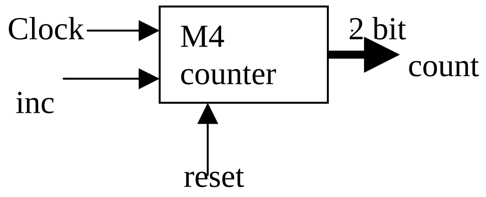
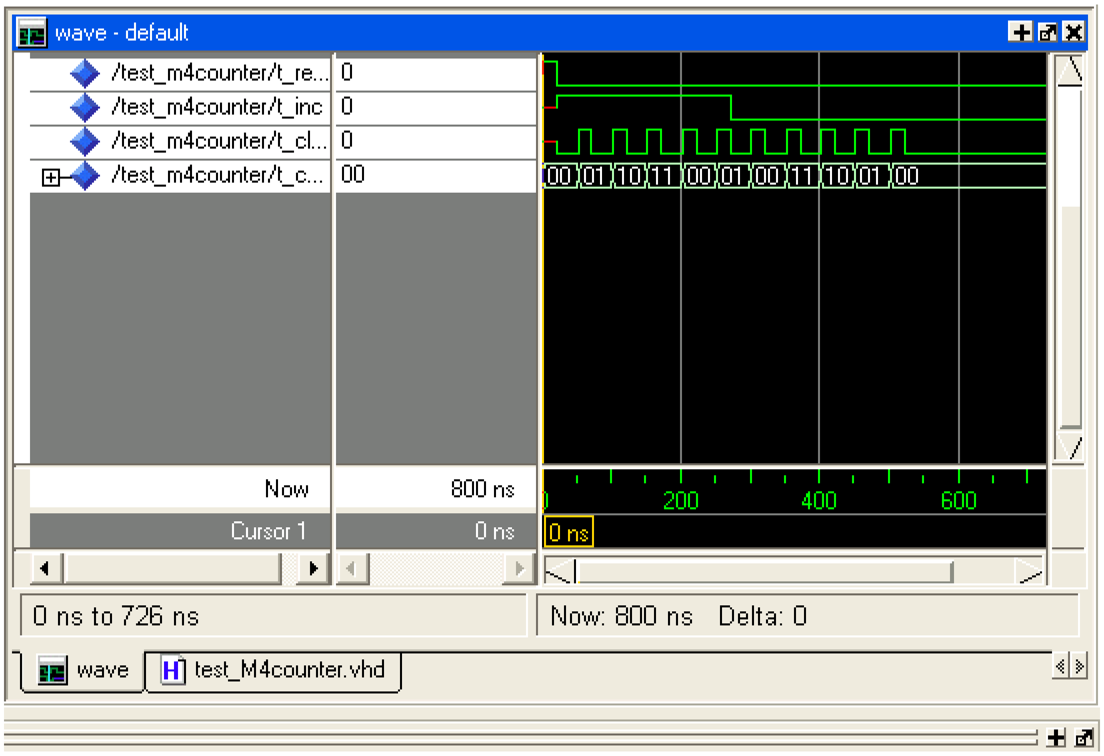
Module 4 increment (decrement) counter



reset	inc	clock	count		
1	X	X	0	0	reset
0	1		0	1	increment
0	1		1	0	
0	1	↑	1	1	
0	1		0	0	
0	1		0	1	
0	0		0	0	decrement
0	0		1	1	
0	0		1	0	
0	0		0	1	
0	0		0	0	


wave - default

/test_m4counter/t_re...	0	
/test_m4counter/t_inc	0	
/test_m4counter/t_cl...	0	
/test_m4counter/t_c...	00	00 01 10 11 00 01 00 11 10 01 00
Now	800 ns	200 400 600
Cursor 1	0 ns	0 ns

0 ns to 726 ns   Now: 800 ns   Delta: 0

wave   test_M4counter.vhd

-- M4counter.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned. all;
use ieee.std_logic_arith. all;

entity M4counter is
port( reset: in std_logic;
                inc: in std_logic;
      clock: in std_logic;
      count: inout std_logic_vector(1 downto 0)
);
end M4counter;

architecture behav1 of M4counter is
begin
process (clock, reset)
begin
  if reset='1' then
     count <= "00" ;
  elsif clock='1' and clock' event then
       if inc='1' then
         count <= count +1;
      else
          count <= count -1;
       end if;
  end if;

end process;
end behav1;
```

-- test_M4counter.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned. all;
use ieee.std_logic_arith. all;
use work.all;

entity test_m4counter is
end test_m4counter;


architecture TB of test_m4counter is
   component m4counter
    port(
                reset: in std_logic;
                inc: in std_logic;
      clock: in std_logic;
      count: inout std_logic_vector(1 downto 0)
          );
    end component;

    signal T_reset:           std_logic;
    signal T_inc:                std_logic;
    signal T_clock:            std_logic;
    signal T_count:             std_logic_vector(1 downto 0) := "UU";


begin

    U_m4counter: m4counter port map ( T_reset,T_inc, T_clock, T_count
);

    process

      variable err_cnt: integer := 0;

   begin
 --initialization
   T_reset<='1';
    wait for 2 ns;
    assert(T_count="00") report "Error!" severity error;
     if (T_count/="00") then
          err_cnt := err_cnt + 1;
      end if;
    wait for 20 ns;
--working cases
     T_reset <= '0' ;
     T_clock <= '0';

--case 1,2,3,4,5 for increament
      T_inc <= '1' ;

   -- case1
       wait for 30 ns;
       T_clock <= '1';
       wait for 10 ns;

          assert (T_count = "01"  ) report "Error1!" severity error;
          if (T_count /= "01" ) then
          err_cnt := err_cnt + 1;
          end if;
       wait for 10 ns;
       T_clock <= '0';

--case2
       wait for 30 ns;
       T_clock <= '1';
       wait for 10 ns;

       assert (T_count = "10"  ) report "Error2!" severity error;
       if (T_count /= "10" ) then
       err_cnt := err_cnt + 1;
       end if;
       wait for 10 ns;
       T_clock <= '0';

 --case3
      wait for 30 ns;
      T_clock <= '1';
      wait for 10 ns;
      assert (T_count = "11"  ) report "Error3!" severity error;
      if (T_count /= "11" ) then
      err_cnt := err_cnt + 1;
      end if;
      wait for 10 ns;
      T_clock <= '0';

 --case4
      wait for 30 ns;
      T_clock <= '1';
      wait for 10 ns;

      assert (T_count = "00"  ) report "Error4!" severity error;
      if (T_count /= "00" ) then
      err_cnt := err_cnt + 1;
      end if;
      wait for 10 ns;
      T_clock <= '0';

  --case5
       wait for 30 ns;
       T_clock <= '1';
       wait for 10 ns;

       assert (T_count = "01"  ) report "Error5!" severity error;
       if (T_count /= "01" ) then
       err_cnt := err_cnt + 1;
       end if;
       wait for 10 ns;
       T_clock <= '0';




-- case6,7,8,9,10 for decreament
              T_inc <= '0';
 --case6
              wait for 30 ns;
              T_clock <= '1';
              wait for 10 ns;
             assert (T_count = "00" ) report "Error6!" severity error;
                 if (T_count /=  "00" ) then
                 err_cnt := err_cnt + 1;
                 end if;
              wait for 10 ns;
              T_clock <= '0';
 --case7
              wait for 30 ns;
              T_clock <= '1';
              wait for 10 ns;
             assert (T_count = "11" ) report "Error7!" severity error;
              if (T_count /=  "11" ) then
              err_cnt := err_cnt + 1;
              end if;
              wait for 10 ns;
              T_clock <= '0';

 --case8
              wait for 30 ns;
              T_clock <= '1';
              wait for 10 ns;
             assert (T_count = "10" ) report "Error8!" severity error;
              if (T_count /=  "10" ) then
              err_cnt := err_cnt + 1;
              end if;
              wait for 10 ns;
              T_clock <= '0';
 --case9
              wait for 30 ns;
              T_clock <= '1';
              wait for 10 ns;
             assert (T_count = "01" ) report "Error9!" severity error;
              if (T_count /=  "01" ) then
              err_cnt := err_cnt + 1;
              end if;
              wait for 10 ns;
              T_clock <= '0';
 --case10
              wait for 30 ns;
              T_clock <= '1';
              wait for 10 ns;
             assert (T_count = "00" ) report "Error10!" severity error;
              if (T_count /=  "00" ) then
              err_cnt := err_cnt + 1;
              end if;
              wait for 10 ns;
              T_clock <= '0';



-- summary of all the tests
      if (err_cnt=0) then
          assert false
          report "Testbench of M4counter completed successfully!"
          severity note;
      else
          assert true
          report "Something wrong, try test again"
          severity error;
      end if;
      wait;
    end process;
end TB;
configuration CFG_TB of test_m4counter is
      for TB
      end for;
end CFG_TB;
```