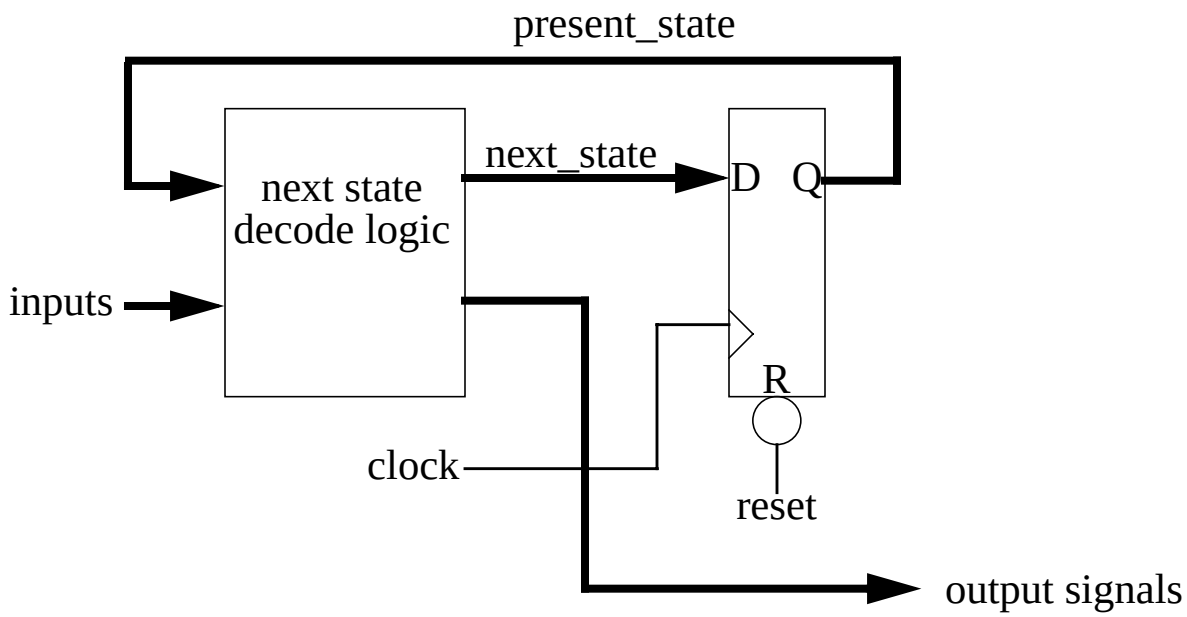
Mealy Outputs

Mealy state machines in VHDL look nearly the same as Moore machines. The difference is in how the output signal is created.

The general structure for a Mealy state machine. Here is the basic Mealy machine structure.



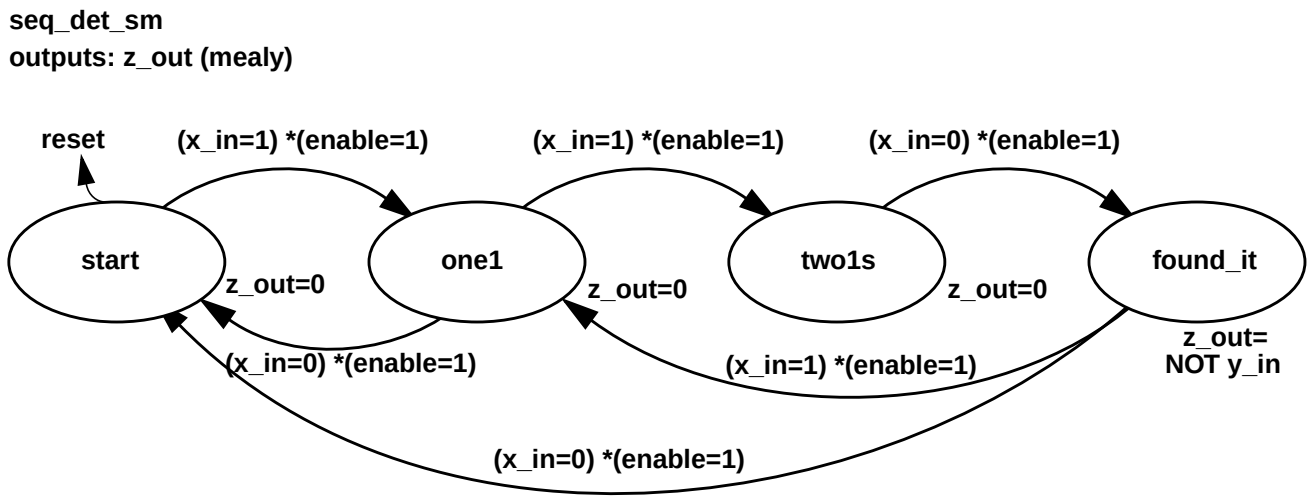
The Mealy state machine uses the next state decode logic to create the output signals. What makes an output a Mealy output is that it is a function of the input signals and the present state. A Mealy machine is really just a Moore machine with the outputs formed differently. As such, you may see a state machine with both Mealy and Moore outputs.

For an example, we will make the following state machine:

A sequence detector is to be built with inputs *clock*, *reset*, *enable*, *x_in*, *y_in* and a single mealy output *z_out*. Once the machine is enabled, it searches for a sequence of “110” on the *x_in* input. When the “110” sequence is found, the value of *z_out* is equal to the complement of the *y_in* input.

Draw the state diagram and write the VHDL code for this state machine.

State Diagram and code for seq_det_sm




```
--Sequence detector
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY seq_det IS
  PORT(
      clock   : IN   STD_LOGIC;  --input clock
      reset   : IN   STD_LOGIC;  --reset async
      enable  : IN   STD_LOGIC;  --data enable
      x_in    : IN   STD_LOGIC;  --input
      y_in    : IN   STD_LOGIC;  --input
      z_out   : OUT  STD_LOGIC   --mealy output
      );
  END seq_det;

 ARCHITECTURE beh OF seq_det IS

 TYPE seq_det_states  IS (start, one1, two1s, found_it);
 SIGNAL seq_det_ps, seq_det_ns: seq_det_states;


 BEGIN
 seq_det_sm:
 PROCESS (clock, reset, enable, x_in, y_in)
   BEGIN
     --clocked part
     IF (reset = '1') THEN
       seq_det_ps <= start;
     ELSIF (clock'EVENT AND clock = '1') THEN
       seq_det_ps <= seq_det_ns;
     END IF;
```

```
 --combinatorial part
    z_out <= '0';  --default value for mealy output
    CASE seq_det_ps  IS
      WHEN start =>
        IF ((x_in = '1') AND (enable = '1')) THEN
          seq_det_ns <= one1;
        ELSE
          seq_det_ns <= start;
        END IF;
      WHEN one1 =>
        IF((x_in = '1') AND (enable = '1')) THEN
          seq_det_ns <= two1s;
        ELSIF((x_in = '0') AND (enable = '1')) THEN
          seq_det_ns <= start;
        ELSE
          seq_det_ns <= one1;
        END IF;
      WHEN two1s =>
        IF((x_in = '0') AND (enable = '1')) THEN
          seq_det_ns <= found_it;
        ELSE
          seq_det_ns <= two1s;
        END IF;
      WHEN found_it =>
        z_out <= NOT y_in;  -- mealy output
        IF((x_in = '0') AND (enable = '1')) THEN
          seq_det_ns <= start;
        ELSIF((x_in = '1') AND (enable = '1')) THEN
          seq_det_ns <= one1;
        ELSE
          seq_det_ns <= found_it;
        END IF;
      END CASE;
   END PROCESS seq_det_sm;
END beh;
```

seq_det_sm after synthesis