VHDL Brush-up

1. 8-bit NAND gate with concurrent
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Ebitnand is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

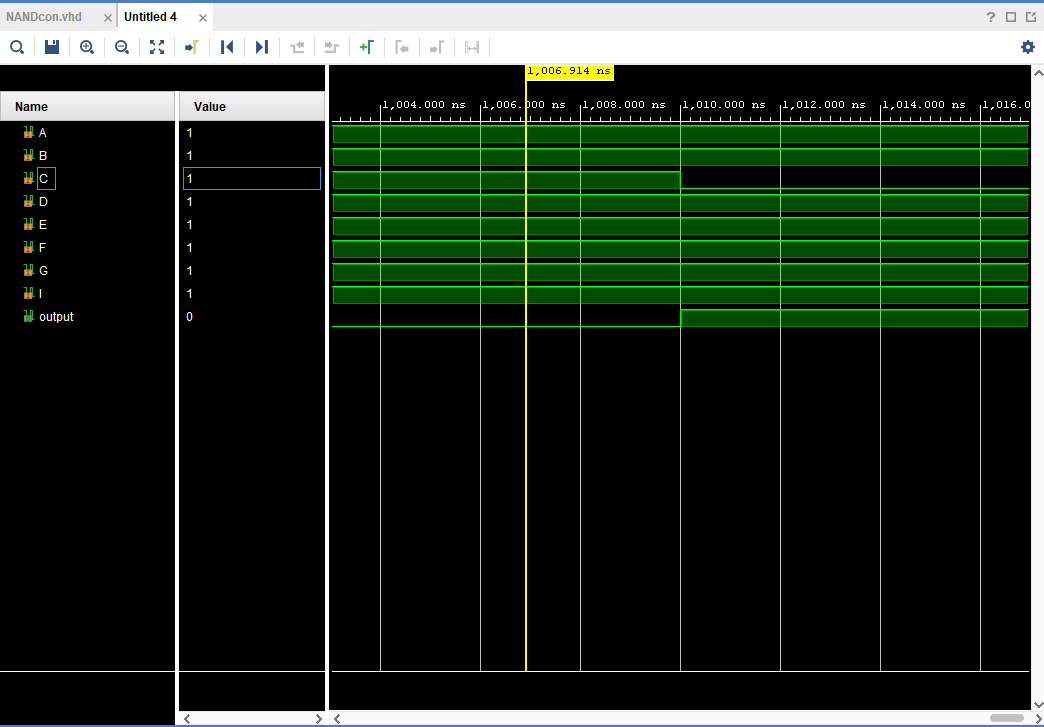
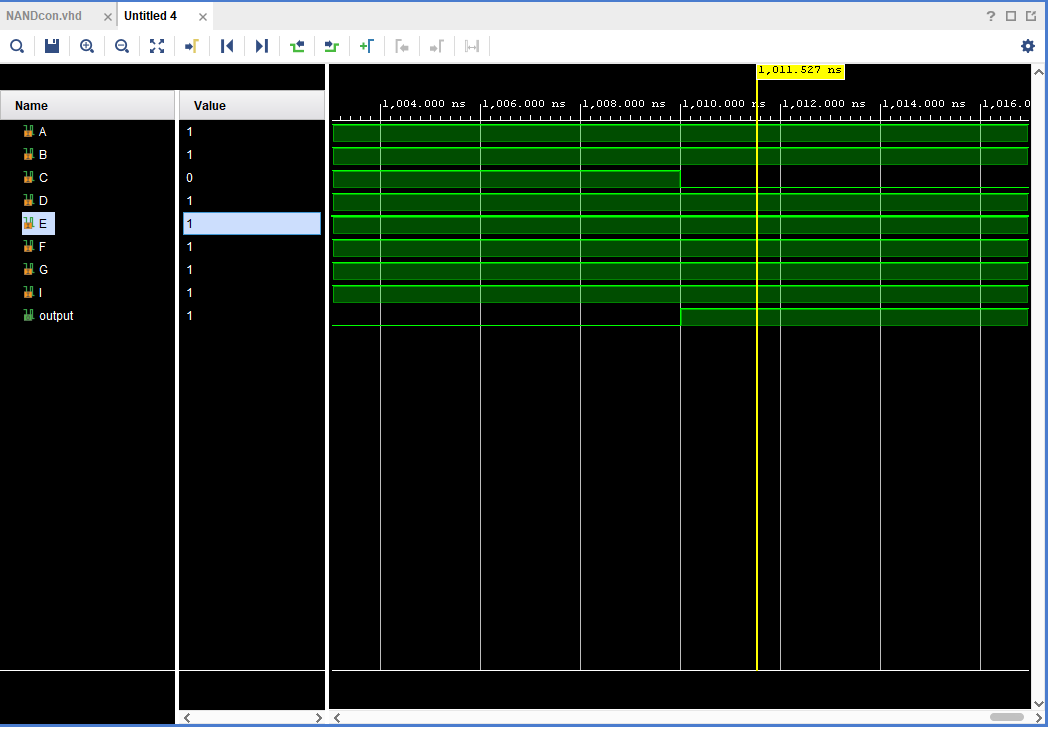
end;

architecture Behavioral of Ebitnand is

begin

output<= not(a and b and c and d and e and f and g and i);

end Behavioral;

* + Simulation:
  + 
  + 

1. 8-bit NAND with conditional
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity NANDcon is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

end NANDcon;

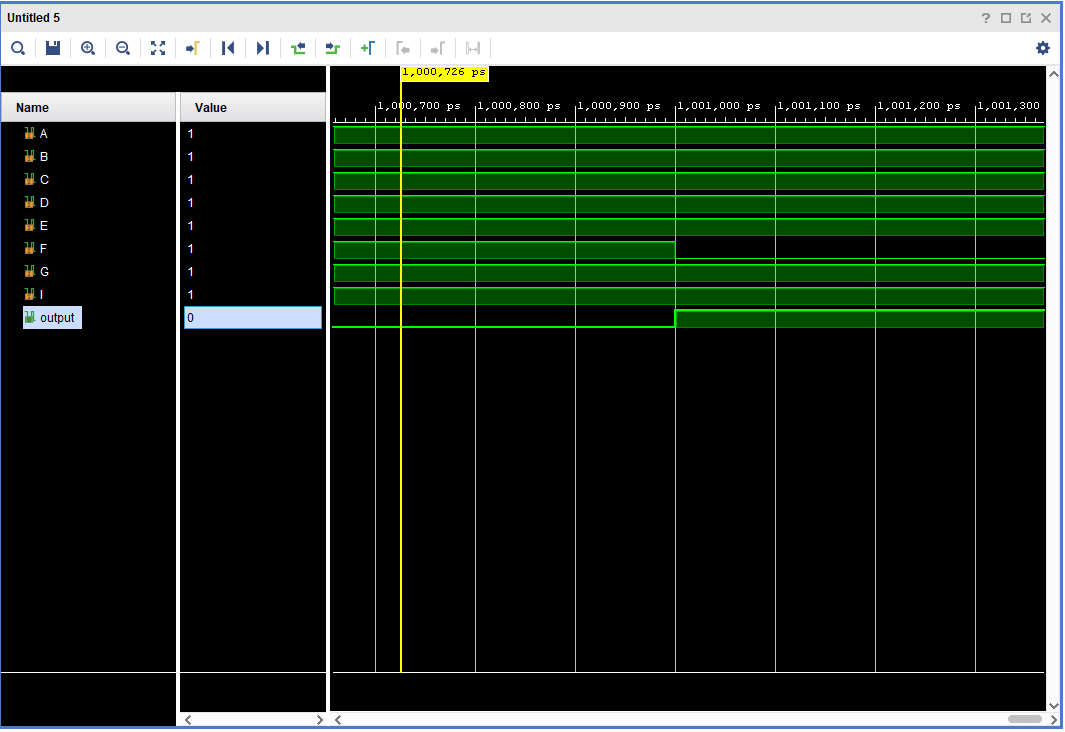
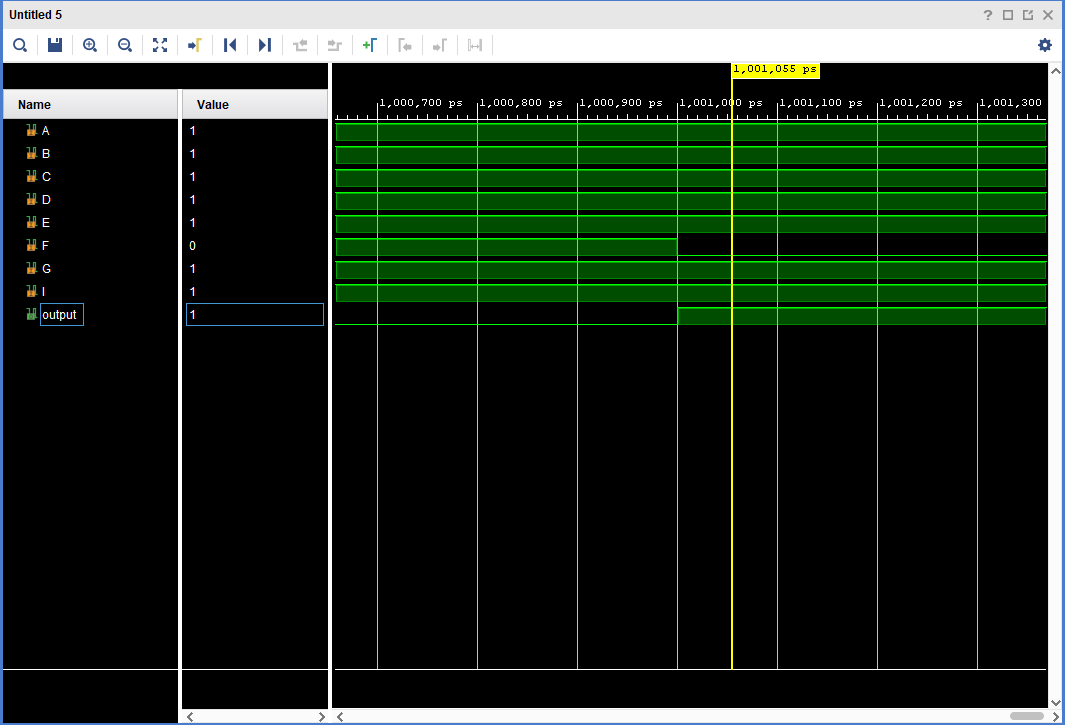
architecture Behavioral of NANDcon is

begin

output<= '0' when (A='1' and B='1' and C='1' and D='1' and E='1' and F='1' and G='1' and I='1') else

'1';

end Behavioral;

* + Simulation:
  + 
  + 

1. 8-bit with selected signal assignment
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity NAND\_selected is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

end NAND\_selected;

architecture Behavioral of NAND\_selected is

signal tmp1:std\_logic\_vector(7 downto 0);

begin

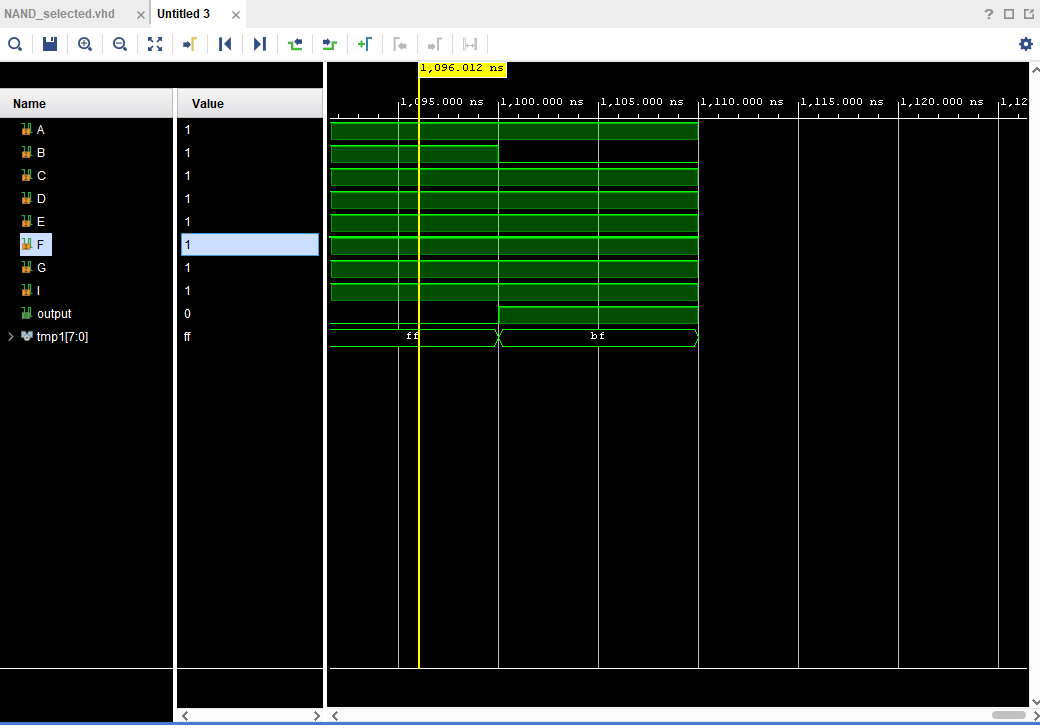
tmp1<= a & b & c & d & e & f & g & i;

with tmp1 select

output<= '0' when "11111111",

'1' when others;

end Behavioral;

* Simulation: 

1. 8-bit nor concurrent
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity nor\_con is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

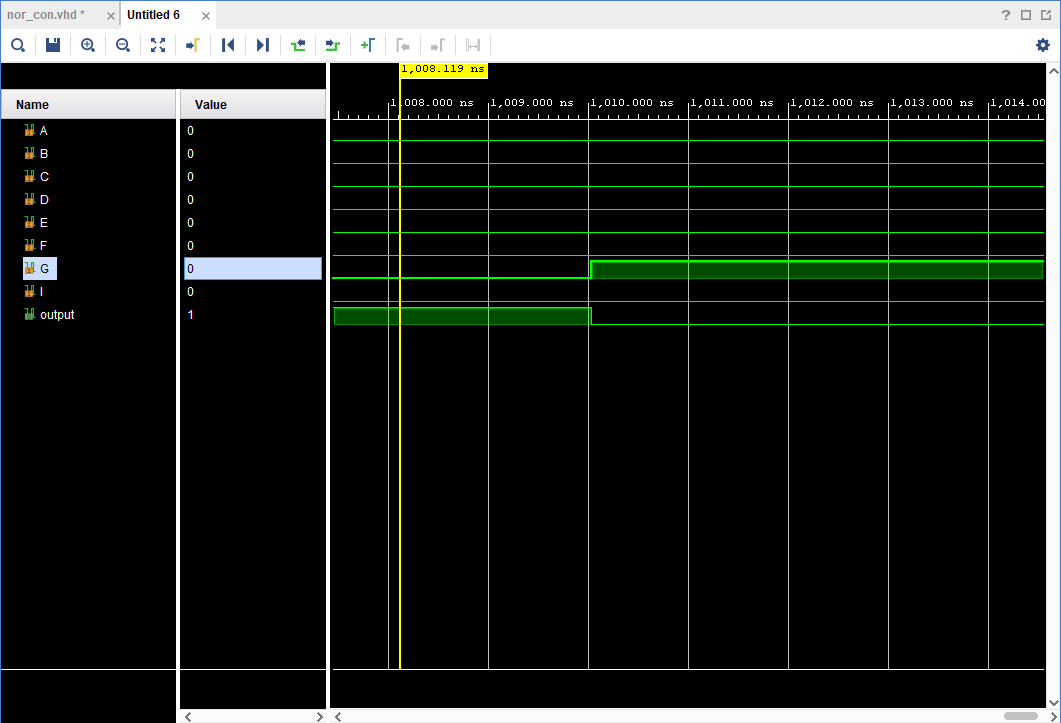
end nor\_con;

architecture Behavioral of nor\_con is

begin

output<= not(a or b or c or d or e or f or g or i);

end Behavioral;

* + simulations: 

1. 8 bit nor conditional
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity nor\_conditional is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

end ;

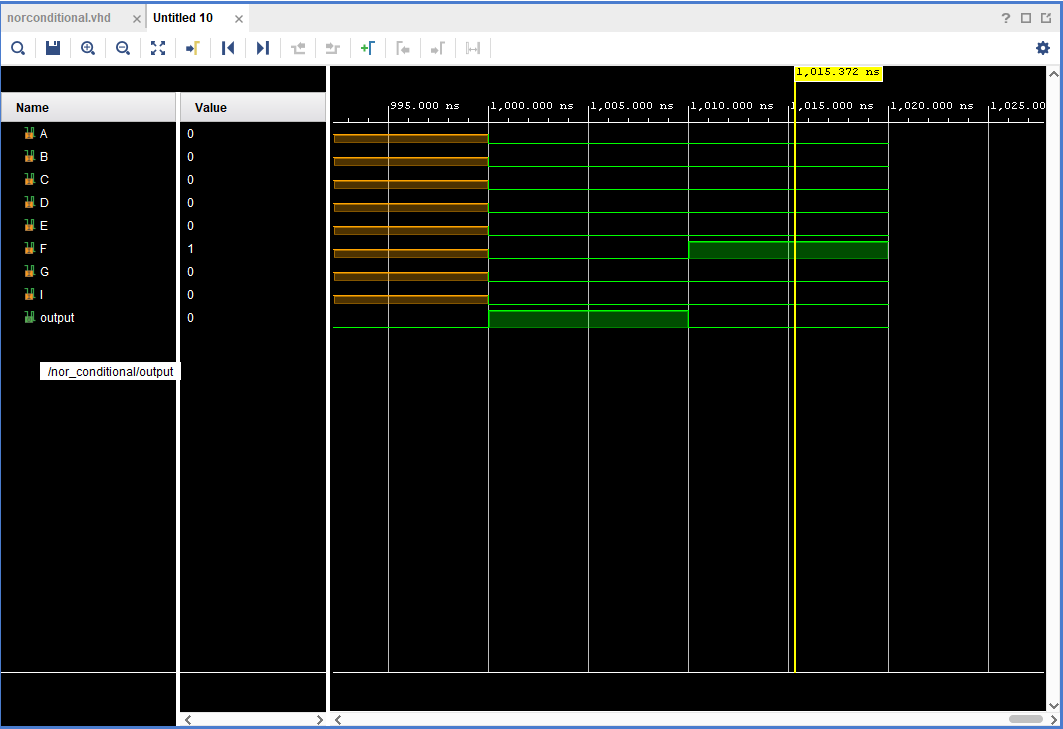
architecture bev of nor\_conditional is

begin

output<= '1' when (A='0' and B='0' and C='0' and D='0' and E='0' and F='0' and G='0' and I='0')

else '0';

end;

* + Simulation: 

1. 8 bit nor with selected signal
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity norselect is

port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

E : in STD\_LOGIC;

F : in STD\_LOGIC;

G : in STD\_LOGIC;

I : in STD\_LOGIC;

output : out STD\_LOGIC);

end norselect;

architecture Behavioral of norselect is

signal tmp1:std\_logic\_vector(7 downto 0);

begin

tmp1<= a & b & c & d & e & f & g & i;

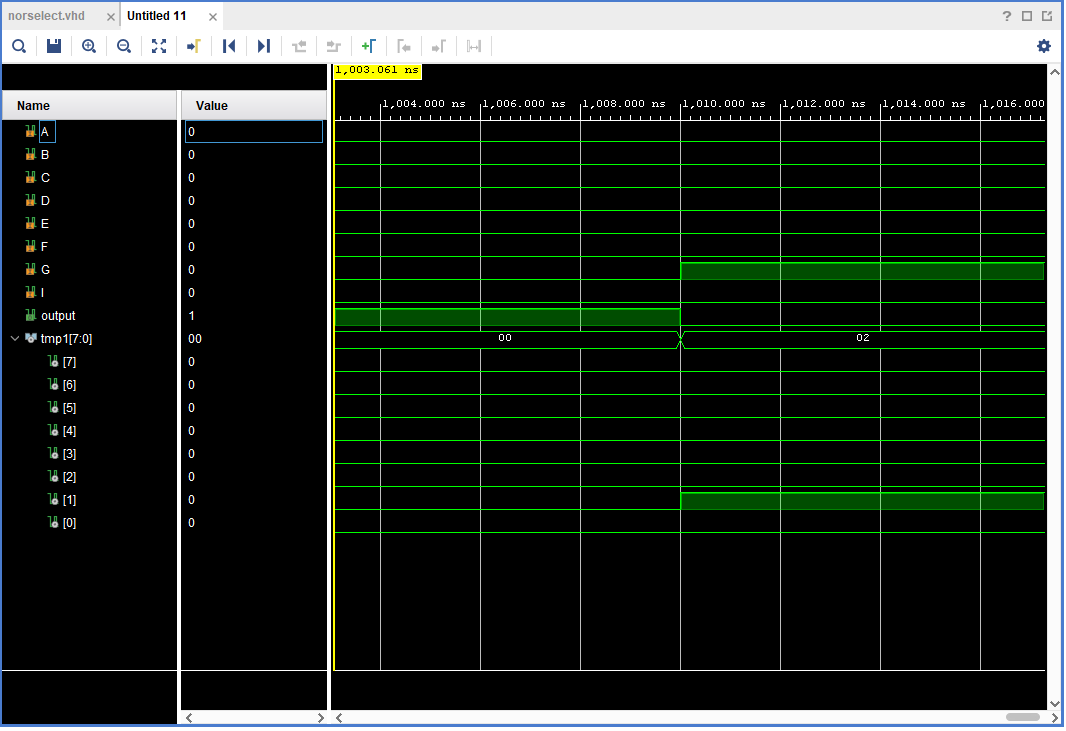
with tmp1 select

output<='1' when "00000000",

'0' when others;

end Behavioral;

* + simulation:



1. NAND gate using a process statement
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity NANDprocess is

port (a,b,c,d,e,f,g,i: in std\_logic;

output: out std\_logic);

end NANDprocess;

architecture Behavioral of NANDprocess is

begin

process(a,b,c,d,e,f,g,i)

begin

if(a='1') and (b='1') and (c='1') and (d='1') and (e='1') and (f='1') and (g='1') and (i='1') then output<='0';

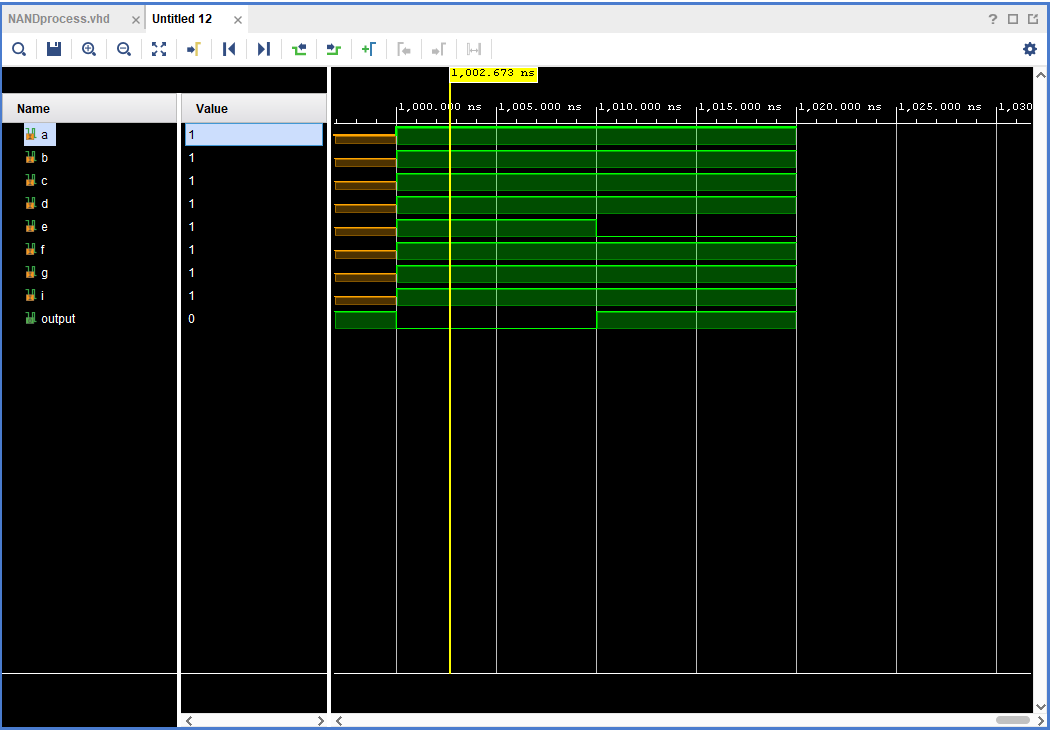
else

output<='1';

end if;

end process;

end;

* + - * Simulation: 

1. Nor with a process statement
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity norprocess is

port (a,b,c,d,e,f,g,i: in std\_logic;

output: out std\_logic);

end norprocess;

architecture beh of norprocess is

begin

process(a,b,c,d,e,f,g,i)

begin

if(a='0') and (b='0') and (c='0') and (d='0') and (e='0') and (f='0') and (g='0') and (i='0') then output<='1';

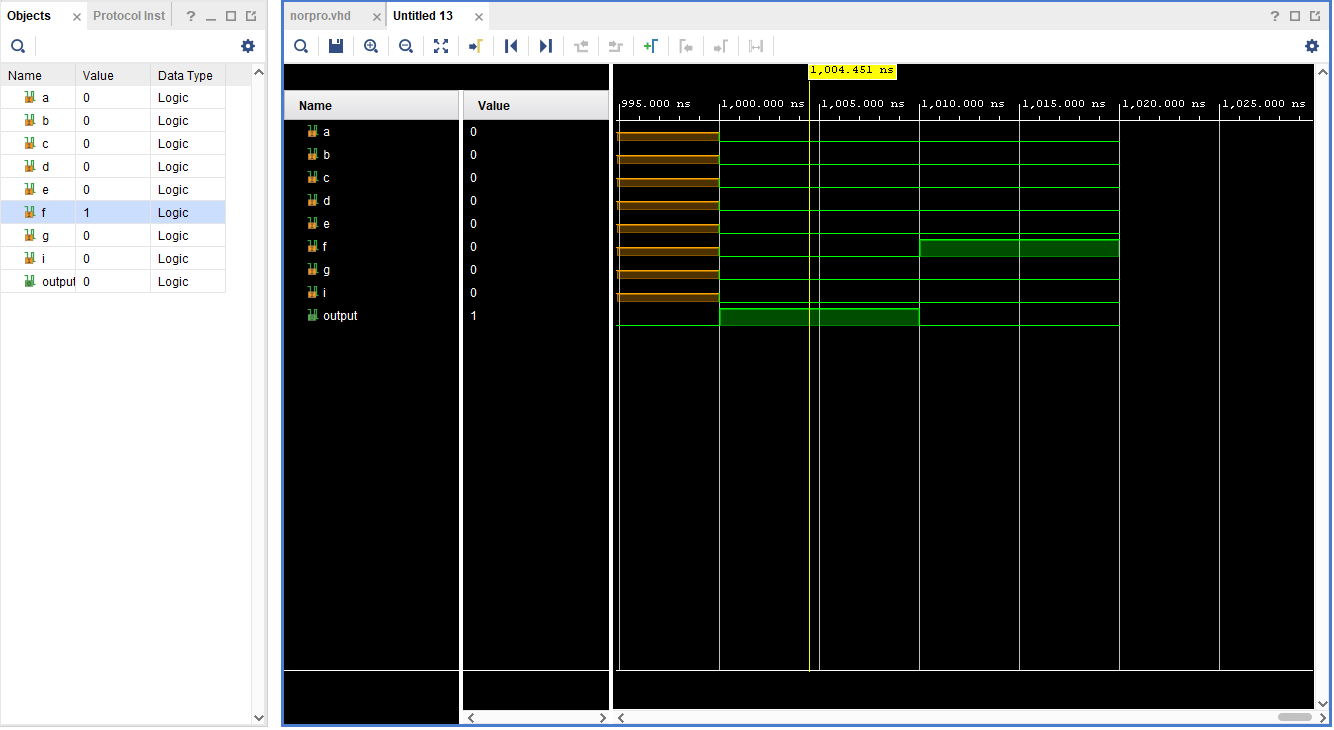
else

output<='0';

end if;

end process;

end;

* + Simulation: 

1. 8 to 1 mux (if statement and case statement)
   * Case statements
   * library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux is

Port ( x : in STD\_LOGIC\_vector(7 downto 0);

sel : in STD\_LOGIC\_vector(2 downto 0);

output : out STD\_LOGIC);

end mux;

architecture Behavioral of mux is

begin

process(sel, x)

begin

case sel is

when "000" => output <=x(0);

when "001" => output <=x(1);

when "010" => output <=x(2);

when "011" => output <=x(3);

when "100" => output <=x(4);

when "101" => output <=x(5);

when "110" => output <=x(6);

when "111" => output <=x(7);

when others=> output<= 'Z';

end case;

end process;

end;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

* + With if statements
  + entity mux is

Port ( x : in STD\_LOGIC\_vector(7 downto 0);

sel : in STD\_LOGIC\_vector(2 downto 0);

output : out STD\_LOGIC);

end mux;

architecture Behavioral of mux is

begin

process(sel, x)

begin

if (sel="000") then output<=x(0);

elsif (sel="001") then output<=x(1);

elsif (sel="010") then output<=x(2);

elsif (sel="011") then output<=x(3);

elsif (sel="100") then output<=x(4);

elsif (sel="101") then output<=x(5);

elsif (sel="110") then output<=x(6);

elsif (sel="000") then output<=x(7);

else

output<='-';

end if;

end process;

end;

* + Simulation: