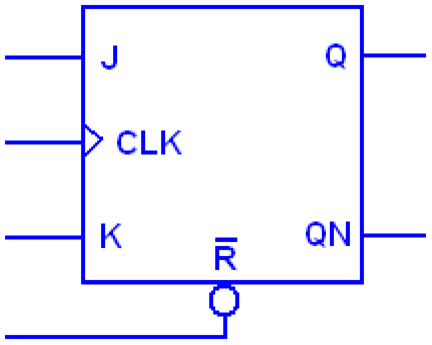
JK flip-flop

The **Fig.-1** below shows the block diagram and truth table for **JK flip-flop**. In the truth table the reset and clock options are not depicted. But as usual when reset is active the output **Q** should be set to '**0**' and **QN** to '**1**'. The output changes will take place only at the rising edge of **CLK** signal. Using **JK flip-flops**, one can build a counter. A **4-bit counter** using the **JK flip-flops** is shown in **Fig.-2**.



JK Flip Flop operation								
Characteristic table				Excitation table				
J	K	Q_{next}	Comment	Q	Q_{next}	J	K	Comment
0	0	Q_{prev}	hold state	0	0	0	X	No change
0	1	0	reset	0	1	1	X	Set
1	0	1	set	1	0	X	1	Reset
1	1	$\overline{Q}_{prev}$	toggle	1	1	X	0	No change

Fig.-1: JK flip-flop

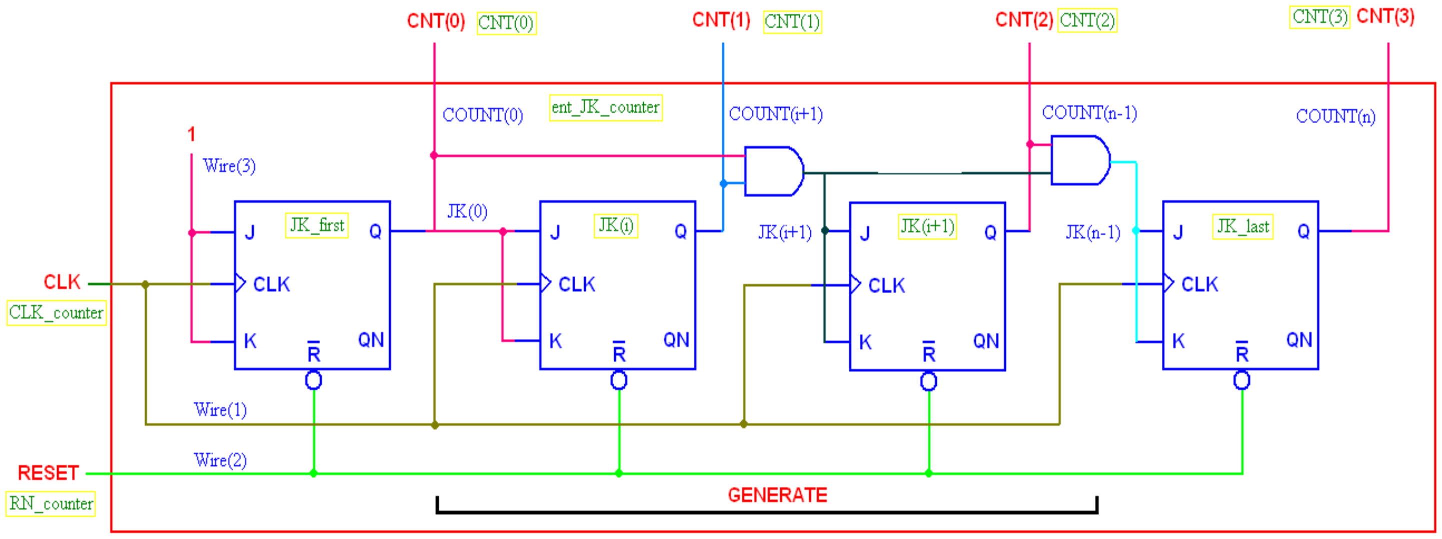


Fig. -2 4-bit counter using JK flip-flop

VHDL CODE

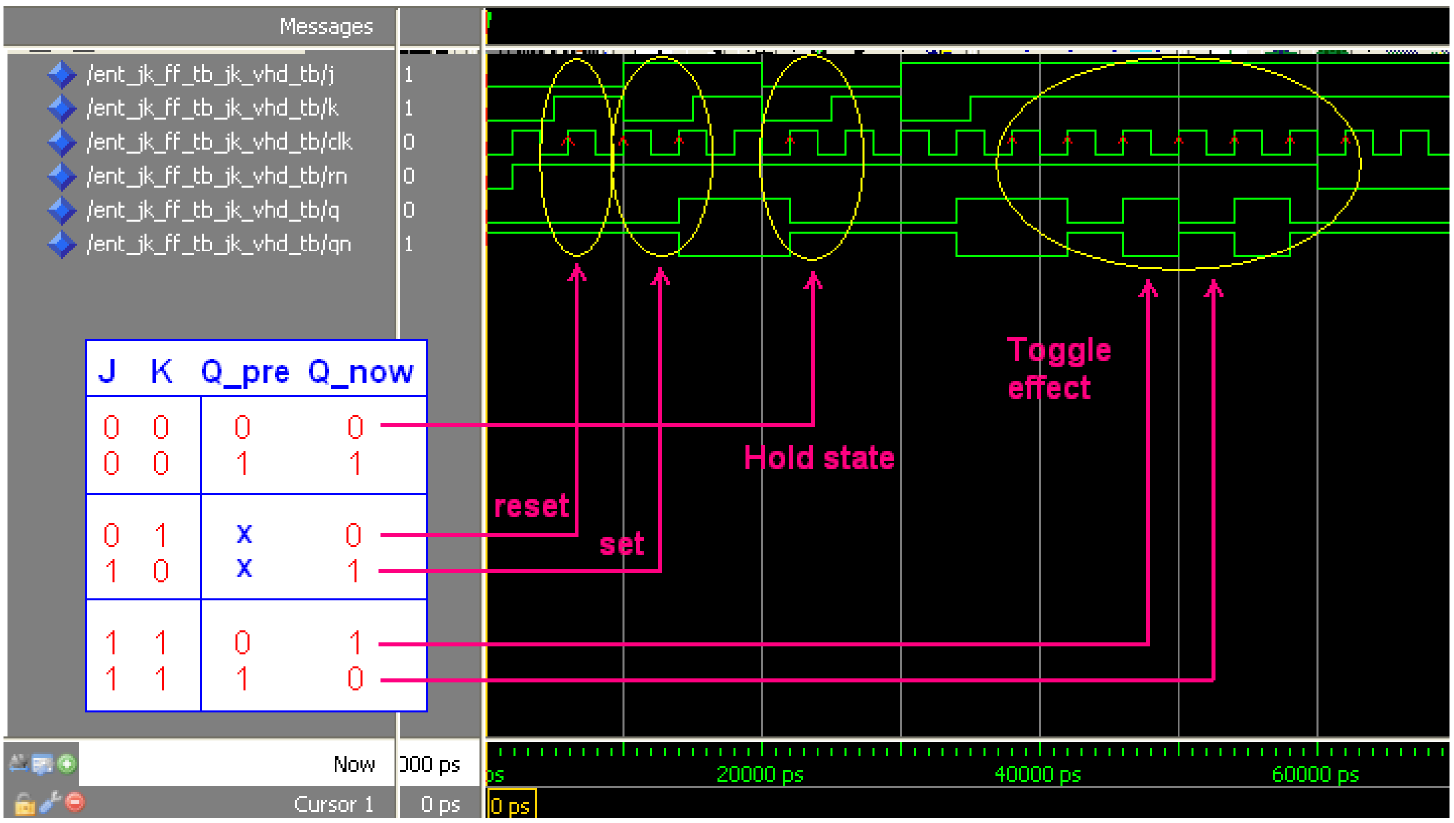
```
================================================================================
JK_FF.vhdl
================================================================================
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Hendra Kesuma

entity ent_JK_FF is
   Port ( J :   in std_logic;
          K :   in std_logic;
         CLK: in std_logic;
         RN :   in std_logic;
          Q :   out std_logic;
        QN :    out std_logic);
end ent_JK_FF;

architecture arch_JK_FF of ent_JK_FF is
signal FF : std_logic  := '0';

begin
process(CLK, RN)
variable Temp_variable : std_logic_vector (1 downto 0);
  begin
   if RN = '0' then
           Q     <= '0';
         QN     <= '1';
   elsif (RN = '1' and CLK = '1' and clk'event) then
       Temp_variable := (J & K);
       case Temp_variable is
         when "00" => FF <= FF;
         when "01" => FF <= '0';
         when "10" => FF <= '1';
         when "11" => FF <= not FF;
         when others => FF <= 'U';
      end case;
         Q <= FF;
         QN <= not FF;
   end if;
end process;

end arch_JK_FF;
```

TESTBENCH CODE

```
==============================================================================
tb_JK.vhd
==============================================================================
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
use work.comp_bit.ALL; -- Include package

-- Hendra Kesuma

ENTITY ent_jk_ff_tb_JK_vhd_tb IS
END ent_jk_ff_tb_JK_vhd_tb;

ARCHITECTURE behavior OF ent_jk_ff_tb_JK_vhd_tb IS

	COMPONENT ent_jk_ff
	PORT(
		J : IN std_logic;
		K : IN std_logic;
		CLK : IN std_logic;
		RN : IN std_logic;
		Q : OUT std_logic;
		QN : OUT std_logic
		);
	END COMPONENT;

	SIGNAL J :  std_logic;
	SIGNAL K :  std_logic;
	SIGNAL CLK :  std_logic;
	SIGNAL RN :  std_logic;
	SIGNAL Q :  std_logic;
	SIGNAL QN :  std_logic;

BEGIN

	uut: ent_jk_ff PORT MAP(
		J => J,
		K => K,
		CLK => CLK,
		RN => RN,
		Q => Q,
		QN => QN
	);
```

```
-- *** Test Bench - User Defined Section ***
   tb1 : PROCESS
   BEGIN
     RN <= '0' after 0 ns, '1' after 2 ns, '0' after 60 ns;
     ApplyData_Procedure(J, "00110011", 5 ns);
     ApplyData_Procedure(K, "01010101", 5 ns);

     wait; -- will wait forever
   END PROCESS;

   tb2 : PROCESS
   BEGIN

   loop
     CLK <= '0';
   wait for 2 ns;
     CLK <= '1';
   wait for 2 ns;
   end loop;

     wait; -- will wait forever
   END PROCESS;

-- *** End Test Bench - User Defined Section ***

END;
```

SIMULATION


Messages
/ent_jk_ff_tb_jk_vhd_tb/j 1
/ent_jk_ff_tb_jk_vhd_tb/k 1
/ent_jk_ff_tb_jk_vhd_tb/clk 0
/ent_jk_ff_tb_jk_vhd_tb/rn 0
/ent_jk_ff_tb_jk_vhd_tb/q 0
/ent_jk_ff_tb_jk_vhd_tb/qn 1
Toggle effect
Hold state
reset
set
Now 000 ps
Cursor 1 0 ps
ps
20000 ps
40000 ps
60000 ps
0 ps

J	K	Q_pre	Q_now
0	0	0	0
0	0	1	1
0	1	x	0
1	0	x	1
1	1	0	1
1	1	1	0

VHDL CODE

```
===============================================================================
JK_counter.vhdl
===============================================================================

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

-- Hendra Kesuma;

entity ent_JK_counter is
   Port ( CLK_counter : in std_logic;
          RN_counter   : in std_logic;
          CNT          : out std_logic_vector(3 downto 0));
end ent_JK_counter;

architecture arch_JK_counter of ent_JK_counter is

component ent_JK_FF
   Port ( J : in std_logic;
          K : in std_logic;
        CLK : in std_logic;
```

```
          RN   : in std_logic;
          Q     : out std_logic;
          QN   : out std_logic);
end component;

constant n: integer := 3 ;
signal Wire       : std_logic_vector(1 to 3 );
signal COUNT  : std_logic_vector(0 to n );
signal JK          : std_logic_vector(0 to n );

begin
JK_first: ent_JK_FF
port map ( J      =>   Wire(3),
                K    =>   Wire(3),
              CLK  =>   Wire(1),
               RN   =>   Wire(2),
                Q    =>  COUNT(0));

JK_gen: for i in 0 to n-2 GENERATE
JK_i: ent_JK_FF
port map ( J      =>  JK(i),
               K      =>   JK(i),
              CLK  =>   Wire(1),
               RN   =>   Wire(2),
                Q    =>   COUNT(i+1));
end GENERATE;

JK_last: ent_JK_FF
port map ( J      =>   JK(n-1),
               K      =>   JK(n-1),
               CLK =>   Wire(1),
               RN   =>   Wire(2),
                 Q   =>   COUNT(n));

Wire(1) <= CLK_counter;
Wire(2) <= RN_counter;

Wire(3) <= '1';
JK(0) <= COUNT(0);

JK(1) <= (COUNT(0) and COUNT(1));

JK(2) <= (JK(1) and COUNT(2));

--======  or ==========
-- for i in 1 to n
--    JK(i+1) <= (JK(i) and COUNT(i+1));
-- end loop
```

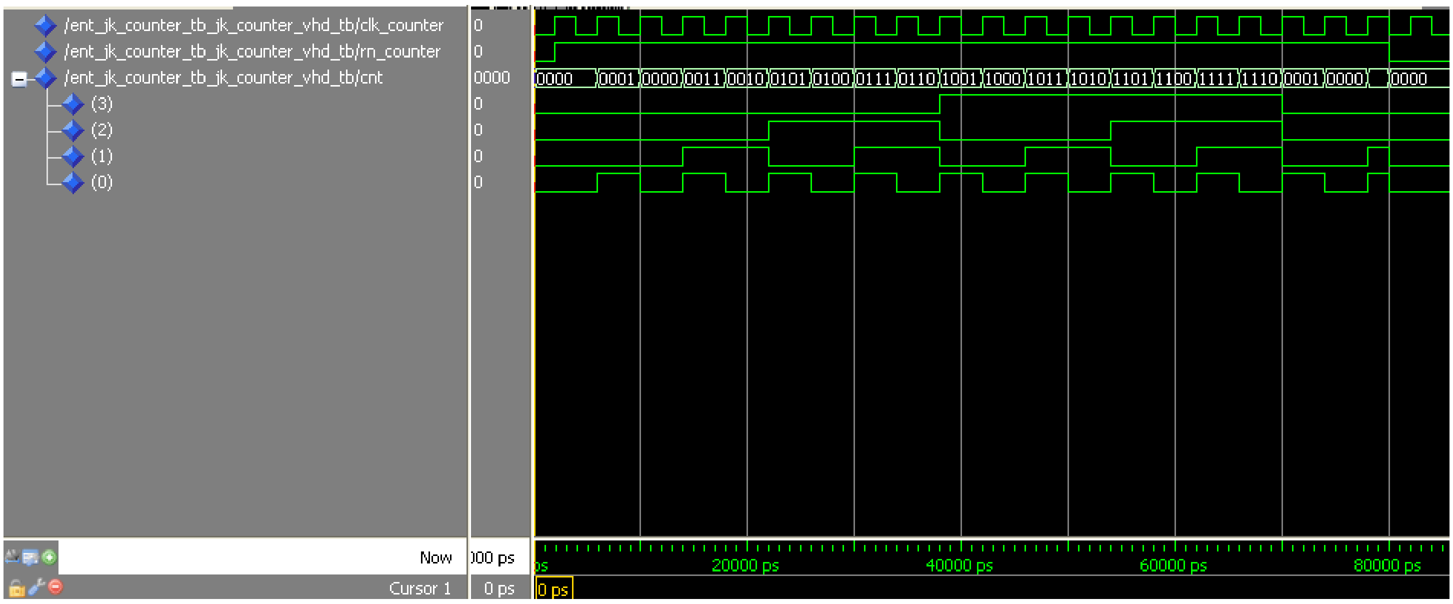
```
CNT(0) <= COUNT(0);
CNT(1) <= COUNT(1);
CNT(2) <= COUNT(2);
CNT(3) <= COUNT(3);

-- ======  or ==========
--   for i in 0 to n
--      CNT(i) <= COUNT(i);
--   end loop

end arch_JK_counter;
```

TESTBENCH CODE

```
==================================================================================
JK_counter.vhdl
==================================================================================

lLIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

-- Hendra Kesuma;

ENTITY ent_jk_counter_tb_JK_counter_vhd_tb IS
END ent_jk_counter_tb_JK_counter_vhd_tb;

ARCHITECTURE behavior OF ent_jk_counter_tb_JK_counter_vhd_tb IS

        COMPONENT ent_jk_counter
        PORT(
                CLK_counter  : IN std_logic;
                RN_counter   : IN std_logic;
                CNT          : OUT std_logic_vector(3 downto 0)
                );
        END COMPONENT;

        SIGNAL CLK_counter  :  std_logic;
        SIGNAL RN_counter   :  std_logic;
        SIGNAL CNT          :  std_logic_vector(3 downto 0);

BEGIN

        uut: ent_jk_counter PORT MAP(
                CLK_counter  => CLK_counter,
                RN_counter   => RN_counter,
                CNT          => CNT
        );
```

```
-- *** Test Bench - User Defined Section ***
   tb1 : PROCESS
   BEGIN
     RN_counter <= '0' after 0 ns, '1' after 2 ns, '0' after 80 ns;
     wait; -- will wait forever
   END PROCESS;

   tb2 : PROCESS
   BEGIN

   loop
     CLK_counter <= '0';
   wait for 2 ns;
     CLK_counter <= '1';
   wait for 2 ns;
   end loop;

     wait; -- will wait forever
   END PROCESS;

-- *** End Test Bench - User Defined Section ***

END;
```

SIMULATION



```
/ent_jk_counter_tb_jk_counter_vhd_tb/clk_counter   0
/ent_jk_counter_tb_jk_counter_vhd_tb/rn_counter    0
/ent_jk_counter_tb_jk_counter_vhd_tb/cnt           0000   0000 0001 0000 0011 0010 0101 0100 0111 0110 1001 1000 1011 1010 1101 1100 1111 1110 0001 0000 0000
  (3)                                              0
  (2)                                              0
  (1)                                              0
  (0)                                              0
Now       )00 ps     ps   20000 ps   40000 ps   60000 ps   80000 ps
Cursor 1   0 ps      0 ps
```