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.

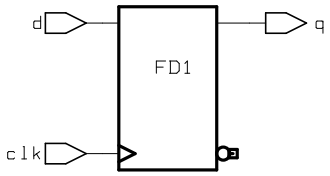
Sequential Circuits in VHDL

The design of sequential circuits in VHDL requires the use of the `process` statement. Process statements can include *sequential* statements that execute one after another as in conventional programming languages. However, for the logic synthesizer to be able to convert a process to a logic circuit, the process must have a very specific structure. You should use only the simple three-line type of process shown below. This is because signal assignments within a process do not happen as you might expect and may lead to strange results. We will only use this specific type of process in this course and we will use it only to generate memory elements (flip-flops or registers).

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

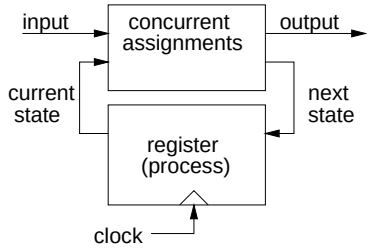
```
entity d_ff is port (
   clk, d : in bit ;
   q : out bit ) ;
end d_ff ;

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. This is how we model "memory" in VHDL. In the process 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 (1) the output and (2) the next state from (a) the current state and (b) the inputs. The process shown above is used to generate the flip-flops that define the current state.

This corresponds to the following block diagram:



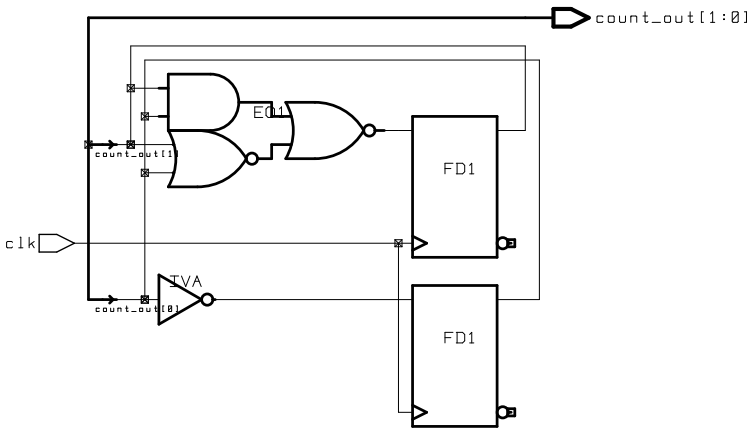
A VHDL description for a 2-bit counter could be written as follows:

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

architecture rtl of count2 is
   signal count, next_count : bit_vector (1 downto 0) ;
begin
   -- combinational logic for next state
   with count select next_count <=
      "01" when "00",
      "10" when "01",
      "11" when "10",
      "00" when others ;

   -- combinational logic for output
   count_out <= count ;

   -- sequential logic
   process(clk)
   begin
      if clk'event and clk = '1' then
         count <= next_count ;
      end if ;
   end process ;
end rtl ;
```

Note that two signals need to be defined for each flip-flop or register in a design: one signal for the input (`next_count` in this case) and one signal for the output (`count`).

Also note that we must define a new signal for an output if its value is to be used within the architecture. This is simply a quirk of VHDL.

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 2-bit counter description to add an up/down control input.