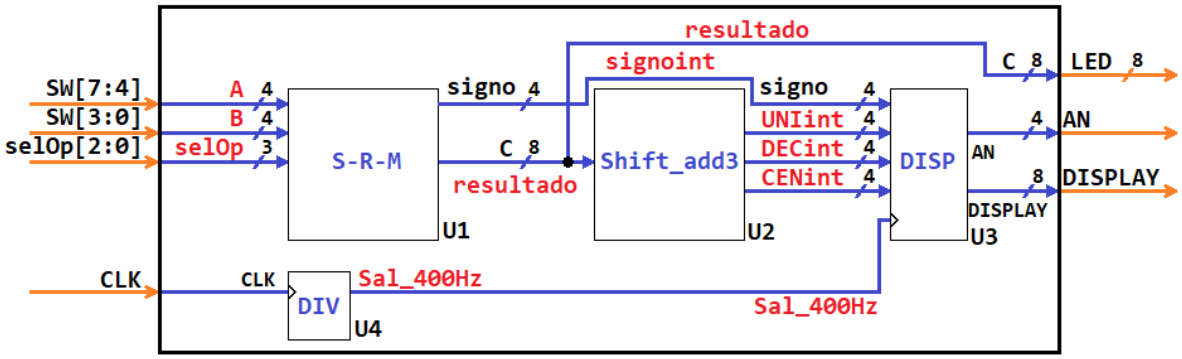
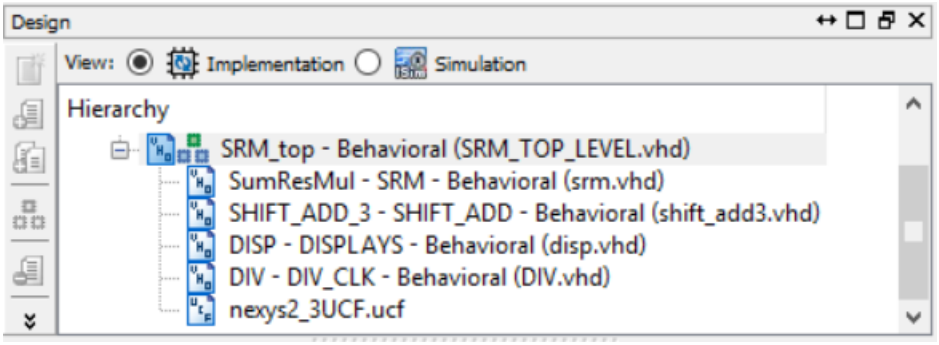
Sumador-Restador-Multiplicador en VHDL con diseño de alto nivel

Se presenta un ejemplo de un SRM de 4 bits con salida a display implementado en la tarjeta nexys, utilizando diseño de alto nivel compuesto por 4 códigos escritos en VHDL, como se presenta en el diagrama de bloques siguiente y el diseño del proyecto en el ISE de Xilinx:



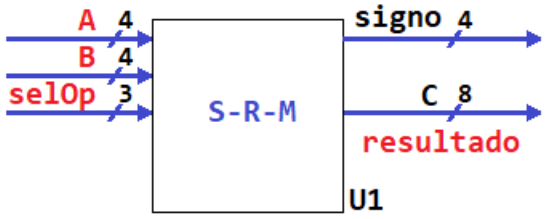



```
----------------------------------------------------------------------------------
-- Top Level Design
-- Sumador, restador y multiplicador
-- con salida a display.
----------------------------------------------------------------------------------

Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

---------------------------------------------------------------
-- Declaración de la entidad

entity SRM_top is
Port (
        CLK: in STD_LOGIC; -- reloj de 50MHz para la nexys 2 y 100MHz para nexys 3
        SW: in STD_LOGIC_VECTOR(7 DOWNTO 0); -- A SW[7:4], B SW[3:0]
        selOp: in STD_LOGIC_VECTOR(2 DOWNTO 0); -- selector de operación S R M
        LED: OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- salida a leds testigos
```

```
        DISPLAY: out STD_LOGIC_VECTOR(7 DOWNTO 0); -- segmentos del display
                                                   --"abcdefgP"
        AN: out STD_LOGIC_VECTOR(3 DOWNTO 0)); -- ánodos del display

end SRM_top;


----------------------------------------------------------------
-- Declaración de la arquitectura

architecture Behavioral of SRM_top is

-- Declaración de señales del divisor
signal SAL_400Hz: std_logic; --salidas 2.5ms

-- Declaración de señales del resultado
signal resultado: STD_LOGIC_VECTOR(7 DOWNTO 0); -- señal para leds y resultado

-- Declaración de señales de la asignación de U-D-C-UM
signal UNIint,DECint,CENint, signoint: std_logic_vector (3 DOWNTO 0); -- U D C signo

BEGIN

LED <= resultado;

-------------------------------------------------------------------------------------------
-- Declaración del srm con signo
-- U1
        SumResMul: ENTITY WORK.SRM PORT MAP(
                A               =>      SW(7 DOWNTO 4),     -- a SW
                B               =>      SW(3 DOWNTO 0),     -- a SW
                C               =>      resultado,          -- a señal p/LD y shift_add3 (U2)
                selOp           =>      selOp,              -- a BTN
                signo           =>      signoint            -- a señal p/displays (U3)
        );
----------------------------------------------------------------


-------------------------------------------------------------------------------------------
-- Declaración del componente que convierte de binario a decimal
-- por la metodología de correr y sumar 3 (shift and add 3)
-- U2
        SHIFT_ADD_3: ENTITY WORK.SHIFT_ADD PORT MAP(
                C => resultado,  -- a señal p/LD y srm (U1)
                UNI => UNIint,   -- a señal p/displays (U3)
                DEC => DECint,  -- a señal p/displays (U3)
                CEN => CENint   -- a señal p/displays (U3)
        );
----------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------
-- Declaración del controlador de display
-- U3
        DISP: ENTITY WORK.DISPLAYS PORT MAP(
                UNI             =>      UNIint,         -- a señal p/shift_add (U2)
                DEC             =>      DECint,         -- a señal p/shift_add (U2)
                CEN             =>      CENint,         -- a señal p/shift_add (U2)
                signo           =>      signoint,       -- a señal p/srm (U1)
                SAL_400Hz       =>      SAL_400Hz,      -- a señal p/div_clk (U4)
                DISPLAY         =>      DISPLAY,        -- a segmentos del display
                AN              =>      AN              -- a ánodos del display
        );
-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------
-- Declaración del componente de los divisores (1ms=1kHz, 2.5ms=400Hz)
-- U4
        DIV: ENTITY WORK.DIV_CLK PORT MAP(
                CLK             =>      CLK,            -- a reloj 50MHz p/nexys2
                SAL_400Hz       =>      SAL_400Hz       -- a señal p/displays (U4)
        );
-------------------------------------------------------------------------------------

end Behavioral; -- fin de la arquitectura TLD
```

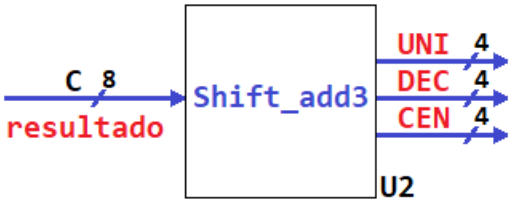
El primer componente (U1) a implementar es el sumador-restador-multiplicador con entradas de datos (A y B) de 4 bits, un selector de operación (selOp) para visualizar la suma, resta o multiplicación, mandando el resultado de 8 bits y el signo, con el siguiente código.




```
-------------------------------------------------------------------------------------
-- programa de un sumador, restador y multiplicador de
-- 4 bits con salida a display, con BCD shift and add3
-- estructura de top level design
-------------------------------------------------------------------------------------

Library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

----------------------------------------------------------------
-- Declaración de la entidad

entity SRM is
Port (
        A,B: in STD_LOGIC_VECTOR(3 DOWNTO 0);       -- entradas con interruptores
        C: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);        -- salida a leds testigos
        signo: out STD_LOGIC_VECTOR(3 DOWNTO 0);  -- salida para el signo
        selOp: in STD_LOGIC_VECTOR(2 DOWNTO 0));  -- a botones
end SRM;


----------------------------------------------------------------
-- Declaración de la arquitectura

architecture Behavioral of SRM is

BEGIN

----------------------OPERACIONES-----------------------------------
-- en este proceso realiza las operaciones según el selector selOp
PROCESS(A,B,selOp)
begin
        case selOp is
        when "001" =>           c <= ("0000" & A) + ("0000" & B);  signo <= x"E"; -- suma
        when "010" =>                                                   -- resta
                        if A >= B then    c <= ("0000" & A) - ("0000" & B); signo <= x"E";
                        else              c <= ("0000" & B) - ("0000" & A); signo <= x"F";
                        end if;
        when "100" =>           c <= A * B; signo <= x"E";          -- multiplicación
        when others =>          c <= (others => '0'); signo <= x"E";
        end case;
end process; --termina el proceso de operaciones

end Behavioral; -- fin de la arquitectura
```

El segundo componente (U2) es el convertidor BCD para el manejador de los datos (unidades-decenas-centenas) al display. El código utiliza el método que se conoce como desplazar y sumar 3 (shift and add 3).

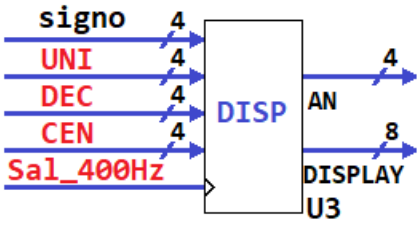



```
----------------------------------------------------------------
-- CONVERTIDOR SHIFT AND ADD 3
----------------------------------------------------------------

Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

----------------------------------------------------------------
-- Declaración de la entidad

entity SHIFT_ADD is
Port (
                C: in std_logic_vector (7 DOWNTO 0):=(others=>'0');     -- 8 bits
                UNI,DEC,CEN: out std_logic_vector (3 DOWNTO 0)          -- digitos unidades,
                );                                                       -- decenas y centenas

end SHIFT_ADD;


----------------------------------------------------------------
-- Declaración de la arquitectura

architecture Behavioral of SHIFT_ADD is

-- Declaración de señales de la asignación de U-D-C
signal P: std_logic_vector (11 DOWNTO 0); -- asigna UNI, DEC, CEN

BEGIN

-----------CONVERTIR DE BIN A BCD-----------------
-- Este proceso contiene un algoritmo recorre y suma 3 para
-- convertir un número binario abcd, que se manda a los displays.
-- El algoritmo consiste en desplazar (shift) el vector inicial
-- (en binario) el número de veces según sea el número de bits,
-- y cuando alguno de los bloques de 4 bits (U-D-C-UM, que es el
-- número de bits necesarios para que cuente de 0 a 9 por cifra)
-- sea igual o mayor a 5 (por eso el >4) se le debe sumar 3
-- a ese bloque, después se continua desplazando hasta que otro
-- (o el mismo) bloque cumpla con esa condición y se le sumen 3.
```

```
-- Inicialmente se rota 3 veces porque es el número mínimo de bits
-- que debe tener para que sea igual o mayor a 5.
-- Finalmente se asigna a otro vector, el vector ya convertido,
-- que cuenta con 3 bloques para las 3 cifras de 4 bits cada una.

PROCESS(C)

        --20 bits para separar las Centenas-Decenas-Unidades
        VARIABLE C_D_U:STD_LOGIC_VECTOR(19 DOWNTO 0);

BEGIN

--ciclo de inicialización------------------------------
        FOR I IN 0 TO 19 LOOP -- manda ceros a los 18 bits +2
                C_D_U(I):='0'; -- se inicializa con 0
        END LOOP;

C_D_U(7 DOWNTO 0):=C(7 downto 0); --c del SRM de 8 bits

--ciclo de asignación C-D-U-------------------------
FOR I IN 0 TO 7 LOOP -- hace 8 veces el ciclo shift-add3

        IF C_D_U(11 DOWNTO 8) > 4 THEN -- U --
        C_D_U(11 DOWNTO 8):= C_D_U(11 DOWNTO 8)+3;
        END IF;

        IF C_D_U(15 DOWNTO 12) > 4 THEN -- D --
        C_D_U(15 DOWNTO 12):= C_D_U(15 DOWNTO 12)+3;
        END IF;

        IF C_D_U(19 DOWNTO 16) > 4 THEN -- C --
        C_D_U(19 DOWNTO 16):= C_D_U(19 DOWNTO 16)+3;
        END IF;

        -- shift -- realiza el corrimiento -----------
        C_D_U(19 DOWNTO 1):= C_D_U(18 DOWNTO 0);

END LOOP;

P<=C_D_U(19 DOWNTO 8); -- guarda en P y en seguida se separan C-D-U

END PROCESS; -- fin del proceso Display

-- separa C-D-U --
-- UNIDADES --
        UNI<=P(3 DOWNTO 0);
-- DECENAS --
        DEC<=P(7 DOWNTO 4);
```

```
-- CENTENAS --
        CEN<=P(11 DOWNTO 8);

-----------------------------------------------
end Behavioral; -- fin de la arquitectura
```

El componente 3 (U3) es el encargado de desplegar el resultado en un display de 4 dígitos conectado en bus (los datos de los segmentos llegan a todos los dígitos) con control de 4 ánodos habilitados en bajo para la Nexys.

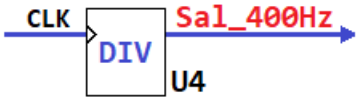



```
-------------------------------------------------------------------------------
-- Controlador del display de 4 digitos
-------------------------------------------------------------------------------

Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

-----------------------------------------------------------------
-- Declaración de la entidad

entity DISPLAYS is
Port (
                UNI,DEC,CEN,signo: in std_logic_vector (3 DOWNTO 0); -- dígitos unidades, decenas,
                                                                     -- centenas y signo
                SAL_400Hz: in STD_LOGIC; -- reloj de 400Hz
                DISPLAY: out STD_LOGIC_VECTOR(7 DOWNTO 0); -- seg dsply "abcdefgP"
                AN: out STD_LOGIC_VECTOR(3 DOWNTO 0)); -- ánodos del display

end DISPLAYS;


-------------------------------------------------------------
-- Declaración de la arquitectura

architecture Behavioral of DISPLAYS is

-- Declaración de señales de la multiplexación y asignación de U-D-C al disp
```

```
signal SEL: std_logic_vector (1 downto 0):="00"; -- selector de barrido
signal D: std_logic_vector (3 downto 0); -- almacena los valores del disp


BEGIN


----------MULTIPLEXOR----------

PROCESS(SAL_400Hz, sel, UNI, DEC,CEN)
BEGIN

        IF SAL_400Hz'EVENT and SAL_400Hz='1' THEN SEL <= SEL + '1';

                CASE(SEL) IS
                when "00" => AN <="0111"; D <= UNI;             -- UNIDADES
                when "01" => AN <="1011"; D <= DEC;             -- DECENAS
                when "10" => AN <="1101"; D <= CEN;             -- CENTENAS
                when "11" => AN <="1110"; D <= SIGNO;           -- signo
                when OTHERS=>AN <="1111"; D <= SIGNO;           -- signo
                END CASE;

        end if;

END PROCESS; -- fin del proceso Multiplexor


----------DISPLAY----------

PROCESS(D)
Begin

        case(D) is                              --    abcdefgP
        WHEN x"0" =>            DISPLAY <= "00000011"; --0
        WHEN x"1" =>            DISPLAY <= "10011111"; --1
        WHEN x"2" =>            DISPLAY <= "00100101"; --2
        WHEN x"3" =>            DISPLAY <= "00001101"; --3
        WHEN x"4" =>            DISPLAY <= "10011001"; --4
        WHEN x"5" =>            DISPLAY <= "01001001"; --5
        WHEN x"6" =>            DISPLAY <= "01000001"; --6
        WHEN x"7" =>            DISPLAY <= "00011111"; --7
        WHEN x"8" =>            DISPLAY <= "00000001"; --8
        WHEN x"9" =>            DISPLAY <= "00001001"; --9
        WHEN x"F" =>            DISPLAY <= "11111101"; --signo
        WHEN OTHERS =>          DISPLAY <= "11111111"; --apagado
        END CASE;
```

```
END PROCESS; -- fin del proceso Display

-----------------------------------------------
end Behavioral; -- fin de la arquitectura
```

El último componente a declarar es U4, un divisor de frecuencia para generar la señal de barrido de los displays a 400Hz (2.5ms).




```
-------------------------------------------------------------------------------
-- Divisor de 2.5ms (400Hz)
-------------------------------------------------------------------------------

Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

-----------------------------------------------------------------
-- Declaración de la entidad

entity DIV_CLK is
Port (
                CLK: in STD_LOGIC; -- reloj de 50MHz para la nexys 2 y 100MHz para nexys 3
                SAL_400Hz: inout STD_LOGIC); --salida 2.5ms,

end DIV_CLK;


-----------------------------------------------------------------
-- Declaración de la arquitectura

architecture Behavioral of DIV_CLK is

-- Declaración de señales de los divisores

signal conta_1250us: integer range 1 to 62_500:=1; -- pulso1 de 1250us@400Hz (0.25ms)

BEGIN

-----------------DIVISOR 2.5ms=400Hz----------------
------------------DIVISOR ÁNODOS------------------
process (CLK) begin
```

```
        if rising_edge(CLK) then
                if (conta_1250us = 62_500) then --cuenta 1250us (50MHz=62500)
                        -- if (conta_1250us = 125000) then --cuenta 1250us (100MHz=125000)
                        SAL_400Hz <= NOT(SAL_400Hz); --genera un barrido de 2.5ms
                        conta_1250us <= 1;
                else
                        conta_1250us <= conta_1250us + 1;
                end if;
        end if;

end process; -- fin del proceso Divisor Ánodos

------------------------------------------------
end Behavioral; -- fin de la arquitectura
```

// Archivo de restricciones de usuario (UCF) //

```
########################################
##Asignación de terminales de la NEXYS 2 NEXYS 2 NEXYS 2

//DISPLAYS
NET "DISPLAY(7)" LOC = "L18"; // a
NET "DISPLAY(6)" LOC = "F18"; // b
NET "DISPLAY(5)" LOC = "D17"; // c
NET "DISPLAY(4)" LOC = "D16"; // d
NET "DISPLAY(3)" LOC = "G14"; // e
NET "DISPLAY(2)" LOC = "J17"; // f
NET "DISPLAY(1)" LOC = "H14"; // g
NET "DISPLAY(0)" LOC = "C17"; // P

//ANODOS
NET "AN(3)" LOC = "F17"; // AN0
NET "AN(2)" LOC = "H17"; // AN1
NET "AN(1)" LOC = "C18"; // AN2
NET "AN(0)" LOC = "F15"; // AN3

//selOp,RESET--PUSH BUTTON
NET "selOp(0)" LOC = "B18"; // btn0 suma
NET "selOp(1)" LOC = "D18"; // btn1 resta
NET "selOp(2)" LOC = "E18"; // btn2 mult
//NET "RESET" LOC = "H13"; // btn3 RST

//CLK reloj 50MHz
NET "CLK" LOC = "B8";

// entrada A y B a SW
```

```
NET "SW(0)" LOC = "G18"; // SW0 B(0)
NET "SW(1)" LOC = "H18"; // SW1 B(1)
NET "SW(2)" LOC = "K18"; // SW2 B(2)
NET "SW(3)" LOC = "K17"; // SW3 B(3)

NET "SW(4)" LOC = "L14"; // SW4 A(0)
NET "SW(5)" LOC = "L13"; // SW5 A(1)
NET "SW(6)" LOC = "N17"; // SW6 A(2)
NET "SW(7)" LOC = "R17"; // SW7 A(3)

// salida a led testigos del resultado C
NET "led(0)" LOC = "J14"; // LD0
NET "led(1)" LOC = "J15"; // LD1
NET "led(2)" LOC = "K15"; // LD2
NET "led(3)" LOC = "K14"; // LD3
NET "led(4)" LOC = "E17"; // LD4
NET "led(5)" LOC = "P15"; // LD5
NET "led(6)" LOC = "F4" ; // LD6
NET "led(7)" LOC = "R4" ; // LD7


########################################
##Asignación de terminales de la NEXYS 3

#//DISPLAYS
#NET "DISPLAY(7)" LOC = "T17"; // a
#NET "DISPLAY(6)" LOC = "T18"; // b
#NET "DISPLAY(5)" LOC = "U17"; // c
#NET "DISPLAY(4)" LOC = "U18"; // d
#NET "DISPLAY(3)" LOC = "M14"; // e
#NET "DISPLAY(2)" LOC = "N14"; // f
#NET "DISPLAY(1)" LOC = "L14"; // g
#NET "DISPLAY(0)" LOC = "M13"; // P
#
#//ANODOS
#NET "AN(3)" LOC = "N16"; // AN0
#NET "AN(2)" LOC = "N15"; // AN1
#NET "AN(1)" LOC = "P18"; // AN2
#NET "AN(0)" LOC = "P17"; // AN3
#
#//A,B--SW,RESET--PUSH BUTTON
#NET "UpDown(1)" LOC = "A8"; // btn0 SUBE
#NET "UpDown(0)" LOC = "C9"; // btn1 BAJA
#NET "RESET" LOC = "D9"; // btn2 RST
#
#//CLK reloj 100MHz
#NET "CLK" LOC = "V10"; #100MHz
#
```

#//C-LEDs salida a leds testigo
#NET "" LOC = "U16"; // LD0
##Asignación de terminales de la nexys3

Por último, se presentan 3 fotos de las operaciones realizadas (resta, suma y multiplicación).