Asynchronous Circuits

- *Asynchronous sequential circuits* do not use a clock or flip-flops for state variables.
- Changes in state occur in response to changes on the inputs.
- This chapter describes *single input change (SIC) fundamental-mode async.* circuits.
- Only one input allowed to change at a time.
- Time between input changes must be sufficient for the circuit to stabilize.

### Terminology

- state table = *flow table*
- state-assigned table = *transition table* or *excitation table.*
Figure 8.90 The general model for a sequential circuit

Figure 9.4 The gated D latch

(a) Circuit

(b) Excitation table

(c) Flow table

(d) State diagram

Figure 9.5 Circuit for the master-slave D flip-flop

(a) Excitation table

(b) Flow table

(c) Flow table with unspecified entries

Figure 9.6 Excitation and flow tables for Example 9.2
Figure 9.7 State diagram for the master-slave D flip-flop

Figure 9.8 Circuit for Example 9.3

Figure 9.9 Excitation and flow tables

Figure 9.10 Modified flow table for Example 9.3

Figure 9.11 State table for Example 9.3

Figure 9.12 Flow table for a simple vending machine
Steps in the Analysis Process

- Cut feedback paths and insert delay element.
  - Input to delay element is next-state, and output is the present-state.
  - Cut set may not be unique.
- Derive next-state and output expressions from the circuit.
- Derive the excitation table.
- Derive a flow table.
- Derive a state diagram, if desired.

Synthesis of Asynchronous Circuits

- Devise a state diagram.
- Derive the flow table.
- Minimize number of states.
- Perform race-free state assignment.
- Derive excitation table.
- Obtain next-state and output expressions.
- Construct a hazard-free circuit.

Figure 9.13 Parity-generating asynchronous FSM

![Parity-generating asynchronous FSM](image)

(a) State diagram

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Flow table

Figure 9.14 State assignment

![State assignment](image)

(a) Poor state assignment

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Good state assignment

Figure 9.15 Circuit that implements an FSM

![Circuit that implements an FSM](image)
Figure 9.16  Synchronous solution for Example 9.4

Figure 9.17  State diagram for a modulo-4 counter

Figure 9.18  Flow and excitation tables for a modulo-4 counter

(a) Flowtable

(b) Excitation table

(c) Output for counting

Figure 9.19  Arbitration example

(a) Arbitration structure

(b) Handshake signaling
Figure 9.21 Implementation of the arbiter

(a) Flow table

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C B C</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>A C C</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>A B C</td>
<td>10</td>
</tr>
</tbody>
</table>

(b) Excitation table

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00 10 01 11</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>00 00 01 10</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>00 00 00 01</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>01 00 10 10</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 9.22 The arbiter circuit

(a) Flow diagram

(b) Modified flow table

Figure 9.23 An alternative for avoiding a critical race

(a) Modified flow table

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00 01 10 11</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>00 00 01 10</td>
<td>01</td>
</tr>
<tr>
<td>00</td>
<td>00 00 00 01</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>01 00 00 01</td>
<td>10</td>
</tr>
</tbody>
</table>

(b) Modified excitation table

Figure 9.24 Mealy model for the arbiter FSM

(a) Flow diagram

(b) Modified flow table

Figure 9.25 Mealy model implementation of the arbiter FSM

State Reduction

- Usually start with primitive flow table (i.e., only a single stable state per row).
- Asynchronous FSMs likely have many unspecified (don’t care) entries.
- Two-step state reduction process:
  - Apply partitioning procedure in which don’t care entries must match.
  - Merge rows exploiting don’t cares.
Merging Procedure

- May be many possibilities for row mergers.
- Two states $S_i$ and $S_j$ are compatible if there are no state conflicts for any input.
  - both $S_i$ and $S_j$ have same successor, or
  - both $S_i$ and $S_j$ are stable, or
  - the successor of $S_i$ or $S_j$ or both is unspecified.
- $S_i$ and $S_j$ must also have same output when specified.
State Reduction Procedure

1. Use partitioning procedure to eliminate equivalent states in primitive flow table.
2. Construct merger diagram.
3. Choose subsets of equivalent states including each state in only one subset.
4. Derive reduced flow table.
5. Repeat 2 to 4 until no reduction.
Figure 9.34 Merger diagram

Figure 9.35 Reduction obtained from the merger diagram

Figure 9.36 Merger diagram

Figure 9.37 Reduced flow table for Example 9.8

Figure 9.38 Flow table for Example 9.9

Figure 9.39 Reduction resulting from the partitioning procedure
(a) Preserving the Moore model

(b) Complete merger diagram

Figure 9.40 Merger diagrams

Figure 9.41 An FSM specified in the form of a Mealy model

Figure 9.42 Reduced flow table for Example 9.9

Figure 9.43 Flow table for Example 9.9

Figure 9.44 Reduction after the partitioning procedure

Figure 9.45 Merger diagram
State Assignment

- Impossible to ensure that a change of two or more state variables occur at the same time.
- To achieve reliable operation, should make state variables change one at a time.
- Hamming distance is number of bits different in two bit strings.
- Ideal state assignment has Hamming distance of 1 for all state transitions.

Transition Diagram

- Transition diagram illustrates all transitions in a flow table.
- Good state assignment means no diagonals in the transition diagram.
- Must embed the transition diagram onto a $k$-dimensional cube.
- $n$ state variables can be embedded onto an $n$-dimensional cube.
Deriving Transition Diagrams

- Derive relabeled flow table.
  - Transitions through unstable states that lead to stable state are given the same number.
- Represent each row of flow table by vertex.
- Join $V_i$ and $V_j$ by edge if they have same number in any column.
- For any column in which $V_i$ and $V_j$ have same number, label edge with that number.

---

**Figure 9.50** Relabeled flow table

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state $r_{ij}$</th>
<th>Output $s_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 2 4 3</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>1 2 4 3</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>1 2 4 5</td>
<td>10</td>
</tr>
</tbody>
</table>

**Figure 9.51** Transition diagrams

(a) Transitions in Figure 9.50
(b) Complete transition diagram

---

**Figure 9.52** Flow tables

(a) Flow table
(b) Relabeled flow table

---

**Figure 9.53** Transition diagrams

(a) First transition diagram
(b) Second transition diagram
(c) Augmented transition diagram

---

**Figure 9.54** Realization of an FSM

(a) Modified flow table
(b) Excitation table
### Figure 9.55
FSM for Example 9.14

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 A</td>
<td>A, C, B</td>
<td>00</td>
</tr>
<tr>
<td>01 B</td>
<td>A, D, B</td>
<td>01</td>
</tr>
<tr>
<td>10 C</td>
<td>C, B, D</td>
<td>10</td>
</tr>
<tr>
<td>11 D</td>
<td>C, A, D</td>
<td>11</td>
</tr>
</tbody>
</table>

**Present**

- state w
- state w
- state z
- state z

**Next state**

- state w
- state w
- state z
- state z

**Output**

- 00
- 01
- 10
- 11

---

### Figure 9.56
Transition diagrams

(a) Transition diagram

(b) Augmented transition diagram

(c) Embedded transition diagram

---

### Figure 9.57 Modified tables for Example 9.14

**Modified flow table**

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 A</td>
<td>A, A, C, B</td>
<td>00</td>
</tr>
<tr>
<td>01 B</td>
<td>A, B, E, B</td>
<td>01</td>
</tr>
<tr>
<td>10 C</td>
<td>C, F, C, G</td>
<td>10</td>
</tr>
<tr>
<td>11 D</td>
<td>G, A, D</td>
<td>11</td>
</tr>
<tr>
<td>01 E</td>
<td>- , D, -</td>
<td>-1</td>
</tr>
<tr>
<td>01 F</td>
<td>- , B, -</td>
<td>01</td>
</tr>
<tr>
<td>01 G</td>
<td>C, - , D</td>
<td>1-</td>
</tr>
</tbody>
</table>

**Excitation table**

- 00
- 01
- 10
- 11

---

### Figure 9.58
Embedded transition diagram if two nodes per row are used

---

### Figure 9.59 Modified flow and excitation tables for Example 9.15

**Modified flow table**

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 A1</td>
<td>A, A, C1, B1</td>
<td>00</td>
</tr>
<tr>
<td>01 A2</td>
<td>A, A, A2, A1, B2</td>
<td>00</td>
</tr>
<tr>
<td>10 B1</td>
<td>A1, B2, B1, D1</td>
<td>01</td>
</tr>
<tr>
<td>10 B2</td>
<td>A2, D2, D2, D2</td>
<td>10</td>
</tr>
<tr>
<td>10 C1</td>
<td>C1, C2, D2, D1</td>
<td>10</td>
</tr>
<tr>
<td>11 C2</td>
<td>C2, B1, D2, D2</td>
<td>11</td>
</tr>
<tr>
<td>11 D1</td>
<td>C1, A2, D2, D2</td>
<td>11</td>
</tr>
<tr>
<td>11 D2</td>
<td>C2, D1, D2, D2</td>
<td>11</td>
</tr>
</tbody>
</table>

**Excitation table**

- 00
- 01
- 10
- 11
Hazards

- In asynchronous circuits, undesirable glitches must not occur.
- Glitches caused by structure of circuit and propagation delays are called hazards.
- Designer must eliminate all hazards from an asynchronous circuit.

Remove Static 1-Hazards

- Hazard exists whenever 2 adjacent 1s in a K-map are not covered by a single product.
- To remove all static hazards, find a cover that includes each pair of adjacent 1s.
Figure 9.63  Two-level implementation of master-slave D flip-flop

Figure 9.64  Function for Example 9.17

Figure 9.65  Static hazard in a POS circuit

Figure 9.66  Circuit with a dynamic hazard

Significance of Hazards

- A glitch in an asynchronous circuit can cause the circuit to enter an incorrect state and possibly become stable in that state.
- Next-state logic must be hazard-free.
- Synchronous circuits can have hazards as long as they are stable by the setup time of the flip-flops.
Vending Machine Controller

- It accepts nickels and dimes.
- A total of 15 cents to release candy.
- No change given if 20 cents is deposited.
- Coins deposited one at a time.

\[ \text{Figure 9.67} \quad \text{Initial state diagram for the vending-machine controller} \]

\[ \text{Figure 9.68} \quad \text{Initial flow table for the vending-machine controller} \]

\[ \text{Figure 9.69} \quad \text{First step in state minimization} \]

\[ \text{Figure 9.70} \quad \text{Merger diagram} \]

\[ \text{Figure 9.71} \quad \text{Reduced flow tables} \]
Figure 9.72  State diagram for the vending machine controller

Figure 9.73  Determination of the state assignment

Figure 9.74  Excitation table

Figure 9.75  Karnaugh maps for the functions in Figure 9.74
Summary

- Analysis of asynchronous circuits.
- Synthesis of asynchronous circuits.
  – State reduction
  – State assignment
  – Hazard-free logic design