Islamic University of Technology

EEE 4483
Digital Electronics & Pulse Techniques

Lecture- 9

Digital Counters

- 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: 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*

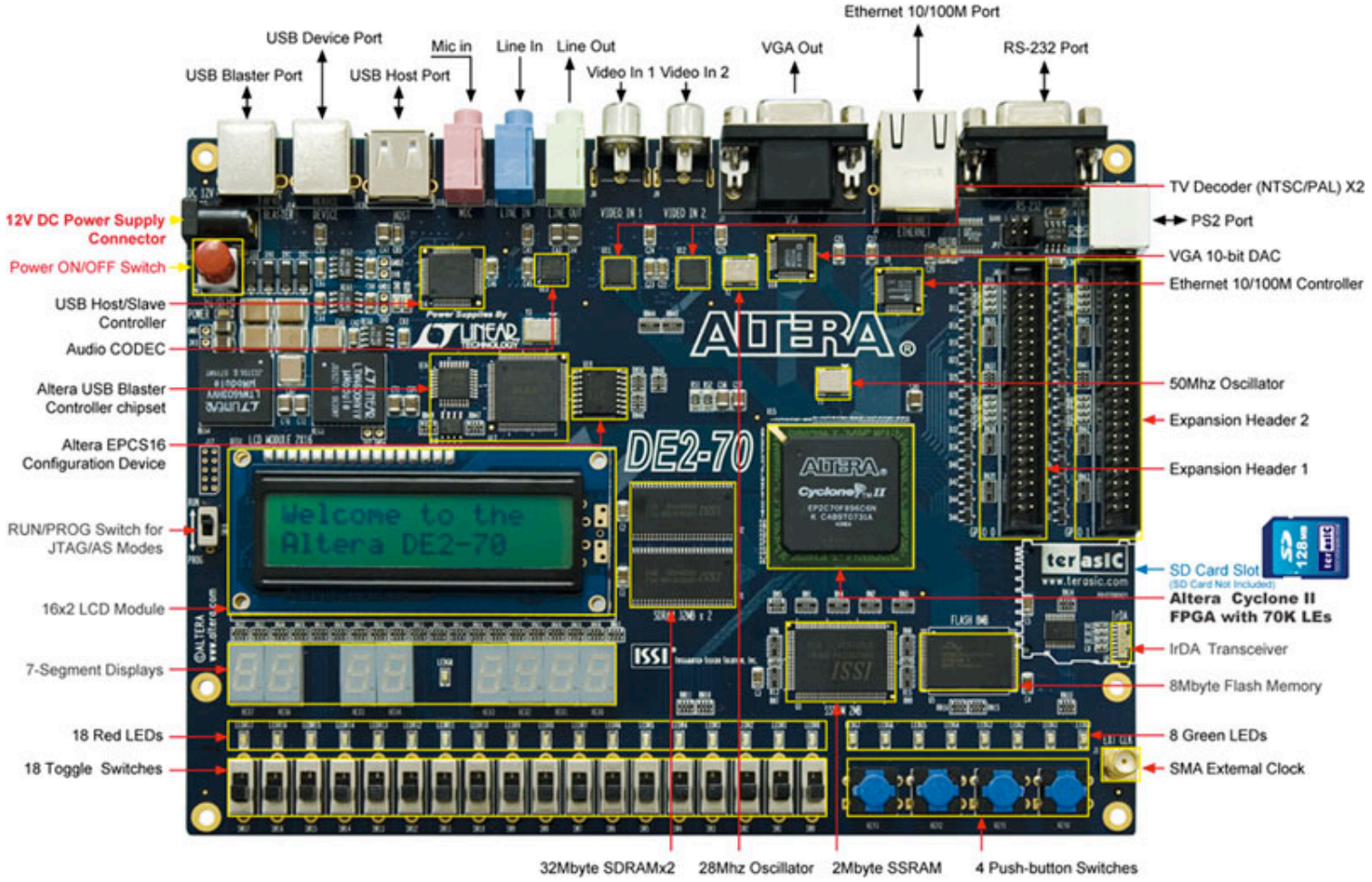
The *O* outputs are assigned the values of *Count* at the end fo the code.

```
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.



Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	27 MHz clock input
CLOCK_50	PIN_N2	50 MHz clock input
EXT_CLOCK	PIN_P26	External (SMA) clock input

Table 4.5. Pin assignments for the clock inputs.

Slow down the Clock

- The Altera DE2 board includes two oscillators that produce 27 MHz and 50 MHz clock signals.

- 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 frequency 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;
```

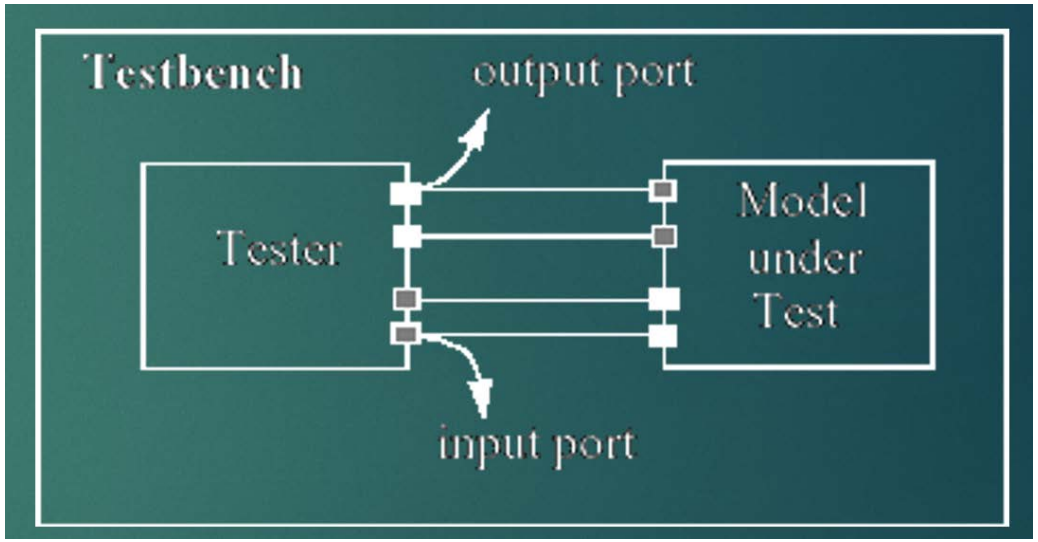
```
architecture Behavioral of top_counterTest is

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;
```

VHDL Testbench

- Testing a design by simulation
- Use a *test bench* model
  - an architecture body that includes an instance of the design under test
  - applies sequences of test values to inputs
  - monitors values on output signals
    - either using simulator
    - or with a process that verifies correct operation

VHDL Testbench : continued ..

- VHDL test bench is VHDL code that produces stimuli to test your design correctness
- It can automatically verify accuracy of the VHDL code
  - Given a known input, does the system generate the expected output
- Verifies that the VHDL code meets the circuits specifications
- Test benches should be easily modified, allowing for future use with other code
- Should be Easy to understand the behavior of the test bench



VHDL Testbench : continued ..

What Is The VHDL Test Bench (TB)?

VHDL test bench (TB) is a piece of VHDL code, which purpose is to verify the functional correctness of HDL model.

The main objectives of TB is to:

– Instantiate the design under test (DUT)

– Generate stimulus waveforms for DUT

– Generate reference outputs and compare them with the outputs of DUT

– Automatically provide a pass or fail indication

Test bench is a part of the circuits specification.

Its a good idea to design the test bench before the DUT, why?

Stimulus and Response

Three ways how TB can generate the stimulus:

– Generate them “on-the-fly”

– Read vectors stored as constants in an array

– Read vectors stored in a separate system file

Response is produced in the test bench.

Response can be stored into file for further processing.

Example:

– Stimulus can be generated with Matlab and TB feeds it into DUT.

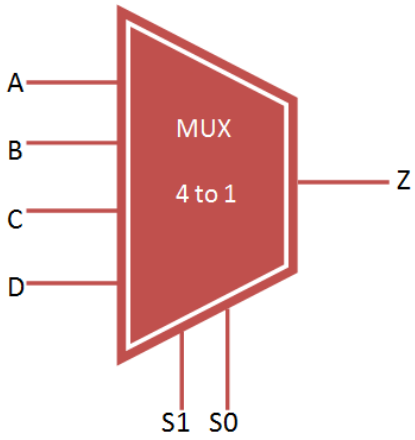
– DUT generates the response and TB stores it into file.

– Result can be compared to Matlab simulations.

Testbench Structures

- TB should be reusable without difficult modifications.
- The structure of the TB should be simple enough so that other people understand its behavior.
- Good test bench propagates all the generics and constants into DUT.
- Question: How to verify that the function of the test bench is correct?

Testbench Example



Input		output
S1	S0	Z
0	0	A
0	1	B
1	0	C
1	1	D

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_mux IS
END tb_mux;

ARCHITECTURE behavioral  OF tb_mux IS
    -- Component Declaration for the Unit
Under Test (UUT)

    COMPONENT mux_4to1
    PORT(
          A, B, C, D : IN  std_logic;
          S0 : IN  std_logic;
          S1 : IN  std_logic;
          Z : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic := '0';
   signal D : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';

    --Outputs
   signal Z : std_logic;
```

```
BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: mux_4to1 PORT MAP (
          A => A, B => B, C => C, D => D, S0 =>
S0, S1 => S1, Z => Z
        );
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

    A <= '1'; B <= '0'; C <= '1'; D <= '0';

    S0 <= '0'; S1 <= '0'; wait for 100 ns;
    S0 <= '1'; S1 <= '0'; wait for 100 ns;
    S0 <= '0'; S1 <= '1'; wait for 100 ns;
    S0 <= '0'; S1 <= '1'; wait for 100 ns;
   end process;

END ARCHITECTURE behavioral;
```