Sequential Logic Design in VHDL

This lecture shows how state machines can be described in VHDL.
After this lecture you should be able to write a VHDL description of a state machine.

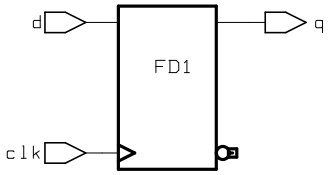
Sequential Circuits in VHDL

Process statements include *sequential* statements that execute one after the other as in conventional programming languages. The design of sequential circuits in VHDL requires the use of the process statement. However, for the logic synthesizer to be able to convert a process to a logic circuit, the process must have a very specific structure. We will use only one specific process in this course and we will use it only to generate memory elements (flip-flops or registers).

For example, we can describe a D flip-flop in VHDL as follows:

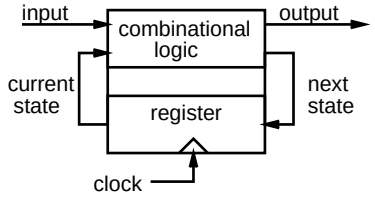
```
entity D_FF is
   port ( clk, d : in bit ;
   q : out bit ) ;
end D_FF ;

architecture rtl of D_FF is
begin
process(clk)
begin
   if clk'event and clk = '1' then
      q <= d ;
   end if ;
end process ;
end rtl ;
```

The expression `clk'event` (pronounced "clock tick event") is true when the value of `clk` has changed since the last time the process was executed. In this case the output `q` is only assigned a value if `clk` changes and the new value is 1. When `clk` = 0 the output retains its previous value. It's necessary to check for `clk`=1 to distinguish between rising and falling edges of the clock.



FSMs in VHDL are implemented by using concurrent assignment statements (e.g. selected assignments) to generate the output and the next state based on the current state (and possibly the inputs). A process is used to implement the flip-flops that define the current state.

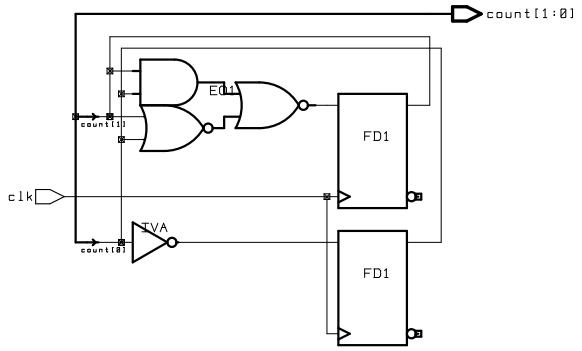
This corresponds to the following block diagram:



For example, a VHDL description for a 2-bit counter could be written as follows:

```
entity count2 is
   port ( clk : in bit ; count :
      out bit_vector (1 downto 0) ) ;
end count2 ;

architecture rtl of count2 is
   signal current, nexts :
      bit_vector (1 downto 0) ;
begin

   -- combinational logic for next state
     with current select
     nexts <=
"01" when "00",
"10" when "01",
"11" when "10",
"00" when others ;

   -- combinational logic for output
     out <= current ;

   -- sequential logic
   process(clk)
   begin
      if clk'event and clk = '1' then
         current <= nexts ;
      end if ;
   end process ;

end rtl ;
```

The synthesized circuit is:



Exercise: Identify the components in the schematic that were created ("*instantiated*") the different parts of the VHDL code.

Exercise: Modify the above description to add an up/down control input.