7.3

ASM Chart Notation

The algorithmic state machine method uses straightforward procedures to create reliable sequential state machines. Via the ASM method we can develop designs with style and elegance. However, the ASM method does not eliminate the need for creativity. Creativity is required to construct the basic algorithm that solves the current problem. At the same time, the ASM method is an excellent assistant during the creative phase. Furthermore, the ASM chart notation consists of only three symbols: state, branch, and conditional output.

State The rectangle is state $s_j$’s symbol (Figure 7.8a). The symbol has only one entrance and one exit.

For a synchronous state machine each active clock transition causes a change of state from the present state to the next state. Given the present state, the next state must be determined without ambiguity for any values of state and input variables. Having arrived at the next state, this next state becomes the present state.

Whereas any number of paths may lead to a state rectangle’s single entry point, only one path may lead away from the state rectangle’s one and only exit point.

Branch The diamond is the branch symbol (Figure 7.8b). The extended diamond is a compact form representing a tree of diamonds (Figures 7.8b and 7.9b). Decision box is an alias for branch diamond.

The diamond symbol with True and False exit paths represents a decision. The condition placed in the box may be any Boolean function of input variables. The True (False) exit path is taken when the condition is True (False). The extended diamond has one exit path per minterm of the input variables involved in the decision. (These are not minterms of all the state machine input variables.) In Figure 7.8b, a decision box with two variables $x, y$ has one exit for each minterm ($xy, x'y, xy'$, and $xy$). Boolean variables such as $x$ in a branch function (Figure 7.8d) usually represent inputs from the outside world. In addition, the inputs can originate from within the same logic machine.

Suppose $y$ is a “don’t care” when $x$ is True. This means the two exits corresponding to minterms $xy'$ and $xy$ merge into one exit (Figure 7.8a). This is why the $y$ diamond on the $x = T$ side of the tree can be removed.

Each branch diamond has only one entry point. A branch exit path leads either to only one other branch diamond or to only one (next) state so that the next state is determined without ambiguity for any values of state and input variables. Only one of the parallel exit paths can be active if the next state is to be uniquely determined.

NOTES TAKEN FROM:
N. L. Pappas
DIGITAL DESIGN
West Publishing Company, 1994
7.3 ASM Chart Notation

(a) State

(b) Branch

(c) Conditional output

(d) ASM block

(e) Trees

Figure 7.8
ASM chart notation
Figure 7.9
Incorrect and correct versions of an ASM binary tree

- (a) Incorrect: always two simultaneous exit paths (x and y independent)

- (b) Correct: always only one exit path at a time
7.3 ASM Chart Notation

An input variable does not have to be associated with every decision. Input variables not involved in a decision are treated as "don't cares" in that decision. For example, there may be five input variables, but only \( x, y \) are associated with some state.

A branch must be associated with a state to have meaning. There must be only one possible next state for each set of input conditions.

**Fundamental Rule**  Every path must lead to only one state. Therefore we activate only one path at a time.

When this is not the case, the logic machine cannot select a unique next state, which is a design error (Figure 7.9a).

The branch is activated during the entire state time. *State time* is the clock period prior to the next clock edge that executes the decision. The exit path selected by the input values immediately before the clock edge occurs determines the next state.

**Unconditional output** One or more unconditional outputs are specified by entering the outputs’ names inside the desired state rectangle (Figure 7.8a). There is no special symbol for an unconditional output; the only symbol is the name.

Unconditional outputs depend solely on the state. They do not depend on inputs in any way. Unconditional outputs are active during the associated state time. (In state diagram terms, an ASM machine with only unconditional outputs is a Moore machine.)

*Note:* When there are no unconditional outputs, a state is an "empty box." We do not delete the empty box from the ASM chart simply because it is empty. There may be other reasons why the state exists.

**Conditional output** The oval is the symbol of conditional output (Figure 7.8c). The oval is always associated with a branch diamond. Conditional output names are entered in the oval. A conditional output is active during a state time if the associated branch condition is True during that state time. The output is conditioned by the state and branch that feeds it. (In state diagram terms, an ASM machine with only conditional outputs is a Mealy machine.) In Figure 7.8d output HELLO is active when the system is in state \( s \), and when \( x \) is True.

*Note:* When there are no conditional outputs, the oval is "empty." The empty oval is deleted because there are no other possible reasons for its presence.
ASM block  An ASM block consists of one entry line, one state rectangle, and any number of associated branch diamonds, conditional output ovals, and mutually exclusive exit lines (Figure 7.8d). Any ASM chart is an assembly of ASM blocks connected by paths (see Example 7.3).

An ASM block representing state \( j \) describes the state machine's operation while the state machine is in state \( j \). The time spent in state \( j \) may be an indefinite number of clock periods when a branch diamond is associated with the state. The time spent in state \( j \) ends when the input variables close the path that returns to the present state \( j \) and open a path to a next state \( k \).

In an ASM chart the operations described in an ASM block are executed simultaneously, i.e., in parallel. In the ASM block of Figure 7.8d the following events occur during time spent in state \( s_7 \): Unconditional output HIT_ME is activated, and, in parallel, input variables \( x \) and \( y \) are evaluated simultaneously, activating the path corresponding to the \( f(x, y) \) minterm that is True. When either minterm \( m_2 \) or \( m_3 \) is True, \( x \) is True. This activates conditional output HELLO. When \( m_0 \) or \( m_4 \) is true, HELLO is not activated and the \( x'y' \) or \( x'y \) exit is taken. Until this ASM block is embedded in an ASM chart, and the characteristics of input \( x \) are known, we do not know how many clock periods are spent in state \( s_7 \). This differs from a flow chart, where the events occur sequentially and event duration is not specified.

---

**Example 7.3 From Statement to ASM Chart**

A state machine is needed to cycle through states \( s_0, s_1, s_2, \) and \( s_3 \) when input variable \( x \) is asserted, and to cycle through states \( s_0 \) and \( s_3 \) when input variable \( x \) is not asserted. The essential deduction is that \( s_1 \) and \( s_2 \) are skipped when \( x \) is not asserted. That is, from present state \( s_0 \) the next state is \( s_1 \) when \( x = T \), and when \( x = F \) the next state is \( s_3 \). Because \( x \) is evaluated in state \( s_0 \), input variable \( x \) is active in, associated with, \( s_0 \). Furthermore, the machine steps unconditionally from \( s_1 \) to \( s_2 \), \( s_2 \) to \( s_3 \), and \( s_3 \) to \( s_0 \). The ASM chart follows.

Two ASM charts are shown [with and without clock (ck) as an input]. When clock is omitted as an input associated with every state, the implicit assumption is that the state machine is a synchronous machine. The ASM chart with clock omitted as an input to every state is not only simplified significantly, it is clear.
EXAMPLE 7.4 From Sequencer Statement to ASM Chart

Suppose a sequential circuit has one input $x$ and one output $g$. Output $g$ is asserted whenever the most recent inputs are 111, where the most
recent input is the last digit in the string. Overlapping of 111 sequences is not allowed, but \textit{"adjacent"} 111 sequences can occur (e.g., . . . 01111110 . . . produces . . . 00010010 . . . ).

The number of states required is not known at the outset. Let state $s_0$ be the rest state. The circuit remains in the $s_0$ state when zeros are received ($x = 0$). Let the machine advance to state $s_1$ when a one is received ($x = 1$). Therefore $s_1$ represents one 1 received. We draw $s_0$ and $s_1$ state boxes and a branch diamond with input $x$ at the $s_0$ exit. We complete the path from $s_0$ to $s_0$ by drawing an arrow from the diamond $x$-equals-0 output to the $s_0$ state box entrance. We draw an arrow from the diamond $x$-equals-1 output to the $s_1$ state box entrance.

If a zero is received while in $s_1$, a return to $s_0$ restarts the sequence detection process. At the $s_1$ state box exit we draw a branch diamond with input $x$. We complete the path from $s_1$ to $s_0$ by drawing an arrow from the diamond $x$-equals-0 output to the $s_0$ state box entrance. If a one is received, we advance to $s_2$ so that $s_2$ represents the fact that a sequence of two 1s has been received. We draw an arrow from the $s_1$ diamond $x$-equals-1 output to the $s_2$ state box entrance.
7.3 ASM Chart Notation

If a zero is received while in $s_2$, a return to $s_0$ restarts the sequence detection process. At the $s_2$ state box exit, we draw a branch diamond with input $x$. We complete the path from $s_2$ to $s_0$ by drawing an arrow from the diamond $x$-equals-0 output to the $s_0$ state box entrance. If a third one is received, we output a one ($g = 1$) on return to $s_0$, which then represents that a sequence of three 1s has been received. We draw an arrow from the $s_2$ diamond $x$-equals-1 output to an oval’s entrance. We enter $g$ in the oval. This will output the one report. We draw an arrow from the oval’s exit to the $s_0$ state box entrance.

When three state numbers are encoded with two binary digits, a fourth state ($s_3$) is possible. Good practice* dictates that all unused states have a next state by design. The “rest” state $s_0$ is a practical next state for unused states. So we add a state box for unused state $s_3$. We draw an arrow from the $s_3$ box exit to the $s_0$ state box entrance.

---

**Exercise 7.5**

Derive the ASM chart for Figure P6.2 on page 326.

*Answer:*

---

* $s_3$ can be entered when the power is turned on, in which case the next clock edge moves the machine to $s_0$ so that it is ready to execute the algorithm correctly.