HW2 + Simulations

FPGA & ASIC

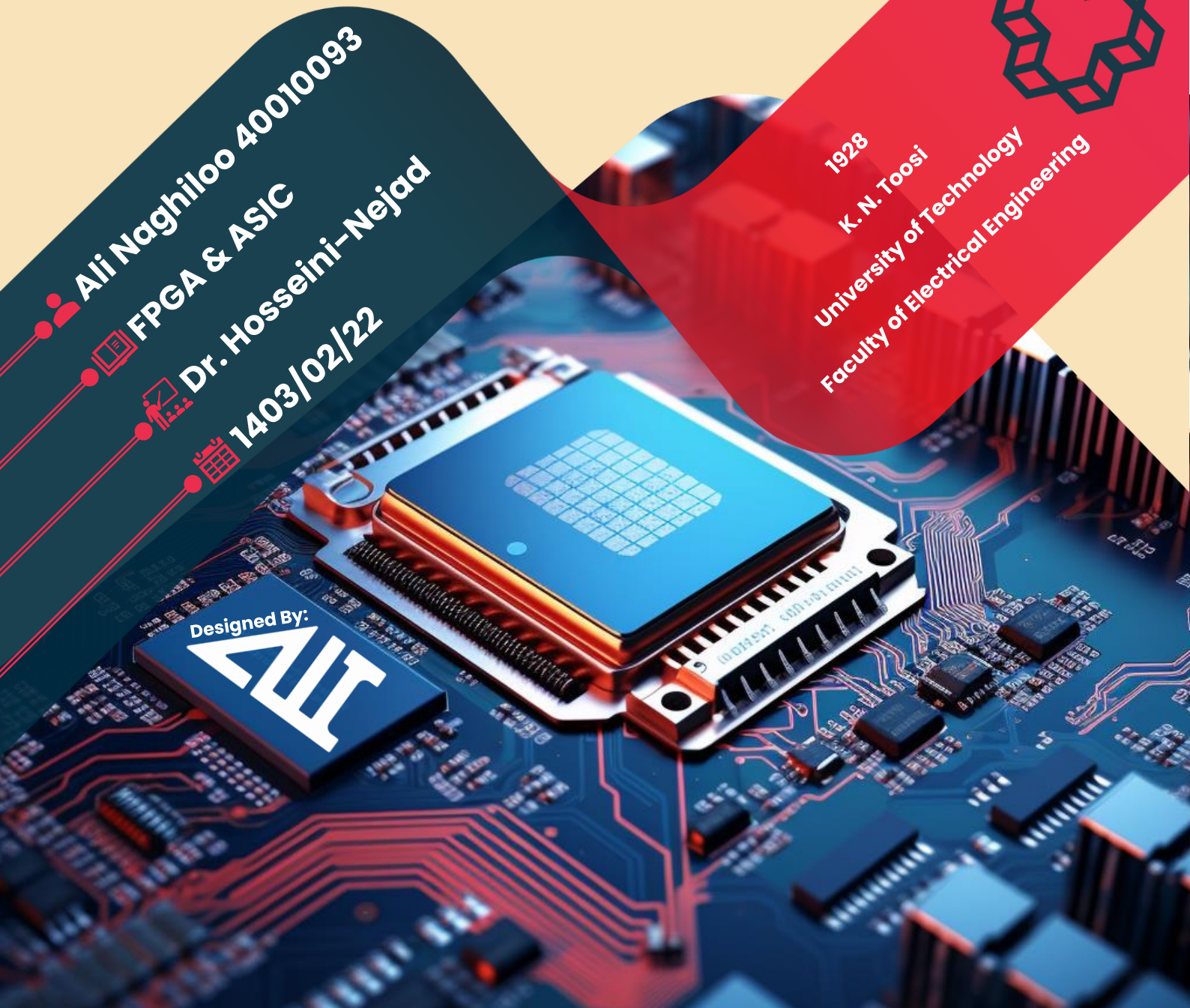
Ali Naghiloo 40010093

FPGA & ASIC

Dr. Hosseini-Nejad

1403/02/22

1928
K. N. Toosi
University of Technology
Faculty of Electrical Engineering



Top-Module Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity convolution is
    generic(conv_width : integer := 8);
    Port ( CLK : in STD_LOGIC;
           input_a : in STD_LOGIC_VECTOR (conv_width-1 downto 0);
           input_x : in STD_LOGIC_VECTOR (conv_width-1 downto 0);
           Y : out STD_LOGIC_VECTOR (19 downto 0));
end convolution;

architecture Behavioral of convolution is
   signal W1,W2 : STD_LOGIC_VECTOR (conv_width-1 downto 0);
   signal W3 : STD_LOGIC_VECTOR (15 downto 0);
   signal W4,W5 : STD_LOGIC_VECTOR (19 downto 0);
   signal W6 : STD_LOGIC;
begin

inA: entity work.reg generic map(8)port map(CLK,'0','1',input_a,W1);
inX: entity work.reg generic map(8)port map(CLK,'0','1',input_x,W2);
multiplier: entity work.multiplier generic map(8)port map(W1,W2,W3);
adder: entity work.adder generic map(16,20)port map(W3,W4,W5);
adder_reg: entity work.reg generic map(20)port map(CLK,W6,'1',W5,W4);
counter: entity work.counter generic map(9) port map(RCO=>W6,CLK=>clk);
output_reg: entity work.reg generic map(20)port map(CLK,'0',W6,W5,Y);
end Behavioral;
```

---------- TB ----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity convolution_TB is
end convolution_TB;

architecture behavior of convolution_TB is
   constant conv_width : integer := 8;
   signal CLK : STD_LOGIC := '0';
   signal input_a : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
   signal input_x : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
   signal Y : STD_LOGIC_VECTOR (19 downto 0);

    component convolution
    generic(conv_width : integer := 8);
    port(
         CLK : in  STD_LOGIC;
         input_a : in  STD_LOGIC_VECTOR (conv_width-1 downto 0);
         input_x : in  STD_LOGIC_VECTOR (conv_width-1 downto 0);
         Y : out  STD_LOGIC_VECTOR (19 downto 0)
        );
    end component;

   -- Clock period definitions
   constant CLK_period : time := 90ns;

begin
```

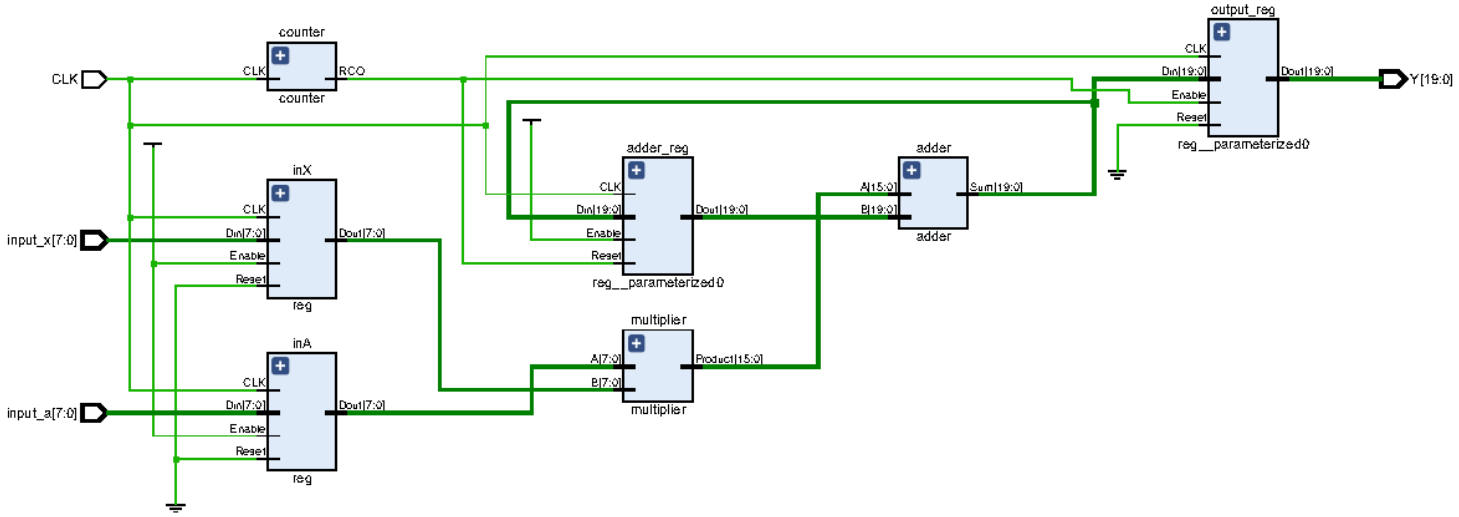
```
   uut: convolution generic map(conv_width) port map (
          CLK => CLK,
          input_a => input_a,
          input_x => input_x,
          Y => Y
        );

CLK <= not CLK after CLK_period/2;

   -- Stimulus process
   stim_proc: process
   begin

      input_a <= b"0000_1011"; input_x <= b"0001_1011";
      wait for CLK_period*2;
      input_a <= b"0001_1011"; input_x <= b"1001_1111";
      wait for CLK_period*3;
      input_a <= b"1001_1000"; input_x <= b"1001_1001";
      wait for CLK_period*4;
      input_a <= b"0000_0000"; input_x <= b"1001_0010";
      wait for CLK_period*2;
      input_a <= b"1001_1011"; input_x <= b"1001_1100";
      wait for CLK_period*5;
      input_a <= b"1001_1000"; input_x <= b"1000_1001";
      wait;
   end process;

end;
```

Utilization-synth & synthesis report is as follows:

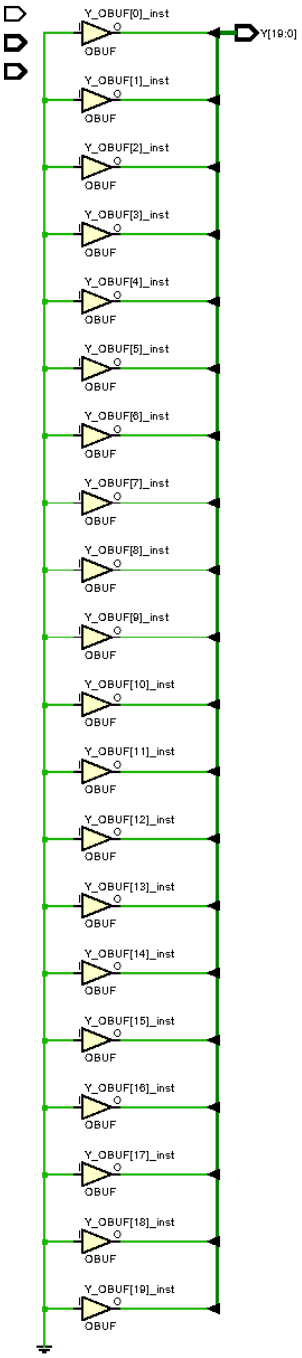
```
Start RTL Component Statistics
---------------------------------------------
Detailed RTL Component Info :
+---Adders :
       2 Input   20 Bit       Adders := 1
+---Registers :
                 20 Bit     Registers := 2
                  8 Bit     Registers := 2
```

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	20	0	0	100	20.00
PHY_CONTROL	0	0	0	2	0.00
PHASER_REF	0	0	0	2	0.00
OUT_FIFO	0	0	0	8	0.00
IN_FIFO	0	0	0	8	0.00
IDELAYCTRL	0	0	0	2	0.00
IBUFDS	0	0	0	96	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	8	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	8	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	100	0.00
ILOGIC	0	0	0	100	0.00
OLOGIC	0	0	0	100	0.00

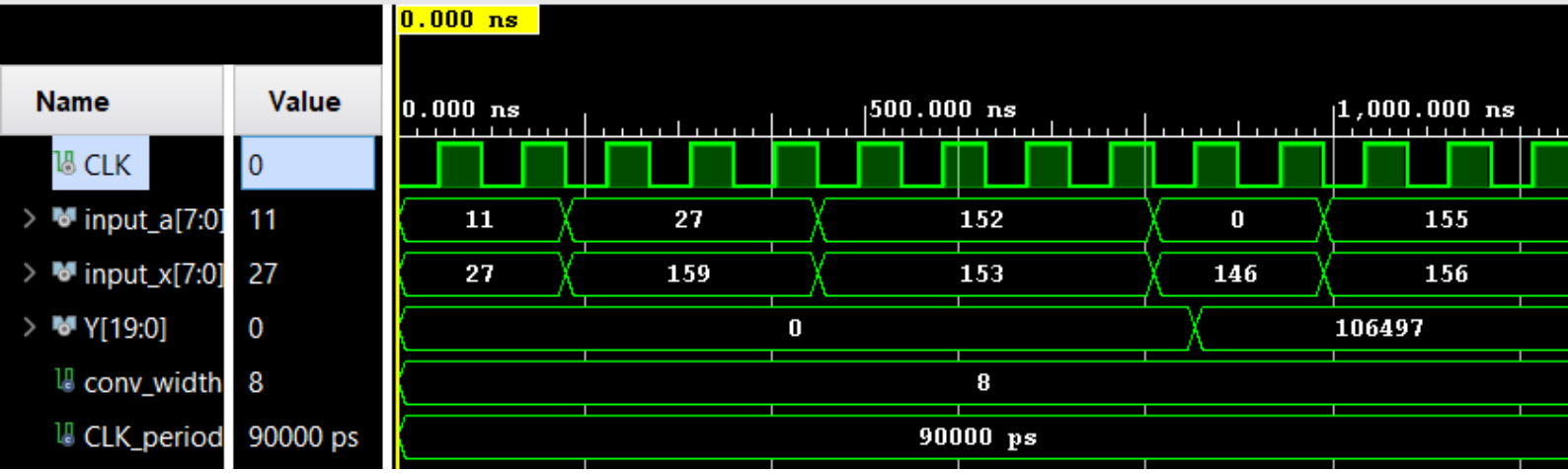
RTL Schematic is as follows:



LUT Schematic:



Waveform


0.000 ns
Name
Value
0.000 ns
500.000 ns
1,000.000 ns
CLK
0
input_a[7:0]
11
11
27
152
0
155
input_x[7:0]
27
27
159
153
146
156
Y[19:0]
0
0
106497
conv_width
8
8
CLK_period
90000 ps
90000 ps

Sub-Modules Code:

REG

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg is
    generic(SR_width : integer := 8);
    Port ( CLK : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Enable : in STD_LOGIC;
           Din : in STD_LOGIC_VECTOR (SR_width-1 downto 0);
           Dout : out STD_LOGIC_VECTOR (SR_width-1 downto 0));
end reg;

architecture Behavioral of reg is
signal temp_dout : STD_LOGIC_VECTOR (SR_width-1 downto 0) :=(others => '0');
begin

process(CLK)
begin


if (clk'event and clk='1') then

    if (reset='1') then
    temp_dout <= (others => '0');
    elsif (enable='1') then
    temp_dout <= din;
    end if;

end if;
end process;
dout <= temp_dout;
end Behavioral;
```

---------- TB ----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_tb is
end reg_tb;

architecture behavior of reg_tb is
    -- Component Declaration for the Unit Under Test (UUT)
    component reg
    generic(SR_width : integer := 8);
    port(
         CLK : in  STD_LOGIC;
         Reset : in  STD_LOGIC;
         Enable : in  STD_LOGIC;
         Din : in  STD_LOGIC_VECTOR (SR_width-1 downto 0);
         Dout : out  STD_LOGIC_VECTOR (SR_width-1 downto 0)
        );
    end component;

   --Inputs
   constant SR_width : integer := 8;
   signal CLK : STD_LOGIC := '0';
   signal Reset : STD_LOGIC := '0';
   signal Enable : STD_LOGIC := '0';
   signal Din : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');

    --Outputs
   signal Dout : STD_LOGIC_VECTOR (7 downto 0);
```

```
    -- Clock period definitions
    constant CLK_period : time := 50 ns;

begin

     -- Instantiate the Unit Under Test (UUT)
    uut: reg generic map (SR_width => SR_width)
         port map (
            CLK => CLK,
            Reset => Reset,
            Enable => Enable,
            Din => Din,
            Dout => Dout
           );

    -- Clock process definitions
    CLK_process :process
    begin
         CLK <= '0';
         wait for CLK_period/2;
         CLK <= '1';
         wait for CLK_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 500 ns.
         Reset <= '1';
       -- insert stimulus here
       wait for CLK_period*6.5;
       Enable <= '1'; Din <= "10011011";
       wait for CLK_period*4;
       Din <= "10000000";
       wait for CLK_period*8;
       Din <= "10000010";
       wait for CLK_period*7;
        Enable <= '0';
       wait for CLK_period*10;
       Din <= "00000000";
       wait for CLK_period*5;
       Din <= "00000001";
       wait;
    end process;

end;
```

Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.std_logic_unsigned.ALL;

entity adder is
    generic(adder_width1 : integer :=16;
            adder_width2 : integer :=20);
    Port ( A : in STD_LOGIC_VECTOR (adder_width1-1 downto 0);
           B : in STD_LOGIC_VECTOR (adder_width2-1 downto 0);
           Sum : out STD_LOGIC_VECTOR (adder_width2-1 downto 0));
end adder;

architecture Behavioral of adder is

begin

Sum <= A + B;

end Behavioral;
```

```
---------------------------------- TB -------------------------------------
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_TB is
--  Port ( );
end adder_TB;

architecture Behavioral of adder_TB is
    constant adder_width1 : integer :=16;
    constant adder_width2 : integer :=20;
    signal A :  STD_LOGIC_VECTOR (adder_width1-1 downto 0);
    signal B :  STD_LOGIC_VECTOR (adder_width2-1 downto 0);
    signal Sum :  STD_LOGIC_VECTOR (adder_width2-1 downto 0);

    component adder is
    generic(adder_width1 : integer :=16;
            adder_width2 : integer :=20);
    Port ( A : in STD_LOGIC_VECTOR (adder_width1-1 downto 0);
           B : in STD_LOGIC_VECTOR (adder_width2-1 downto 0);
           Sum : out STD_LOGIC_VECTOR (adder_width2-1 downto 0));
    end component;

begin

DUT1: adder generic map(adder_width1,adder_width2) port map(A=>A,B=>B,Sum=>sum);

A <= b"0000_1011",b"1001_1111" after 10ns,b"0001_1101" after 20ns;
B <= b"0010_1011",b"0001_1001" after 5ns,b"0001_0001" after 18ns,b"1001_1101" after 25ns;

end Behavioral;
```

Multiplier:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.std_logic_unsigned.ALL;

entity multiplier is
    generic(mult_width : integer :=8);
    Port ( A : in STD_LOGIC_VECTOR (mult_width-1 downto 0);
           B : in STD_LOGIC_VECTOR (mult_width-1 downto 0);
           Product : out STD_LOGIC_VECTOR (2*mult_width-1 downto 0));
end multiplier;

architecture Behavioral of multiplier is
signal extra : std_logic_vector(2*mult_width-1 downto 0);
begin

product <= conv_std_logic_vector(conv_integer(A * B),2*mult_width);

end Behavioral;
```

---------- TB ----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity multiplier_TB is
--  Port ( );
end multiplier_TB;

architecture Behavioral of multiplier_TB is

    constant mult_width : integer :=8;
    signal A : STD_LOGIC_VECTOR (mult_width-1 downto 0);
    signal B : STD_LOGIC_VECTOR (mult_width-1 downto 0);
    signal Product : STD_LOGIC_VECTOR (2*mult_width-1 downto 0);

component multiplier is
    generic(mult_width : integer :=8);
    Port ( A : in STD_LOGIC_VECTOR (mult_width-1 downto 0);
           B : in STD_LOGIC_VECTOR (mult_width-1 downto 0);
           Product : out STD_LOGIC_VECTOR (2*mult_width-1 downto 0));
end component;

begin

DUT1: multiplier generic map(mult_width=>mult_width) port map(A=>A,B=>B,Product=>Product);

A <= b"0000_1011",b"0000_1111" after 10ns,b"0001_1101" after 20ns;
B <= b"0000_1011",b"0000_1001" after 5ns,b"0000_0001" after 18ns,b"0000_0001" after 25ns;


end Behavioral;
```

Counter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.std_logic_unsigned.ALL;

entity counter is
    generic(interval : integer :=10);
    Port ( CLK : in STD_LOGIC;
           RCO : out STD_LOGIC);
end counter;

architecture Behavioral of counter is
signal count : integer range 0 to interval-1 :=0;
signal temp_RCO :std_logic :='0';
begin
process(clk)
begin

if (clk'event and clk='1') then

    count <= count + 1;
    if (count = interval) then
    count <= 0;
    end if;

end if;
end process;

temp_RCO <= '1' when count=interval else '0';
RCO <= temp_RCO;

end Behavioral;
```

---------- TB ----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity counter_TB is
--  Port ( );
end counter_TB;

architecture Behavioral of counter_TB is
signal CLK : STD_LOGIC :='0';
signal RCO : STD_LOGIC;
component counter is
    generic(interval : integer);
    Port ( CLK : in STD_LOGIC;
           RCO : out STD_LOGIC);
end component;
constant period : time := 90ns;
begin

clk <= not clk after (period/2);
DUT1: counter generic map(10) port map(CLK=>CLK,RCO=>RCO);

end Behavioral;
```