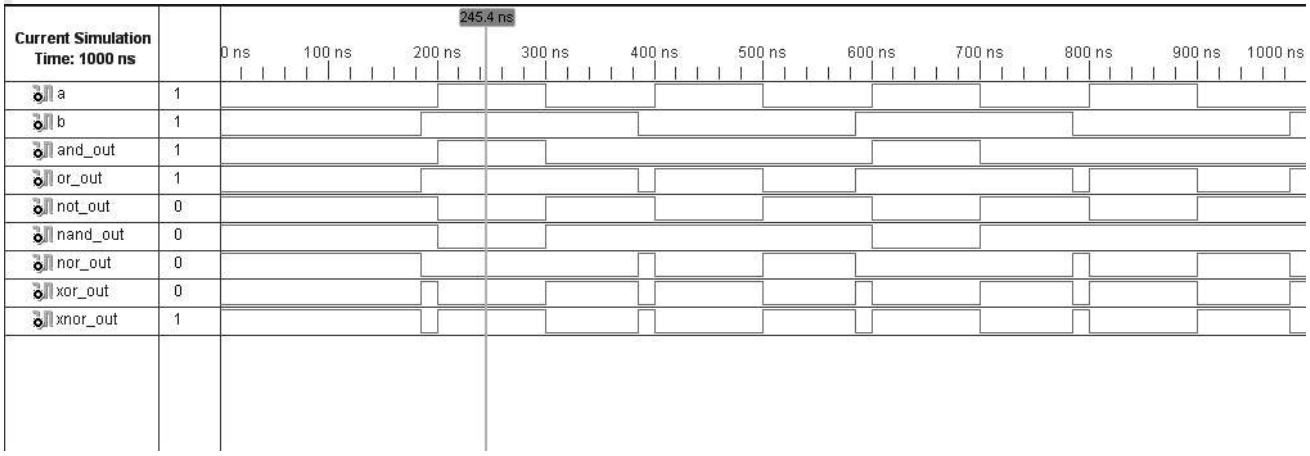
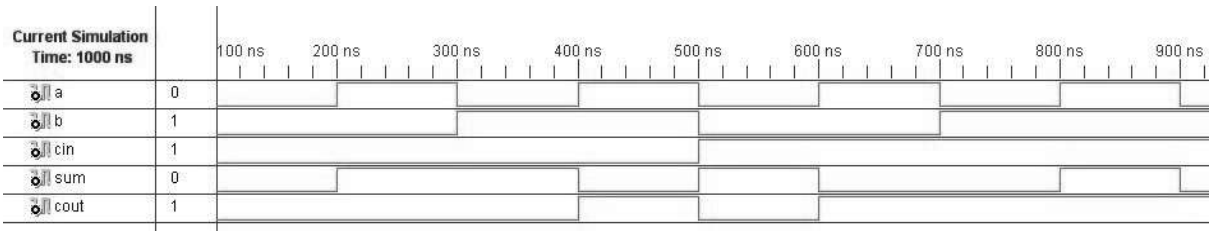
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity logic_gates is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           and_out, or_out, not_out, nand_out, nor_out, xor_out, xnor_out : out  STD_LOGIC);
end logic_gates;

architecture Behavioral of logic_gates is

begin

   and_out <= a and b;
   or_out <= a or b;
   not_out <= not a;
   nand_out <= a nand b;
   nor_out <= a nor b;
   xor_out <= a xor b;
   xnor_out <= a xnor b;

end Behavioral;
```



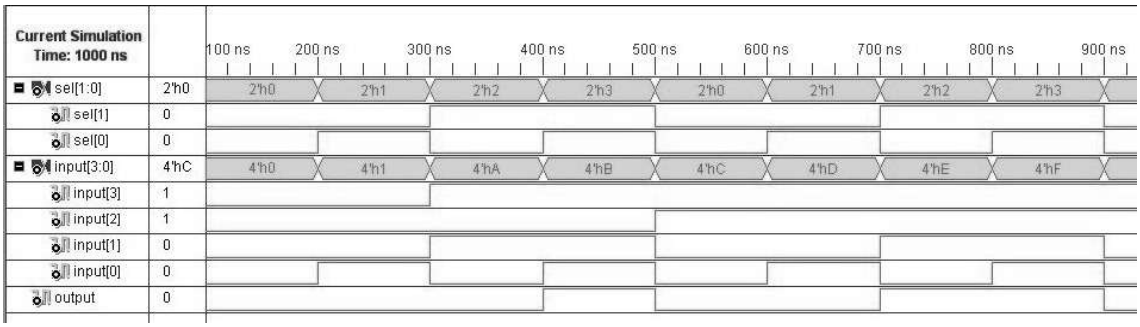
Full Adder

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

entity fulladder is
      port (      a,b,cin: in std_logic;
                  sum, cout: out std_logic);
end fulladder;

architecture Behavioral of fulladder is

begin
      sum <= (a xor b) xor cin;
      cout <= (a and b) or (b and cin) or (a and cin);

end Behavioral;
```



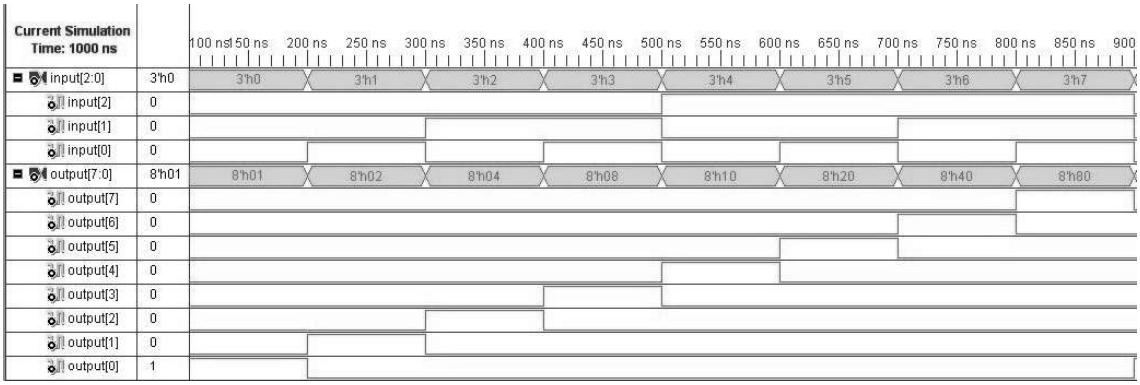
Multiplexer

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

entity mux is
     Port (     sel : in STD_LOGIC_VECTOR ( 1 downto 0 );
                input : in STD_LOGIC_VECTOR ( 3 downto 0);
                output : out STD_LOGIC);
end mux;

architecture Behavioral of mux is

begin
     with sel select
          output <=  input(0) when "00",
                     input(1) when "01",
                     input(2) when "10",
                     input(3) when "11",
                     '0' when others;
end Behavioral;
```



Decoder

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

entity decoder is
     port (     input: in std_logic_vector (2 downto 0);
                output: out std_logic_vector (7 downto 0));
end decoder;

architecture Behavioral of decoder is

begin
     output(0) <=  '1' when input = "000" else '0';
     output(1) <=  '1' when input = "001" else '0';
     output(2) <=  '1' when input = "010" else '0';
     output(3) <=  '1' when input = "011" else '0';
     output(4) <=  '1' when input = "100" else '0';
     output(5) <=  '1' when input = "101" else '0';
     output(6) <=  '1' when input = "110" else '0';
     output(7) <=  '1' when input = "111" else '0';
end Behavioral;
```



Seven Segment Display

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

entity sevensegment is
     port (     a: in std_logic_vector (1 downto 0);
                b: in std_logic_vector (3 downto 0);
                an: out std_logic_vector (3 downto 0);
                seg: out std_logic_vector (6 downto 0));
end sevensegment;

architecture Behavioral of sevensegment is

begin
     seg<= "0000001" when b = "0000" else
           "1001111" when b = "0001" else
           "0010010" when b = "0010" else
           "0000110" when b = "0011" else
           "1001100" when b = "0100" else
           "0100100" when b = "0101" else
           "0100000" when b = "0110" else
           "0001111" when b = "0111" else
           "0000000" when b = "1000" else
           "0000100" when b = "1001" else
           "0001000" when b = "1010" else
           "0000000" when b = "1011" else
           "0110001" when b = "1100" else
           "0000001" when b = "1101" else
           "0110000" when b = "1110" else
           "0111000" when b = "1111" else
           "1111111";

     an <= "1110" when a = "00" else
           "1101" when a = "01" else
           "1011" when a = "10" else
           "0111" when a = "11" else
           "1111";


end Behavioral;
```

Note: This program doesn't contain any diagrams, the program was directly implemented in hardware

D-FlipFlop

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

entity dflipflop is
      Port( clr,clk,d: in std_logic;
                  q: out std_logic );
end dflipflop;

architecture Behavioral of dflipflop is

begin
      process(clk,clr)
            begin
                  if (clr='0') then q<='0';
                  elsif rising_edge(clk) then q<=d;
                  end if;
            end process;

end Behavioral;
```

JK-FlipFlop

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

entity jkflipflop is
     Port ( j,k,clr,clk: in STD_LOGIC;
                    q,q0: out STD_LOGIC);
end jkflipflop;

architecture Behavioral of jkflipflop is
signal qtemp: std_logic;
begin

     process (clr,clk)
          begin
               if (clr='0') then qtemp<='0';
               elsif rising_edge(clk) then
                    if (j='0' and k='0') then NULL;
                    elsif (j='0' and k='1') then qtemp<='1';
                    elsif (j='1' and k='0') then qtemp<='0';
                    elsif (j='1' and k='1') then qtemp<= not qtemp;
                    end if;
               end if;
     end process;
     q<=qtemp;
     q0<=not qtemp;

end Behavioral;
```

Hex-Counter

counter.vhd

```
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;
            count: out STD_LOGIC_VECTOR (3 downto 0)
      );
end counter;

architecture Behavioral of counter is
signal c: STD_LOGIC_VECTOR (29 downto 0);
begin
      process(clk)
      begin
            if(rising_edge(clk)) then
                  c <= c + 1;
            end if;
      end process;
      count <= c(28 downto 25);
end Behavioral;
```

sevensegment.vhd

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

entity sevensegment is
      port (     a: in std_logic_vector (1 downto 0);
                        b: in std_logic_vector (3 downto 0);
                        an: out std_logic_vector (3 downto 0);
                        seg: out std_logic_vector (6 downto 0));
end sevensegment;

architecture Behavioral of sevensegment is
```

```
begin
      seg <="0000001" when b = "0000" else
                  "1001111" when b = "0001" else
                  "0010010" when b = "0010" else
                  "0000110" when b = "0011" else
                  "1001100" when b = "0100" else
                  "0100100" when b = "0101" else
                  "0100000" when b = "0110" else
                  "0001111" when b = "0111" else
                  "0000000" when b = "1000" else
                  "0000100" when b = "1001" else
                  "0001000" when b = "1010" else
                  "0000000" when b = "1011" else
                  "0110001" when b = "1100" else
                  "0000001" when b = "1101" else
                  "0110000" when b = "1110" else
                  "0111000" when b = "1111" else
                  "1111111";

      an <= "1110" when a = "00" else
                  "1101" when a = "01" else
                  "1011" when a = "10" else
                  "0111" when a = "11" else
                  "1111";

end Behavioral;
```

hexcounter.vhd

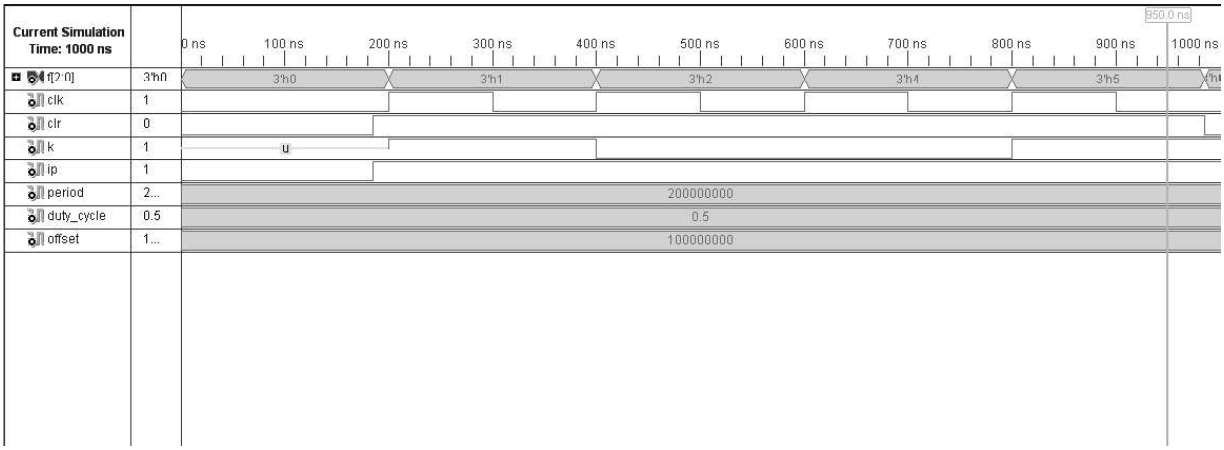
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity hexcounter is
      Port (
            clk_50MHz: in STD_LOGIC;
            an: out STD_LOGIC_VECTOR(3 downto 0);
            seg: out STD_LOGIC_VECTOR(6 downto 0)
      );
end hexcounter;

architecture Behavioral of hexcounter is
component counter  is
```

```
      Port (
            clk: in STD_LOGIC;
            count: out STD_LOGIC_VECTOR (3 downto 0)
      );
end component;
component sevensegment is
      port (     a: in std_logic_vector (1 downto 0);
                       b: in std_logic_vector (3 downto 0);
                       an: out std_logic_vector (3 downto 0);
                       seg: out std_logic_vector (6 downto 0)
                 );
end component;
signal s: STD_LOGIC_VECTOR(3 downto 0);
begin
      c1: counter port map (clk=>clk_50MHz, count=>s);
      l1: sevensegment port map (a=>"00", b=>s, seg=>seg, an=>an);

end Behavioral;
```

STATE MACHINE

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

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity FSM is
port(F : out STD_LOGIC_VECTOR (2 downto 0);
       clk, clr: in STD_LOGIC;
     k     : out STD_LOGIC ;
       ip    : in STD_LOGIC);
end FSM;

architecture Behavioral of FSM is

signal count: integer;
type state_type is (idle, s1,s2,s4,s5,s7);
signal state:state_type;
begin
process(clk,clr)

begin

if (clr = '0') then state <= idle;

elsif rising_edge(clk) then
count <= count +1;
if(count = 50) then
count <= 0;
if(state = idle and ip = '1') then
      state <= s1;
      k <= '1';

elsif(state = idle and ip = '0') then
      NULL;

elsif(state = s1 and ip = '1') then
      state <= s2;
      k <= '0';


elsif(state = s1 and ip = '0') then
      NULL;

elsif(state = s2 and ip = '1') then
```

```
      state <= s4;
      k <= '0';

elsif(state = s2 and ip = '0') then
      NULL;

elsif(state = s4 and ip = '1') then
      state <= s5;
      k <= '1';

elsif(state = s4 and ip = '0') then
      NULL;

elsif(state = s5 and ip = '1') then
      state <= s7;
      k <= '1';

elsif(state = s5 and ip = '0') then
      NULL;

elsif(state = s7 and ip = '1') then
      state <= s1;
      k <= '1';

elsif(state = s7 and ip = '0') then
      NULL;

else
      NULL;

end if;
end if;
end if;

end process;
F <=  "000" WHEN state = idle else
            "001" when state = s1 else
            "010" when state = s2 else
            "100" when state = s4 else
            "101" when state = s5 else
            "111" when state = s7 else
            "000";

end Behavioral;
```


Current Simulation Time: 1000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
1000 ns
950.0 ns
i[2:0]
3'h0
3'h0
3'h1
3'h2
3'h4
3'h5
clk
clr
k
ip
period
200000000
duty_cycle
0.5
offset
100000000