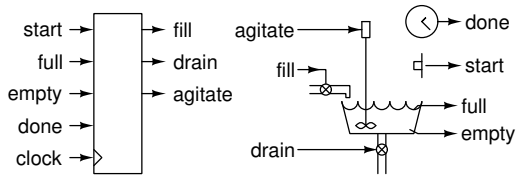
ELEX 2117 : Digital Techniques 2
2020 Winter Term

Solutions to Quiz 3 - State Machines

Description

The controller for a washing machine:



has three active-high outputs:

- **`fill`** opens a valve to fill the tub
- **`drain`** opens a valve to drain the tub
- **`agitate`** causes the agitator to turn

and three active-high inputs:

- **`start`** is used by the user to start or terminate a wash cycle
- **`full`** indicates the tub is full
- **`empty`** indicates the tub is empty
- **`done`** indicates that the agitation cycle has been running for a sufficiently long time

The controller operates as follows:

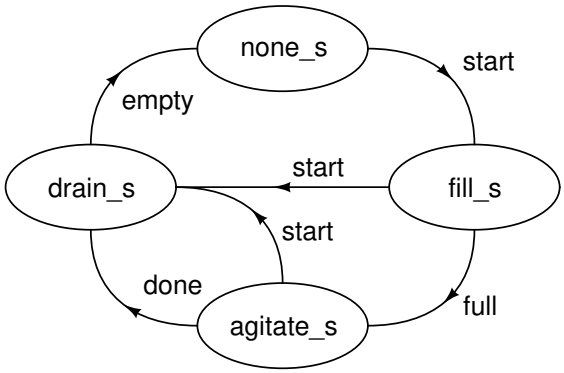
- nothing happens until the user presses **`start`**
- then **`fill`** is asserted until **`full`** is asserted
- then **`agitate`** is asserted until **`done`** is asserted
- then **`drain`** is asserted until **`empty`** is asserted; this completes the wash cycle
- if **`start`** is pressed while a wash is in progress (i.e. while **`fill`**'ing or **`agitate`**'ing) then **`drain`** is asserted until **`empty`** is asserted and this completes the wash cycle

You may assume the **`start`** signal is synchronized to the clock and lasts exactly one clock period.

Solution

(a) There are four combinations of output values: (none asserted, only fill asserted, only agitate asserted, only drain asserted). We can try a solution using only four states with names: **`none_s`**, **`fill_s`**, **`agitate_s`** and **`drain_s`**[1].

(b) Each of the six listed behaviors can be implemented with a state transition as shown in the following state transition diagram:



(c) In this case the outputs are only a function of the current state (any unambiguous syntax would be considered correct):

```
fill <= '1' when state = fill_s else '0' ;
agitate <= '1' when state = agitate_s else '0' ;
drain <= '1' when state = drain_s else '0' ;
```

The VHDL for this solution and a simulation testbench are given below. The simulation waveforms are shown in Figure 1.

[1] I'll use the suffix **`_s`** to avoid name conflicts with the names of the inputs

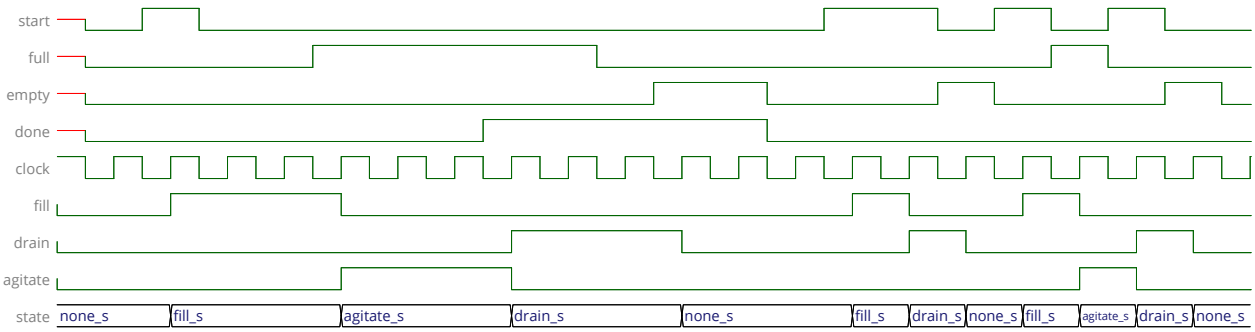


Figure 1: Simulation Results

```
-- ELEX 2117 202010
-- quiz 3 solution
-- Ed.Casas 2020-02-23

library ieee ;
use ieee.std_logic_1164.all ;

entity quiz3 is
  port (
    start, full, empty, done: in std_logic;
    clock: in std_logic;
    fill, drain, agitate: out std_logic ) ;
end ;

architecture rtl of quiz3 is

  type state_t is ( none_s, fill_s, agitate_s, drain_s ) ;
  signal state, state_next : state_t ;
  signal none_s_next, fill_s_next, agitate_s_next,
    drain_s_next : state_t ;

begin

  none_s_next <= fill_s when start = '1' else
                 none_s ;
  fill_s_next <= agitate_s when full = '1' else
                 drain_s when start = '1' else
                 fill_s ;
  agitate_s_next <= drain_s when done = '1' else
                    drain_s when start = '1' else
                    agitate_s ;
  drain_s_next <= none_s when empty = '1' else
                  drain_s ;

  with state select state_next <=
    none_s_next when none_s,
    fill_s_next when fill_s,
    agitate_s_next when agitate_s,
    drain_s_next when drain_s ;

  state <= state_next when rising_edge(clock);

  -- outputs

  fill <= '1' when state = fill_s else '0' ;
  agitate <= '1' when state = agitate_s else '0' ;
  drain <= '1' when state = drain_s else '0' ;

end rtl ;
```

```
-- testbench for quiz 3 solution

library ieee;
use ieee.std_logic_1164.all;

entity quiz3_tb is

end ;

architecture rtl of quiz3_tb is
  signal start, full, empty, done: std_logic;
  signal clock: std_logic := '1' ;
  signal fill, drain, agitate:  std_logic ;
  signal done_sim: boolean := false ;
begin

  dut: entity work.quiz3 port map (
    start, full, empty, done, clock,
    fill, drain, agitate ) ;

  process
    type tv_t is array (natural range <>) of
      std_logic_vector (3 downto 0) ;
    variable tv: tv_t (1 to 21) := (
      "0000", "1000", "0000", "0000", -- fill
      "0100", "0100", "0100", "0101", -- agitate
      "0101", "0001", "0011", "0011", -- drain
      "0000", "1000", "1000", "0010", -- abort from fill
      "1000", "0100", "1000", "0010", -- " from agitate
      "0000" ) ;
  begin

    for i in tv'low to tv'high loop
      wait until falling_edge(clock);
      ( start, full, empty, done ) <= tv(i) ;
    end loop ;

    done_sim <= true ;
  end process ;

 clock <= not clock after 5 sec when not done_sim ;

end rtl ;
```