FSM Encoding for Low Power, Reduced Area and Increased Testability using Iterative Algorithms

Faisal Nawaz Khan
COE, KFUPM
Agenda

- Theory of State Encoding
- State Encoding for Increased Testability
- State Encoding for Reduced Area
- State Encoding for Low Power
FSM Encoding

- To encode p states using k bits, the number of possible assignments are

\[
\frac{(2^k - 1)!}{(2^k - p)!k!}
\]

- Encoding governs the mutual dependence of the state variables. Thus effecting the number of literals for next-state functions, their interconnection and inter-dependence.

\[
Y_1 = f_1(y_1, \ldots, y_n, x_1, \ldots, x_m) \\
Y_2 = f_2(y_1, y_2, x_1, \ldots, x_m) \\
Y_3 = f_3(y_3, y_4, x_1, \ldots, x_m) \\
Y_4 = f_4(y_3, y_4, x_1, \ldots, x_m)
\]
## Introductory Example

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0</td>
<td>X=1</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>
$Y_1 = x'y_1 + x_1y'_1 = f(x, y_1)$

$Y_2 = x'y_1 + xy_2 = f(x, y_1, y_2)$

$z = xy_2' = f(x, y_2)$

<table>
<thead>
<tr>
<th>y1y2</th>
<th>Y1Y2</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0</td>
<td>X=1</td>
</tr>
<tr>
<td>A -&gt; 00</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>B -&gt; 01</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>C -&gt; 11</td>
<td>11</td>
<td>01</td>
</tr>
<tr>
<td>D -&gt; 10</td>
<td>11</td>
<td>00</td>
</tr>
</tbody>
</table>
Thus, the choice of assignment affects the complexity of the circuit and determines the dependency of the next-state variables and the overall structure of the machine.

Thus we need to find out tools in order to derive assignments that result in reduced dependencies among the state variables.

Such assignments generally yield simpler logic equations and circuits.
Partitions

- State assignment problem can also be viewed as partitioning problem
- A partition consists of blocks of states.
- E.g. in Encoding-1, we have
  - \( Y_1 = 1 \) for \( C \) and \( D \); 0 for \( A \) and \( B \);
  - \( Y_2 = 1 \) for \( B \) and \( C \); 0 for \( A \) and \( D \);
- We say
  - \( Y_1 \) induces a partition \( T_1 = \{ A, B; C, D \} \)
  - \( Y_2 \) induces a partition \( T_2 = \{ A, D; B, C \} \)
- In this case,
  - \( T_1 . T_2 = \pi(0) \)
  - Where \( \pi(0) = \{ A; B; C; D \} \) is called 0-partition.
- The 0-partition describes that we have successfully assigned a unique code to each state.
- Thus, our aim in state encoding is to find set of partitions such that their product results in 0-partition.
- Here ‘\( T \)’ is a general partition that is induced by a state variable.
Closed Partitions

- Closed partitions are represented with $\pi$.
- A partition $\pi$ is said to be closed if for every two states, $S_i$ and $S_j$ which are in the same block of $\pi$ and any input $I_k$, the states $I_kS_i$ and $I_kS_j$ are in a common block of $\pi$.
- For the sample machine shown, the following partitions are closed:
  - $\pi_1 = \{AB; CD\}$
  - $\pi_2 = \{AC; BD\}$
- The successor relationship can be described using a graph.
- Clearly, it can be seen that the knowledge of the present block of the machine and the input is sufficient to determine uniquely the next block.

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0</td>
<td>X=1</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>
Closed Partitions

- In other words, we can say that the state variables assigned to blocks of a partition are independent of the remaining state variables.
- For e.g., partition \( \pi(3) \) requires 2 state variables, say \( y_1 \) and \( y_2 \); the encoding of variables is independent of other variables.

\[
\begin{align*}
\pi (0) &= \{A; B; C; D; E; F; G; H\} \\
\pi (1) &= \{ABCD; EFGH\} \\
\pi (2) &= \{ADEH; BCFG\} \\
\pi (3) &= \{ AD; BCFG; EH\} \\
\pi (4) &= \{ ADEH; BC; FG\} \\
\pi (5) &= \{ AD; BC; EH; FG\} \\
\pi (6) &= \{ ABCDEFGH\} = \pi (I)
\end{align*}
\]

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X=0</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>G</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
</tr>
<tr>
<td>G</td>
<td>B</td>
</tr>
<tr>
<td>H</td>
<td>D</td>
</tr>
</tbody>
</table>

Machine: M2
M2 has eight states => 3 variables are required

\( \pi(5) \) requires 2 state variables.

We can partition the machine such into two blocks such that predecessor components has two variables, say y1 and y2, that are assigned to partition \( \pi(5) \), while the successor component has a single variable y3, which can distinguish the states in the blocks of \( \pi(5) \).

To do so, we need to find a partition such that

\( \pi(5). T = \pi(0) \)

A sample partition could be \{ABEF; CDGH\}

Information Flow

\[
\begin{align*}
\pi(0) &= \{A; B; C; D; E; F; G; H\} \\
\pi(1) &= \{ABCD; EFGH\} \\
\pi(2) &= \{ADEH; BCFG\} \\
\pi(3) &= \{AD; BCFG; EH\} \\
\pi(4) &= \{ADEH; BC; FG\} \\
\pi(5) &= \{AD; BC; EH; FG\} \\
\pi(6) &= \{ABCDEFGH\} = \pi(I)
\end{align*}
\]
However, maximal reduction in dependency (which is a good measure of area as well) of the state variables would be achieved if we could find three two-blocks closed partitions whose product is 0-partition.

Then each state closed partition would be represented with a state variable – which would be independent of other state variables.

We only have two 2-block partitions $\pi(1)$ and $\pi(2)$.

So we need to find out partition to fill out the missing information, such that

$\pi(1) \cdot \pi(2) \cdot T = \pi(0)$

\[
\begin{align*}
\pi(0) &= \{A; B; C; D; E; F; G; H\} \\
\pi(1) &= \{ABCD; EFGH\} \\
\pi(2) &= \{ADEH; BCFG\} \\
\pi(3) &= \{AD; BCFG; EH\} \\
\pi(4) &= \{ADEH; BC; FG\} \\
\pi(5) &= \{AD; BC; EH; FG\} \\
\pi(6) &= \{ABCDEFGH\} = \pi(I)
\end{align*}
\]
Let $T = \{ABGH; CDEF\}$

Then

- $y_1$ is assigned to $\pi(0)$
- $y_2$ is assigned to $\pi(1)$
- $y_3$ is assigned to $T$

Now, $y_1$ and $y_2$, that are assigned to closed partitions are clearly self-dependent, while $y_3$, which is assigned to $T$, will be a function of external inputs and all three state variables.

This is proved with the logical equations that are derived from the encoding.

\[ Y_1 = x'y_1' \]
\[ Y_2 = x'y_2 + xy_2' \]
\[ Y_3 = xy_3 + x'y_1'y_2y_3' + y_1'y_2'y_3 + x'y_1y_2'y_3' \]
Parallel/Serial decompositions

- If the product of $n$ closed partitions results in $0$-partition then the machine can be realized with $n$ parallel components (independent subsets)
  \[ \pi(1) \cdot \pi(2) \cdots \pi(n) = \pi(0) \]

- If the above is not true, we need to incorporate a partition which is not closed. Such a partition result in a machine that is dependant on independent subsets.
  \[ \pi(1) \cdot \pi(2) \cdots T = \pi(0) \]
Two Implementation for a machine

\[ \pi(1) = \{ABC; DEF\} \]
\[ \pi(2) = \{AE; BF; CD\} \]
\[ \pi(1). \pi(2) = \pi(0) \]

\[ T(Y2) = \{AE; BCDF\} \]
\[ T(Y3) = \{ACDE; BF\} \]

\[ T(Y2).T(Y3) = \pi(2) \]
\[ \pi(1).T(Y2).T(Y3) = \pi(0) \]

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>F</td>
</tr>
</tbody>
</table>
**Implementation - 1**
- Consider a parallel decomposition of a machine
  \[ \pi(1) \pi(2) = \pi(0) \]
  \[ Y_1 = f(x_1, y_1) \]
  \[ Y_2 = f(x_1, x_2, y_2, y_3) \]
  \[ Y_3 = f(x_1, x_2, y_2, y_3) \]
- 30 Diodes (gates)

**Implementation - 2**
- The same machine can be implemented as
  \[ \pi(1) T(Y_2) T(Y_3) = \pi(0) \]
  \[ Y_1 = f(x_1, y_1) \]
  \[ Y_2 = f(x_1, x_2, y_3) \]
  \[ Y_3 = f(x_1, x_2, y_2) \]
- 20 Diodes (gates)

- Partitions \( T(Y_2) \) and \( T(Y_3) \) are cross dependant.
- In implementation-1, we have two closed partitions. However, in implementation-2, we have only 1.
- We see
  - That next block for Partition \( T(Y_2) \) lie in partition \( T(T3) \) and vice versa
  - \( T(Y2) \).\( T(Y3) \) results in a closed partition – and they should be since together they are independent of the rest and form a self-dependant subset for the machine.
- Thus, we need to have a more general tool for evaluating such cross dependencies
Partition Pairs

- Partition Pair is a set of two partitions such that they are cross dependant.
- \((T, T')\) are said to be partition pairs if for any two states in any block in \(T\), the next state for both lie in some block of \(T'\).
- Thus \(T'\) consists of all the successor blocks implied by \(T\).
- A closed partition can now be thought of as a special case for a partition pair such that \(T' = T\).
Partial Ordering on Partition Pairs

- (T1, T1’) and (T2, T2’) are partition pairs then (T1 + T2, T1’ + T2’) and (T1.T2, T1’.T2’) are also partition pairs.

- Intuitively, if two states, Si and Sj are in the same block of T1.T2, then they must also be in the same blocks of T1 and T2. Thus (T1.T2, T1’.T2’) is a partition pair.

- Similar observation can also be derived for considering (T1+T2, T1’+T2’) as a partition pair.

- We say that (T1 + T2, T1’ + T2’) is the least upper bound (lub) for partition pairs (T1, T1’) and (T2, T2’).

- Similarly, (T1.T2, T1’.T2’) is the greatest lower bound (glb) for partition pairs (T1, T1’) and (T2, T2’).
M(T’) and m(T’)

- M (T’) = Σ T_i, where the sum is over all T_i such that (Ti, T’) is a partition pair.
- M (T’) is the largest partition the successors of whose blocks are contained in the blocks of T’.
- M (T’) can be said as lub of all Ti such that (Ti, T’) is a partition pair.

- m (T) = π Ti’, where the product is over all Ti’ such that (T, Ti’) is a partition pair
- m (T) is the smallest partition containing all the successors of the blocks of T.
- m (T) can be said as glb of all Ti’ such that (T, Ti’) is a partition pair.
Let $T_{ab}$ be the partition that includes a block $(ab)$ and leaves all other states in separate blocks. Then $m(T_{ab})$ is the smallest partition containing the blocks implied by the identification of $(ab)$. $(T_{ab}, m(T_{ab}))$ is a partition pair.

In other words $m(T_{ab})$ represents smallest partition (maximum amount of information) such that the next states of partition $T_{ab}$ are contained in it.

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>D</td>
</tr>
</tbody>
</table>

- $m(T_{AB}) = \{ACE, BD\} = T'_1$
- $m(T_{AC}) = m(T_{DE}) = \{ACD, BE\} = T'_2$
- $m(T_{AD}) = m(T_{CE}) = \{A; B; CE; D\} = T'_3$
- $m(T_{AE}) = m(T_{CD}) = \pi(I)$
- $m(T_{BC}) = m(T_{BE}) = \{A; BCDE\} = T'_4$
- $m(T_{BD}) = \{AC; BD; E\} = T'_5$
- \( m(T_{AB}) = \{ACE, BD\} = T'_1 \)
- \( m(T_{AC}) = m(T_{DE}) = \{ACD, BE\} = T'_2 \)
- \( m(T_{AD}) = m(T_{CE}) = \{A; B; CE; D\} = T'_3 \)
- \( m(T_{AE}) = m(T_{CD}) = \pi(I) \)
- \( m(T_{BC}) = m(T_{BE}) = \{A; BCDE\} = T'_4 \)
- \( m(T_{BD}) = \{AC; BD; E\} = T'_5 \)

- \( M(T'_1) = T_{AB} + T_{AD} + T_{CD} + T_{BD} = \{ABD; CE\} = T_1 \)

In other words, \( M(T'_1) \) is the largest partition from which the block of \( T'_1 \) containing the next state of the machine can be determined.

- \( M(T') \) represents least amount of information such that \((M(T'), T')\) can be partition pair.
Information Flow Inequality

- If the next state variable, $Y_i$, can be computed from the external inputs and a subset $P_i$ of the variables then

\[
\pi T (y_j) \leq M [T (y_i)]
\]

Where the product is taken over all $T (y_j)$, such that $y_j$ is contained in the subset $P_i$.

- Verbally

Smallest partition (Max. no. of blocks) that contains the next state induced by variable(s) $Y_j$ \leq Largest partition (least no. of blocks) containing the next state of partition induced by $Y_i$. 