Laboratorio 1:

Lógica combinacional

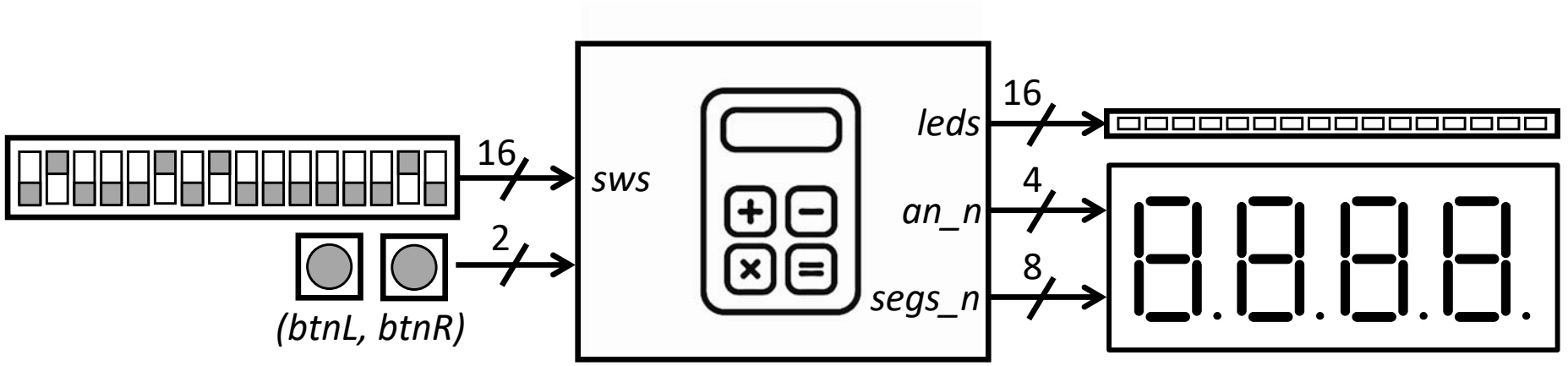
aritmética y acceso a dispositivos elementales de E/S

Diseño automático de sistemas

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Presentación

- Diseñar una ALU combinacional de 8 bits capaz de sumar, restar, multiplicar y calcular el opuesto.
- Leerá los operandos y visualizará los resultados del siguiente modo:
  - El operando derecho lo leerá de los 8 switches menos significativos.
  - El operando izquierdo lo leerá de los 8 switches más significativos.
  - El código de operación lo leerá de los pulsadores (btnL,btnR):
    - "00" : Suma, "01" : Resta, "10" : Opuesto del operando derecho, "11" : Multiplicación.
  - El resultado lo visualizará en binario en los 16 leds.
  - Los 4 bits menos significativos del resultado los visualizará en hexadecimal en el display 7-segs derecho.

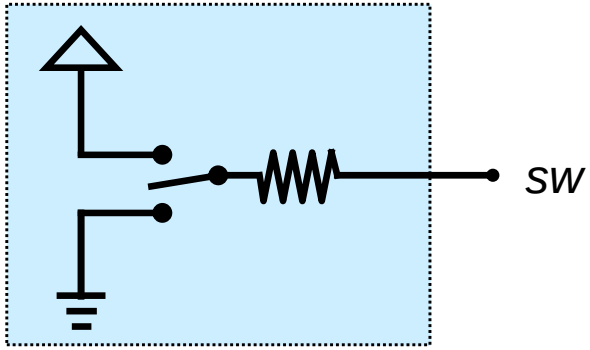


J.M. Mendias
2024

Entrada elemental

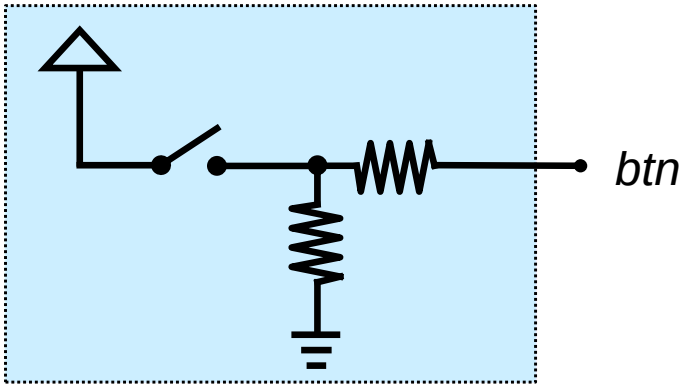
interruptores y pulsadores

- La Basys 3 dispone de 16 interruptores.
  - Conectados a los pines de la FPGA a través de resistencias limitadoras.



- Además dispone de 5 pulsadores en configuración de lógica directa.
  - Con resistencias de pull-down y conectados a los pines de la FPGA a través de resistencias limitadoras.

lógica directa
(al pulsar, X=1)

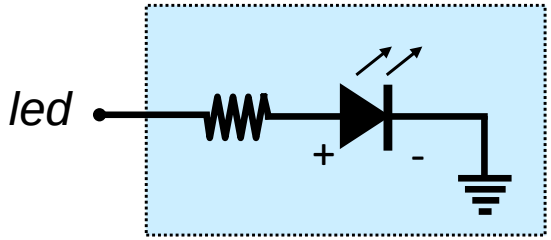


J.M. Mendias
2024

Salida elemental

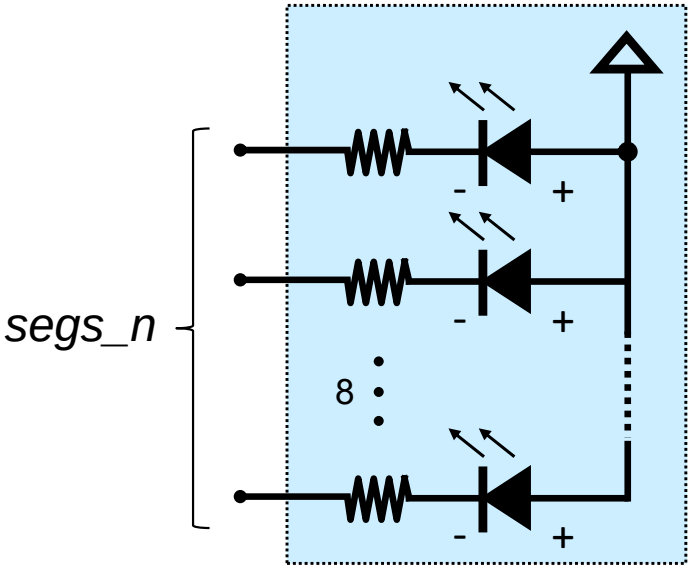
leds y displays 7 segmentos

- La Basys 3 dispone de 16 leds en configuración de lógica directa.
  - Conectados a los pines de la FPGA a través de resistencias limitadoras que aseguran los niveles de voltaje e intensidad requeridos para que el led se ilumine.

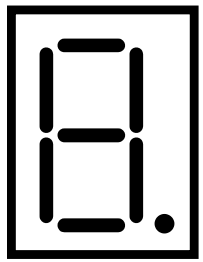


lógica directa
(se ilumina con X=1)

- Además dispone de 4 displays 7-segs en configuración de ánodo común.



lógica inversa
(se ilumina con X=0)

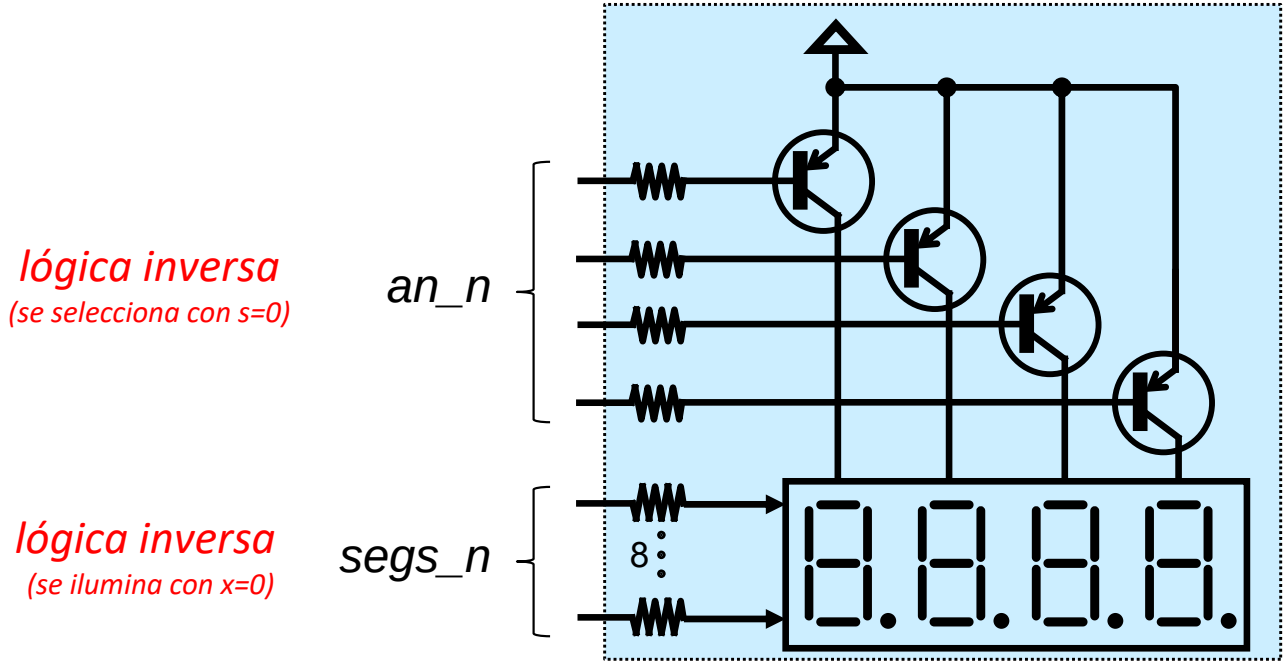


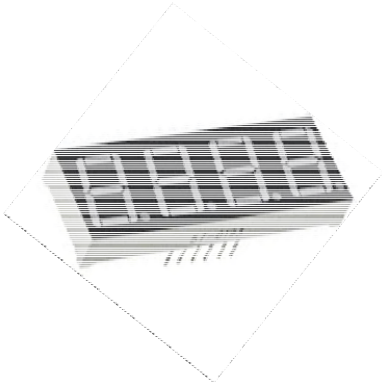
J.M. Mendías
2024

Salida elemental

banco de displays 7 segmentos: organización

- Para reducir el pineado los 4 displays 7 segs se agrupan en bancos.
  - Todos los displays comparten las mismas 8 señales de datos.
  - La activación de cada display se controla individualmente conectando su ánodo a Vcc a través de transistores bipolares pnp
    - Si todos los displays se activan, todos mostrarán el mismo dígito.
    - Si solo un display se activa, solo él mostrará el dígito enviado por las líneas de datos.

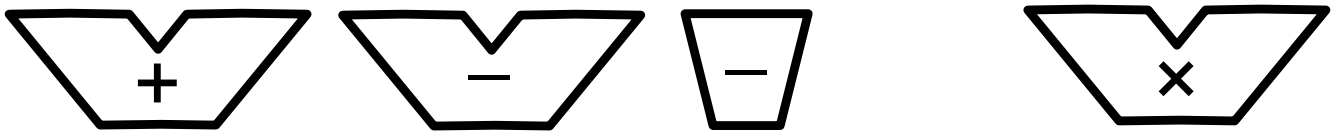




J.M. Mendías
2024

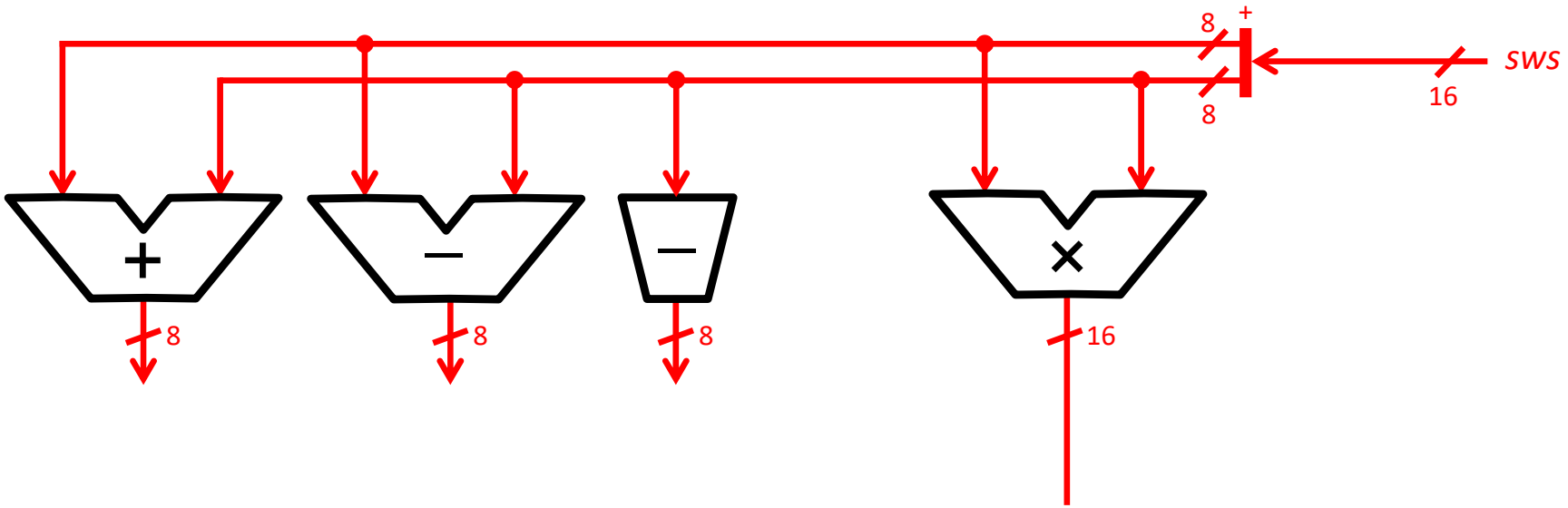
Diseño principal

esquema RTL



Diseño principal

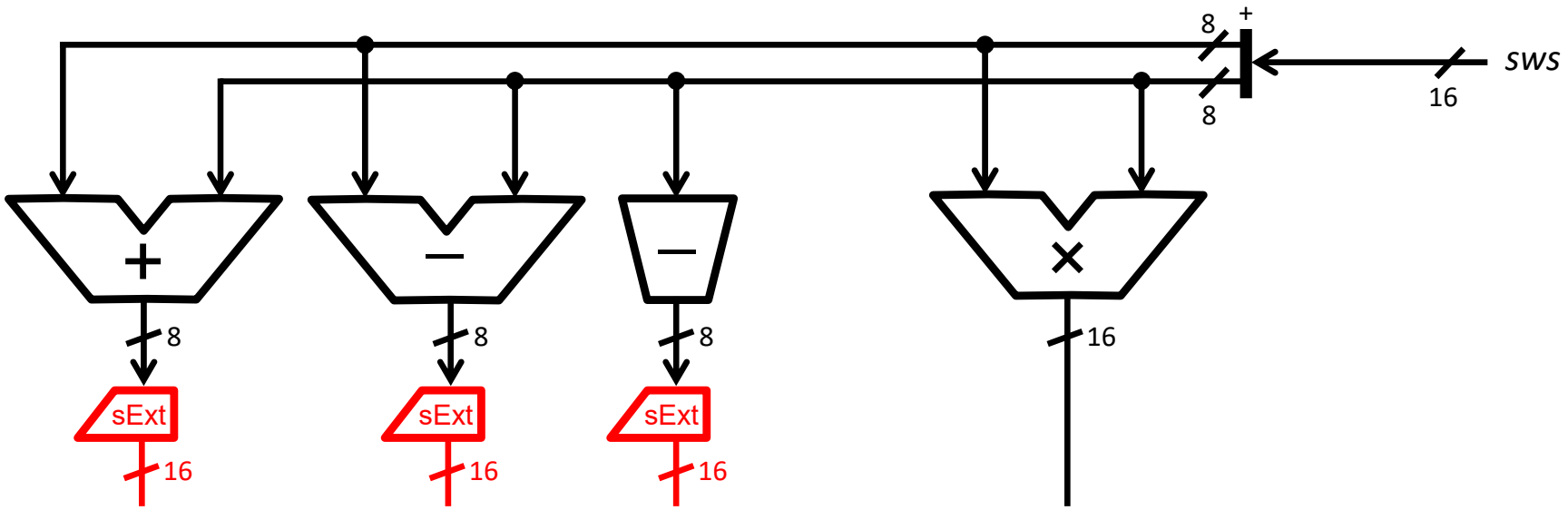
esquema RTL



J.M. Mendías
2024

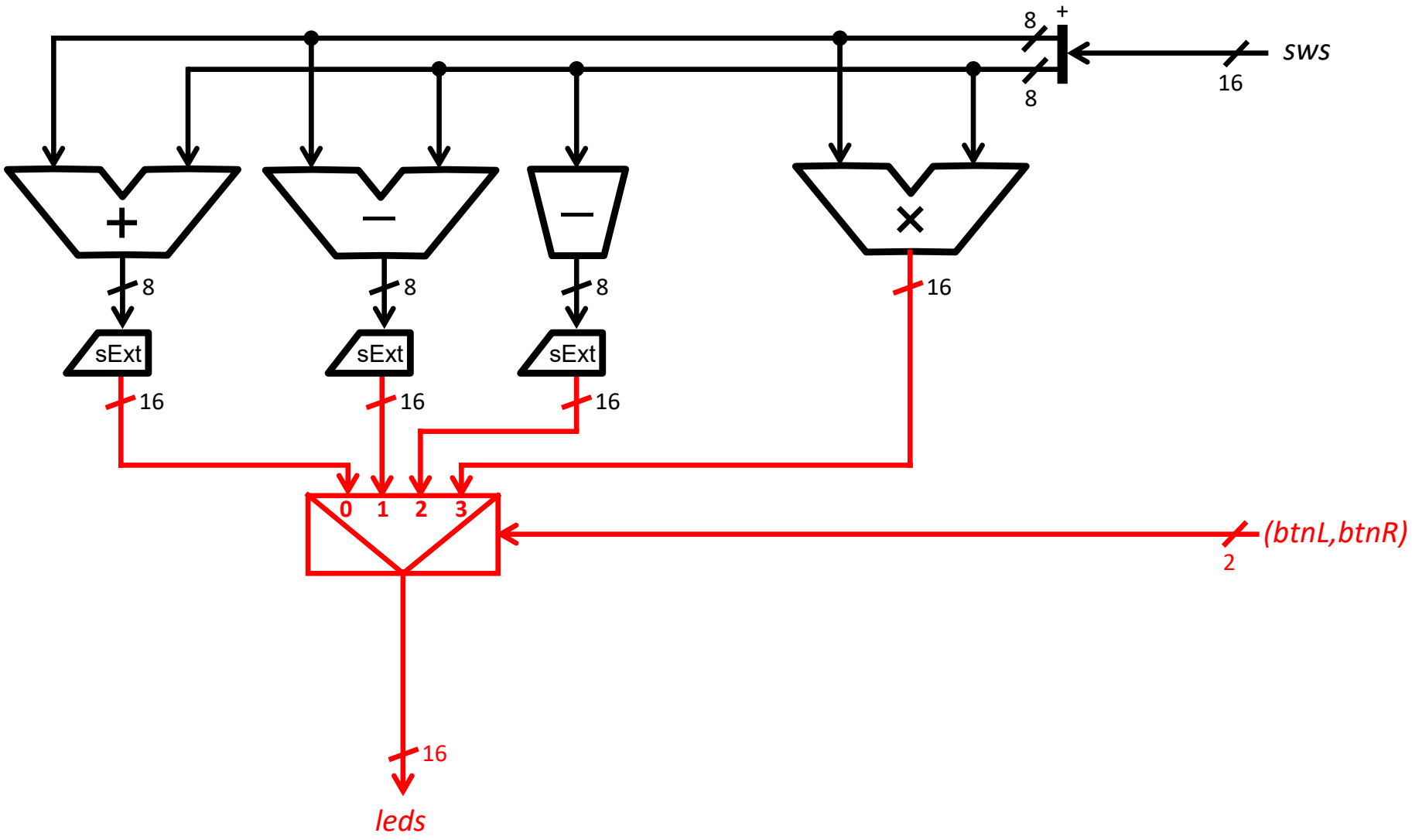
Diseño principal

esquema RTL



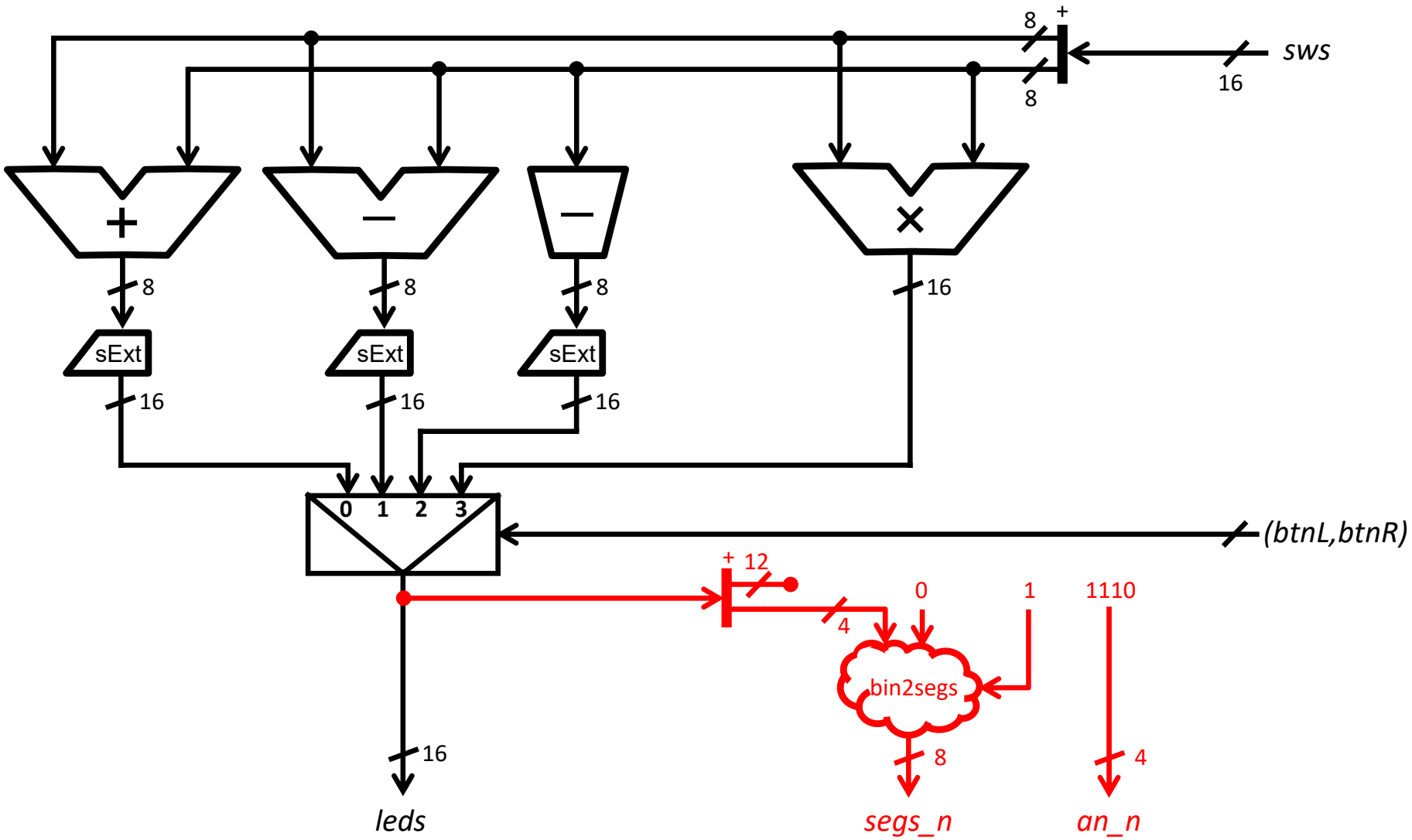
Diseño principal

esquema RTL



Diseño principal

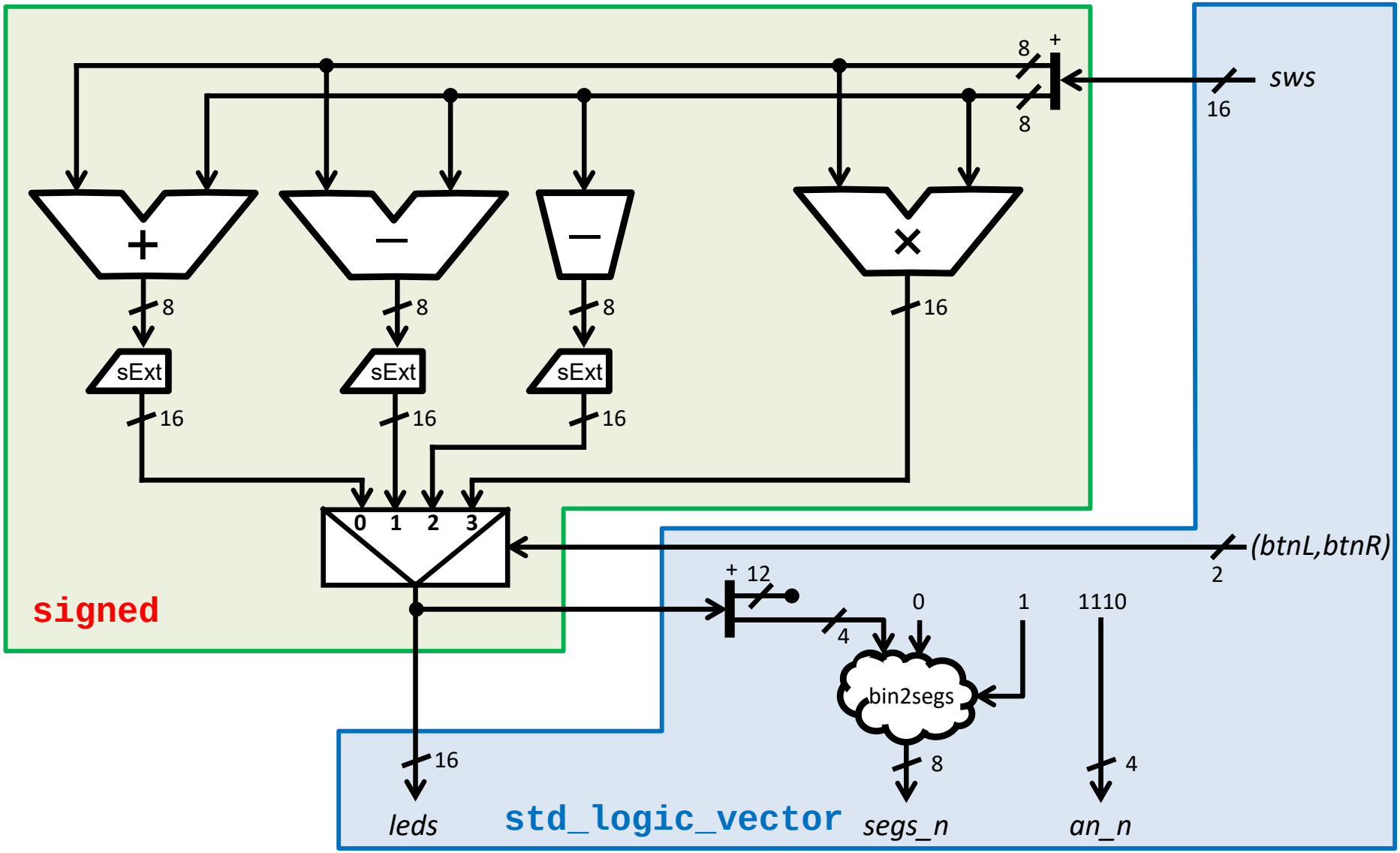
esquema RTL



J.M. Mendias
2024

Diseño principal

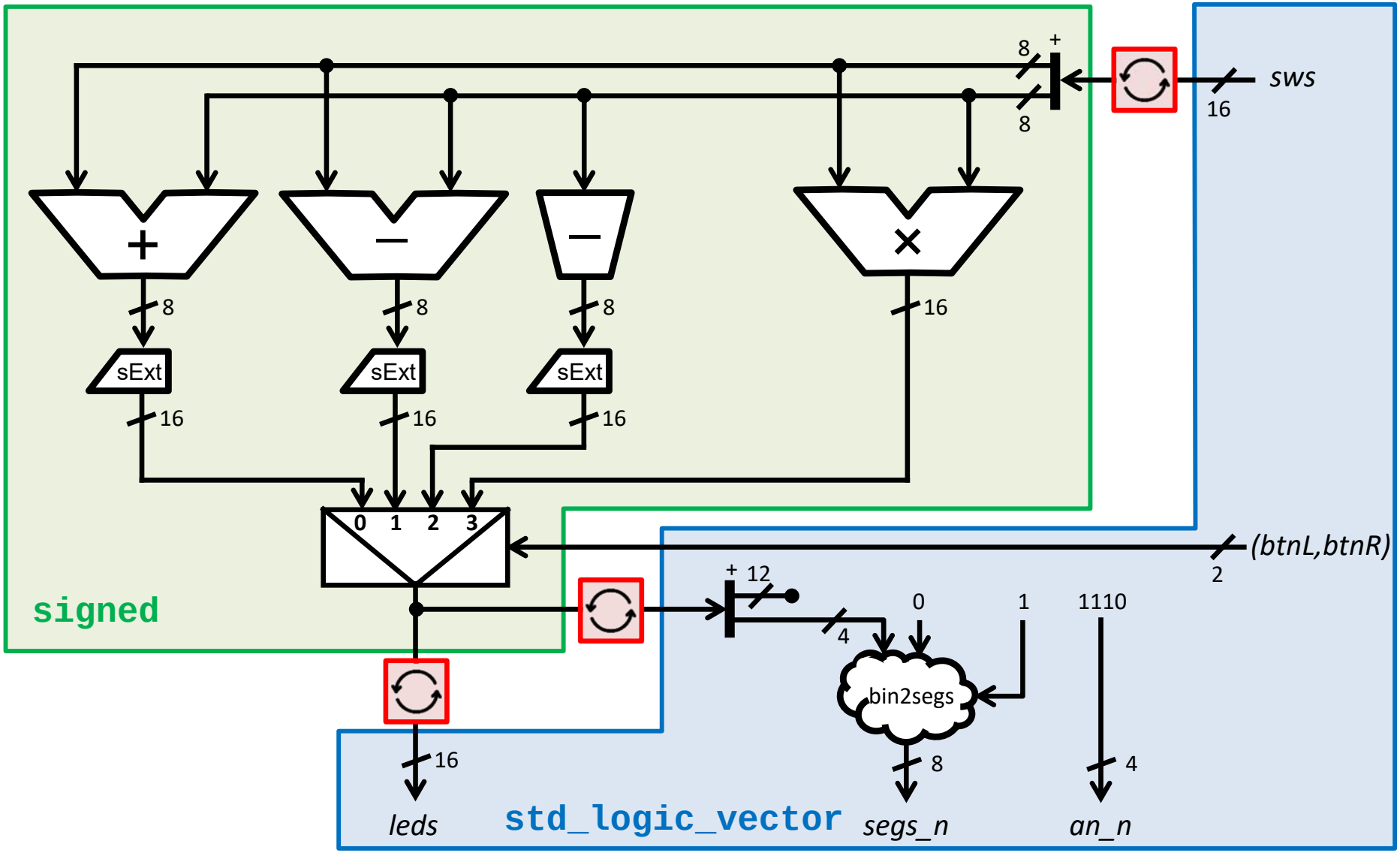
esquema RTL



J.M. Mendias
2024

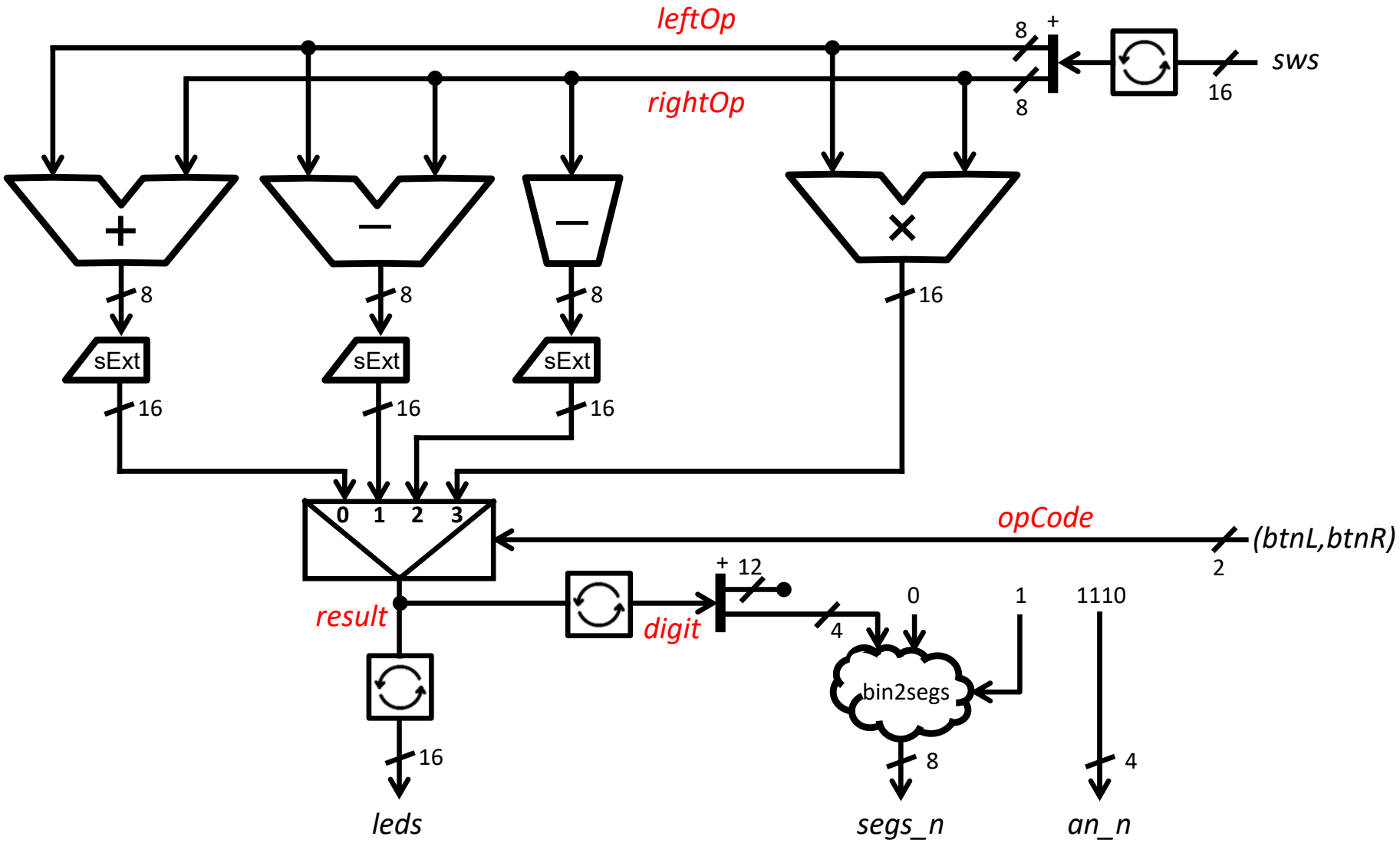
Diseño principal

esquema RTL



Diseño principal

esquema RTL



J.M. Mendias
2024

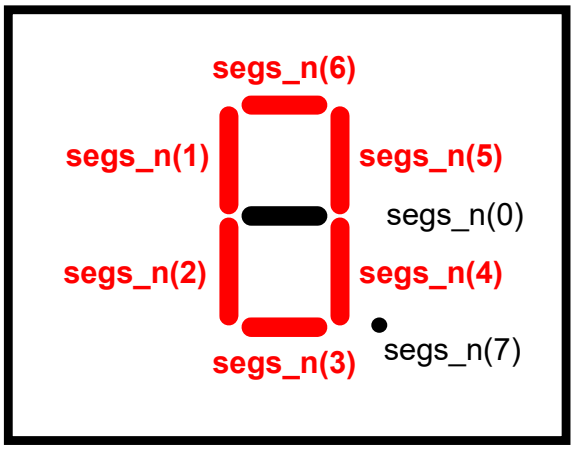
Módulo conversor 7-segmentos

bin2segs.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity bin2segs is
  port (
    -- host side
    en     : in std_logic;                          -- capacitacion
    bin    : in std_logic_vector(3 downto 0);       -- codigo binario
    dp     : in std_logic;                          -- punto
    -- leds side
    segs_n : out std_logic_vector(7 downto 0)       -- codigo 7-segmentos
);
end bin2segs;

architecture syn of bin2segs is
  signal segs : std_logic_vector(7 downto 0);
begin
  segs(7) <= ...;
  with bin select
    segs(6 downto 0) <=
      "0000001" when X"0",·····  visualiza el 0 en lógica inversa
      "......." when X"1",
         ...
      "......." when others;
  segs_n <= ... when ... else ...;
end syn;
```



Utilidades y componentes

common.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;

package common is

  constant YES  : std_logic := '1';
  constant NO   : std_logic := '0';
  constant HI   : std_logic := '1';
  constant LO   : std_logic := '0';
  constant ONE  : std_logic := '1';
  constant ZERO : std_logic := '0';

  -- Calcula el logaritmo en base-2 de un numero.
  function log2(v : in natural) return natural;
  ...

  -- Convierte codigo binario a codigo 7-segmentos
  component bin2segs
    port
    (
      -- host side
      en     : in std_logic;                      -- capacitacion
      bin    : in std_logic_vector(3 downto 0);   -- codigo binario
      dp     : in std_logic;                      -- punto
      -- leds side
      segs_n : out std_logic_vector(7 downto 0)   -- codigo 7-segmentos
    );
  end component;

end package common;
```

declaración de funciones de utilidad

declaración de componentes reusables

J.M. Mendías
2024

Utilidades y componentes

common.vhd

```
package body common is

  function log2(v : in natural) return natural is
    variable n    : natural;
    variable logn : natural;
  begin
    n := 1;
    for i in 0 to 128 loop
      logn := i;
      exit when (n >= v);
      n := n * 2;
    end loop;
    return logn;
  end function log2;

  ...

end package body common;
```

funciones orientadas a trabajar con argumentos constantes en expresiones computables

Diseño principal

lab1.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity lab1 is
  port (
    sws    : in  std_logic_vector(15 downto 0);
    btnL   : in  std_logic;
    btnR   : in  std_logic;
    leds   : out std_logic_vector(15 downto 0);
    an_n   : out std_logic_vector(3 downto 0);
    segs_n : out std_logic_vector(7 downto 0)
  );
end lab1;

library ieee;
use ieee.numeric_std.all;
use work.common.all;

architecture syn of lab1 is

  signal opCode  : std_logic_vector(1 downto 0);
  signal leftOp  : signed(7 downto 0);
  signal rightOp : signed(7 downto 0);
  signal result  : signed(15 downto 0);
  signal digit   : std_logic_vector(3 downto 0);

begin
  ...
end syn;
```

permite usar las utilidades y las componentes

declaración de los tipos de las señales internas

Diseño principal

lab1.vhd

```
begin

  opCode  <= ...;
  leftOp  <= ...;
  rightOp <= ...;

  ALU:
  with opCode select
    result <=
      ...;

  leds  <= ...;
  digit <= ...;

  an_n  <= ...;

  converter : bin2segs
  port map ( en => ..., bin => ..., dp => ..., segs => ... );

end syn;
```

adapta puertos de entrada a señales internas

núcleo operativo

adapta señales internas a puertos de salida

usar siempre asociación por nombre

J.M. Mendías
2024

Tareas

1. Crear la carpeta **DAS** y en ella las carpetas **source** y **projects**.
2. En la carpeta **DAS/source** crear las carpetas **common** y **lab1**.
3. Descargar de la Web los ficheros en:
   - **common:** **common.vhd** y **bin2segs.vhd**
   - **lab1:** **lab1.vhd** y **lab1.xdc**
4. Completar el código omitido en los ficheros:
   - **bin2segs.vhd** y **lab1.vhd**
5. En la carpeta **DAS/projects** crear el proyecto **lab1**.
6. Incluir en el proyecto (sin copiarlos) los siguientes fuentes:
   - **common.vhd**, **bin2segs.vhd**, **lab1.vhd** y **lab1.xdc**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Conectar la placa y encenderla.
9. Volcar el fichero de configuración **lab1.bit**

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/