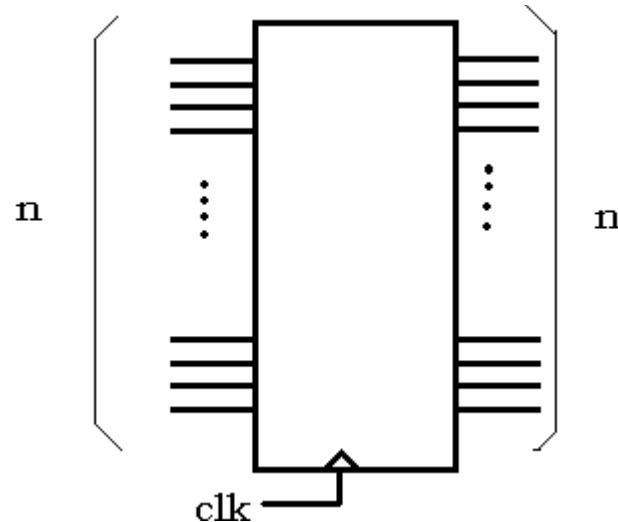


**ENTIDAD Y ARQUITECTURA DONDE EL USUARIO
TENGA LA CAPACIDAD DE GENERAR UN REGISTRO DE
“N” BITS.**



Primero creamos un flip flop tipo D síncrono, el cual se usa para este tipo de registros

1° ffd.vhd

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

```
entity ffd is
  Port ( d : in std_logic;
        clk : in std_logic;
        q : out std_logic);
end ffd;
```

```
architecture algortimo of ffd is
```

```
begin
process (clk)
begin
if clk='1' and clk'event then
  q<=d;
end if;
end process;
end algortimo;
```

2° el archivo que va a usar este flip flop y el cual va a hacer configurable el registro
registro_n.vhd

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

```

use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity registro_n is
    generic (n: positive);
    Port ( inn : in std_logic_vector((n-1) downto 0);
          outn : out std_logic_vector((n-1) downto 0);
          clk : in std_logic);
end registro_n;

architecture estructural of registro_n is
    component ffd is
        port (d,clk:in std_logic;
              q: out std_logic);
    end component;
begin
    paso: for i in 0 to n-1 generate
        d: ffd port map (inn(i),clk,outn(i));
    end generate paso;
end estructural;

```

3º Ahora para probar la capacidad de esta implementación, vamos a desarrollar un archivo que use esto: prueba_problema2.vhd

```

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

entity prueba_problema2 is
    Port ( entrada : in std_logic_vector(31 downto 0);
          salida : out std_logic_vector(31 downto 0);
          clk : in std_logic);
end prueba_problema2;

architecture Behavioral of prueba_problema2 is
    component registro_n is
        generic (n:positive);
        Port ( inn : in std_logic_vector((n-1) downto 0);
              outn : out std_logic_vector((n-1) downto 0);
              clk : in std_logic);
    end component;
begin
    A: registro_n generic map (32) port map (entrada,salida,clk);
end Behavioral;

```

