```
process (en, sel, a)
begin
    -- tristate driver
    if en = '1' then
        -- multiplexer
        if sel = '0' then
            z <= a;
        else
            z <= b;
        end if;
    else
        z <= (others => 'Z');
    end if;
end process;
```

Note how the combinational part of the behaviour has been kept separate from the tristate driver part, with the multiplexer logic completely contained within the first branch of the outer if statement which represents the tristate driver.

12.2 FINITE STATE MACHINES

The basic form of a finite state machine (FSM) is a sequential circuit in which the next state and the circuit outputs depend on the current state and the inputs. The most common application for FSMs is in control circuits. The basic form is shown in Figure 12.4.

An FSM can be modelled in VHDL as a combinational block and a register block, and in that sense is nothing special. However, most synthesisers have the capability of performing state optimisation on FSMs to minimise the circuit area. This optimisation is only available if the FSM model fits a special template.

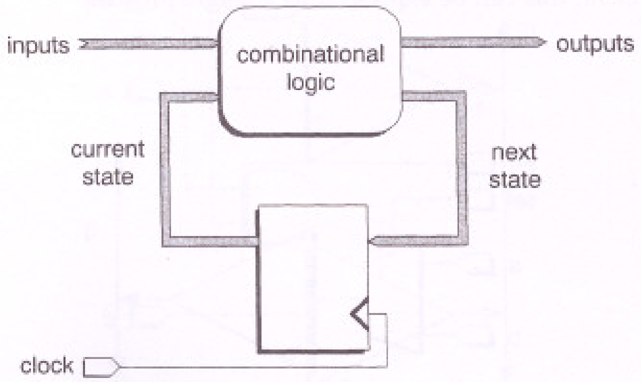

Figure 12.4 Finite state machine.

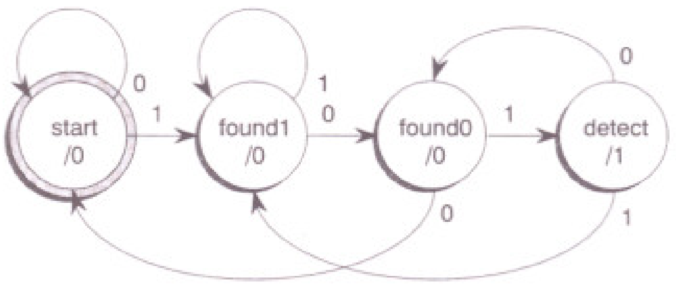

Figure 12.5 Signature detector state transition diagram.

As usual, the template varies slightly from one synthesiser to another and some synthesisers support several variants of the template, but the template presented here is a common denominator.

The key feature of the FSM template is that the current state and next state are represented by an enumeration type, with one value for each state. The state signal may need to be identified as such by an attribute, so the synthesiser knows to apply state optimisation to the state machine. The inputs and outputs can be of any type.

The example is a very simple state machine which detects a certain bit sequence (a signature) on a 1-bit-wide serial input. The state machine is defined by the state transition diagram in Figure 12.5. The state transition diagram shows the states as circles with the state name inside. The states themselves are also annotated with the output value which is required from the state machine when it is in that state. The state transitions are labelled with the input value which causes the transition. This example is a Moore machine, since the output only depends on the current state and is independent of the input.

The example in VHDL is shown in context in a separate architecture. In general, state machines can be mixed with other circuitry and the separation has been done purely for clarity:

```
library ieee;
use ieee.std_logic_1164.all;
entity signature_detector is
   port (d : in std_logic;
         ck : in std_logic;
         found : out std_logic);
end;

architecture behaviour of signature_detector is
   type state_type is (start, found1, found0, detect);
   attribute enum_encoding of state_type : type is
      "00 01 11 10";
   signal current_state, next_state : state_type;
```

```
begin
   -- register block
   process
   begin
      wait until ck'event and ck = '1';
      current_state <= next_state;
   end process;

   -- combinational logic block
   process (current_state, d)
   begin
      case current_state is
         when start =>
            found <= '0';
            if d = '1' then
               next_state <= found1;
            else
               next_state <= start;
            end if;
         when found1 =>
            found <= '0';
            if d = '0' then
               next_state <= found0;
            else
               next_state <= found1;
            end if;
         when found0 =>
            found <= '0';
            if d = '1' then
               next_state <= detect;
            else
               next_state <= start;
            end if;
         when detect =>
            found <= '1';
            if d = '1' then
               next_state <= found1;
            else
               next_state <= found0;
            end if;
      end case;
   end process;

end;
```

The attribute enum_encoding that has been attached to the state_type used to define the state signal will vary from one synthesiser to another. Some synthesisers will not support it at all. This attribute allows the encoding to be explicitly stated—in this case a Gray code has been used. In the absence of an explicit encoding, the synthesiser will allocate states automatically.

This template should be adhered to closely. The combinational logic block should be modelled as a process with a case statement branching on the current value of the state. The contents of each branch of the case statement should contain simple assignments of values to the next state and output signals. Branches in the state transition diagram are modelled by if statements within a branch of the case statement, as in this example. To model a Mealy machine, the outputs would also be conditional on the inputs, so the assignments to the outputs would also be inside the if statements.

An alternative template, which fits some design styles better than this one, has the next state logic in the registered process and only the output decoding in the combinational process. The above example using this alternative style is:

```
architecture behaviour of signature_detector is
   type state_type is (start, found1, found0, detect);
   attribute enum_encoding of state_type : type is
      "00 01 11 10";
   signal state : state_type;
begin

   -- register block
   process
   begin
      wait until ck'event and ck = '1';
      case state is
         when start =>
            if d = '1' then
               state <= found1;
            else
               state <= start;
            end if;
         when found1 =>
            if d = '0' then
               state <= found0;
            else
               state <= found1;
            end if;
         when found0 =>
            if d = '1' then
               state <= detect;
            else
               state <= start;
            end if;
         when detect =>
            if d = '1' then
               state <= found1;
            else
               state <= found0;
            end if;
      end case;
   end process;

   -- combinational logic block
   process (state)
   begin
      case state is
         when start => found <= '0';
         when found1 => found <= '0';
         when found0 => found <= '0';
         when detect => found <= '1';
```

```
        end case;
    end process;

end;
```

This template makes the distinction between Moore and Mealy state machines more obvious. In this example the combinational logic block depends only on the state and does not have any other inputs. This makes it a Moore machine. If the combinational output also depended in any way on the inputs to the state machine (in this case port d), then it would be a Mealy machine.

An FSM should be designed to be either resettable, re-entrant, or both. A reset, either synchronous or asynchronous, can easily be added to the register part of the template and has no effect on the combinational part. For example, a synchronous reset to the start state could be incorporated by adding a reset input to the entity and rewriting the register part:

```
-- register block
process
begin
    wait until ck'event and ck = '1';
    if rst = '1' then
        current_state <= start;
    else
        current_state <= next_state;
    end if;
end process;
```

To make a state machine re-entrant, all states must be considered, including those not shown in the VHDL. In the VHDL, the enumeration type representing the state can have any number of values. The hardware implementation, however, may have more states than the VHDL because it has to be implemented using a binary encoding, which for n flip-flops will have 2^n permutations. Any difference between the number of permutations of the bits in the hardware implementation and the number of states in the VHDL model are extra, or undefined, states. These extra states, which do not even exist in the VHDL model, can be mopped up by an others clause at the end of the case statement.

In this example, four states are used and a Gray encoding using two bits has been specified, so there are no undefined states. However, consider an example with five states. This will be encoded in 3 bits, giving eight permutations, so there are three undefined states in the VHDL. In this case an others clause could be added to the case statement to specify what happens if the state machine gets into an undefined state. This branch is unreachable in simulation, so cannot be tested (this is the main weakness in this strategy), but it does ensure that the state machine is re-entrant after synthesis.