16.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Div3 is
  port(
    clk   : in  std_logic;
    reset : in  std_logic;
    X     : in  std_logic; -- input bit
    Z     : out std_logic  -- output: 1 iff total # of 1's seen so far is divisible by 3
  );
end entity;

architecture beh of Div3 is
  type state_t is (S0, S1, S2);
  signal cs, ns : state_t;
begin
  -- Next-state logic
  process(cs, X)
  begin
    case cs is
      when S0 =>
        if X = '1' then
          ns <= S1;
        else
          ns <= S0;
        end if;

      when S1 =>
        if X = '1' then
          ns <= S2;
        else
          ns <= S1;
        end if;

      when S2 =>
        if X = '1' then
          ns <= S0;
        else
          ns <= S2;
        end if;
    end case;
```

```
  end process;


  Z <= '1' when ns = S0 else '0';

  -- State register (synchronous)
  process(clk)
  begin
    if rising_edge(clk) then
      if reset = '1' then
        cs <= S0;
      else
        cs <= ns;
      end if;
    end if;
  end process;
end architecture;
```

18.

```
library ieee;
use ieee.std_logic_1164.all;

entity odd0 is
  port(
    clk   : in  std_logic;
    reset : in  std_logic;
    X     : in  std_logic;
    Z     : out std_logic
  );
end entity;

architecture beh of odd0 is
  signal s0, s1, s2, s3, s4, s5 : std_logic;
begin

  -- Next-state and state register (synchronous)
  process(clk)
  begin
    if rising_edge(clk) then
      if reset = '1' then
```

```
        s0 <= '1';
        s1 <= '0';
        s2 <= '0';
        s3 <= '0';
        s4 <= '0';
        s5 <= '0';
      else
        -- Default all low, then set new one-hot state
        s0 <= '0'; s1 <= '0'; s2 <= '0'; s3 <= '0'; s4 <= '0'; s5 <= '0';
        -- State transitions
        if s0 = '1' then
          if X='0' then s1 <= '1'; else s2 <= '1'; end if;
        elsif s1='1' then
          if X='0' then s0 <= '1'; else s3 <= '1'; end if;
        elsif s2='1' then
          if X='0' then s3 <= '1'; else s4 <= '1'; end if;
        elsif s3='1' then
          if X='0' then s2 <= '1'; else s5 <= '1'; end if;
        elsif s4='1' then
          if X='0' then s5 <= '1'; else s2 <= '1'; end if;
        elsif s5='1' then
          if X='0' then s4 <= '1'; else s3 <= '1'; end if;
        end if;
      end if;
    end if;
  end process;

  --- output
  Z <= s5;

end architecture;
```