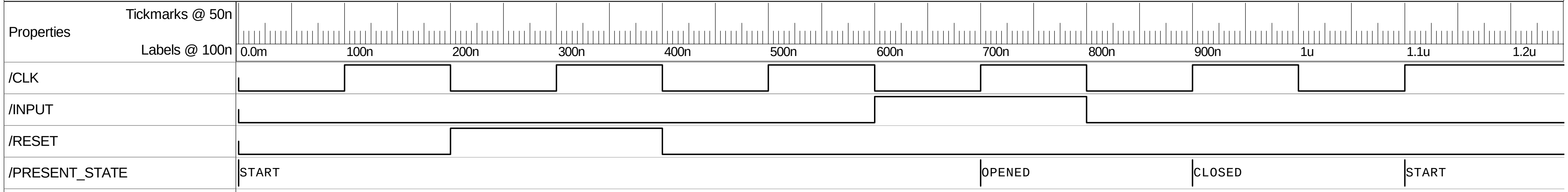
Practical VHDL Samples 2

The following is a list of files used as examples in the ESD3 lectures.
The files are included overleaf with simulations and post-synthesis schematics where applicable.

Source Name	**Entity Name**	**Description**	**Synthesisable?**	**Comments**
ssmach.vhd	ssmach	Simple State Machine		Implements a basic state machine with no outputs. Illustrates the basic structure of a state machine in VHDL.
ssmach2.vhd	ssmach2	State Machine With Outputs	✓	Implements a simple state machine with outputs. Illustrates outputs being a function of present state etc.
upcount.vhd	upcounter	Simple counter.	✓	Demonstrates the use of the numeric_std package for arithmetic. Also demonstrates the use of conversion routines present in numeric_std for conversion to std_logic.
rom.vhd	rom	ROM	✓	Implements a basic 16 address by 5 bit ROM. Illustrates the creation of memory structures and also arrays in VHDL.
testalu..vhd	testbench_alu	Structural schematic.		Illustrates the use of structural VHDL for schematic designs.

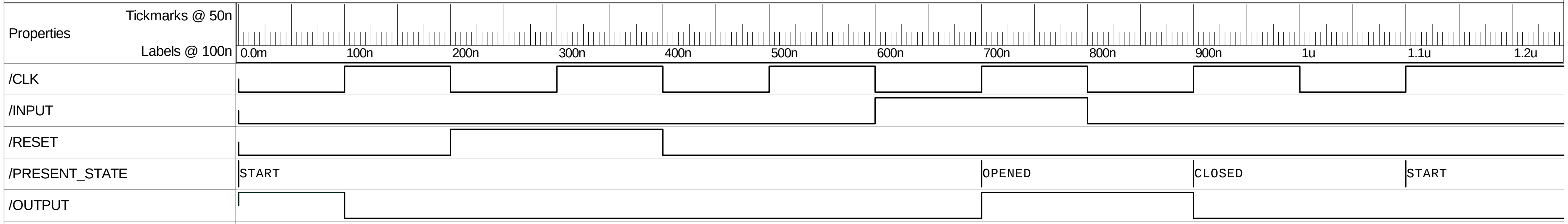
```
-------------------------------------------------------------------------------
-- Title         : Simple State Machine Implementation in VHDL
-- Project       : Digital Design IV
-------------------------------------------------------------------------------
-- File          : ssmach.vhd
-- Author        :   <Craig Slorach@HANDEL>
-- Created       : 1999/02/17
-- Last modified : 1999/02/17
-------------------------------------------------------------------------------
-- Description :
-- Implements a simple state machine with no outputs in VHDL
-------------------------------------------------------------------------------
-- Modification history :
-- 1999/02/17 : created
-------------------------------------------------------------------------------

-- NOTE THIS DESIGN IS FOR DEMONSTRATION/ SIMULATION  PURPOSES ONLY- IT CANNOT
-- BE SYNTHESISED SINCE THERE ARE NO SYSTEM OUTPUTS AND AS SUCH THE
-- OPTIMISATION STEP IN SYNTHESIS WILL YIELD NO CIRCUIT !


library ieee;
use ieee.std_logic_1164.all;

entity SSMACH is

    port (input : in std_logic;         -- system input to start the system
          clk,reset : in std_logic);    -- clock and reset inputs

end SSMACH;

-- purpose: Implement main architecture for SSMACH
architecture BEHAVIOR of SSMACH is

    type STATES is (START, OPENED, CLOSED);  -- possible states
    signal PRESENT_STATE : STATES;      -- present state

begin  -- BEHAVIOR

    -- purpose: Main process
    process (clk, reset)

    begin  -- process

        -- activities triggered by asynchronous reset (active high)
        if reset = '1' then

            PRESENT_STATE <= START;     -- default state

        -- activities triggered by rising edge of clock
        elsif clk'event and clk = '1' then

            case PRESENT_STATE is

                when START =>
                    if INPUT='1' then
                        PRESENT_STATE <= OPENED;
                    else
                        PRESENT_STATE <= START;
                    end if;

                when OPENED =>
                    PRESENT_STATE <= CLOSED;

                when CLOSED =>
                    PRESENT_STATE <= START;

                when others =>
                    PRESENT_STATE <= START;

            end case;
        end if;
    end process;
end BEHAVIOR;
```

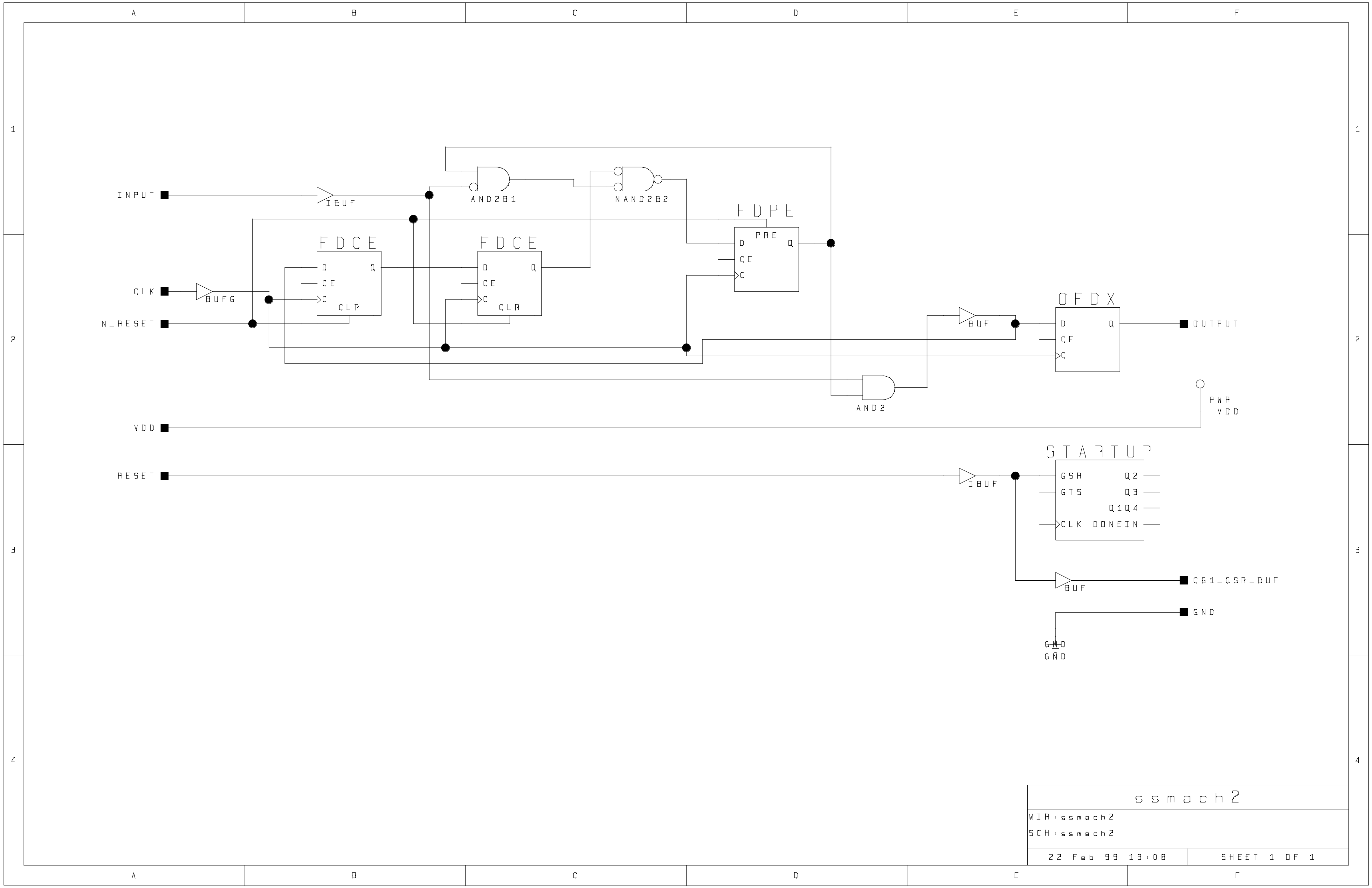
Plot of SSMACH.VHD



```
-------------------------------------------------------------------------------
-- Title         : Simple State Machine Implementation (with outputs) in VHDL
-- Project       : Digital Design IV
-------------------------------------------------------------------------------
-- File          : ssmach2.vhd
-- Author        :   <Craig Slorach@HANDEL>
-- Created       : 1999/02/17
-- Last modified : 1999/02/17
-------------------------------------------------------------------------------
-- Description :
-- Implements a simple state machine with outputs in VHDL
-------------------------------------------------------------------------------
-- Modification history :
-- 1999/02/17 : created
-- 2000/01/13 : Updated for default output behaviour
-------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity SSMACH2 is

  port (input      : in  std_logic;     -- system input to start the system
        output     : out std_logic;     -- system output
        clk, reset : in  std_logic);    -- clock and reset inputs

end SSMACH2;

-- purpose: Implement main architecture for SSMACH2
architecture BEHAVIOR of SSMACH2 is

  type STATES is (START, OPENED, CLOSED);  -- possible states
  signal PRESENT_STATE : STATES;           -- present state

begin  -- BEHAVIOR

  -- purpose: Main process
  process (clk, reset)
  begin  -- process

    -- activities triggered by asynchronous reset (active high)
    if reset = '1' then
      PRESENT_STATE <= START;           -- default state
      OUTPUT        <= '0';

      -- activities triggered by rising edge of clock
    elsif clk'event and clk = '1' then

      OUTPUT        <= '0';             -- default output
      PRESENT_STATE <= OPENED;          -- default state

      case PRESENT_STATE is

        when START =>
          if INPUT = '1' then
            PRESENT_STATE <= OPENED;
            OUTPUT        <= '1';
          else
            PRESENT_STATE <= START;
            OUTPUT        <= '0';
          end if;

        when OPENED =>
          PRESENT_STATE <= CLOSED;
          OUTPUT        <= '0';

        when CLOSED =>
          PRESENT_STATE <= START;
          OUTPUT        <= '0';

        when others =>
          PRESENT_STATE <= START;
          OUTPUT        <= '0';

      end case;
    end if;
  end process;
end BEHAVIOR;
```

Plot of SSMACH2.VHD





ssmach2

WIR:ssmach2

SCH:ssmach2

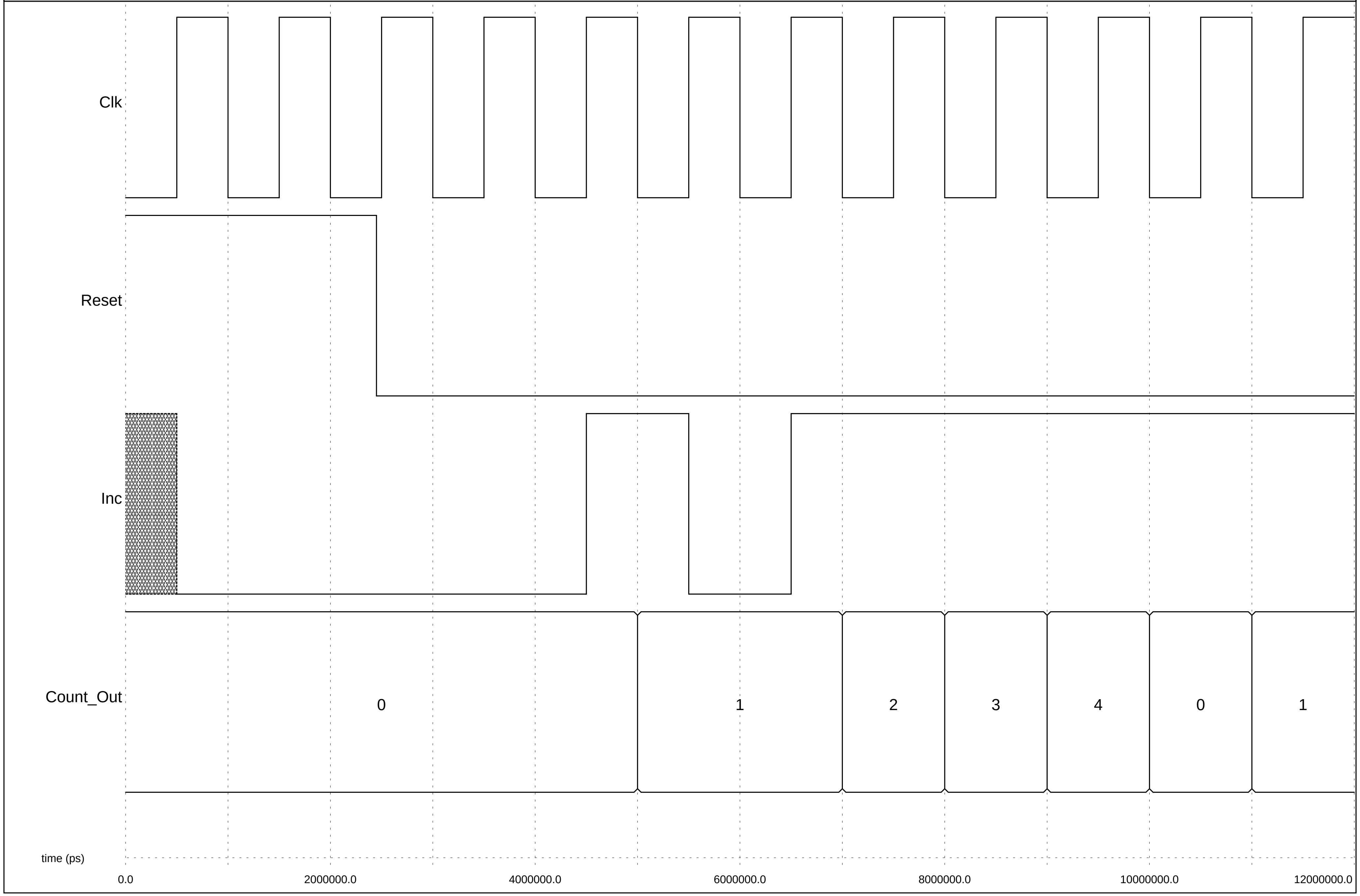
22 Feb 99 18:08

SHEET 1 OF 1

```
-------------------------------------------------------------------------------
-- Title      : Simple counter in VHDL
-- Project    : Digital Design IV
-------------------------------------------------------------------------------
-- File       : upcounter.vhd
-- Author     :   <Craig Slorach@HANDEL>
-- Company    :
-- Last update: 2000/01/13
-- Platform   :
-------------------------------------------------------------------------------
-- Description: Implements a simple counter in VHDL which counts up only
--              when an input 'inc' = '1'
-------------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2000/01/13  1.0      Craig Slorach   Created
-------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity UPCOUNTER is

  port (
    inc        : in  std_logic;                      -- increment input
    count      : out std_logic_vector (3 downto 0);  -- output
    clk, reset : in  std_logic);                     -- clock and reset

end UPCOUNTER;

architecture BEHAVIOR of UPCOUNTER is

  signal internal_count : unsigned (3 downto 0);  -- internal counter

begin  -- BEHAVIOR

  -- purpose: Main process
  -- type   : sequential
  -- inputs : clk, reset, inc
  -- outputs: internal_count
  process (clk, reset)
  begin  -- process
    if reset = '1' then                 -- asynchronous reset (active high)

      internal_count <= "0000";

    elsif clk'event and clk = '1' then  -- rising clock edge

      if inc = '1' then

        if internal_count = "0100" then
          internal_count <= "0000";     -- reset back

        else
          internal_count <= internal_count + 1;  -- increment

        end if;

      else
        null;

      end if;

    end if;

    -- drive the counter output with the internal value
    count <= std_logic_vector(internal_count);

  end process;


end BEHAVIOR;
```

SimWave 3.17-E

Fri Jan 14 08:21:41 2000



```
-------------------------------------------------------------------------------
-- Title      : ROM Implementation in VHDl
-- Project    : Digital Design IV
-------------------------------------------------------------------------------
-- File       : rom.vhd
-- Author     :   <Craig Slorach@HANDEL>
-- Company    :
-- Last update: 2000/01/13
-- Platform   :
-------------------------------------------------------------------------------
-- Description: Implements a 16 x 5 ROM in VHDL with a combinational logic
--              approach.
-------------------------------------------------------------------------------
-- Revisions  :
-- Date        Version  Author  Description
-- 2000/01/13  1.0      Craig Slorach   Created
-------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ROM is

  port (
    ADDR : in  std_logic_vector (3 downto 0);   -- address input
    DOUT : out std_logic_vector (4 downto 0));  -- data output

end ROM;

architecture BEHAVIOR of ROM is

  type ROMTABLE is array (0 to 15) of std_logic_vector (4 downto 0);
                                        -- internal table

  constant romdata : romtable := (

    "10101",                            -- data for address 0
    "11111",                            -- data for address 1
    "10101",
    "11110",
    "10110",
    "10101",
    "11010",
    "10010",
    "10110",
    "11101",
    "10110",
    "10111",
    "00110",
    "11101",
    "10010",
    "10110"                             -- data for address 15
    );

begin  -- BEHAVIOR

  -- purpose: Main process
  -- type   : combinational
  -- inputs : ADDR
  -- outputs: DOUT
  process (ADDR)
  begin  -- process

    DOUT <= romdata(to_integer(unsigned(ADDR)));

  end process;

end BEHAVIOR;
```

```
--Netlist:
--Library=dd49900lab1,Cell=testbench_alu,View=schematic
--Time:Thu Dec  9 11:39:48 1999
--By:craigs

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ARCHITECTURE schematic OF testbench_alu IS
    COMPONENT alu
        PORT(
            carryout : OUT std_logic;
            result : OUT std_logic_vector(3 DOWNTO 0);
            a : IN std_logic_vector(3 DOWNTO 0);
            b : IN std_logic_vector(3 DOWNTO 0);
            clk : IN std_logic;
            inst : IN std_logic_vector(1 DOWNTO 0);
            rst_n : IN std_logic
        );
    END COMPONENT;

    COMPONENT stim_alu
        PORT(
            stim_a : OUT std_logic_vector(3 DOWNTO 0);
            stim_b : OUT std_logic_vector(3 DOWNTO 0);
            stim_clock : OUT std_logic;
            stim_inst : OUT std_logic_vector(1 DOWNTO 0);
            stim_reset : OUT std_logic
        );
    END COMPONENT;

    SIGNAL reset : std_logic;
    SIGNAL clock : std_logic;
    SIGNAL inst : std_logic_vector(1 DOWNTO 0);
    SIGNAL a : std_logic_vector(3 DOWNTO 0);
    SIGNAL b : std_logic_vector(3 DOWNTO 0);

    FOR ALL: alu USE ENTITY dd4lab1.alu(behavior);
    FOR ALL: stim_alu USE ENTITY dd4lab1.stim_alu(behavior);


BEGIN


    \I2\ : alu
        PORT MAP(
            carryout => alucarryout,
            result(3 DOWNTO 0) => aluout(3 DOWNTO 0),
            a(3 DOWNTO 0) => a(3 DOWNTO 0),
            b(3 DOWNTO 0) => b(3 DOWNTO 0),
            clk => clock,
            inst(1 DOWNTO 0) => inst(1 DOWNTO 0),
            rst_n => reset
        );

    \I1\ : stim_alu
        PORT MAP(
            stim_a(3 DOWNTO 0) => a(3 DOWNTO 0),
            stim_b(3 DOWNTO 0) => b(3 DOWNTO 0),
            stim_clock => clock,
            stim_inst(1 DOWNTO 0) => inst(1 DOWNTO 0),
            stim_reset => reset
        );

END schematic;
```