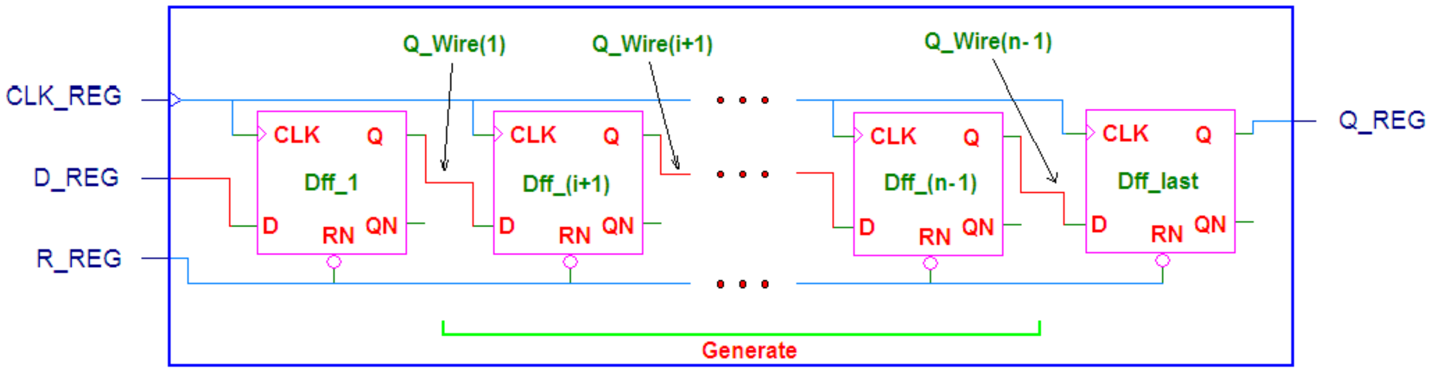
n-stage Shift Register with „Generate“ Keyword

With the increase of number of components of same type, the component instantiation becomes more complex and lengthy. For example the circuit shown below represents the n-stage shift register. As the sequential depth of the shift register increases the component instantiations of ***D*** flip-flop become more complex and lengthy. For this type of problems there exists a keyword „***Generate***“ in the VHDL syntax.



n-stage shift register

VHDL CODE

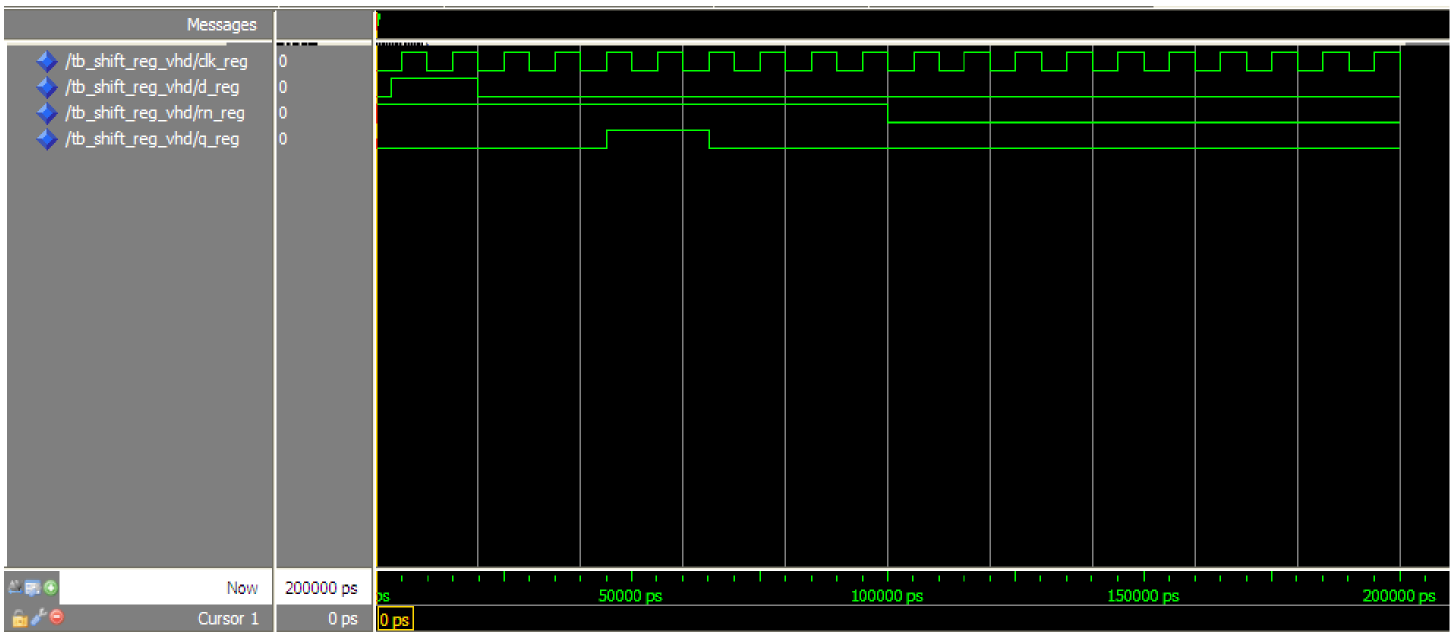
```
==============================================================================
SHIFT_REG.vhdl
==============================================================================
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Hendra Kesuma

entity SHIFT_REG_ent is
    Port ( CLK_REG : in  STD_LOGIC;
           D_REG : in  STD_LOGIC;
           RN_REG : in  STD_LOGIC;
           Q_REG : out  STD_LOGIC);
end SHIFT_REG_ent;

architecture SHIFT_REG_arch of SHIFT_REG_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;
  constant n: integer := 5;
  signal Q_Wire : std_logic_vector(1 to n-1);

  begin
    Dff_first: D_FF_ent
    port map
       ( D => D_REG,
        CLK => CLK_REG,
        RN => RN_REG,
        Q => Q_Wire(1));

    Dff_gen: for i in 1 to n-2 GENERATE
    Dff_i: D_FF_ent
    port map
       (D => Q_Wire(i),
        CLK => CLK_REG,
        RN => RN_REG,
        Q => Q_Wire(i+1));
    end GENERATE;

    Dff_last : D_FF_ent
    port map
       ( D => Q_Wire(n-1),
        CLK => CLK_REG,
         RN => RN_REG,
        Q => Q_REG);

end SHIFT_REG_arch;
```

TESTBENCH CODE

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

-- Hendra Kesuma

ENTITY TB_SHIFT_REG_vhd IS
END TB_SHIFT_REG_vhd;

ARCHITECTURE behavior OF TB_SHIFT_REG_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT SHIFT_REG_ent
        PORT(
                CLK_REG : IN std_logic;
```

```vhdl
			D_REG : IN std_logic;
			RN_REG : IN std_logic;
			Q_REG : OUT std_logic
			);
	END COMPONENT;

	--Inputs
	SIGNAL CLK_REG :  std_logic := '0';
	SIGNAL D_REG :  std_logic := '0';
	SIGNAL RN_REG :  std_logic := '0';

	--Outputs
	SIGNAL Q_REG :  std_logic;

BEGIN

	-- Instantiate the Unit Under Test (UUT)
	uut: SHIFT_REG_ent PORT MAP(
		CLK_REG => CLK_REG,
		D_REG => D_REG,
		RN_REG => RN_REG,
		Q_REG => Q_REG
	);

	tb : PROCESS
	BEGIN

			-- Wait 100 ns for global reset to finish
			--wait for 100 ns;

			-- Place stimulus here
			RN_REG  <= '1' after 0 ns, '0' after 100 ns;
			D_REG     <= '0' after 0 ns, '1' after  3 ns,
			                   '0' after 20 ns;
			loop
			CLK_REG <= '0';
			wait for 5 ns;
			CLK_REG <= '1';
			wait for 5 ns;
			end loop;

			wait; -- will wait forever
	END PROCESS;

END;
```

SIMULATION



Messages

/tb_shift_reg_vhd/clk_reg 0
/tb_shift_reg_vhd/d_reg 0
/tb_shift_reg_vhd/rn_reg 0
/tb_shift_reg_vhd/q_reg 0

Now 200000 ps
Cursor 1 0 ps