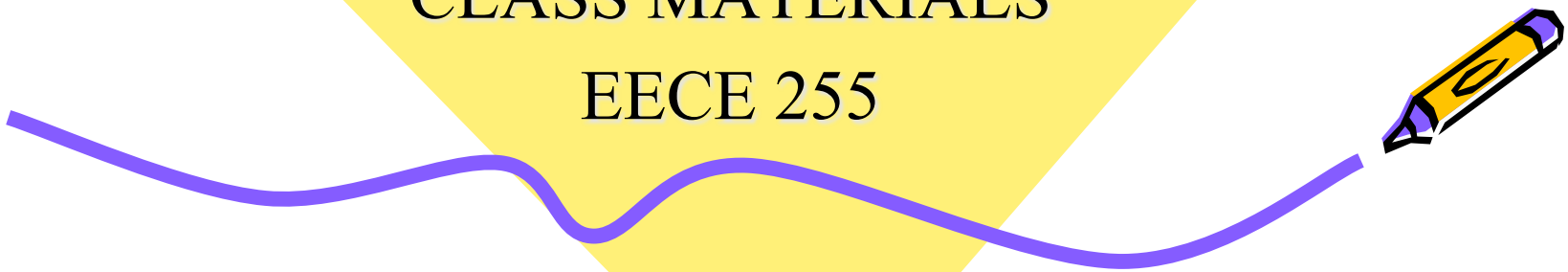
Introduction to Counter in VHDL

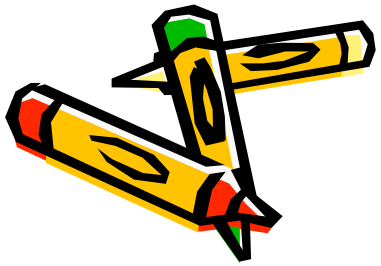
CLASS MATERIALS

EECE 255



Counter ♪

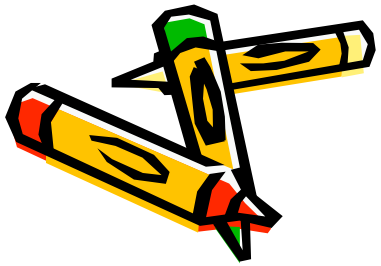
- In electronics, counters can be implemented quite easily using register -type circuits such as the flip-flop, and a wide variety of designs exist, e.g.:
  - Asynchronous (ripple) counters
  - Synchronous counters
  - Johnson counters
  - Decade counters
  - Up-Down counters
  - Ring counters
- There are several ways to create counter circuits, such as using T flip-flop, D flip-flop, JK flip-flop. In this class, we will introduce a simply way to write code in VHDL for the counter.



VHDL Example: Gated D Latch

The code in Figure 7.36 defines an entity named latch, which has the inputs D and Clk and the output Q. The process uses an if-then-else statement to define the value of the Q output. When Clk=1, Q takes the value of D. When Clk = 0, Q will retain its current value in this case, and the code describes a gated D latch.

The process sensitivity list includes both Clk and D because these signals can cause a change in the values of the Q output.



```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY latch IS
    PORT (  D, Clk  : IN    STD_LOGIC ;
            Q       : OUT   STD_LOGIC) ;
END latch ;

ARCHITECTURE Behavior OF latch IS
BEGIN
    PROCESS ( D, Clk )
    BEGIN
        IF Clk = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 7.36. Code for a gated D latch.

VHDL Example: D Flip Flop

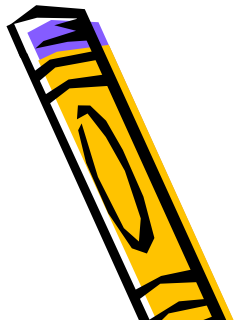
This is a example for a positive-edge-triggered D flip-flop.

1. The process sensitivity list contains only the clock signal because it is the only signal that cause a change in the Q output.
2. The syntax Clock'EVENT uses a VHDL construct called an attribute. With condition Clock = 1, here it means that "the value of the Clock signal has just changed, and the value is now equal to 1", which refers to a positive clock edge.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
    PORT (  D, Clock : IN    STD_LOGIC ;
            Q        : OUT   STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS ( Clock )
    BEGIN
        IF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 7.37. Code for a D flip-flop.

VHDL Example: D Flip Flop



This process uses the statement WAIT UNTIL Clock'EVENT AND Clock='1'.

This statement has the same effect as the IF statement. However, the process sensitivity list is omitted.

In our use of VHDL, which is for synthesis of circuits, a process can use a WAIT UNTIL statement only if this is the first statement in the process.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
      PORT (  D, Clock  : IN       STD_LOGIC ;
              Q         : OUT      STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
     PROCESS
     BEGIN
          WAIT UNTIL Clock'EVENT AND Clock = '1' ;
          Q <= D ;
     END PROCESS ;
END Behavior ;
```

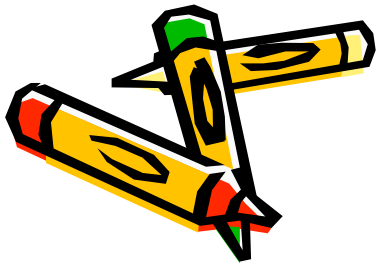
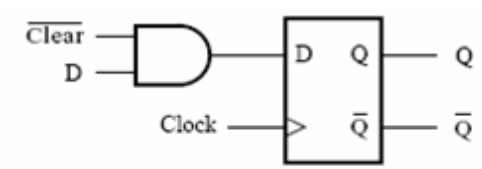


Figure 7.38. Equivalent code for Figure 7.37, using a WAIT UNTIL statement.

VHDL Example: Synchronous Clear

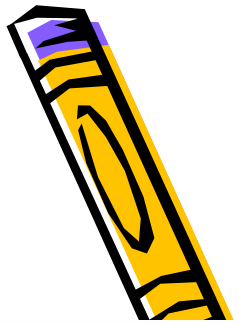
Here is a example shows how a D flip-flop with a synchronous reset input can be described. In this case, the reset signal is acted upon only when a positive cloc k edge arrives.




```
ENTITY flipflop IS
    PORT ( D, Resetn, Clock   : IN     STD_LOGIC ;
           Q                  : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Resetn = '0' THEN
            Q <= '0' ;
        ELSE
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 7.40. D flip-flop with synchronous reset.

VHDL Example: A Four Bit Up-counter



Resetn: Reset input

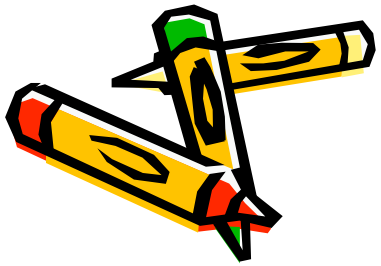
E: enable input

In the architecture body the flip-flops in the counter are represented by the signal named *Count*

If E=1, the count is incremented

If E=0, the code explicitly assigns *Count*<=*Count*

The *O* outputs are assigned the values of *Count* at the end fo the code.

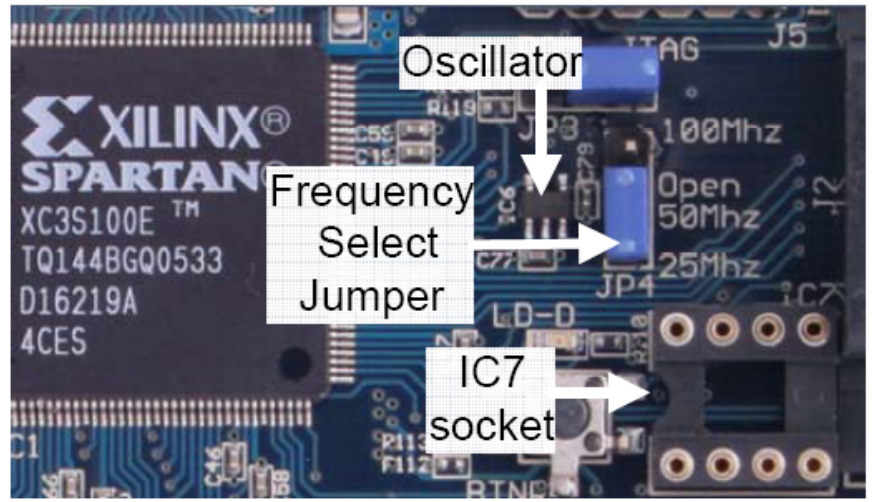


```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY upcount IS
    PORT ( Clock, Resetn, E   : IN     STD_LOGIC ;
           Q                  : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
    SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Count <= "0000" ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF E = '1' THEN
                Count <= Count + 1 ;
            ELSE
                Count <= Count ;
            END IF ;
        END IF ;
    END PROCESS ;
    Q <= Count ;
END Behavior ;
```

Figure 7.52. Code for a four-bit up-counter.

Introduction to Clock

In electronics and especially synchronous digital circuits, a clock signal is a signal used to coordinate the actions of two or more circuits. A clock signal oscillates between a high and a low state and is usually in the form of a square wave.



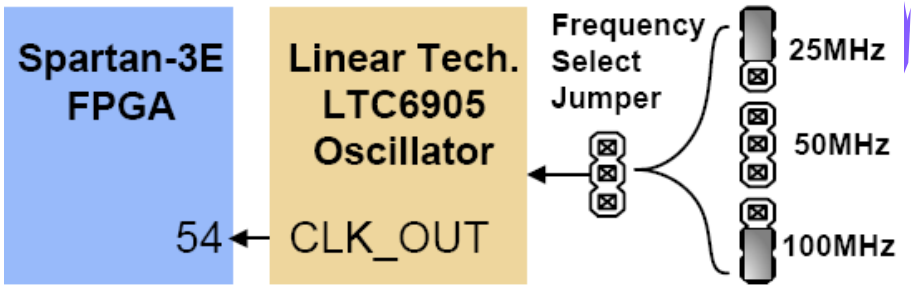
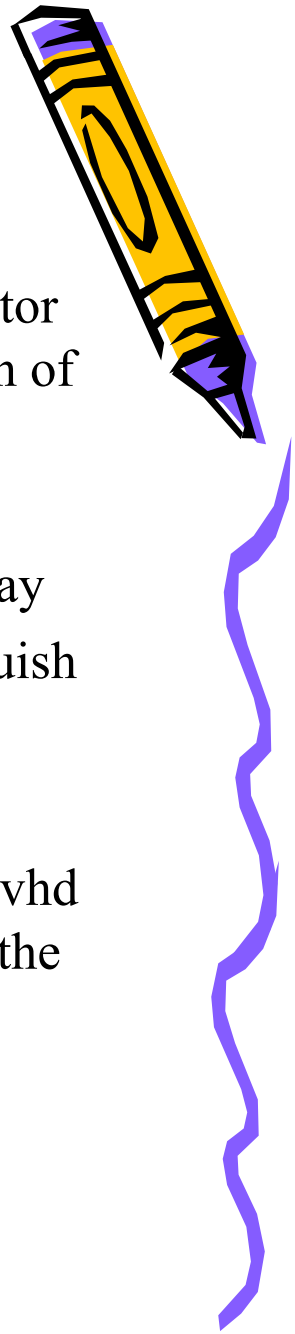


Figure 5. Basys oscillator circuits

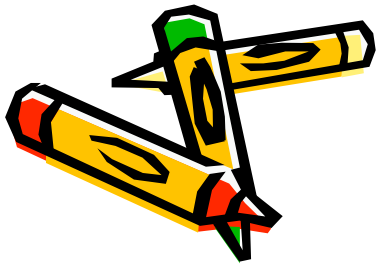
Slow down the Clock



- The Basys board includes a primary, user-settable silicon oscillator that produces 25MHz, 50MHz, or 100MHz based on the position of the clock select jumper at JP4.

- However, the high frequency will make the seven segment display looks like on all the time, and the eyes of human can not distinguish the change.

  One way to slow down the clock frequency is to write a DivClk.vhd file, with the help of IF-ELSE statement and a variable to count the high frequency signal to generate a low freqency signal.



Structure Descriptions in VHDL

- Once we have defined the basic building blocks of our design using entities and their associated architectures, we can combine them together to form other designs.

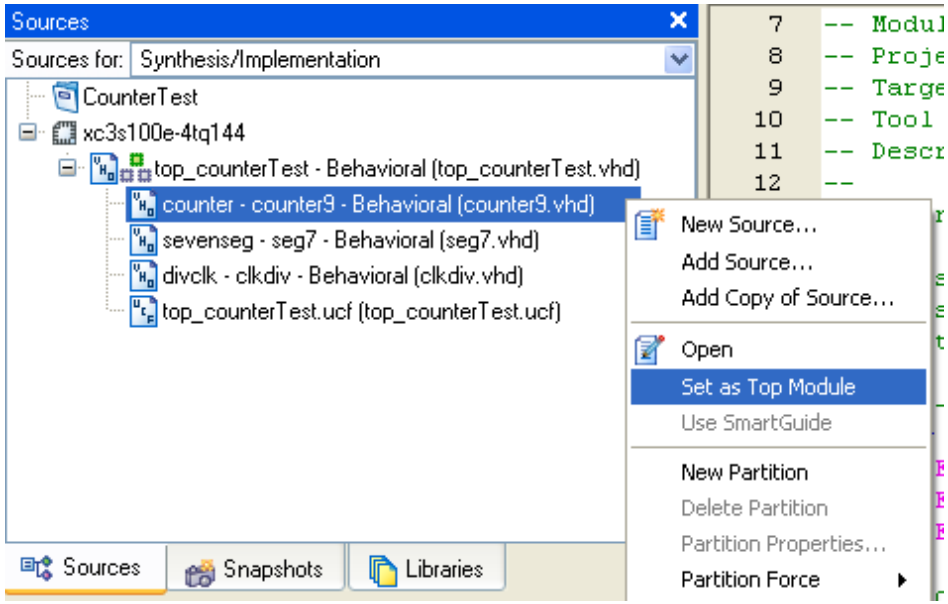
```
entity counter9 is
 port(clock, resetn, E   : IN STD_LOGIC;
      Q                  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
 );
end counter9;
```

```
architecture Behavioral of top_counterTest is

component counter9 is
    port(clock, resetn, E   : IN STD_LOGIC;
         Q                  : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
end component;
--The component declarations (for count9) must match the corresponding entity declarations
-- exactly with respect to the names, order and types of the ports
--...

signal count : std_logic_vector(3 downto 0);
--Signals in an architecture are associated with ports on a component using a port map.
--In effect, a port map makes an electrical connection between "pieces of wire" in an
--architecture (signals) and pins on a component (ports). The same signal may be associated
--with several ports. This is the way to define interconnections between components
--...

begin
   CountDigit: counter9 port map (clk_in, rst, E, count);
   --The instance labels (CountDigit) identify a specific instance of the component, and are
   --mandatory. The component name (counter9) is reference to design entities defined elsewhere.
   --...

   process(clk100)
   begin
      --...
   end process;
end Behavioral;
```

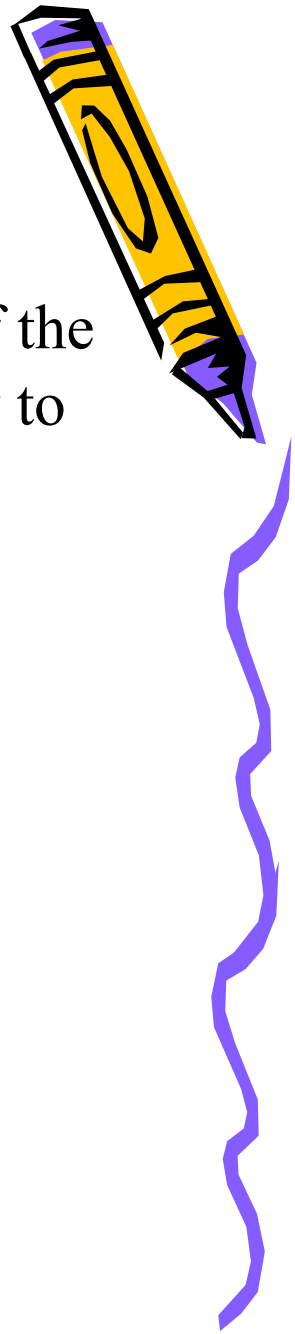
Structure Descriptions in VHDL

The port map clause specifies what signals of the design to connect to the interface of the component in the same order as they are listed in the component declaration. The instance connects clk_in to clock, rst to resetn, E to E, and count to Q.

In Xilinx ISE, you can right click on a certain vhdl file and choose set as top module. Combined with component declaration and port mapping,



The end



In this project, the AN3, AN2, AN1, AN0 are the ID of the four digits display. You will need to figure out a way to output the two digit number on the 7-seg display

VHDL Reference: Textbook, Appendix A.11 Common Errors in VHDL Code.

Advise: Start early and have fun! ♪