and not guaranteed to succeed. In this work, we propose precise conditions on 2-level logic minimization for faithful 3-valued simulation. These are similar to Eichlerber’s [8] and Nowick’s [9] cube covering formulations for hazard-freedom.

Background:
Consider the functionally initializable machine $M$ in Fig. 1[4]. At startup, the machine can be in any state, i.e. the initial state group consists of all states: $\{S_1, S_2, S_3, S_4\}$. $I = 1001$ is a synchronizing sequence of $M$. The following is the state of group states that result as the input sequence 1001 is applied to $M$:

$$(S_1, S_2, S_3, S_4) \rightarrow (S_2, S_4) \rightarrow^0 (S_1, S_3) \rightarrow^0 (S_1, S_4) \rightarrow^1 (S_4)$$

Associated with each state group, after state assignment, is its smallest containing cube, called the group face. Each group face is represented by a 3-valued vector. Thus, if state encoding $(S_1 : 01, S_2 : 00, S_3 : 10, S_4 : 11)$ were used, the following group faces would result: $(XX) \rightarrow (XX) \rightarrow^0 (XX) \rightarrow^0 (XX) \rightarrow^1 (XX)$. This sequence does not converge to a single state, therefore the circuit realized above is logically uninitializable. A 3-valued simulator can only simulate group faces, not state groups, so there is a loss of information. Therefore, our aim is to produce a state assignment that allows the sequence of group faces to “track” the sequence of state groups, and therefore ensure logical initializability.

Cheng and Agrawal state that the problem can be solved by introducing an additional set of face-embedding constraints into the state encoding step. Each of these is a $k \rightarrow 1$ dichotomy constraint, $(G_i; s_j)^2$. In particular, a $k \rightarrow 1$ dichotomy of the form $(G_i; s_j)$ is introduced for every symbolic state $s_j$ not present in the state group $G_i$. That is, a symbolic state that does not belong to a state group is forbidden from being embedded in it. This applies to all state groups encountered when a synchronizing input is applied to the machine. For the example of Fig. 1, the dichotomies generated are: $\{(S_2; S_1), (S_2; S_3; S_4), (S_1; S_3; S_2), (S_1; S_4; S_2), (S_1; S_3; S_4)\}$. A state assignment satisfying these dichotomies is: $\{S_1 : 11, S_2 : 00, S_3 : 10, S_4 : 01\}$, resulting in a correct

---

*This work was supported by an NSF CAREER Award MIP-9501880 and by an Alfred P. Sloan Research Fellowship.

1Here we are really talking about the physical circuit corresponding to $M$, not $M$ itself.

2An $n \rightarrow k$ dichotomy constraint $(X; Y)$ [7] is the stipulation that the smallest containing cubes of $X$ and $Y$ not intersect. Here, $X$ and $Y$ are sets of states of cardinality $n$ and $k$ respectively. A dichotomy is satisfied if some state bit has value 1 for all states in $X$ and value 0 for all states in $Y$, or vice versa.
3-valued simulation: \( (XX) \xrightarrow{1} (0X) \xrightarrow{0} (1X) \xrightarrow{0} (X1) \xrightarrow{1} (01) \).

**State Assignment:**

In this section, we show that, (1) when considering encoding of symbolic states, the constraints of the Cheng and Agrawal method (“the \( C/A \) method”) are unnecessarily restrictive, and (2) when considering assignment of don’t-care next-state entries, the \( C/A \) method is, in general, unsafe. In fact, we demonstrate that applying \( C/A \) may result in an uninitialized circuit.

Consider the machine in Fig. 2. A synchronizing sequence for the machine is 100, resulting in the state group sequence: \((S_1, S_2, S_3, S_4) \xrightarrow{1} (S_1, S_2, S_3) \xrightarrow{0} (S_1, S_4) \xrightarrow{0} (S_4)\). The dichotomy constraints produced by the \( C/A \) method are \\{\((S_1S_2S_3; S_4), (S_1S_4; S_2), (S_1S_4; S_3)\)\} and a minimum length state encoding satisfying these constraints requires 3 bits. In fact, the dichotomy \((S_1S_2S_3; S_4)\) is unnecessary. At the second time instant, the next-state value of \( S_4 \) is \( S_4 \), which happens to belong to the next-state group, \((S_1, S_1)\). Therefore, if we let \( S_4 \) be embedded in the group-face of \((S_1, S_2, S_3)\), no harm would accrue. Thus, it is safe to delete the dichotomy \((S_1S_2S_3; S_4)\). The resulting state assignment is \((S_1 : 00, S_2 : 01, S_3 : 11, S_4 : 10)\), still yielding a correct 3-valued simulation: \( (XX) \xrightarrow{1} (XX) \xrightarrow{0} (X1) \xrightarrow{0} (01) \). This solution required only two bits to encode the states, whereas the \( C/A \) solution requires three. Therefore, we prune the list of dichotomies proposed by \( C/A \)—we delete the dichotomy \((G_i, s_j)\) whenever \( NS(s_j, I_i) \in G_{i+1} \), because this is a “safe” embedding.

Second, we demonstrate that the \( C/A \) conditions are, in general, unsafe. In particular, we show that the issue of filling in don’t-care entries is important, and cannot be left entirely to the later stages of synthesis.

Consider the state machine of Fig. 3a. It has a single-vector initialization sequence \( I = 1 \). Fig. 3c shows a state encoding satisfying the face-embedding constraints (there are none). It uses two bits to encode three states, thereby introducing a fourth state \((11)\) that has no symbolic equivalent (a non-symbolic state). If a later step assigns state \( 11 \) a next-state (NS) entry of \( 10 \) on input \( 1 \), then the machine is no longer initializable; the 3-valued simulation trace is: \( (XX) \xrightarrow{1} (X0) \). Simulation fails because NS entry, 10, for state \( 11 \) lies outside of the state group being simulated, \((S_1)\), thus throwing initialization off course. Observe that if the NS value was assigned \( S_1 \) instead, the circuit would have been initializable: \( (XX) \xrightarrow{1} (00) \). Therefore, if non-symbolic states can be assigned arbitrary NS values (during logic synthesis), a non-initializable circuit may result.
Figure 4: Second example illustrating the issue of don’t-care NS assignment.

In general, however, there are examples where using C/A (or our earlier constraints) results in a circuit that is uninitializable for every assignment of don’t-cares. Consider the example of Fig. 4. Applying a synchronizing sequence gives the following state groups: \((S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9) \xrightarrow{11} (S_1, S_2, S_3) \xrightarrow{00} (S_7, S_8, S_9) \xrightarrow{01} (S_4, S_5, S_6) \xrightarrow{00} (S_3)\). A state encoding that satisfies the above constraints is shown in Fig. 4b. Bit-vector 0111 is a non-symbolic state with an as yet unassigned NS value. A careful analysis shows that this non-symbolic state cannot be assigned any NS value for input 00 while preserving initializability. Indeed, state 0111 is embedded in the group-faces of \((S_1, S_2, S_3)\) as well as \((S_4, S_5, S_6)\). The latter embedding mandates the NS value of this non-symbolic state to be set to \(S_3\). The former embedding requires the NS value to be set to a state in the column containing \((S_7, S_8, S_9)\). These two conditions are not simultaneously satisfiable. Assuming the NS value is set to \(S_3\), the resulting simulation trace is: \((XXXX) \xrightarrow{11} (011X) \xrightarrow{00} (XXXX) \xrightarrow{01} (XXXX)\).

To insure that there are no conflicting demands on the assignment to a don’t-care, we can draw up pairs of group-faces and force them to be disjoint. Then, no non-symbolic state can belong to both of them, so the above problem is circumvented. One way to do this is to introduce an \(n \to k\) dichotomy constraint between \(G_j\) and \(G_k\), represented as \((G_j; G_k)\), if \(G_j\) and \(G_k\) are followed by the same input vector but \(G_{j+1}\) and \(G_{k+1}\) do not intersect. In the above example, we therefore add dichotomy \((S_1 S_2 S_3; S_4 S_5 S_6)\).

The above conditions are sufficient to produce a state encoding that insures logic initializability. These can be formalized as a unate covering problem [7]. We have developed two alternative solutions which are less restrictive, which can be formalized as binate covering problems [7].

Logic Minimization:

In [5], Cheng and Agrawal mention a final issue of the actual logic realization influencing 3-valued simulation, and hence initializability. They surmise that initializability can be preserved by separately optimizing each output, as opposed to doing multi-output optimization. However, we discovered that this is neither necessary nor sufficient.

Based on the above, we have developed an iterative algorithm that selectively assigns don’t-care values to insure logic initializability. In particular, we demonstrate that the problem of 3-valued logic initializability can be cast as a hazard-free minimization problem [8,9]. For the 2-level case, necessary and sufficient conditions correspond to Eichelberger’s static hazard conditions [8]. We propose that similar results apply to the multi-level case.

Our approach exhibits a strong similarity to the work of Chakrada et al. [6] on initializable synthesis for asynchronous circuits. Preliminary results show that our method may be more efficient.

Results: We will discuss results on a small number of examples in the presentation.

References:


---

*This state machine is incompletely-specified, but our analysis also applies to completely-specified machines.*