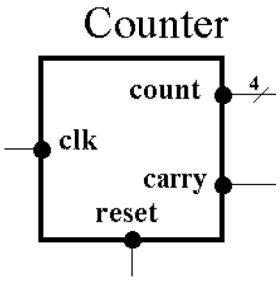
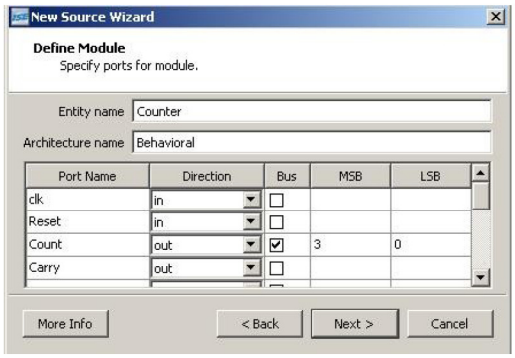
Example VHDL Entity - 0 to 9 Counter



- **Define input & output ports**



VHDL Model

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

entity Counter is
Port(clk : in STD_LOGIC;
     Reset : in STD_LOGIC;
     Count : out STD_LOGIC_VECTOR (3 downto 0);
     Carry : out STD_LOGIC);
end Counter;

architecture Behavioral of Counter is
signal count_int : std_logic_vector(3 downto 0);
-- define interal register
begin

process (reset, clk)
begin
  if reset = '1' then
    count_int <= "0000"; -- set counter, and
    carry <= '0'; -- carry to zero
  elsif clk'event and clk = '1' then
    if count_int <= "1000" then -- check count
       count_int <= count_int + "1"; --increment
       carry <= '0'; -- show still below 9
    else -- else we are at 9
       count_int <= "0000"; -- roll over count
       carry <= '1'; -- flag roll over
    end if;
  end if;
end process;
count <= count_int; -- send value to the outside

end Behavioral;
```

VHDL Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY Counter_TB IS
END Counter_TB;

ARCHITECTURE behavior OF Counter_TB IS

 -- Component Declaration for UUT)
 COMPONENT Counter
 PORT(
       clk : IN std_logic;
       Reset : IN std_logic;
       Count : OUT std_logic_vector(3 downto 0);
       Carry : OUT std_logic );
  END COMPONENT;

 --Inputs
 signal clk : std_logic := '0';
 signal Reset : std_logic := '0';

 --Outputs
 signal Count : std_logic_vector(3 downto 0);
 signal Carry : std_logic;

 -- Clock period definitions
 constant clk_period : time := 50 us; -- = 20KHz

 BEGIN
 -- Instantiate the Unit Under Test (UUT)
 uut: Counter PORT MAP (
       clk => clk,
       Reset => Reset,
       Count => Count,
       Carry => Carry );

 clk_process :process -- Clock process definitions
   begin
     clk <= '0';
     wait for clk_period/2;
     clk <= '1';
     wait for clk_period/2;
   end process;

 setup : PROCESS -- generate short reset pulse
   BEGIN
     reset <= '0';
     wait for 3 ns;
     reset <= '1';
     wait for 3 ns;
     reset <= '0';
     wait;
   end PROCESS;
 END;
```