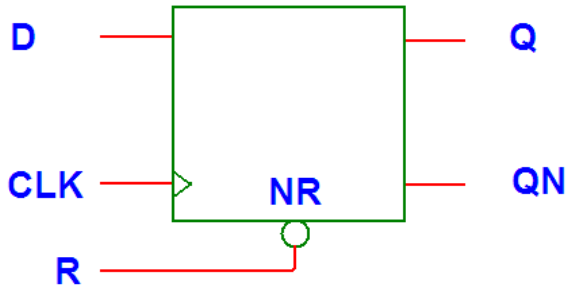
Data or Delay (D-Type) flip-flop

Data or Delay (***D***-Type) flip-flop have a single input line (***D***), instead of two input lines as the case of ***RS*** and ***JK*** flip-flops. The ***D***-Type flip-flop is shown below.



In ***D***-Type of flip-flop whatever input is applied to ***D*** will be transferred to the ***Q*** output as soon as the clock goes high, provided that the reset is not active, (*i.e reset is low = 0*). The ***Q*** output will be immediately reset to ***0*** irrespective of the low or high data in ***D***-input if reset is active (*i.e reset is high = 1*). The truth table for ***D***-Type of flip-flop is given below.

Reset (**R**)	Not Reset (**NR**)	Data (**D**)	Clock (**CLK**)	Output (**Q**)	Not Output (**NQ**)
1	**0**	**X**	**X**	**0**	**1**
0	**1**	**0**	(**CLK** = '1' and **CLK**'event)	**0**	**1**
0	**1**	**1**	(**CLK** = '1' and **CLK**'event)	**1**	**0**

VHDL CODE

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

-- Hendra Kesuma;

entity D_FF_ent is
    Port ( D : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           RN : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           QN : out  STD_LOGIC);
end D_FF_ent;
```

```
architecture D_FF_arch of D_FF_ent is
begin
  process(CLK, RN)
  begin
      if RN = '0' then
        Q <= '0';
        QN <= '1';
     elsif (RN = '1' and CLK = '1' and clk'event) then
        Q <= D;
        QN <= not D;
     end if;
  end process;
end D_FF_arch;
```

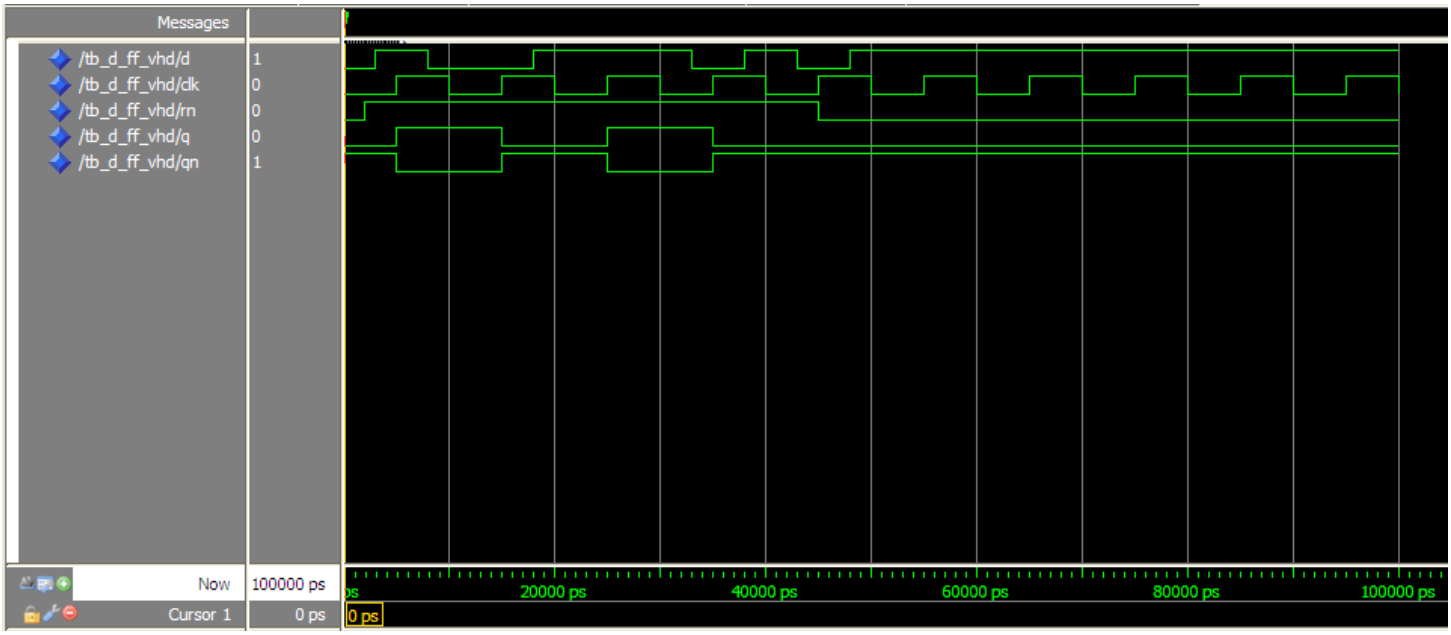
TESTBENCH CODE

TB_D_FF.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

-- Hendra Kesuma

ENTITY TB_D_FF_vhd IS
END TB_D_FF_vhd;

ARCHITECTURE behavior OF TB_D_FF_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT D_FF_ent
        PORT(
                D : IN std_logic;
                CLK : IN std_logic;
                RN : IN std_logic;
                Q : OUT std_logic;
                QN : OUT std_logic
                );
        END COMPONENT;

        --Inputs
        SIGNAL D :  std_logic := '0';
        SIGNAL CLK :  std_logic := '0';
        SIGNAL RN :  std_logic := '0';

        --Outputs
        SIGNAL Q :  std_logic;
```

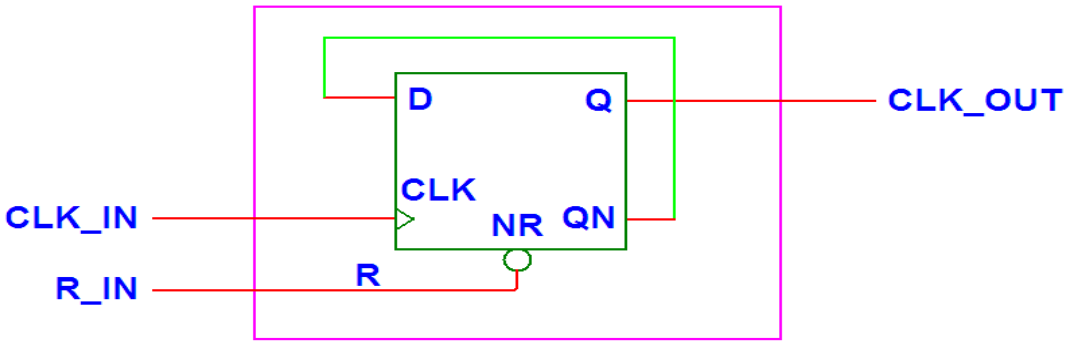
```
        SIGNAL QN :  std_logic;

BEGIN
        -- Instantiate the Unit Under Test (UUT)
           uut: D_FF_ent
           PORT MAP(
                 D => D,
                 CLK => CLK,
                 RN => RN,
                 Q => Q,
                 QN => QN
            );

        tb1 : PROCESS
        BEGIN
                 -- Wait 100 ns for global reset to finish
                 -- wait for 100 ns;

                 -- Place stimulus here
                 D  <= '0' after 0 ns,
                         '1' after 3 ns,  '0' after 8 ns,
                         '1' after 18 ns, '0' after 33 ns,
                         '1' after 38 ns, '0' after 43 ns,
                         '1' after 48 ns;

                  RN <= '0' after 0 ns, '1' after 2 ns, '0' after 45 ns;

                 wait; -- will wait forever

        END PROCESS;

        tb2 : PROCESS
        BEGIN

             loop
             CLK <= '0';
             wait for 5 ns;
             CLK <= '1';
             wait for 5 ns;
             end loop;

              wait; -- will wait forever

        END PROCESS;

END;
```

SIMULATION



Frequency Divider with (D-Type) flip-flip

Data or Delay (***D***-Type) can be used to divide the clock input frequency by two, by connecting the ***QN*** output to the ***D*** input as shown in the figure below..



VHDL CODE

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

-- Hendra Kesuma

entity CLK_DIV_ent is
    Port ( CLK_IN : in  STD_LOGIC;
           RN_IN : in  STD_LOGIC;
           CLK_OUT : out  STD_LOGIC);
end CLK_DIV_ent;

architecture CLK_DIV_arch of CLK_DIV_ent is

  component D_FF_ent
       Port ( D : in  STD_LOGIC;
              CLK : in  STD_LOGIC;
              RN : in  STD_LOGIC;
              Q : out  STD_LOGIC;
              QN : out  STD_LOGIC);
  end component;

  signal Wire1 : std_logic;

begin

      CLK_DIV: D_FF_ent
      port map(
            D => Wire1,
            CLK => CLK_IN,
            RN => RN_IN,
            Q => CLK_OUT,
            QN => Wire1 );

end CLK_DIV_arch;
```

TESTBENCH CODE

```
================================================================================
TB_CLK_DIV.vhd
================================================================================
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

-- Hendra Kesuma

ENTITY TB_CLK_DIV_vhd IS
END TB_CLK_DIV_vhd;

ARCHITECTURE behavior OF TB_CLK_DIV_vhd IS

	-- Component Declaration for the Unit Under Test (UUT)
	COMPONENT CLK_DIV_ent
	PORT(
		CLK_IN : IN std_logic;
		RN_IN : IN std_logic;
		CLK_OUT : OUT std_logic
		);
	END COMPONENT;

	--Inputs
	SIGNAL CLK_IN :  std_logic := '0';
	SIGNAL RN_IN :  std_logic := '0';

	--Outputs
	SIGNAL CLK_OUT :  std_logic;

BEGIN
	-- Instantiate the Unit Under Test (UUT)
	uut: CLK_DIV_ent PORT MAP(
		CLK_IN => CLK_IN,
		RN_IN => RN_IN,
		CLK_OUT => CLK_OUT
	);

	tb1 : PROCESS
	BEGIN
		-- Wait 100 ns for global reset to finish
		-- wait for 100 ns;

		-- Place stimulus here
		RN_IN  <= '0' after 0 ns, '1' after 2 ns, '0' after 60 ns;

	 wait; -- will wait forever
```

```
        END PROCESS;

        tb2 : PROCESS
        BEGIN

            loop
            CLK_IN <= '0';
            wait for 5 ns;
            CLK_IN <= '1';
            wait for 5 ns;
            end loop;

            wait; -- will wait forever

        END PROCESS;
END;
```

SIMULATION