Chapter 1

Algorithmic state machines

The counter designed in the last section could easily be described in terms of state changes. Most sequential systems are more complex and require a formal notation to fully describe their functionality. From this formal notation, a state table and hence Boolean expressions can be derived. There are a number of types of formal notation that may be used. We will briefly refer to one before introducing the principle technique used in this book – the *algorithmic state machine (ASM) chart*.

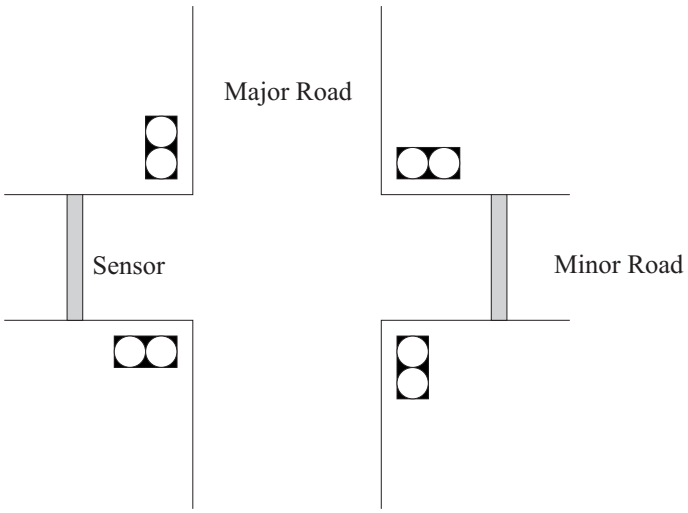

Figure 1.1: Traffic signal problem

The form of an ASM chart is best introduced by an example. Let us design a simple controller for a set of traffic signals, as shown in Figure 1.1. This example is significantly simpler than a real traffic signal controller (and would probably be more dangerous than an uncontrolled junction!).

The traffic signals have two lights each – red and green. The major road normally has a green light, while the minor road has a red light. If a car is detected on the minor road, the signals change to red for the major road and green for the minor road. When the lights change, a timer is started. Once that timer completes a 'TIMED' signal is asserted, which causes the lights to change back to their default state.

The functionality of this system can be described by the state machine diagram of Figure 1.2. This form of diagram is commonly used, but can be unclear. For some systems, such diagrams are sufficient. In this book, however, we will generally use ASM charts, which are much less ambiguous. The ASM chart for the traffic signal controller is shown in Figure 1.3.

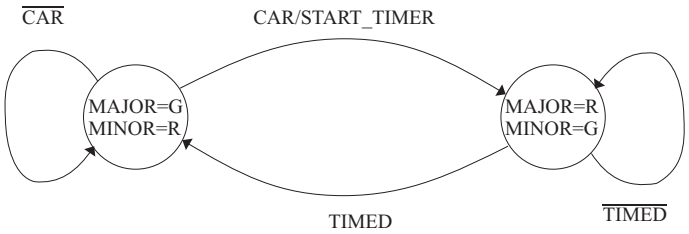

Figure 1.2: State machine of traffic signal controller

ASM charts resemble flow charts, but contain implicit timing information – the clock signal is not explicitly shown in Figure 1.3. It should be noted that ASM charts represent physical hardware. Therefore all transitions within the ASM chart must form closed paths – hardware cannot suddenly start or stop (the only exception to this might be a reset state to which the system never returns).

The basic component of an ASM chart is the state box, shown in Figure 1.4(a). The state takes exactly one clock cycle to complete. At the top left-hand corner the name of the state is shown. At the top right-hand corner the state assignment (see below) may be given. Within the state box, the output signals are listed. The signals take the values shown for the duration of the clock cycle and are reset to their default values for the next clock cycle. If a signal does not have a value assigned to it (e.g. Y), that signal is asserted (logic 1) during the state and is deasserted elsewhere. The notation $X \leftarrow 1$ means that the signal is assigned at the *end* of the state (i.e. during the next clock cycle) and holds its value until otherwise set elsewhere.

A decision box is shown in Figure 1.4(b). Two or more branches flow from the decision box. The decision is made from the value of one or more input signals. The decision box *must* follow and be associated with a state

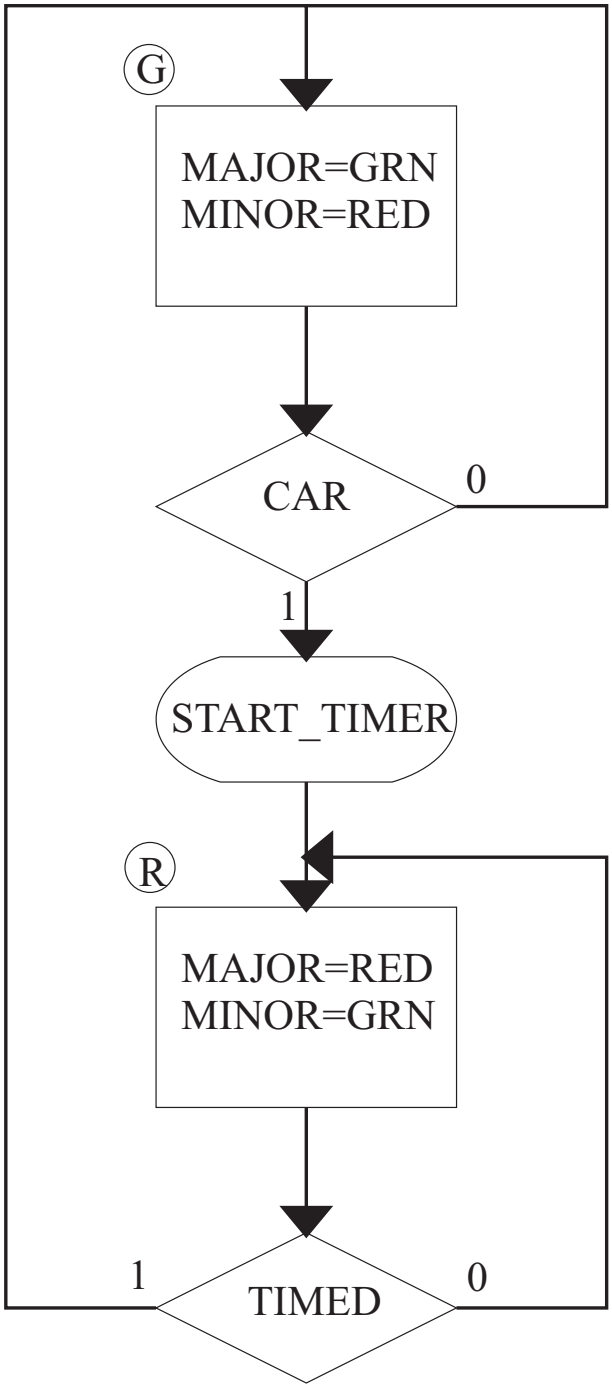

Figure 1.3: ASM chart of traffic signal controller

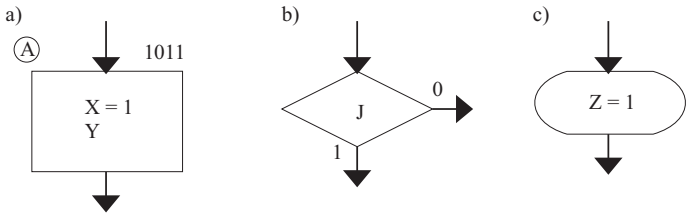

Figure 1.4: ASM chart symbols

box. Therefore the decision is made in the same clock cycle as the other actions of the state. Hence the input signals must be valid at the start of the clock cycle.

A conditional output box is shown in Figure 1.4(c). A conditional output must follow a decision box. Therefore the output signals in the conditional output box are asserted in the same clock cycle as those in the state box to which it is attached (via one or more decision boxes). The output signals can change during that state as a result of input changes. The conditional output signals are sometimes known as Mealy outputs because they are dependent on input signals, as in a Mealy machine.

It can therefore be seen that one state, or clock cycle, consists of more than just the state box. Decision boxes and conditional output boxes also form part of the state. Figure 1.3 can be redrawn, as in Figure 1.5, where all the components of a state are enclosed within dashed lines.

The difference between state boxes and conditional output boxes is illustrated in Figure 1.6. In Figure 1.6(a), there are two states. Output Y is asserted during the first state if input C is true or becomes true. In Figure 1.6(b) there are three states. The difference can be seen in the timing diagrams of Figure 1.7.

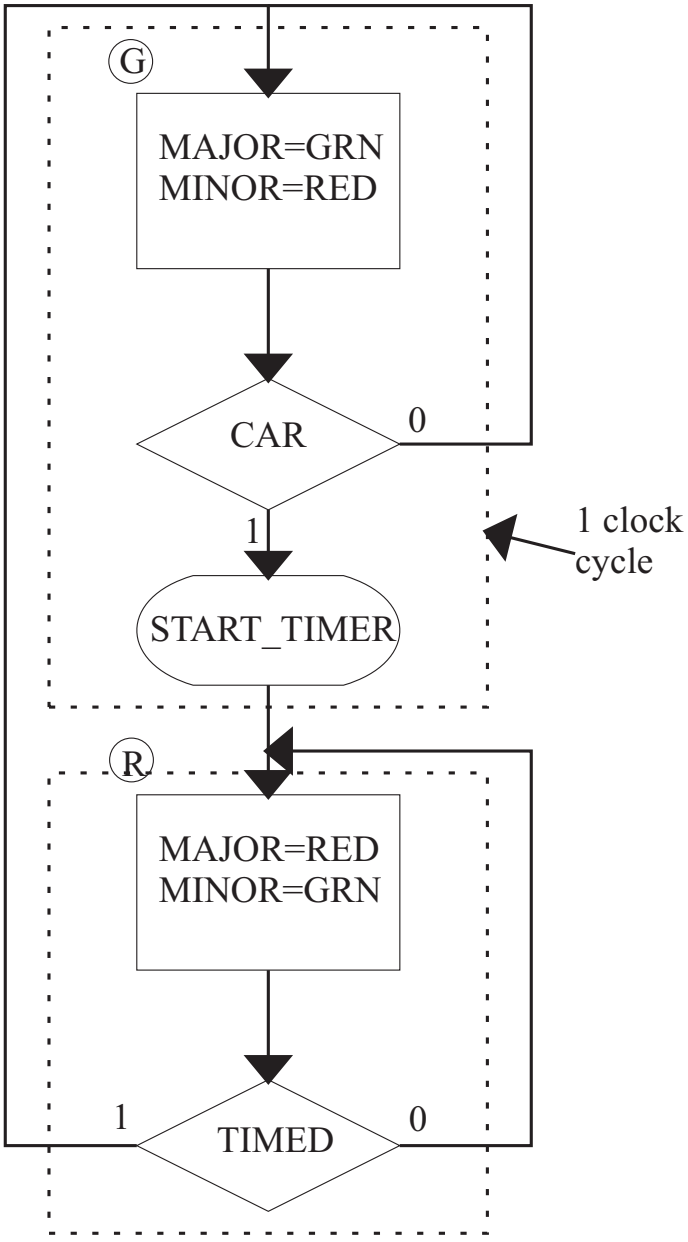

Figure 1.5: ASM chart showing clock cycles

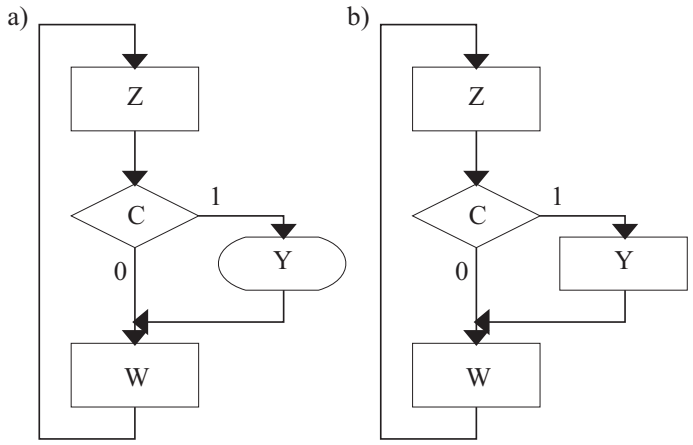

Figure 1.6: Conditional and unconditional outputs

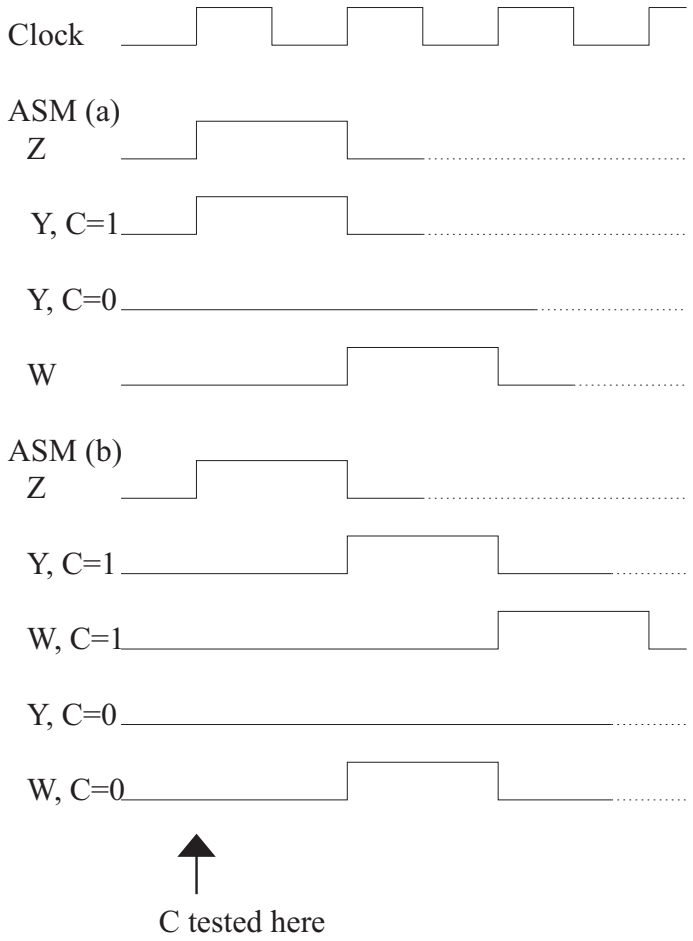

Figure 1.7: Timing diagram for Figure 1.6

Chapter 2

Linked state machines

In principle, any synchronous sequential system could be described by an ASM chart. In practice, this does not make sense. The states of a system, such as a microprocessor, include all the possible values of all the data that might be stored in the system. Therefore it is usual to partition a design in some way. In this chapter, we will show first how an ASM chart, and hence the SystemVerilog model of the state machine, can be partitioned, and second how a conceptual split may be made between the *datapath* of a system, i.e. the components that store and manipulate data, and the state machine that controls the functioning of those datapath components.

A large body of theory covers the optimal partitioning of state machines. In practice, it is usually sufficient to identify components that can easily be separated from the main design and implemented independently. For example, let us consider again the traffic signal controller.

If a car approaches the traffic signals on the minor road, a sensor is activated that causes the major road to have a red light and the minor road to have a green light for a fixed interval. Once that interval has passed, the major road has a green light again and the minor road has a red light. In Chapter 1, we simply assumed that a signal would be generated after the given interval had elapsed. Let us now assume that the clock frequency is such that the timed interval is completed in 256 clock cycles. We can draw an ASM chart for the entire system as shown in Figure 2.1 (states 1 to 254 and the outputs are not shown, for clarity). Although this is a simple example, the functionality of the system is somewhat lost in the profusion of states that implement a simple counting function. It would be clearer to separate the traffic light controller function from the timer.

One way of doing this is shown in Figure 2.2, in which there are two ASM charts. The ASM chart on the left is the traffic light controller, in which a signal, START, is asserted as a conditional output when a car is detected.

This signal acts as an input to the second state machine, allowing that state machine to move from the IDLE state into the counting sequence. When the second state machine completes the counting sequence, the signal TIMED is asserted, which acts as an input to the first state machine, allowing the latter to move from state R to state G. The second state machine moves back into the IDLE state.

A state machine of the form of the second state machine of Figure 2.2 can be thought of as a 'hardware subroutine'. In other words, any state machine may be partitioned in this way. Unlike a software subroutine, however, a piece of hardware must exist and must be doing something, even when it is not being used. Hence, the IDLE state must be included to account for the time when the 'subroutine' is not doing a useful task.

An alternative way to implement a subsidiary state machine is shown in Figure 2.3. This version does not correspond to the 'hardware subroutine' model, but represents a conventional counter. The use of standard components will be discussed further in the next section.

From the ASM chart of Figure 2.1 it is quite clear that the system takes 256 clock cycles to return to state G after a car has been detected. The sequence of operations may be harder to follow in Figure 2.3. In state s255, TIMED is asserted as a conditional output. This causes the left-hand state machine to move from state R to state G. In state R, ENABLE is asserted which allows the right-hand state machine to advance through its counting sequence. A timing diagram of this is shown in Figure 2.4.

At first glance this timing diagram may appear confusing. The ENABLE signal causes the TIMED signal to be asserted during the final state of the right-hand diagram. The TIMED signal causes the left-hand state machine to move from state R to state G. According to ASM chart convention, both these signals are asserted at the beginning of a state and deasserted at the end of a state. In fact, of course, the signals are asserted some time after a clock edge and also deasserted after a clock edge. Therefore, a more realistic timing diagram is given in Figure 2.5. The changes to TIMED and ENABLE happen after the clock edges. This, of course, is necessary in order to satisfy the setup and hold times of the flip-flops in the system. The clock speed is limited by the propagation delays through the combinational logic of both state machines. In that sense, a system divided into two or more state machines behaves no differently to a system implemented as a single state machine.

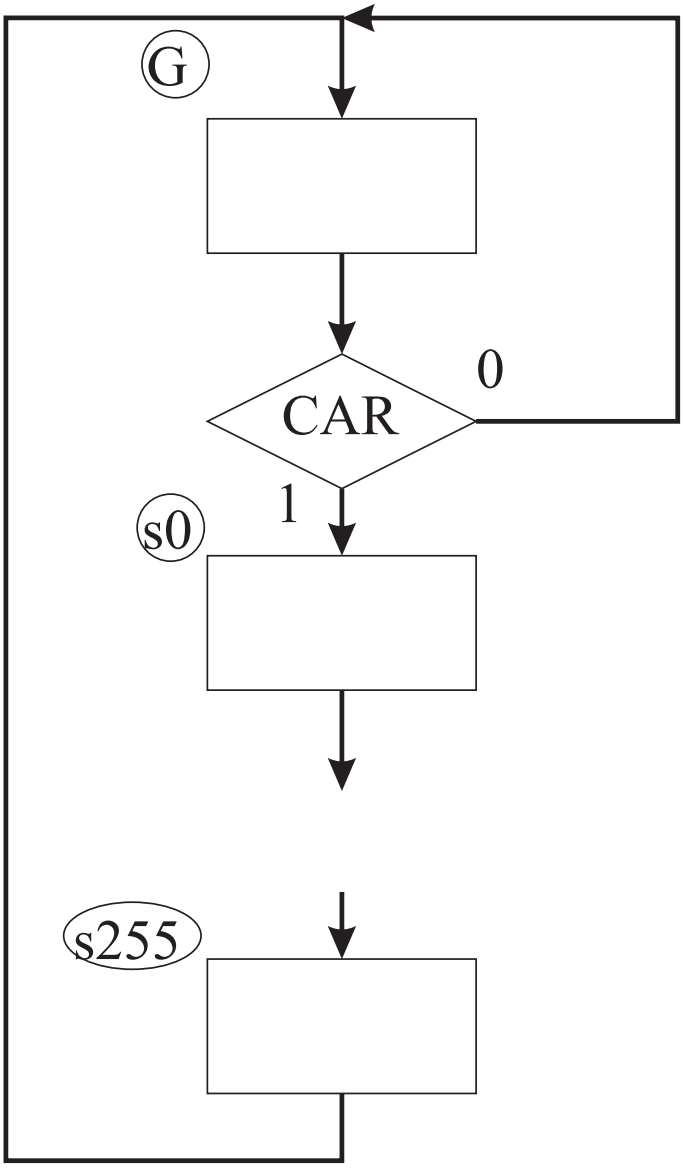

Figure 2.1: ASM chart of traffic signal controller including the timer.

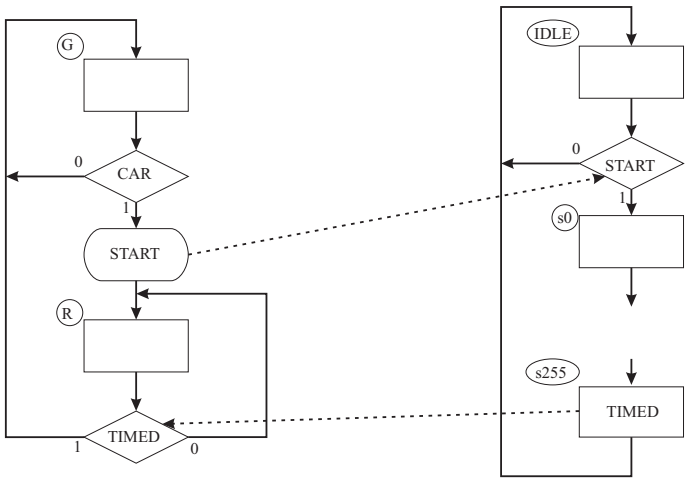

Figure 2.2: Linked ASM charts for traffic signal controller.

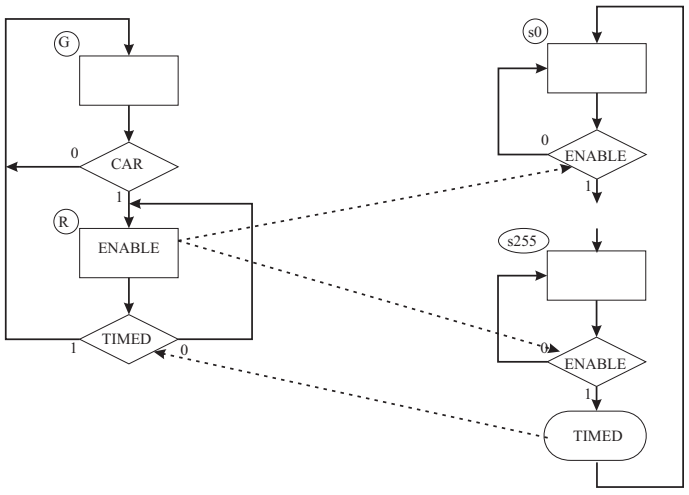

Figure 2.3: ASM chart of traffic signal controller with counter.

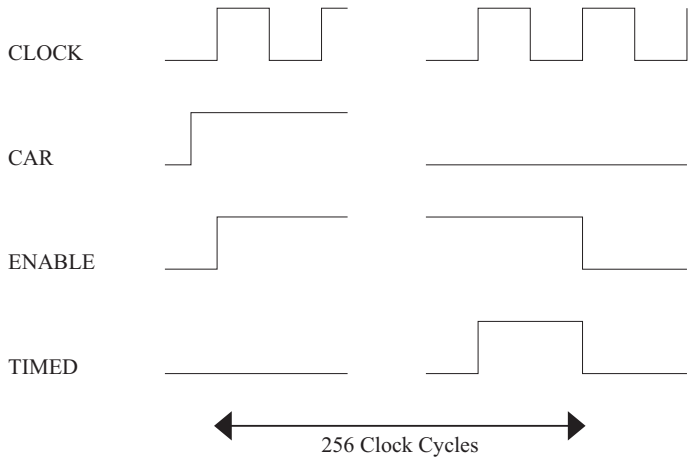

Figure 2.4: Timing diagram of linked ASM charts.

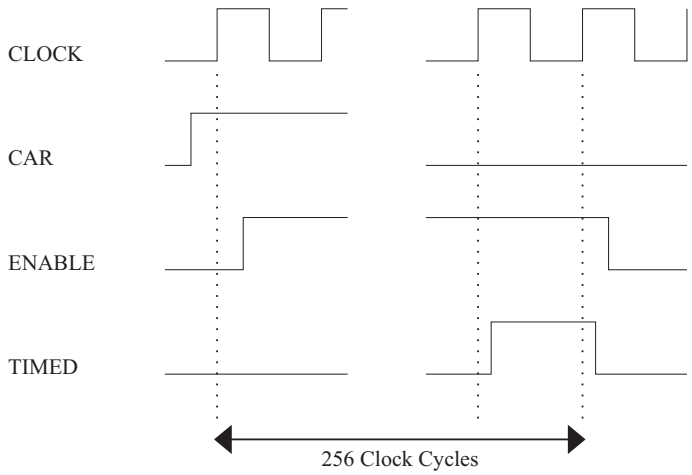

Figure 2.5: Timing diagram showing delays.

Chapter 3

Datapath/controller partitioning

Although any synchronous sequential system can be designed in terms of one or more state machines, in practice this is likely to result in the 'reinvention of the wheel' on many occasions. For example, the right-hand state machine of Figure 2.3 is simply an 8-bit counter. Given this, it is obviously more effective to reuse an existing counter, either as a piece of hardware or as a SystemVerilog model. It is therefore convenient to think of a sequential system in terms of the *datapath*, i.e. those components that have been previously designed (or that can be easily adapted) and that can be reused, and the *controller*, which is a design-specific state machine. A model of a system partitioned in this way is shown in Figure 3.1.

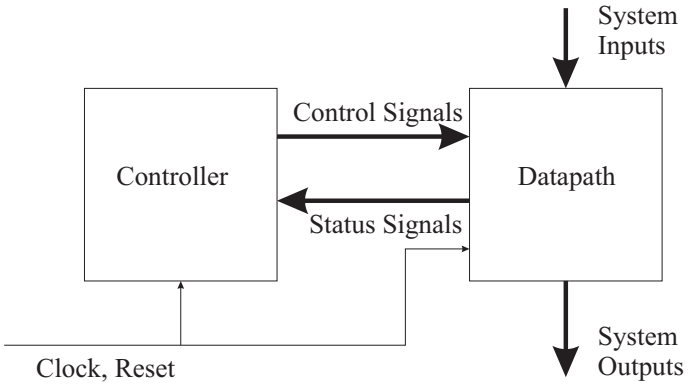

Figure 3.1: Controller/datapath partitioning.

Returning to the example of Figure 2.4, it can be seen that the left-hand state machine corresponds to a controller, while the right-hand state machine, the counter, corresponds to the datapath. The TIMED signal is a

status signal, as shown in Figure 3.1, while the ENABLE signal is a control signal.

The datapath would normally contain registers. As the functionality of the system is mainly contained in the datapath, the system can be described in terms of *register transfer operations*. These register transfer operations can be described using an extension of ASM chart notation. In the simplest case a registered output can be indicated as shown in Figure 3.2(a). This notation means that Z takes the value 1 *at the end* of the state indicated, and *holds that value* until it is reset. If, in this example, Z is reset to 0 and it is only set to 1 in the state shown, the registered output would be implemented as a flip-flop and multiplexer, as shown in Figure 3.2(b), or simply as an enabled flip-flop as shown in Figure 3.2(c). In either implementation, the ENABLE signal is only asserted when the ASM is in the indicated state. Thus the ASM chart could equally include the ENABLE signal, as shown in Figure 3.2(d).

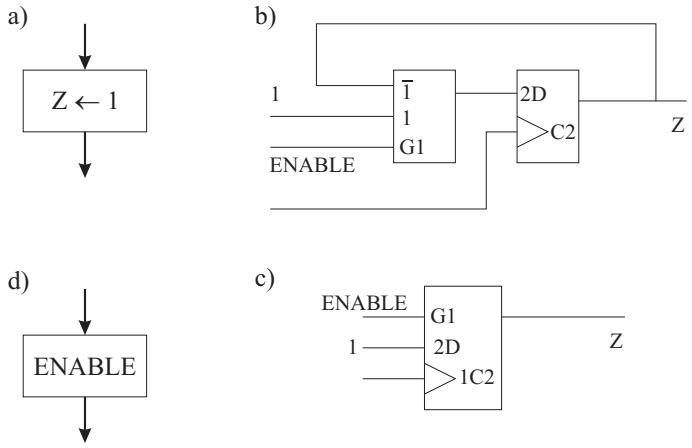

Figure 3.2: Extended ASM chart notation.

A more complex example is shown in Figure 3.3. In state 00, three registers, B_0, B_1 and B_2 are loaded with inputs X_0, X_1 and X_2, respectively. Input A then determines whether a shift left, or multiply by 2, is performed (A=0) or a shift right, or divide by 2 (A=1) in the next state. If a divide by 2 is performed, the value of the least significant bit is tested, so as always to round up. From the ASM chart we can derive next state equations for the controller, either formally or by inspection:

$S_0^+ = \bar{S}_0.\bar{S}_1.(\bar{A} + \bar{X}_0)$

$S_1^+ = \bar{S}_0.\bar{S}_1.A$

The datapath part of the design can be implemented using registers for B_0, B_1 and B_2 and multiplexers, controlled by S_0 and S_1, to select the inputs

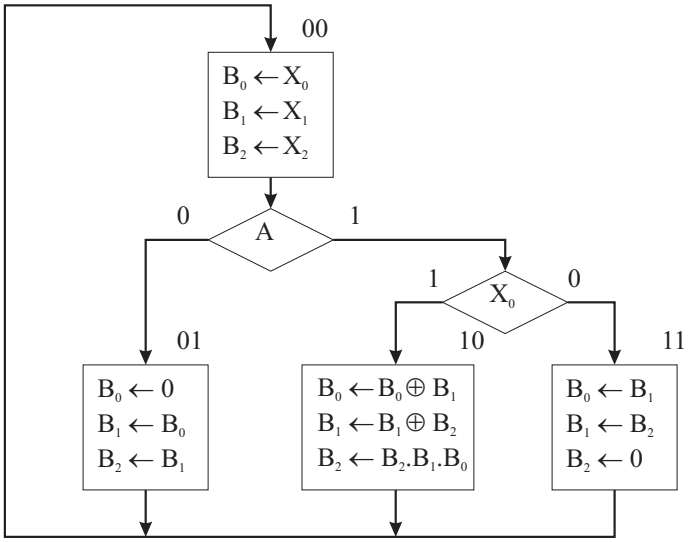

Figure 3.3: ASM chart of partitioned design.

to the registers, as shown in Figure 3.4. It is also possible to implement the input logic using standard gates and thus to simplify the logic slightly.

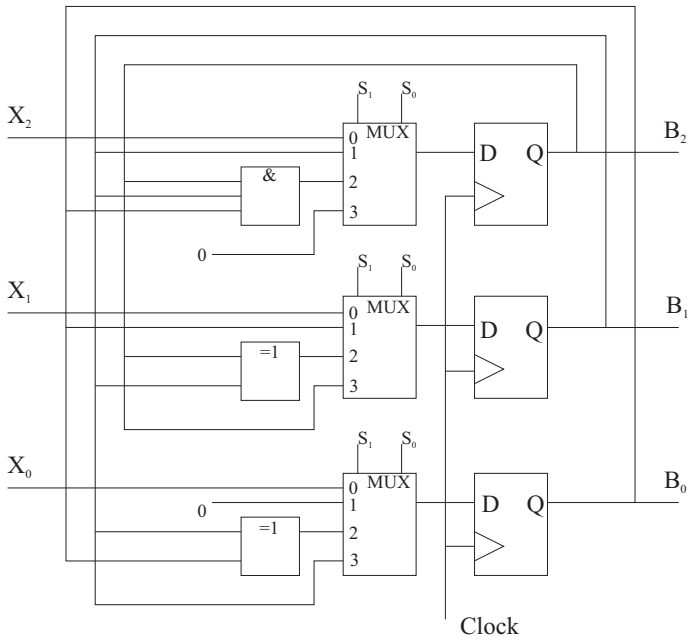

Figure 3.4: Implementation of datapath.