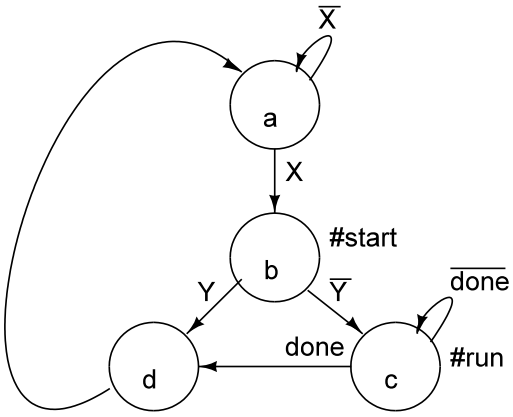
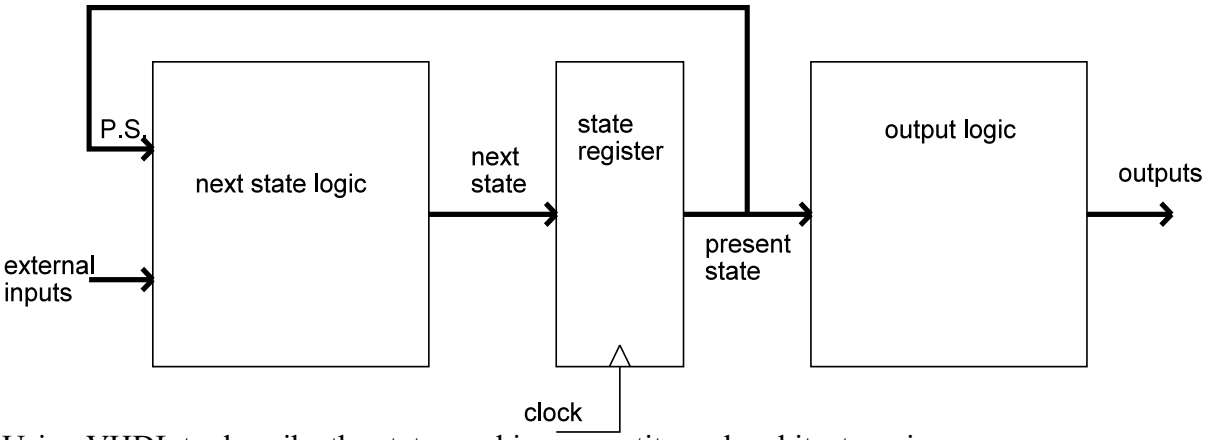
Example State Machine Description in VHDL

Assume a simple state machine needs to be created as described by this state diagram:




A state machine can be described using VHDL as follows. Assume that the required signal names have been declared as standard logic either in an entity or with a signal statement. Also, in this example symbolic state names are used leaving the synthesizer to assign state numbers (it is possible to explicitly define the state number for each state, but that is not done here).

Recall the fundamental state machine model (Moore style):




Using VHDL to describe the state machine, an entity and architecture is:

```
entity fsm_example1 is
    Port ( clk : in  STD_LOGIC;
           X,Y,done : in STD_LOGIC;
           start,run : out STD_LOGIC);
end fsm_example1;

architecture Behavioral of fsm_example1 is
   type state_type is (state_a,state_b,state_c,state_d);
   signal state_reg, nxt_state : state_type;
begin
```

```
   -- **** state register  ****
   process(clk)
   begin
      if (clk'event and clk='1') then
         state_reg <= nxt_state;
      end if;
   end process;

   -- **** next state logic ****
   process(state_reg, X, Y, done)
   begin
      case state_reg is
         when state_a =>
            if X = '1' then
               nxt_state <= state_b;
            else
               nxt_state <= state_a;
            end if;
         when state_b =>
            if Y = '1' then
               nxt_state <= state_d;
            else
               nxt_state <= state_c;
            end if;
         when state_c =>
            if done = '1' then
               nxt_state <= state_d;
            else
               nxt_state <= state_c;
            end if;
         when state_d =>
            nxt_state <= state_a;
      end case;
   end process;

   -- **** output logic ****
   process (state_reg)
   begin
      start <= '0';      -- default value
      run <= '0';        -- default value
      case state_reg is
         when state_a =>
         when state_b =>
            start <= '1';
         when state_c =>
            run <= '1';
         when state_d =>
      end case;
   end process;
end Behavioral;
```