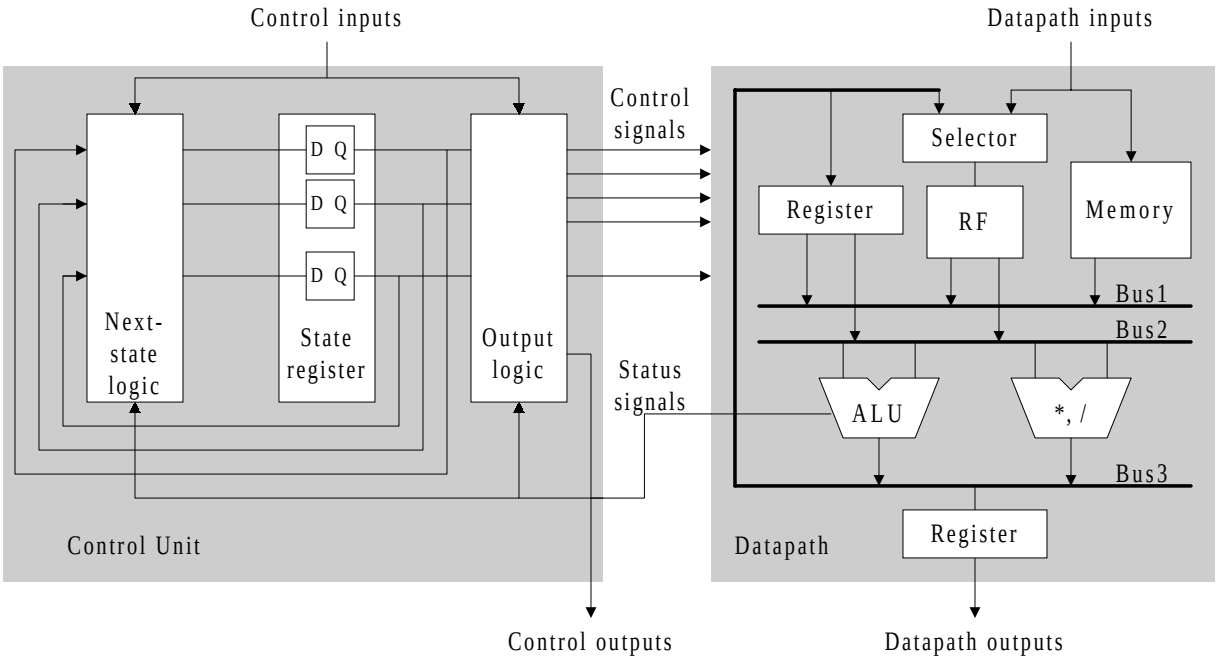
## 8.1 Register-Transfer Design Model

Every digital design consists of a control unit and a datapath.




In the FSM model, each state assigns values to a set of datapath control signals which completely specifies the behavior of the datapath.

However, when there are too many control signals it is difficult to realize what and how the datapath will operate. (See figure 7.30)

To improve on the FSM model, we use variable assignment statements to indicate changes in variable values stored in the datapath.

An **FSMD** (**FSM with datapath**) is a FSM model with assignment statements added to each state.

Formally, a finite-state machine with datapath is a 6-tuple defined as follows:

$$P = <S, s_0, I \cup STAT, O \cup A, f, h>$$

compare: FSM is defined as $P = <S, s_0, I, O, f, h>$

where:

- $S = \{s_0, \ldots, s_m\}$ is a finite set of *states*
- $s_0$ is the *reset state*.
- $I = \{i_j\}$ is a set of primary input values.
- $STAT = \{Rel(a, b) : a, b \in EXP\}$ is a set of status signals as logical relations between two expressions from the set *EXP*.
- $EXP = \{f(x,y,z,\ldots) : x,y,z,\ldots \in VAR\}$ is a set of expressions.
- *VAR* is a set of storage variables.
- $O = \{o_k\}$ is a set of primary output values.
- $A = \{x \Leftarrow e : x \in VAR, e \in EXP\}$ is a set of storage assignments.
- *f* is a *state transition* function that maps a cross product of *S* and $I \cup STAT$ into *S*.
- *h* is the *output function* that maps a cross product of *S* and $I \cup STAT$ into $O \cup A$ for Mealy models or *S* into $O \cup A$ for Moore models.

So instead of a separate next-state table and output table, we combine the two to give us a "next-state and output table with variable assignments."

| Current State | Next State Start, Data=0 | Control Output | Datapath Output | Datapath Variables |
|---|---|---|---|---|
| | 00 01 10 11 | Done | Output | |
| $s_0$ | $s_0\ s_0\ s_1\ s_1$ | 0 | $Z$ | |
| $s_1$ | $s_2\ s_2\ s_2\ s_2$ | 0 | $Z$ | Data = Input |
| $s_2$ | $s_3\ s_3\ s_3\ s_3$ | 0 | $Z$ | Ocount = 0 |
| $s_3$ | $s_4\ s_4\ s_4\ s_4$ | 0 | $Z$ | Mask = 1 |
| $s_4$ | $s_5\ s_5\ s_5\ s_5$ | 0 | $Z$ | Temp = Data AND Mask |
| $s_5$ | $s_6\ s_6\ s_6\ s_6$ | 0 | $Z$ | Ocount = Ocount + Temp |
| $s_6$ | $s_4\ s_7\ s_4\ s_7$ | 0 | $Z$ | Data = Data >> 1 |
| $s_7$ | $s_0\ s_0\ s_0\ s_0$ | 1 | Ocount | |

Figure 8.3 (b) State and output table with variable assignments – FSMD specification of one's counter.

We assume that variables retain their old values if no new value is specified in a particular state.

- Note that although expression evaluation is performed in one state, the new variable value could not be used until the next state at the earliest.

Similarly, we do not have to specify the next state for every control input and every status signal, but only for those that affect the next state selection.

- Thus, we can simplify the next-state column by specifying in each state only the condition and the next state that the control unit will enter if the condition is true.

We can also use assignment statements for datapath and control unit output ports.

- Unlike variables in the datapath, output ports do not retain their value beyond the present state since the values are not stored in registers or memory.

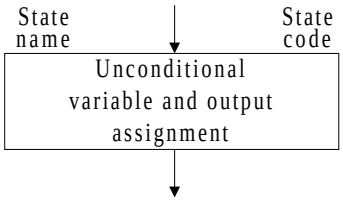
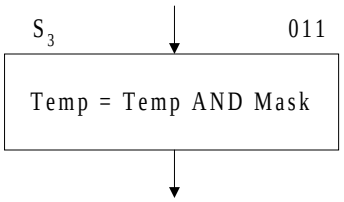
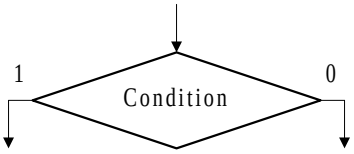
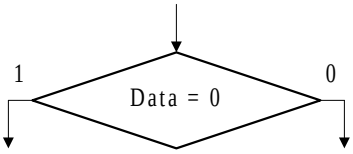
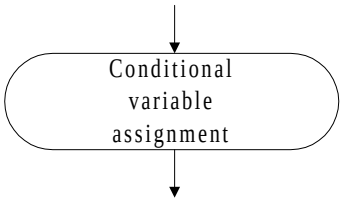
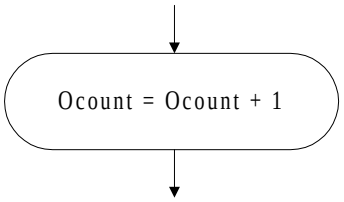
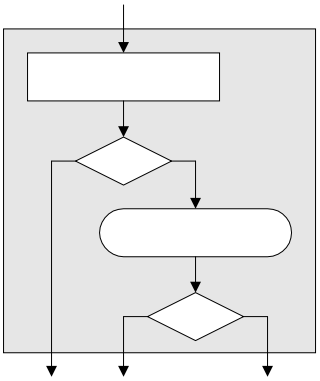
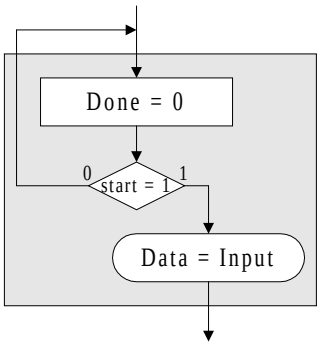
From the above observations, we obtain a reduced table called a **state-action table** for specifying FSMDs.

| Current State | Next State | Control and Datapath Actions |
|---|---|---|
| | Condition, State | Condition, Actions |
| $s_0$ | $\begin{bmatrix} Start = 0, s_0 \\ Start = 1, s_1 \end{bmatrix}$ | $\begin{bmatrix} Done = 0 \\ Output = Z \end{bmatrix}$ |
| $s_1$ | $s_2$ | Data = Input |
| $s_2$ | $s_3$ | Ocount = 0 |
| $s_3$ | $s_4$ | Mask = 1 |
| $s_4$ | $s_5$ | Temp = Data AND Mask |
| $s_5$ | $s_6$ | Ocount = Ocount + Temp |
| $s_6$ | $\begin{bmatrix} Data \neq 0, s_4 \\ Data = 0, s_7 \end{bmatrix}$ | Data = Data >> 1 |
| $s_7$ | $s_0$ | $\begin{bmatrix} Done = 1 \\ Output = Ocount \end{bmatrix}$ |

Figure 8.3 (c) State-action table – FSMD specification of one's counter.

## 8.3 Algorithmic-state machine (ASM) chart

ASM chart is an alternative graphic form for specifying FSMDs and are equivalent to the state-action tables.

| Name | Definition | Example | Comment |
|---|---|---|---|
| State box (Unconditional assignment) |  |  | Each state is indicated by a state box.<br>Contains the set of unconditional assignments to variables and output ports in the datapath. |
| Decision box |  |  | To select specific actions in the datapath and the next state. |
| Condition box (Conditional assignment) |  |  | Variable or output assignments that are executed under conditions specified by one or more decision boxes. |
| ASM block |  |  | Each ASM block describes the operations executed in one state.<br>This is equivalent to a row in the state-action table. |

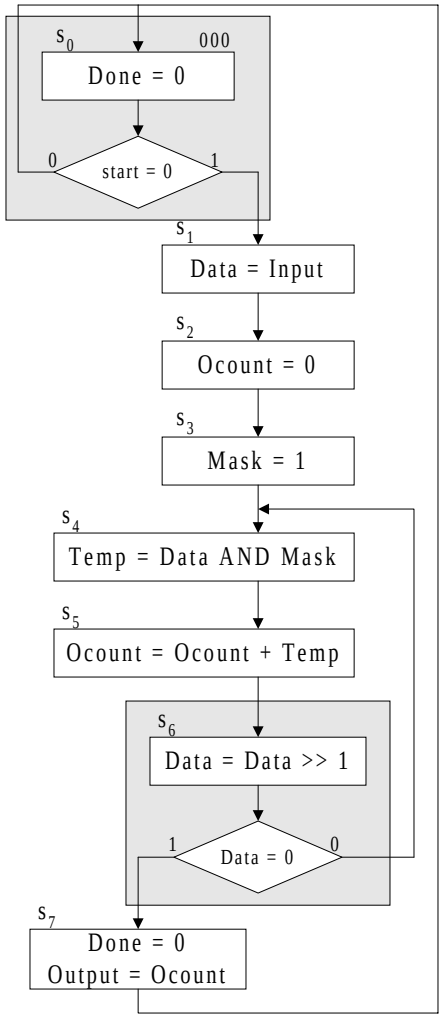



Figure 8.5 ASM chart for one's counter.

Example 8.1 Redesign the one's counter using a custom datapath.

Solution.

- Need only two variables: Data and Ocount.
- Need only two operations: shift the value in Data, and increment the value in Ocount.
- In the state-based (Moore) version of the FSMD:
  - all the variable assignments must be executed unconditionally in a state and only next states are to be selected conditionally.
  - has six states → more state registers → next-state logic more complex.
  - output logic is simpler since it is dependent only on the present state.
- In the input-based (Mealy) version of the FSMD:
  - the variable assignments can be executed conditionally together with the conditional selection of next states.
  - has four states → simpler next-state logic.
  - output logic is more complex since it includes external and internal conditions.

State-base (Moore)

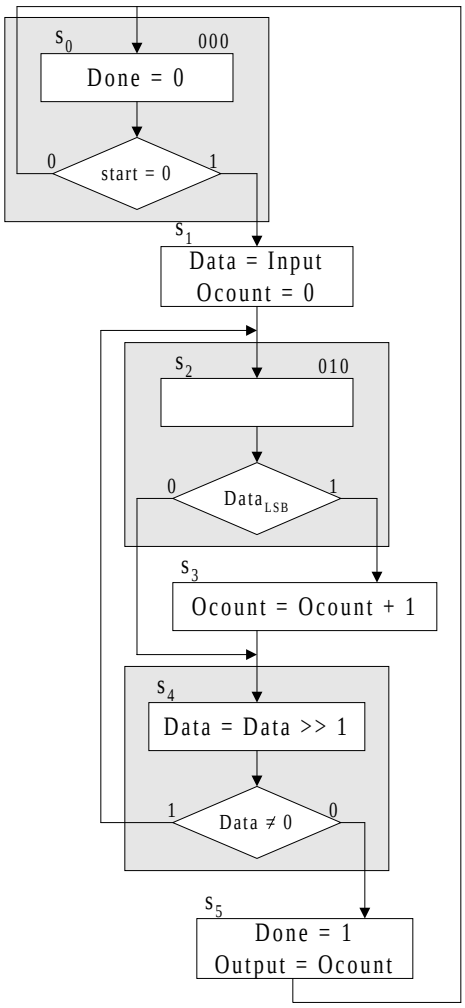



Figure 8.6 (a) State-based (Moore) ASM chart.

| Current State $Q_2Q_1Q_0$ Name | Next State Condition, State | Control and Datapath Actions Condition, Actions |
|---|---|---|
| 0 0 0 $s_0$ | $\begin{bmatrix} Start=0, s_0 \\ Start=1, s_1 \end{bmatrix}$ | $\begin{bmatrix} Done=0 \\ Output=Z \end{bmatrix}$ |
| 0 0 1 $s_1$ | $s_2$ | $\begin{bmatrix} Data=Input \\ Ocount=0 \end{bmatrix}$ |
| 0 1 0 $s_2$ | $\begin{bmatrix} Data_{LSB}=1, s_3 \\ Data_{LSB}=0, s_4 \end{bmatrix}$ | |
| 0 1 1 $s_3$ | $s_4$ | Ocount = Ocount + 1 |
| 1 0 0 $s_4$ | $\begin{bmatrix} Data \neq 0, s_2 \\ Data=0, s_5 \end{bmatrix}$ | Data = Data >> 1 |
| 1 0 1 $s_5$ | $s_0$ | $\begin{bmatrix} Done=1 \\ Output=Ocount \end{bmatrix}$ |

Figure 8.7 (a) State-based state-action table for the one's counter.

State encodings:

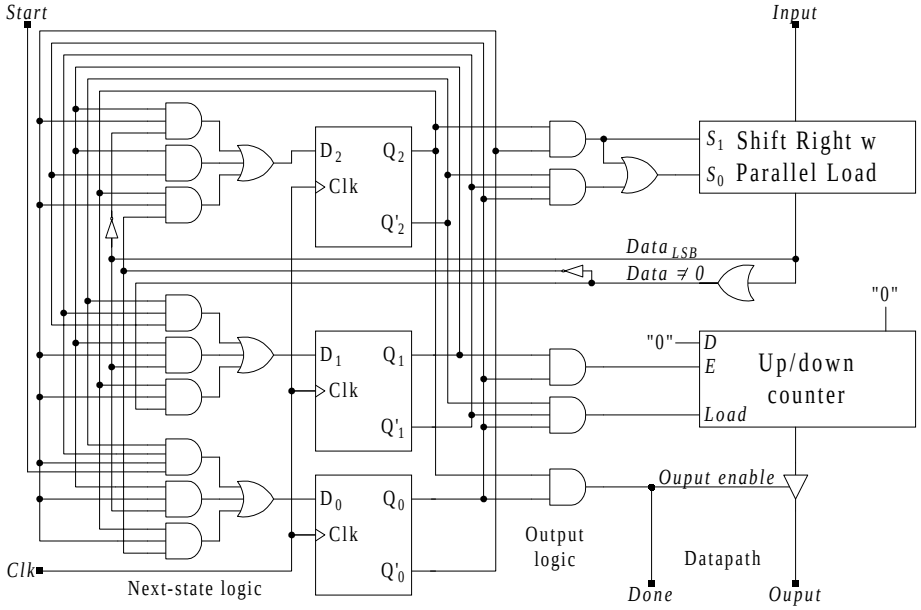
$s_0=Q_2'Q_1'Q_0'$ $s_1=Q_2'Q_1'Q_0$ $s_2=Q_1Q_0'$
$s_3=Q_1Q_0$ $s_4=Q_2Q_0'$ $s_5=Q_2Q_0$

Next-state equations using D flip-flops, $D_i = Q_{i(\text{next})}$

$D_2 = Q_{2(\text{next})}$ = (since $Q_2$ is encoded in $s_4$ and $s_5$, therefore look for next states being $s_4$ and $s_5$)
$= s_2Data_{LSB}' + s_3 + s_4(Data=0)$
$= Q_1Q_0'Data_{LSB}' + Q_1Q_0 + Q_2Q_0'(Data=0)$

$D_1 = Q_{1(\text{next})}$ = ($Q_1$ is encoded in $s_2$ and $s_3$)
$= s_1 + s_2Data_{LSB} + s_4(Data\neq 0)$
$= Q_2'Q_1'Q_0 + Q_1Q_0'Data_{LSB} + Q_2Q_0'(Data\neq 0)$

$D_0 = Q_{0(\text{next})}$ = ($Q_0$ is encoded in $s_1$, $s_3$and $s_5$)
$= s_0Start + s_2Data_{LSB} + s_4(Data=0)$
$= Q_2'Q_1'Q_0'Start + Q_1Q_0'Data_{LSB} + Q_2Q_0'(Data=0)$




Output equations:

Recall from 7.3 – select lines:
$S_1S_0 = 01$ = load data
$S_1S_0 = 11$ = shift right

Thus, when $S_1 = 1$ (we want to do shift which is done in state $s_4$)
$S_1 = s_4 = Q_2Q_0'$
$S_0 = 1$ (load & shift – in $s_1$&$s_4$)
$= s_1+s_4 = Q_2'Q_1'Q_0 + Q_2Q_0'$
$E = 1$ (needed in $s_3$)
$= s_3 = Q_1Q_0$
$Load = s_1 = Q_2'Q_1'Q_0$
$Done = Output\ enable$
$= s_5 = Q_2Q_0$

Input-base (Mealy)

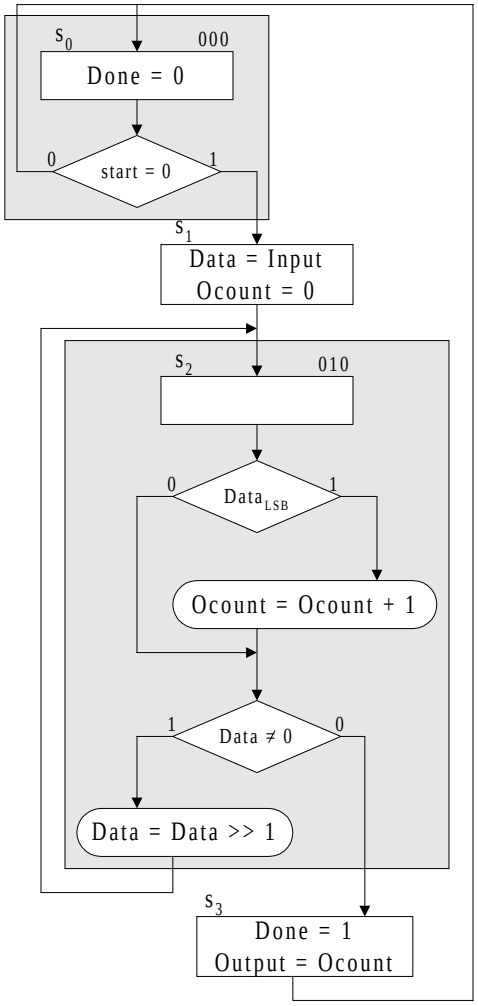


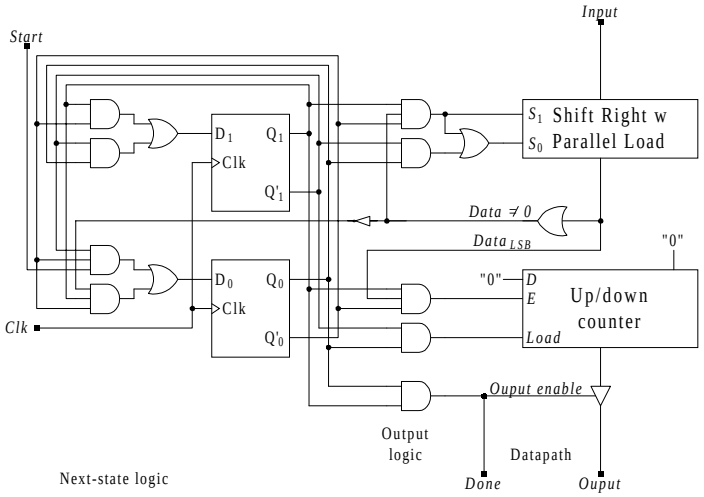

Figure 8.6 (b) Input-based (Mealy) ASM chart.

| Current State $Q_1Q_0$ | Next State<br>Condition, State | Control and Datapath Actions<br>Condition, Actions |
|---|---|---|
| 0 0 $s_0$ | $\begin{bmatrix} Start = 0, s_0 \\ Start = 1, s_1 \end{bmatrix}$ | $\begin{bmatrix} Done = 0 \\ Output = Z \end{bmatrix}$ |
| 0 1 $s_1$ | $s_2$ | $\begin{bmatrix} Data = Input \\ Ocount = 0 \end{bmatrix}$ |
| 1 0 $s_2$ | $\begin{bmatrix} Data \neq 0, s_2 \\ Data = 0, s_3 \end{bmatrix}$ | $\begin{bmatrix} Data_{LSB} = 1, Ocount = Ocount + 1 \\ Data \neq 0, Data = Data >> 1 \end{bmatrix}$ |
| 1 1 $s_3$ | $s_0$ | $\begin{bmatrix} Done = 1 \\ Output = Ocount \end{bmatrix}$ |

Figure 8.7 (b) Input-based state-action table for the one's counter.

State encodings:
$s_0$=$Q_1$'$Q_0$' $s_1$=$Q_1$'$Q_0$ $s_2$=$Q_1Q_0$' $s_3$=$Q_1Q_0$

Next-state equations using D flip-flops, $D_i = Q_{i(\text{next})}$
$D_1 = Q_{1(\text{next})}$ = (since $Q_1$ is encoded in $s_2$ and $s_3$, therefore look for next states being $s_2$ and $s_3$)
= $s_1 + s_2(Data=0) + s_2(Data\neq 0) = s1 + s2$
= $Q_1$'$Q_0 + Q_1Q_0$'
$D_0 = Q_{0(\text{next})}$ = ($Q_0$ is encoded in $s_1$ and $s_3$)
= $s_0 Start + s_2(Data=0)$
= $Q_1$'$Q_0$'$Start + Q_1Q_0$'($Data$=0)




Output equations:
Recall from 7.3 – select lines:
$S_1S_0$ = 01 = load data
$S_1S_0$ = 11 = shift right

Thus, when $S_1$ = 1 (we want to do shift which is done in state $s_2$ and when $Data\neq 0$)
$S_1 = s_2(Data\neq 0)$
= $Q_1Q_0$'($Data\neq 0$)
$S_0$ = 1 (load & shift – in $s_1$&$s_2$)
= $s_1+s_2(Data\neq 0)$
= $Q_1$'$Q_0 + Q_1Q_0$'($Data\neq 0$)
$E$ = 1 (increment needed in $s_2$ and when $Data_{LSB}$=1)
= $s_2(Data_{LSB})$
= $Q_1Q_0$'($Data_{LSB}$)
$Load = s_1 = Q_1$'$Q_0$
$Done = Output\ enable$
= $s_3 = Q_1Q_0$