Laboratorio 10:

Interfaces programables de E/S

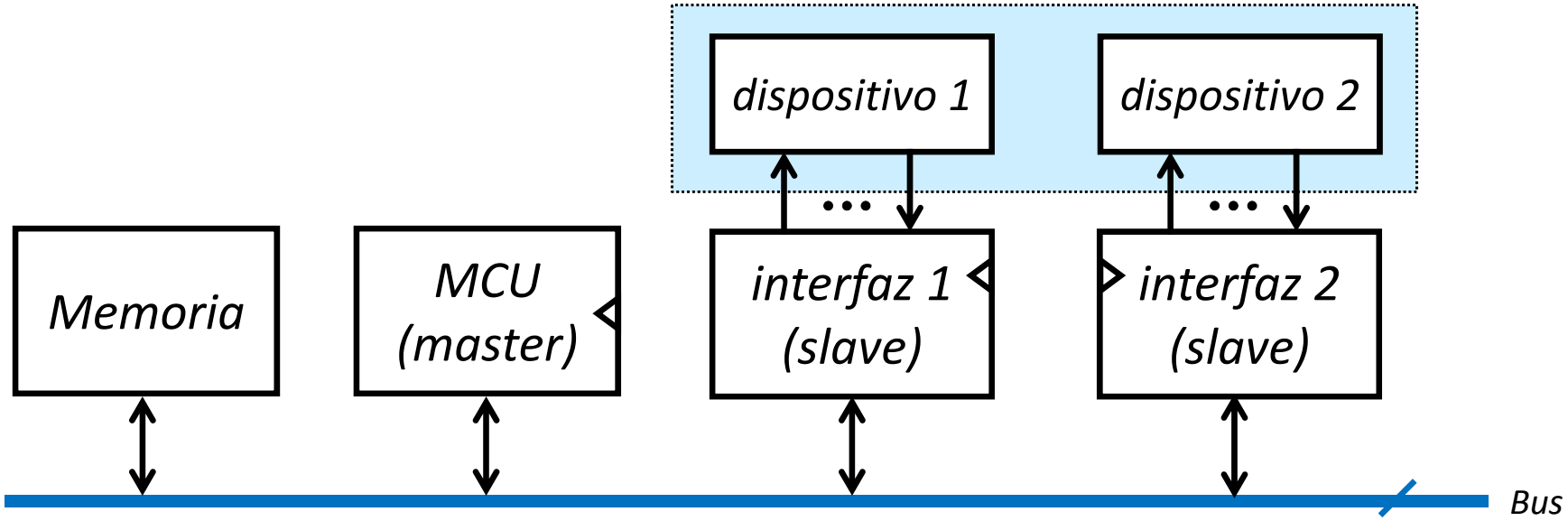
mapeo en el espacio de direcciones de un microcontrolador

Diseño automático de sistemas

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

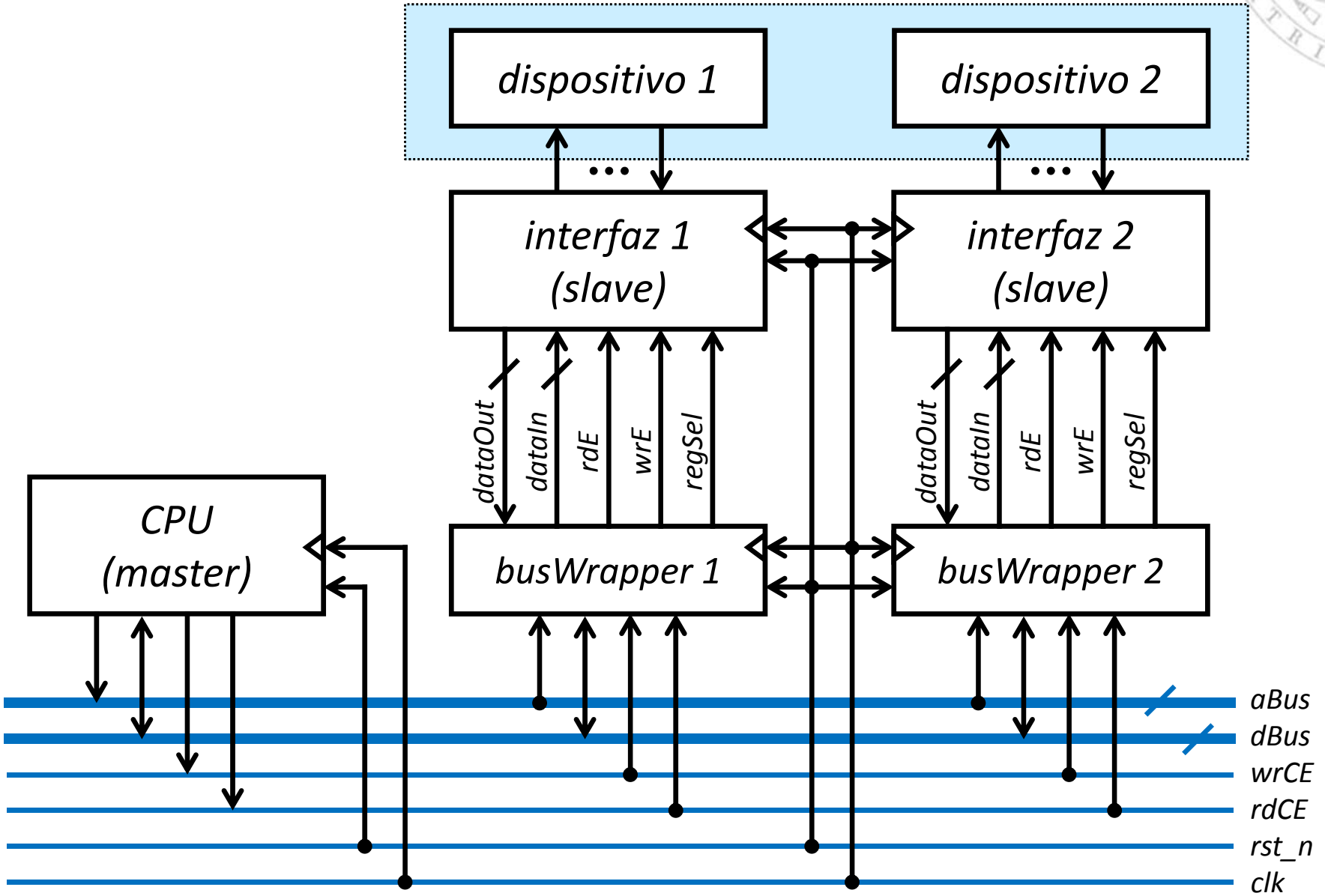
Presentación

- Diseñar un esquema genérico de interconexión de interfaces programables de dispositivos a un bus síncrono.
  - Conceptualmente análogo al usado por Xilinx EDK para interconectar sistemas hardware a un microcontrolador.
  - Cada interfaz se mapeará en un rango de direcciones dado y actuará como esclavo del bus realizando transferencias de datos desde/hacia el microcontrolador



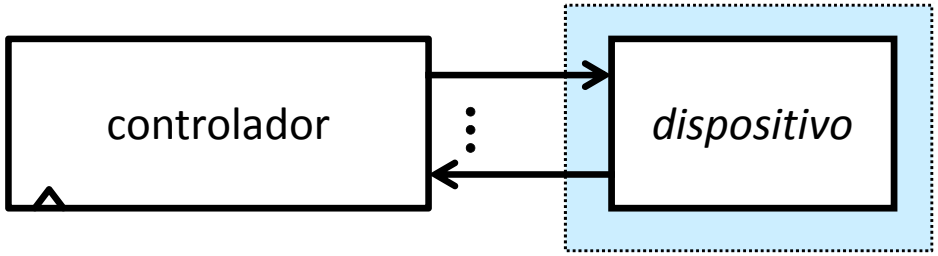
J.M. Mendías
2017

Modelo de sistema



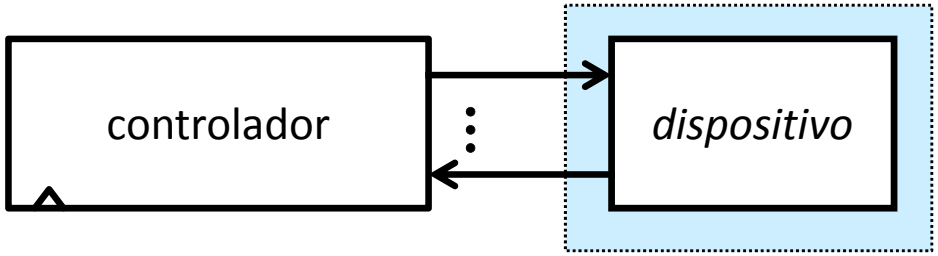
J.M. Mendías
2017

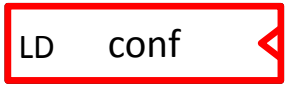
Modelo de interfaz programable

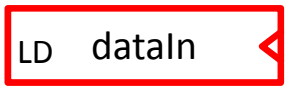


J.M. Mendías
2017

Modelo de interfaz programable

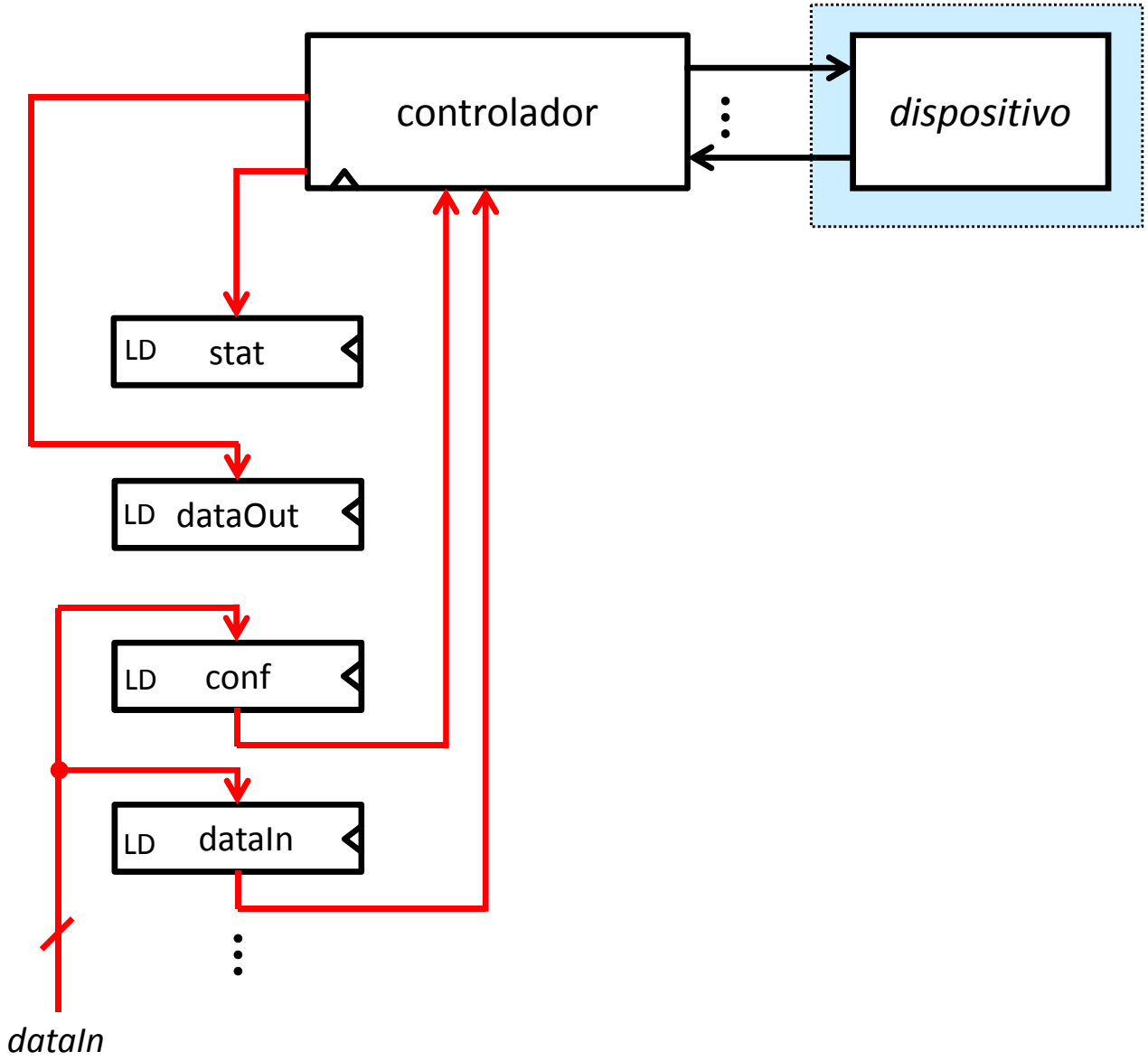






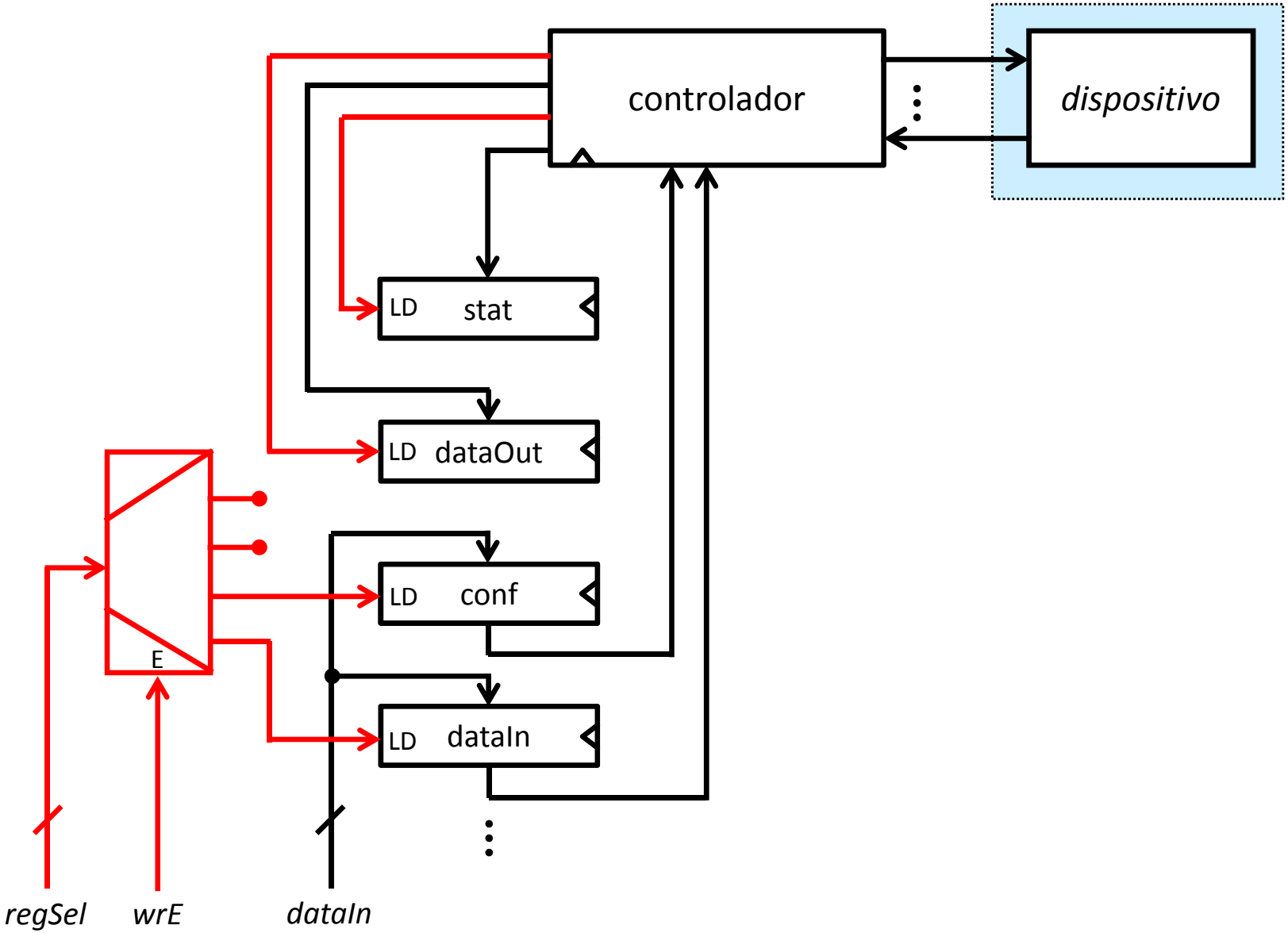
J.M. Mendías
2017

Modelo de interfaz programable



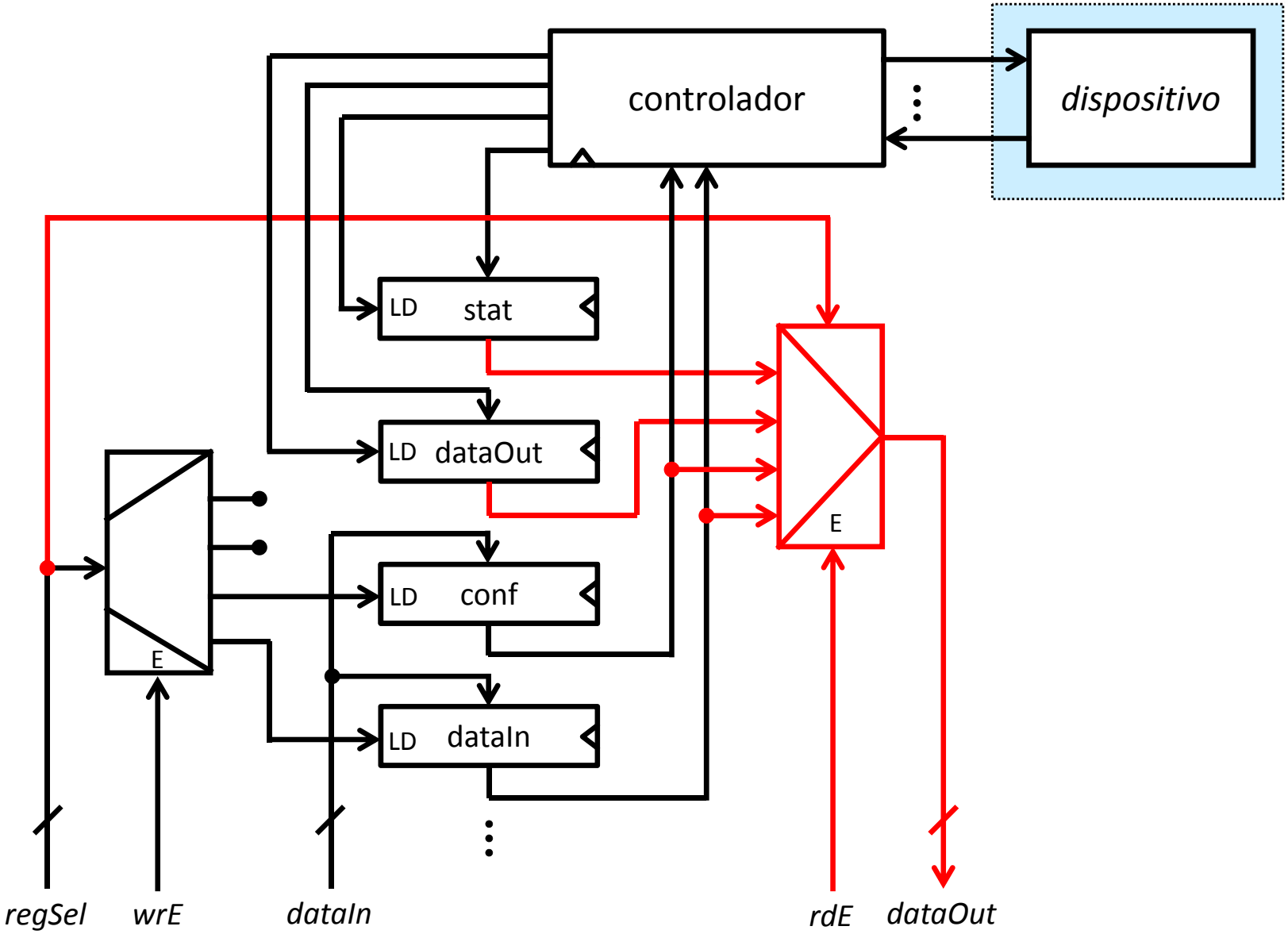
J.M. Mendías
2017

Modelo de interfaz programable



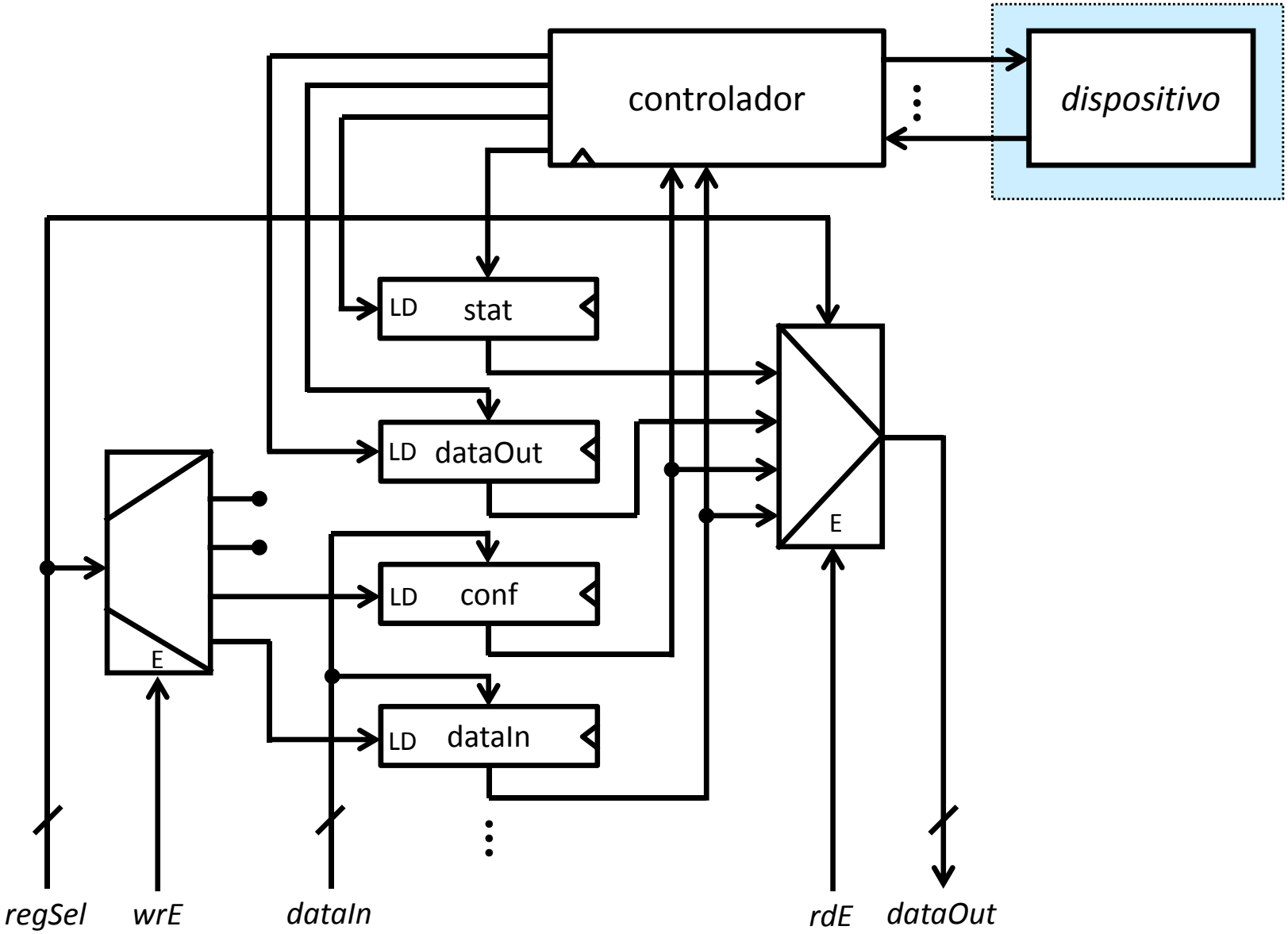
J.M. Mendías
2017

Modelo de interfaz programable



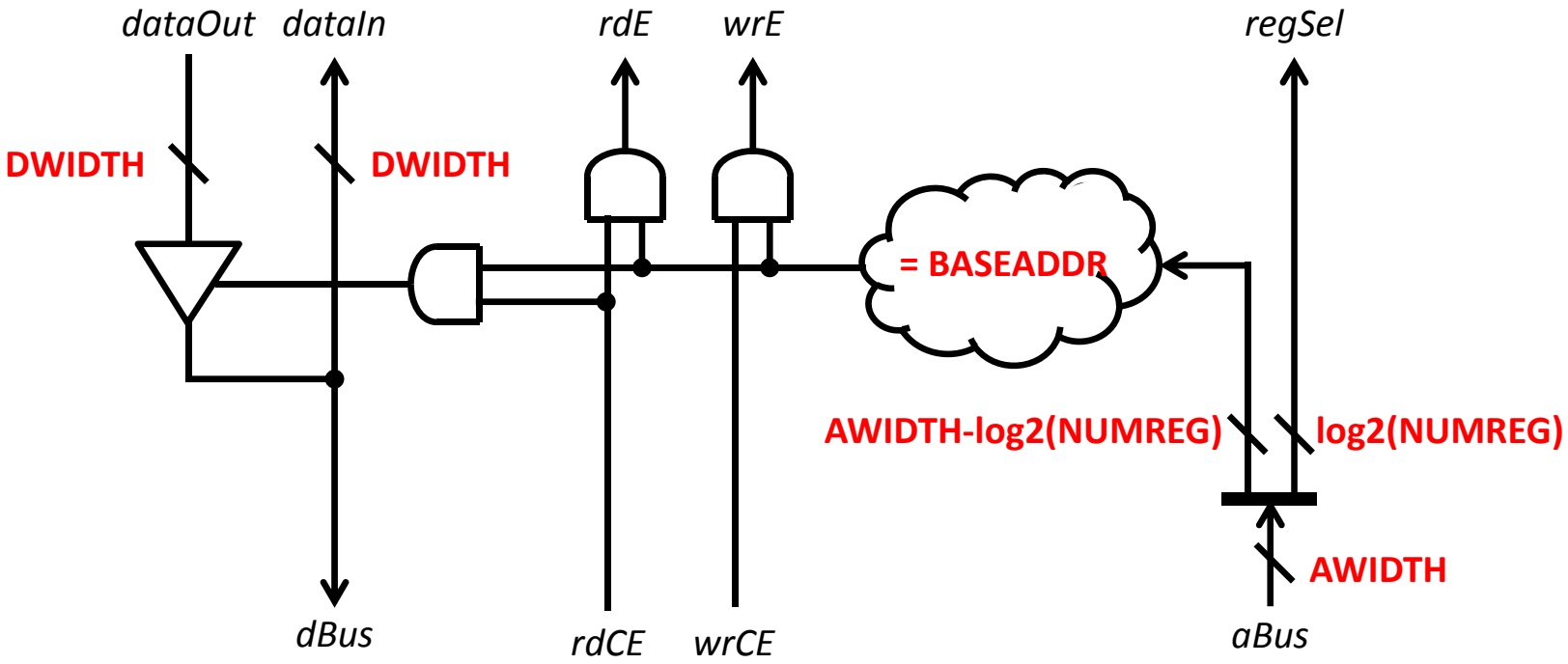
J.M. Mendías
2017

Modelo de interfaz programable



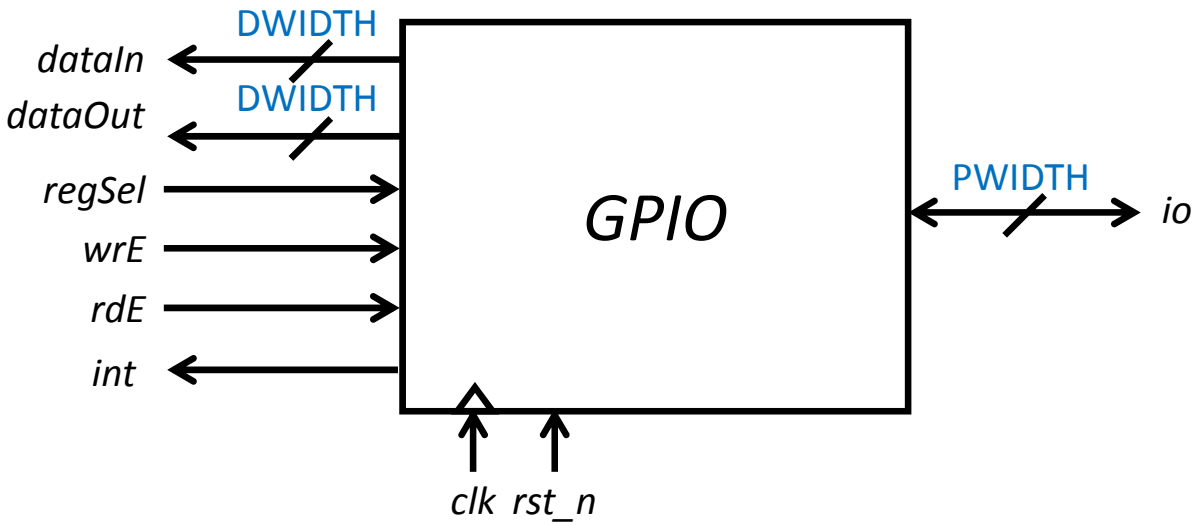
busWrapper
diagrama RTL

- Diseñar un adaptador de bus genérico:
  - Permitirá mapear en direcciones distintas los registros de un interfaz programable
  - Serán configurables:
    - el número de registros del interfaz (NUMREG)
    - la dirección base (BASEADDR) a partir de la cual están ubicados los registros
    - La anchura del bus de datos (DWIDTH) y direcciones (AWIDTH)



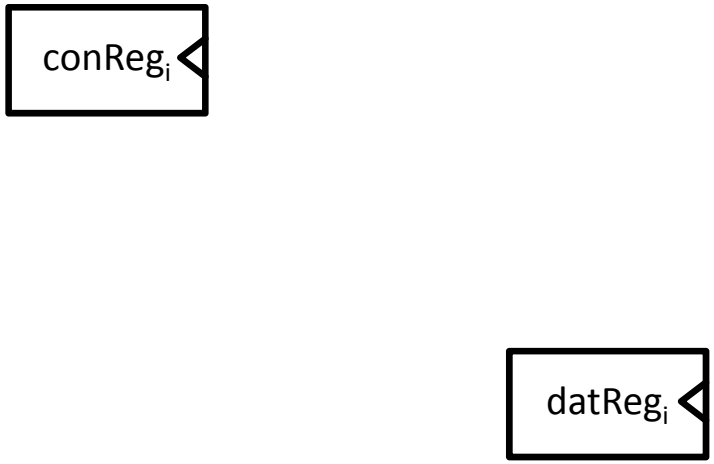
GPIO

- Diseñar un puerto de entrada/salida programable:
  - Dispondrá de un registro de configuración (accesible si regSel vale 1):
    - Cada bit configura la correspondiente línea de io como entrada/salida (1/0)
  - Dispondrá de un registro de datos (accesible si regSel vale 0):
    - Si el puerto está configurado como salida, almacenará el valor escrito por la CPU. Su salida estará conectada al puerto.
    - Si el puerto está configurado como entrada, en todo ciclo cargará el valor presente en el puerto. Su entrada estará conectada al puerto.
  - Adicionalmente dispondrá de una salida de interrupción
    - Se activará durante un ciclo si hay cambio en algún puerto configurado como entrada.



GPIO

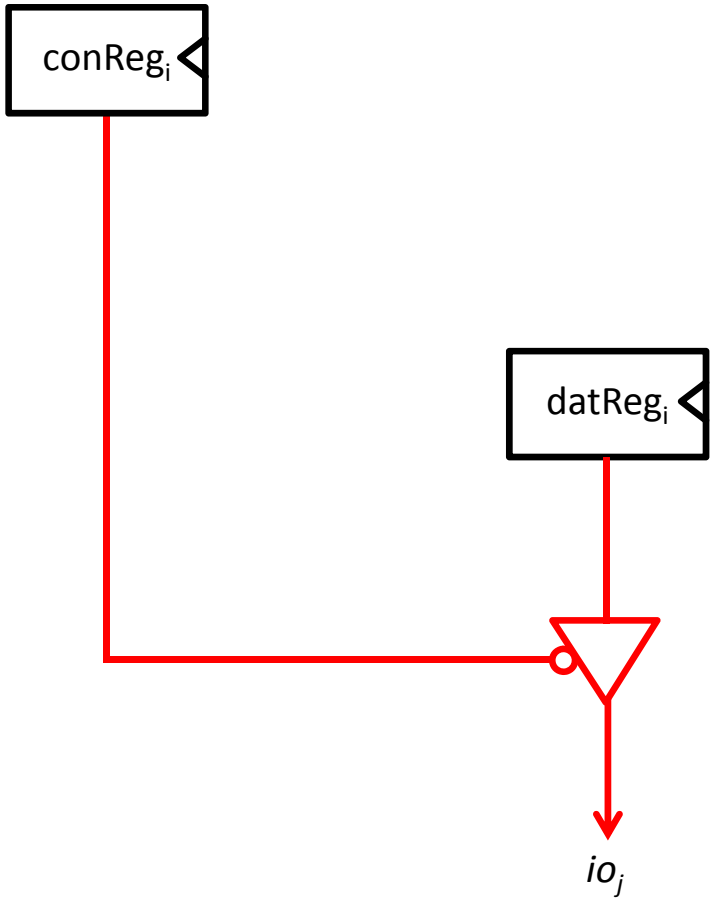
diagrama RTL



J.M. Mendías
2017

GPIO

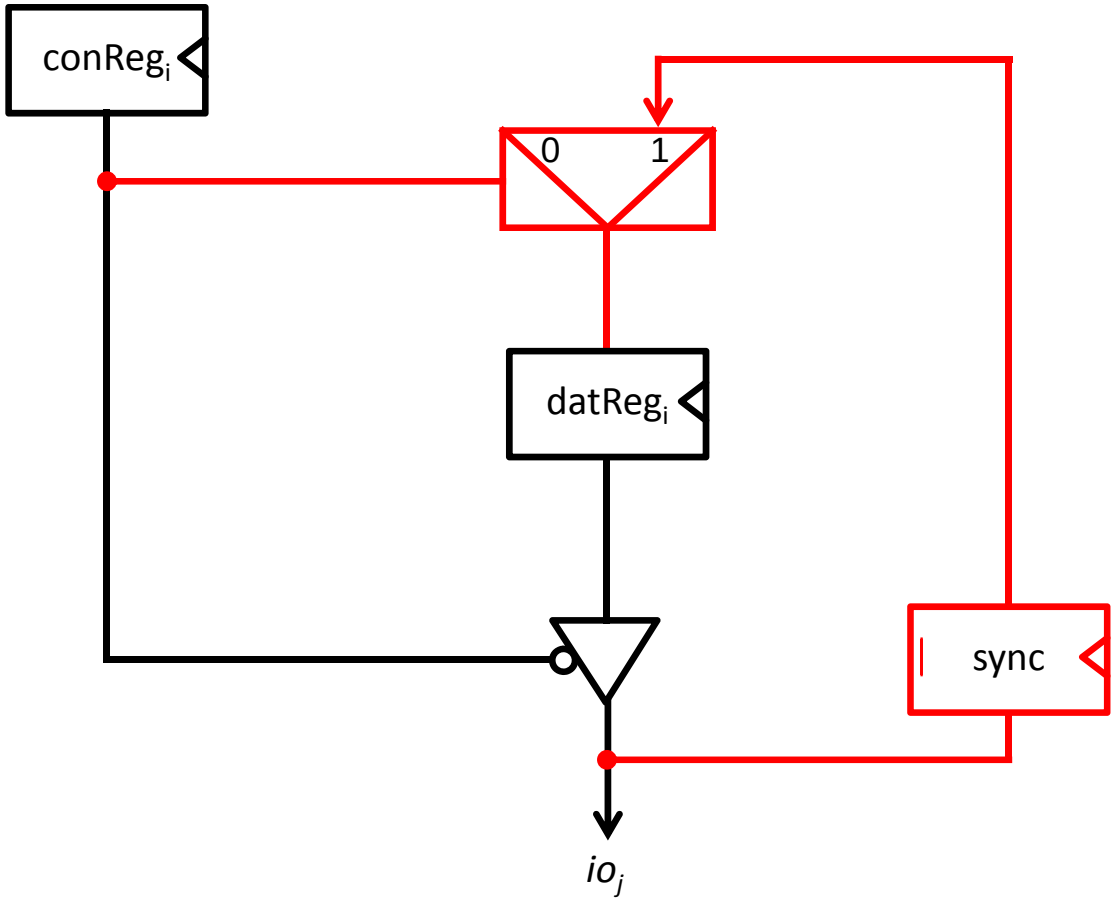
diagrama RTL



J.M. Mendías
2017

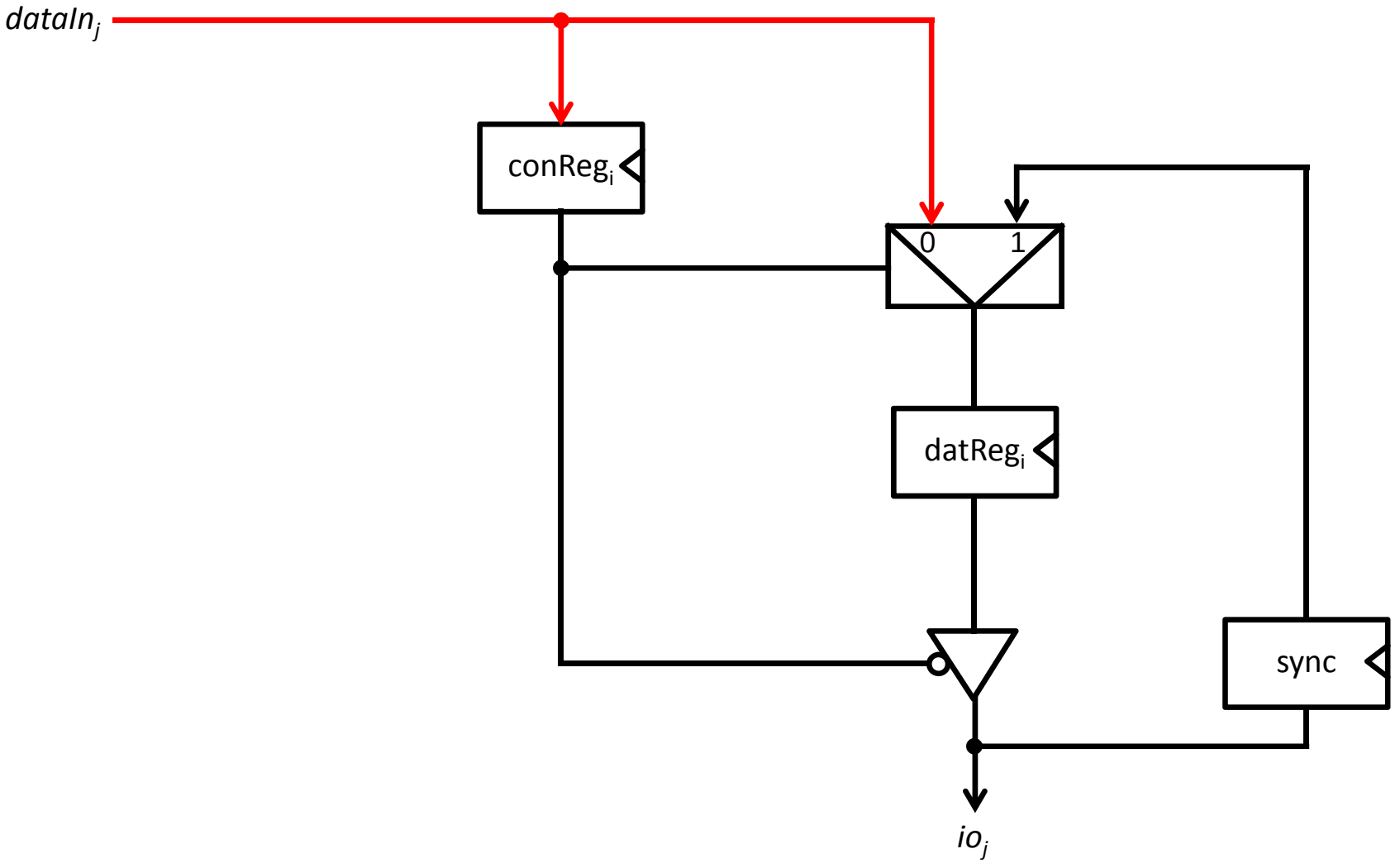
GPIO

diagrama RTL



GPIO

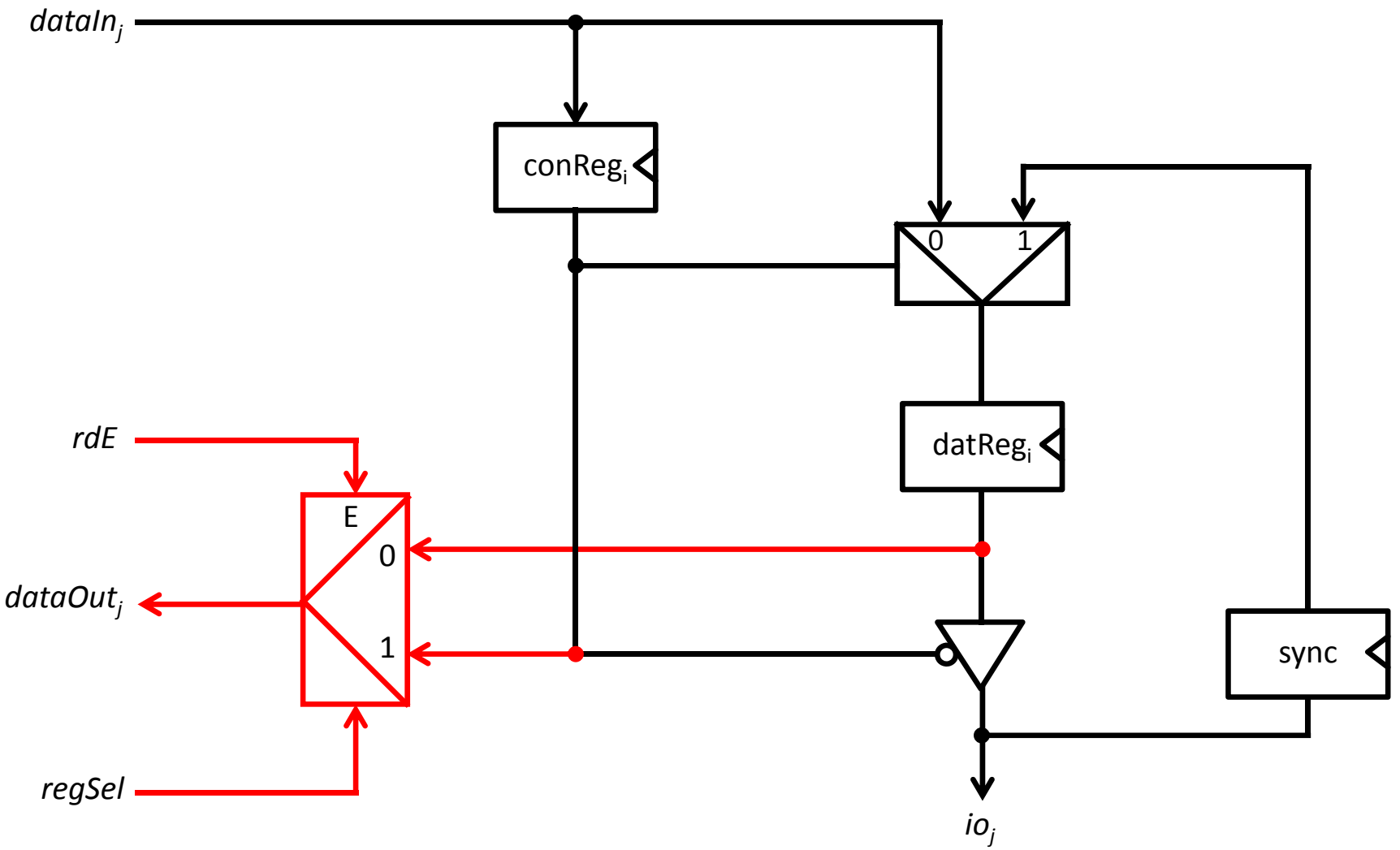
diagrama RTL



J.M. Mendías
2017

GPIO

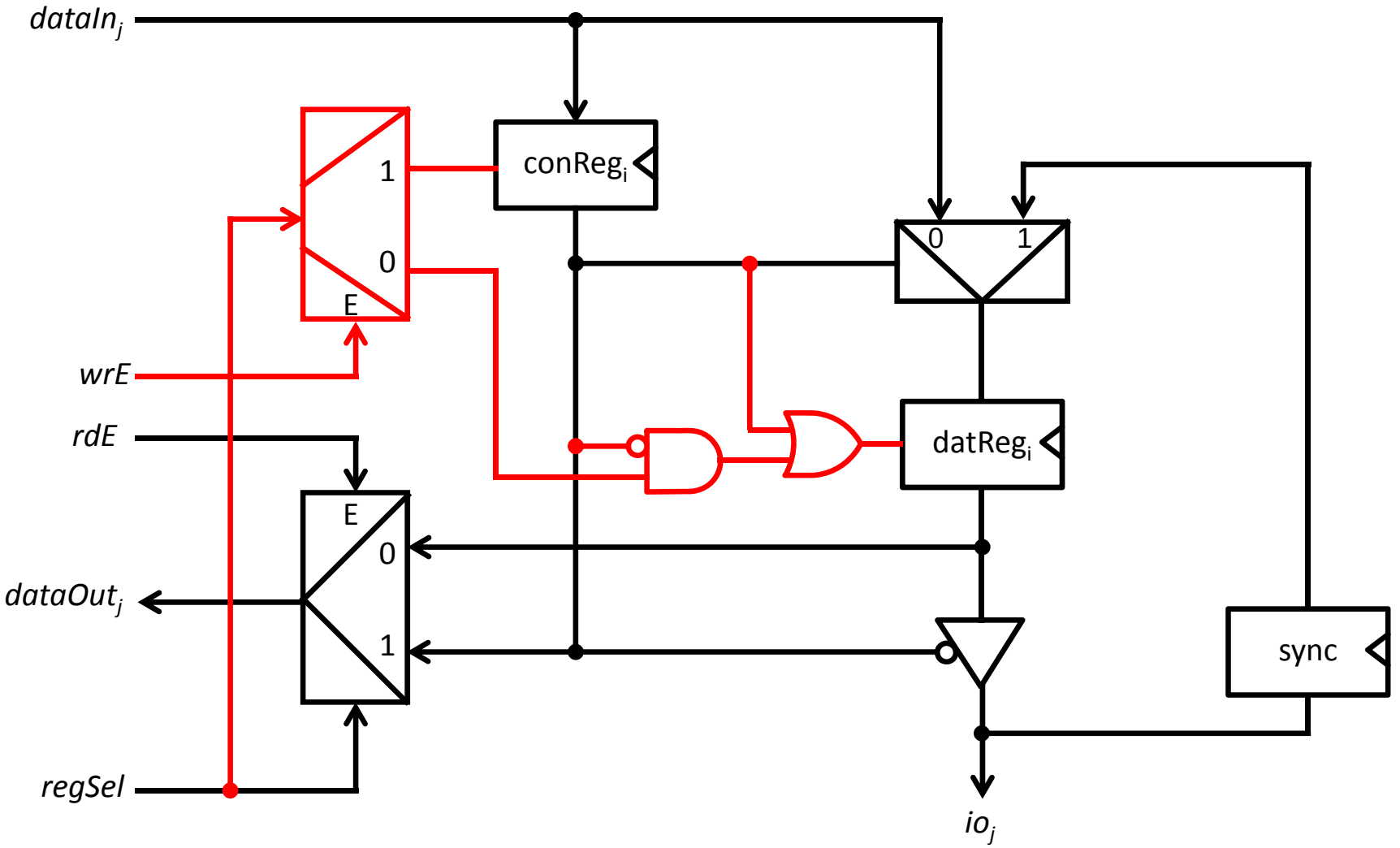
diagrama RTL



J.M. Mendías
2017

GPIO

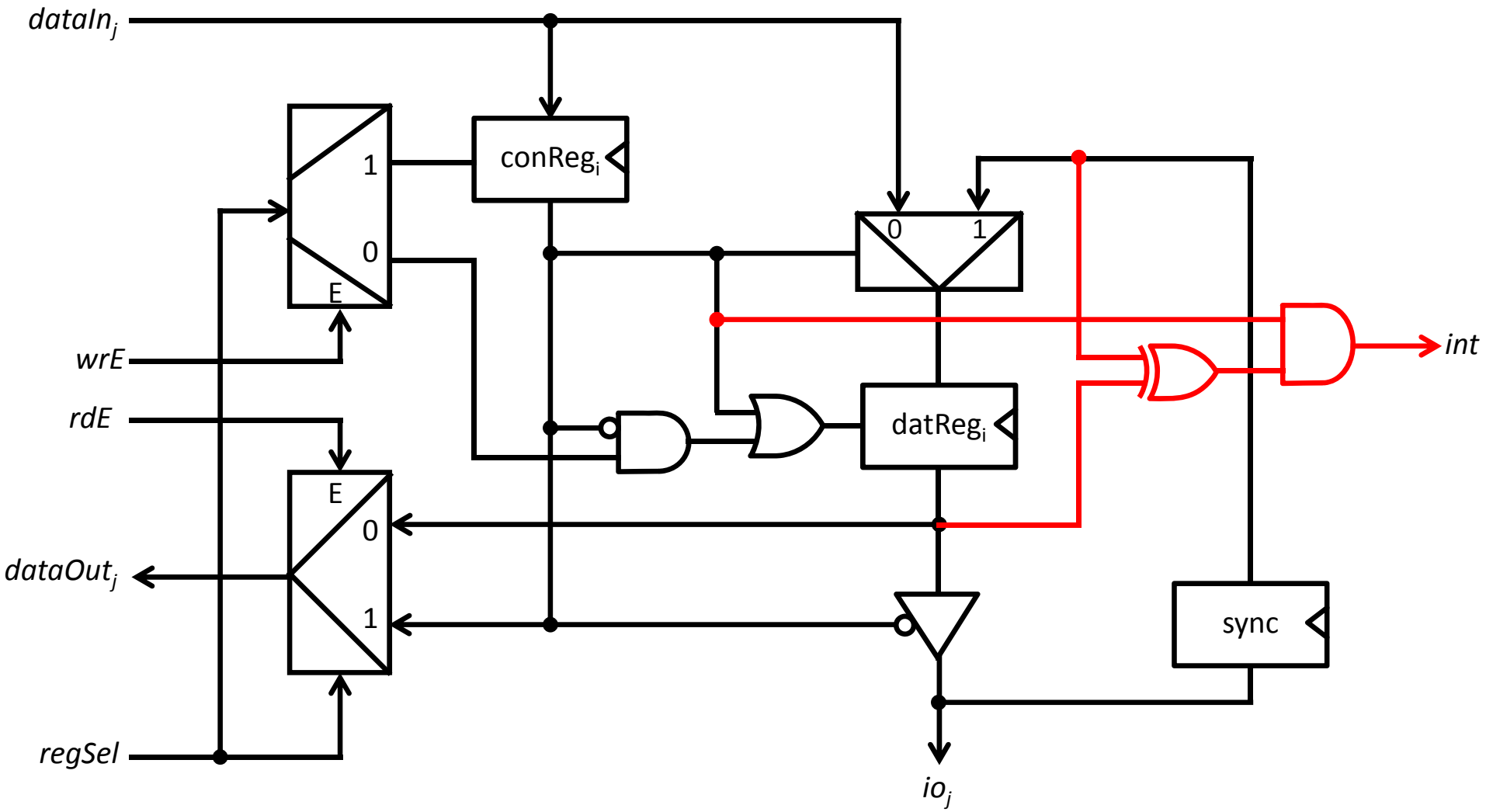
diagrama RTL



J.M. Mendías
2017

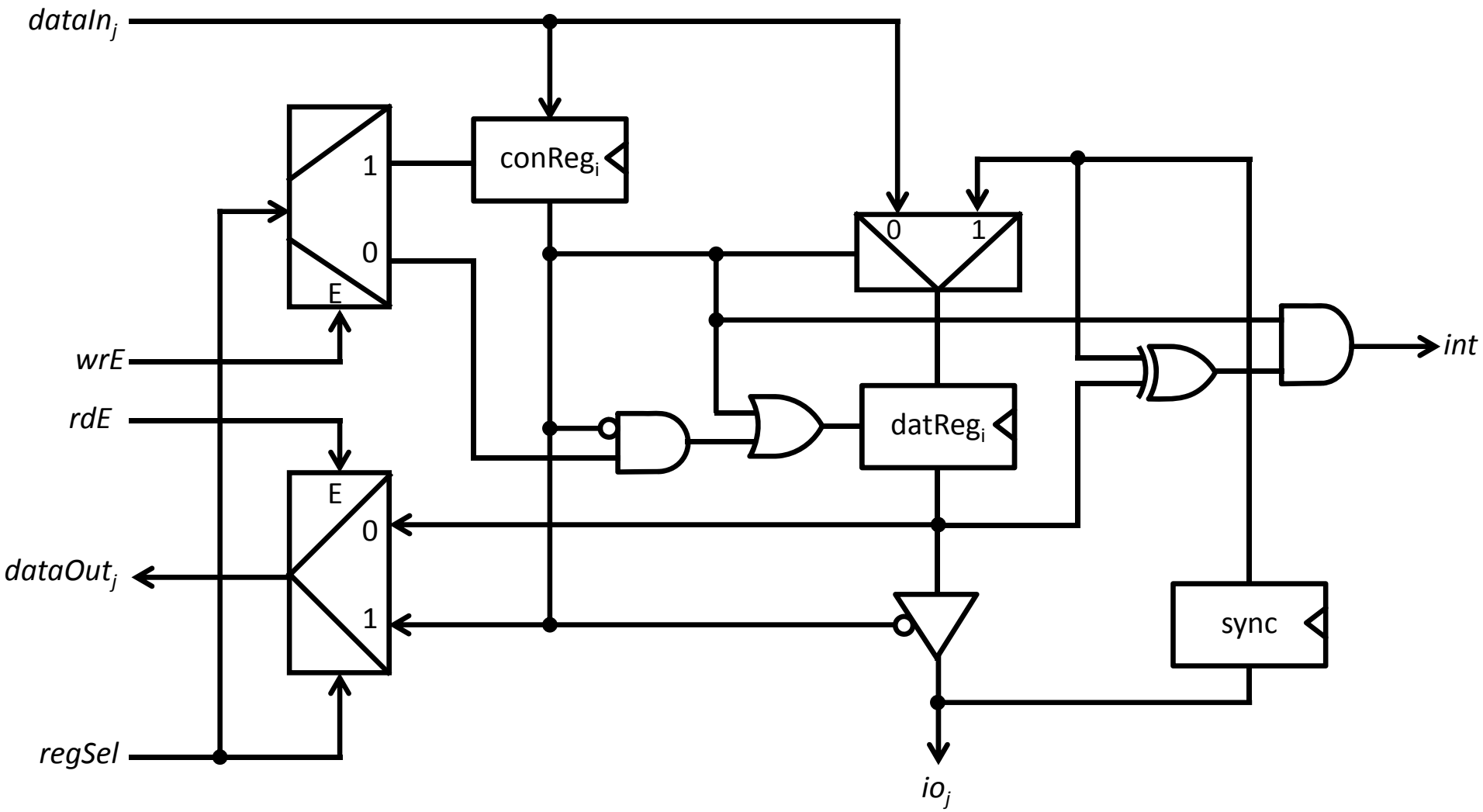
GPIO

diagrama RTL



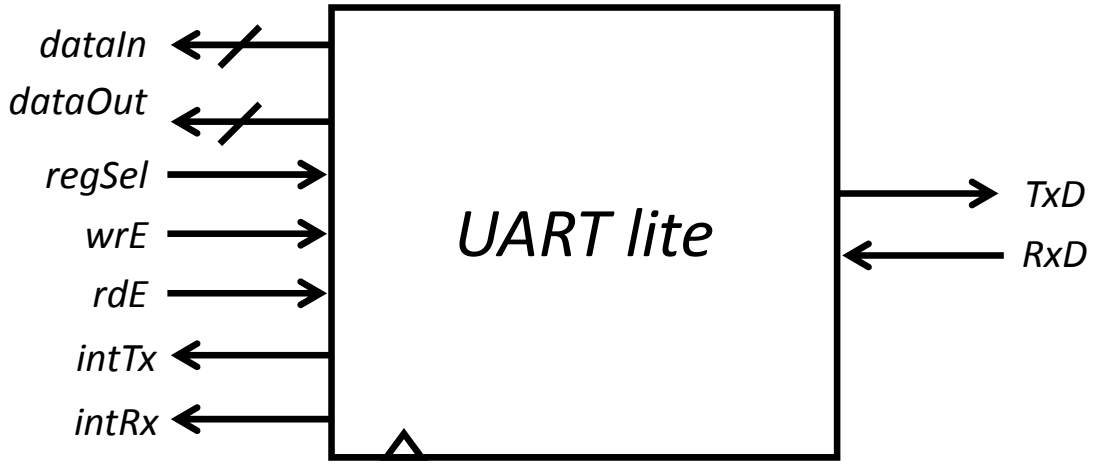
GPIO

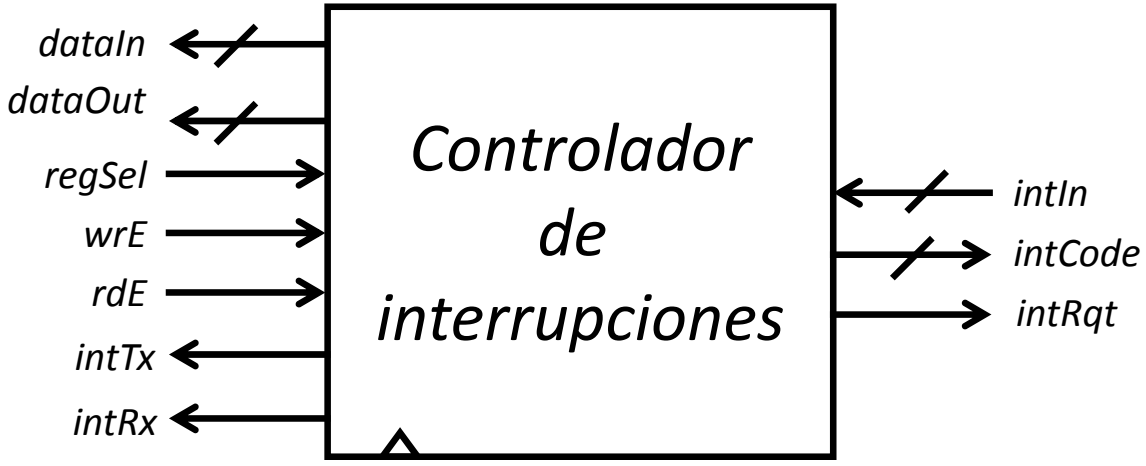
diagrama RTL



Otros interfaces

- Siguiendo el mismo patrón es sencillo diseñar cualquier tipo de interfaz programable partiendo de un controlador de dispositivo





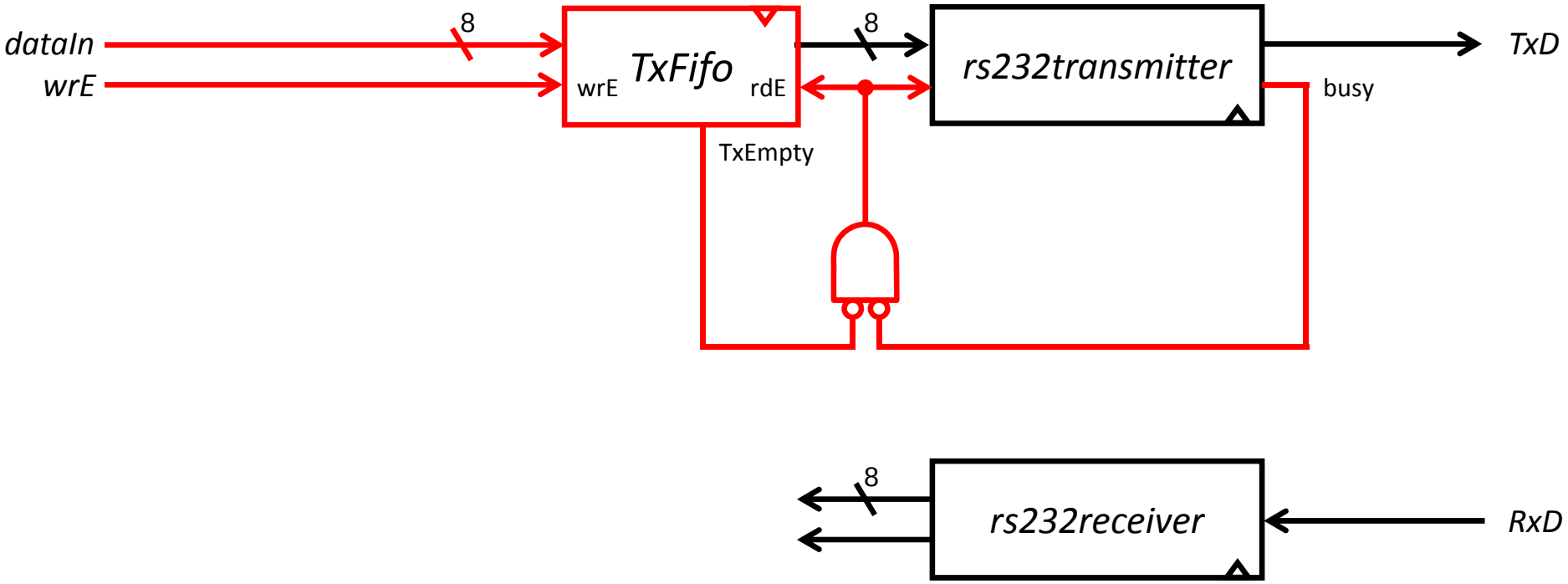
UART lite

diagrama RTL

J.M. Mendías
2017

UART lite

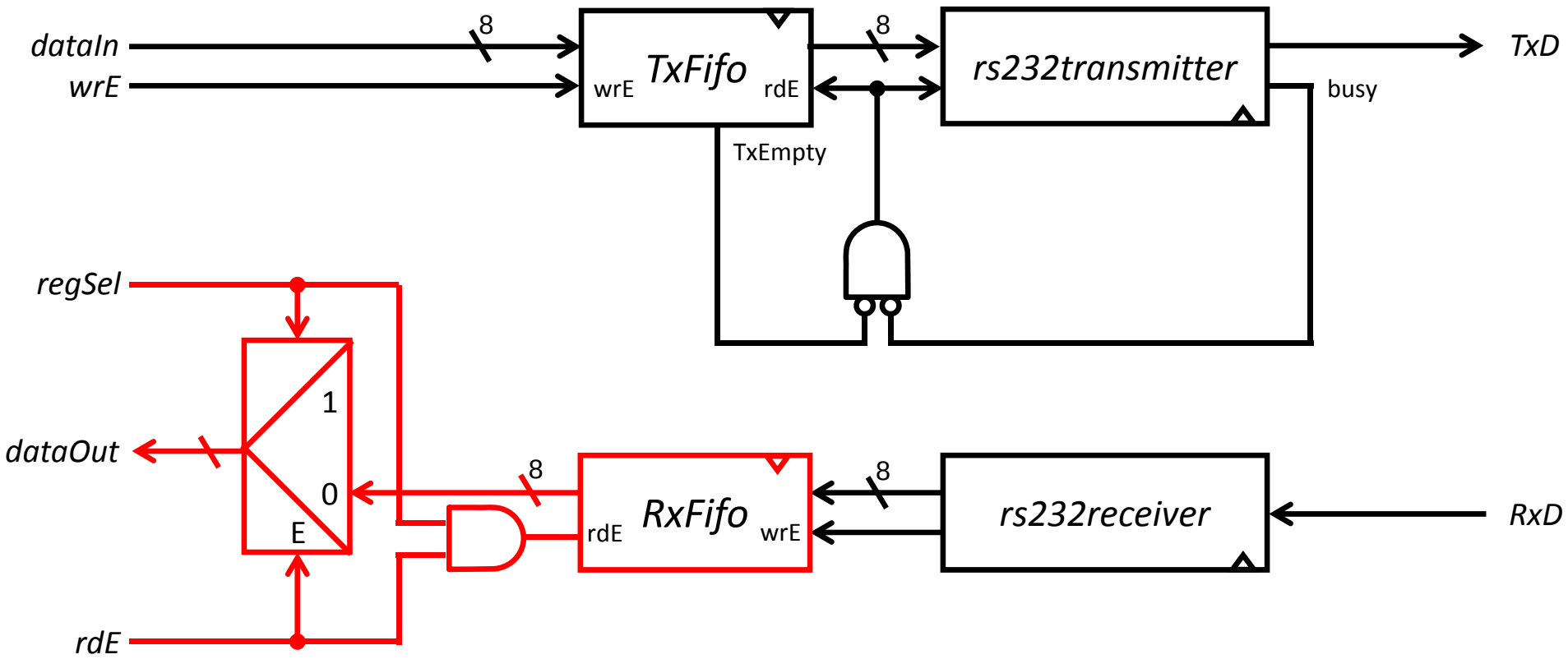
diagrama RTL



J.M. Mendías
2017

UART lite

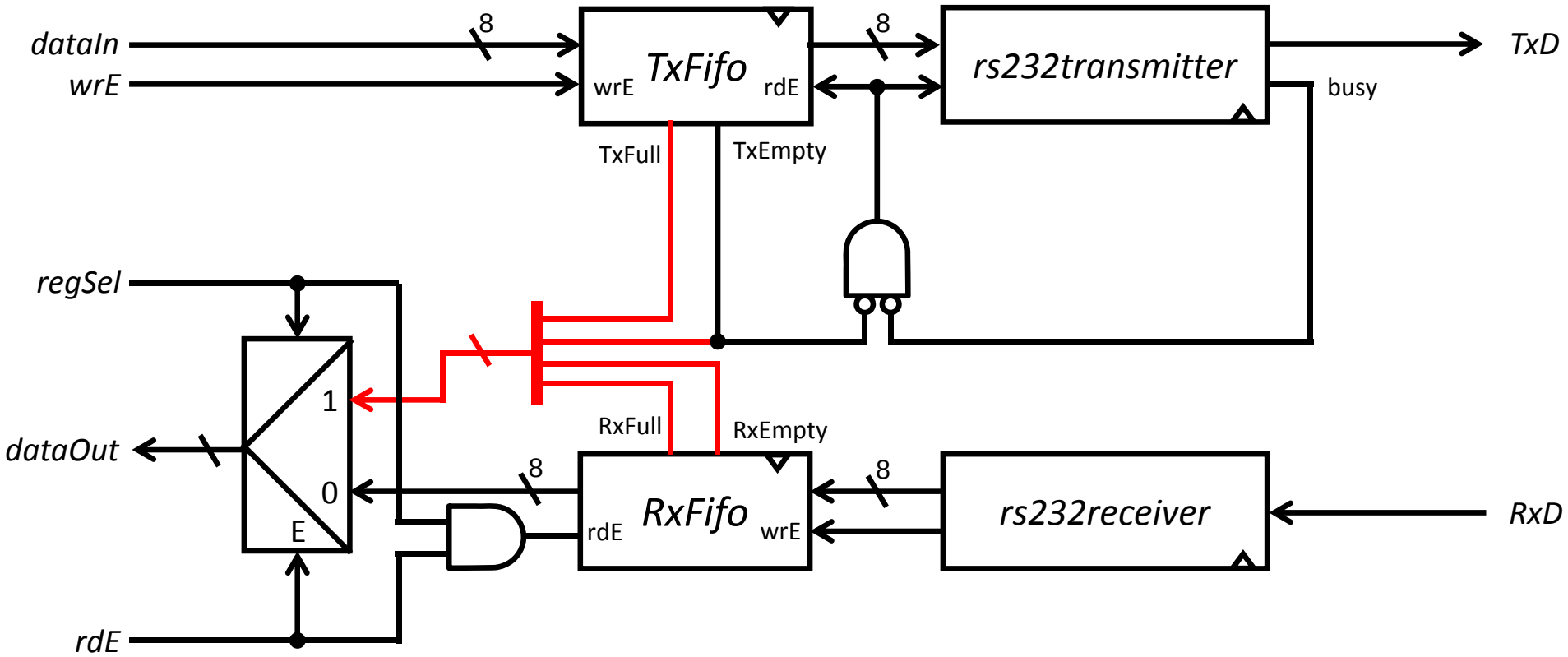
diagrama RTL



J.M. Mendías
2017

UART lite

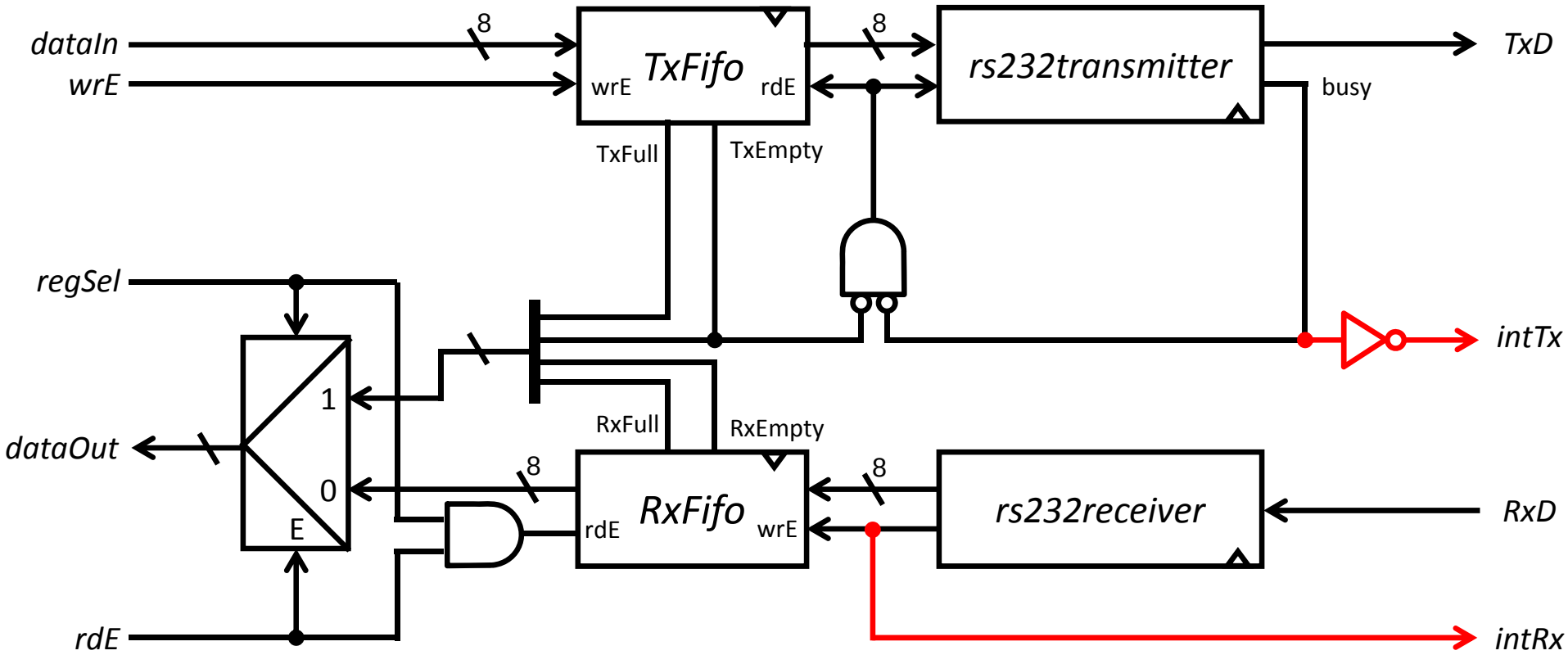
diagrama RTL



J.M. Mendías
2017

UART lite

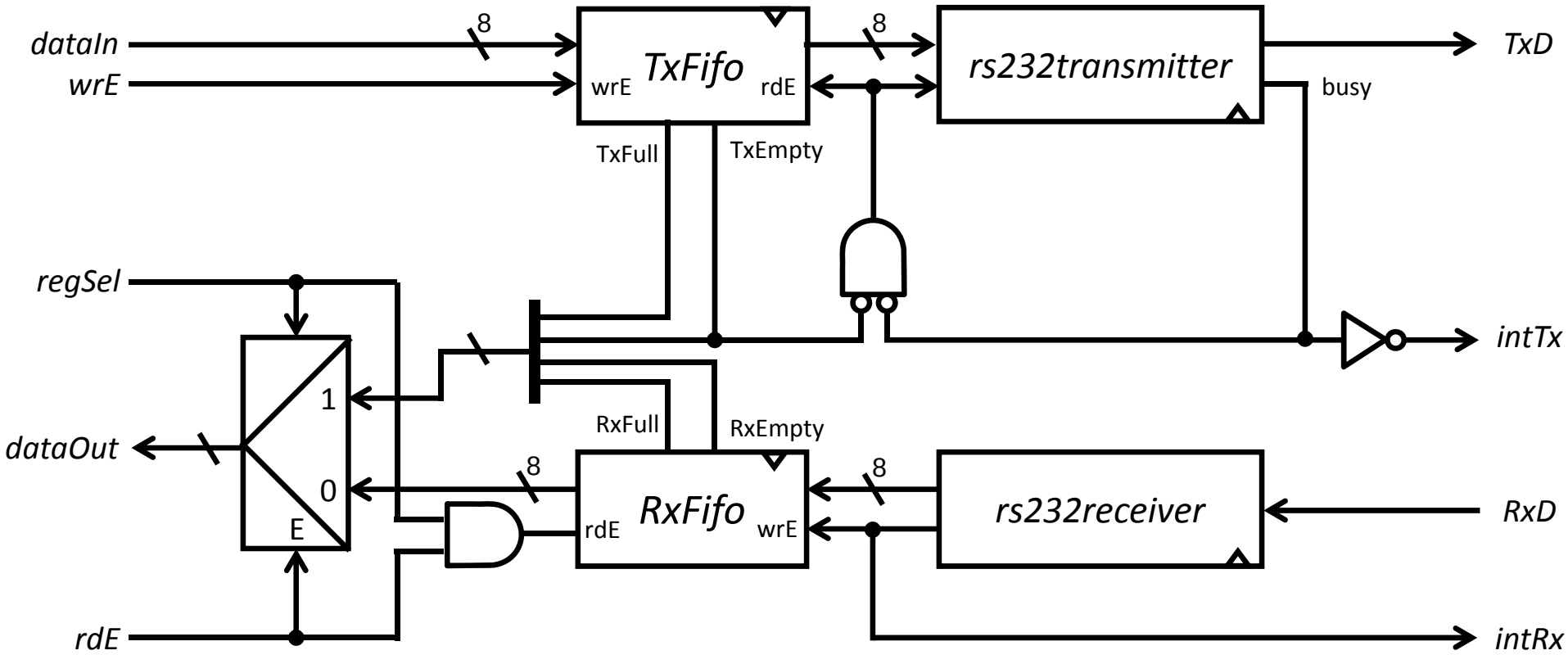
diagrama RTL



J.M. Mendías
2017

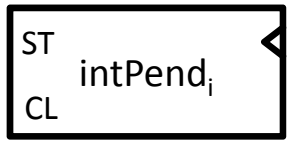
UART lite

diagrama RTL



Controlador de interrupciones

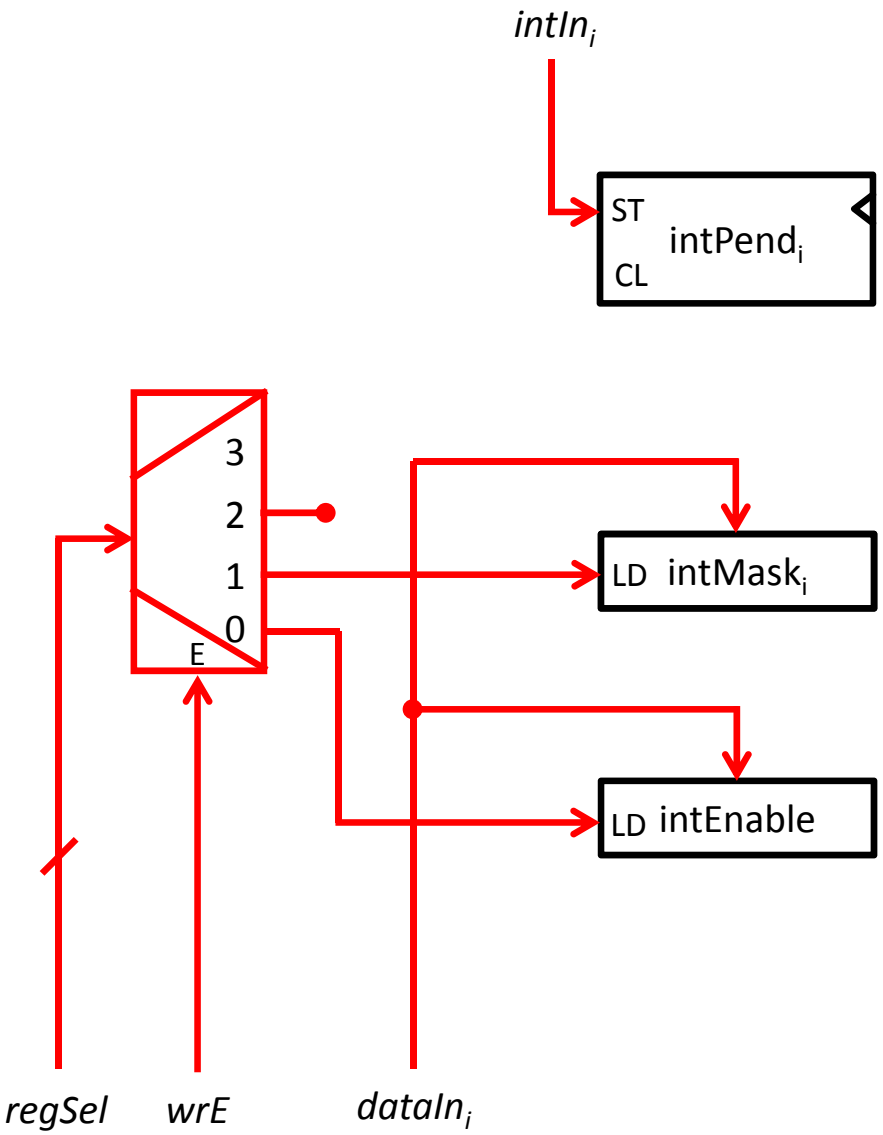
diagrama RTL



J.M. Mendías
2017

Controlador de interrupciones

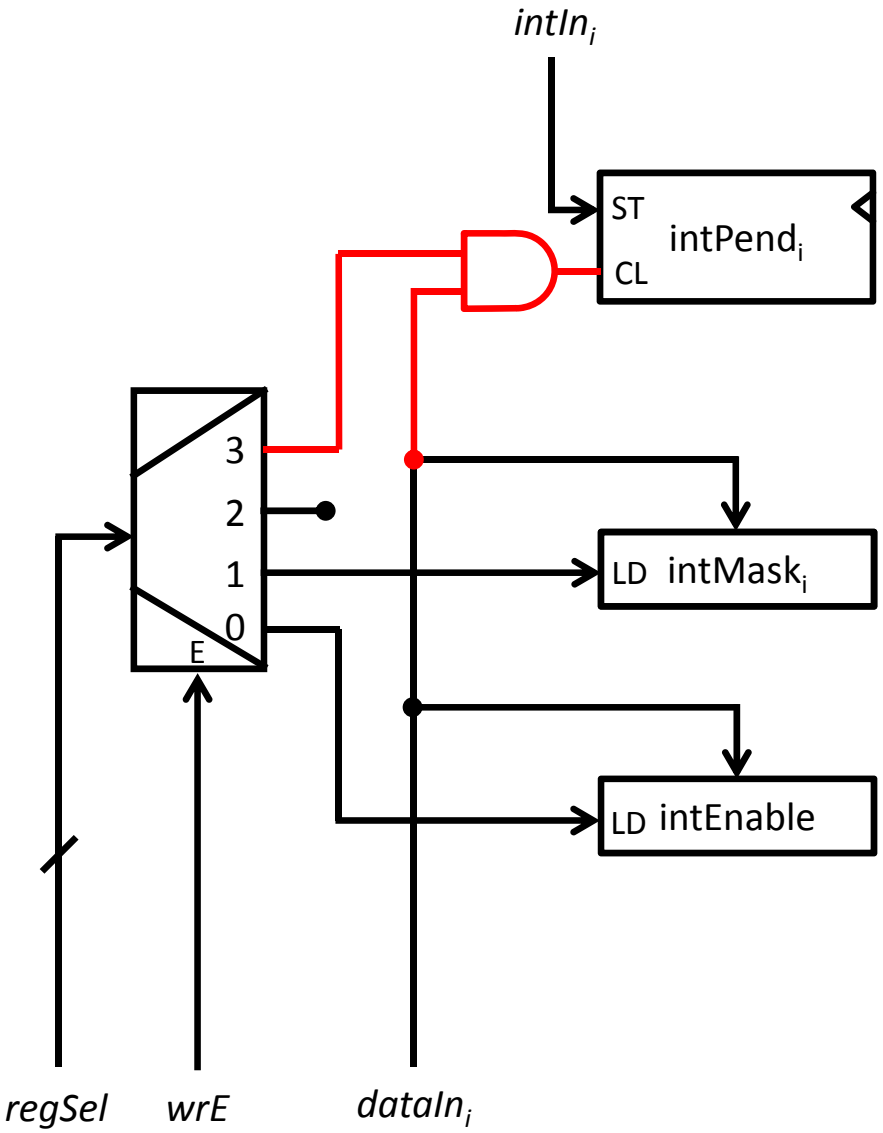
diagrama RTL



J.M. Mendías
2017

Controlador de interrupciones

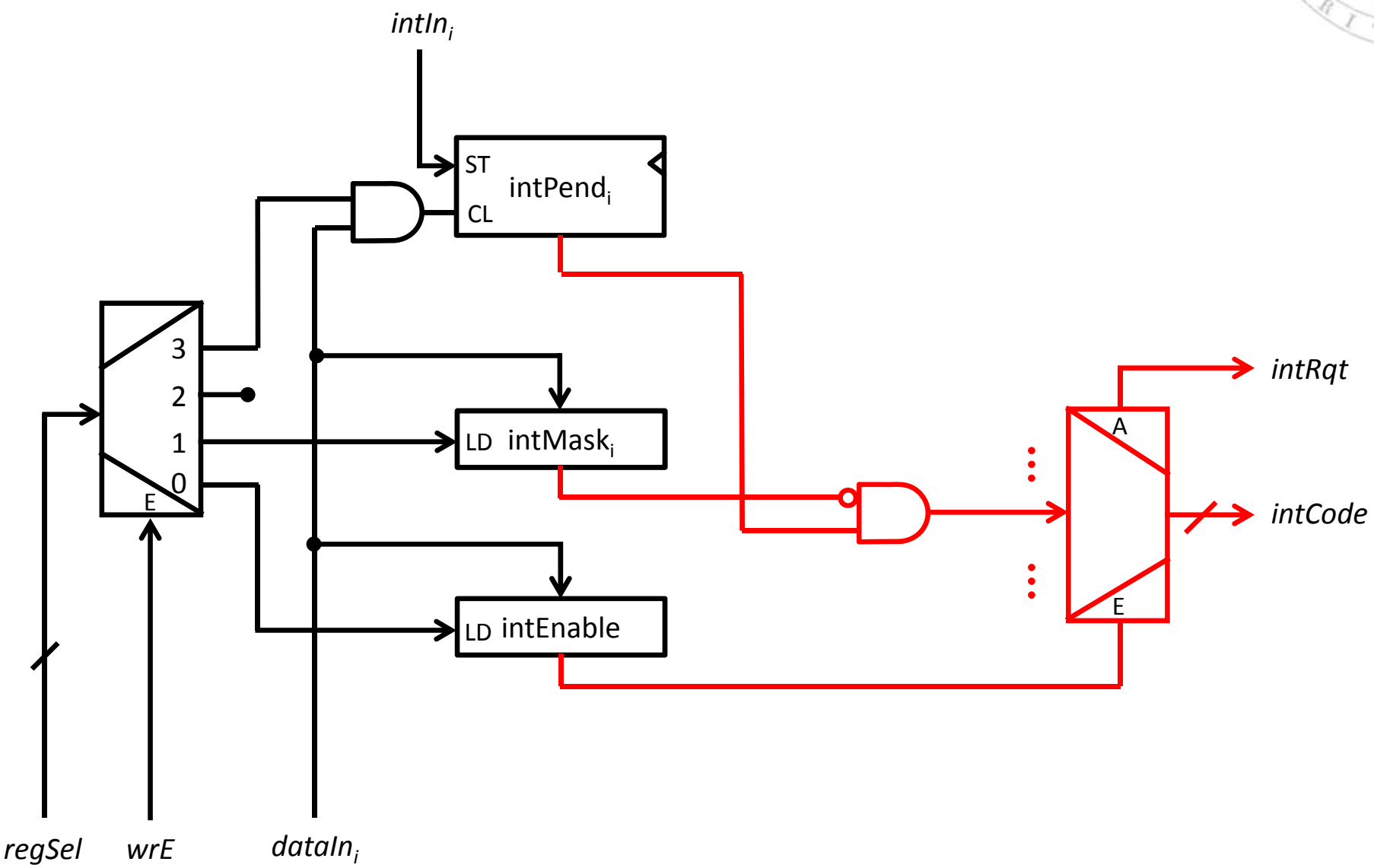
diagrama RTL



J.M. Mendías
2017

Controlador de interrupciones

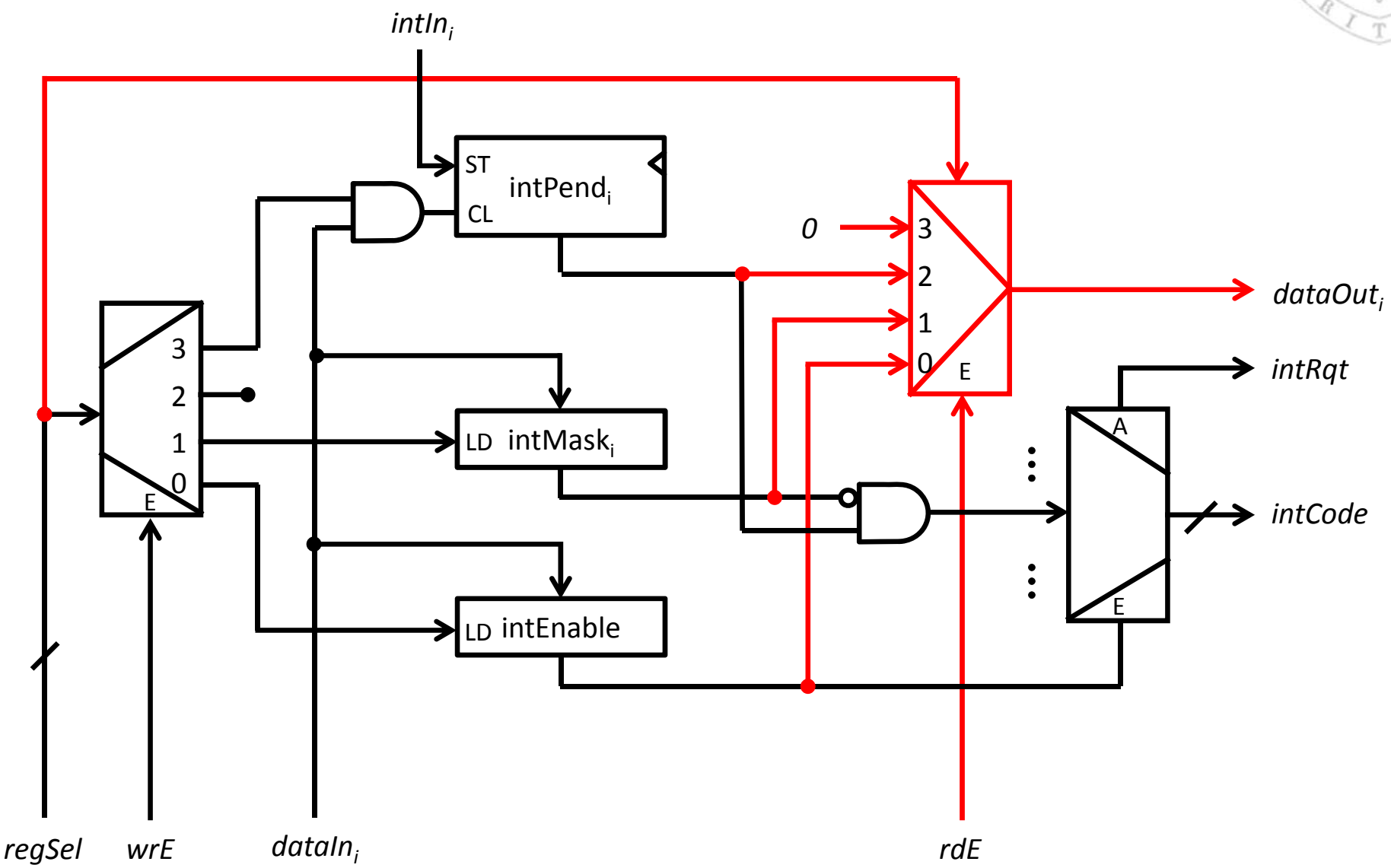
diagrama RTL



J.M. Mendías
2017

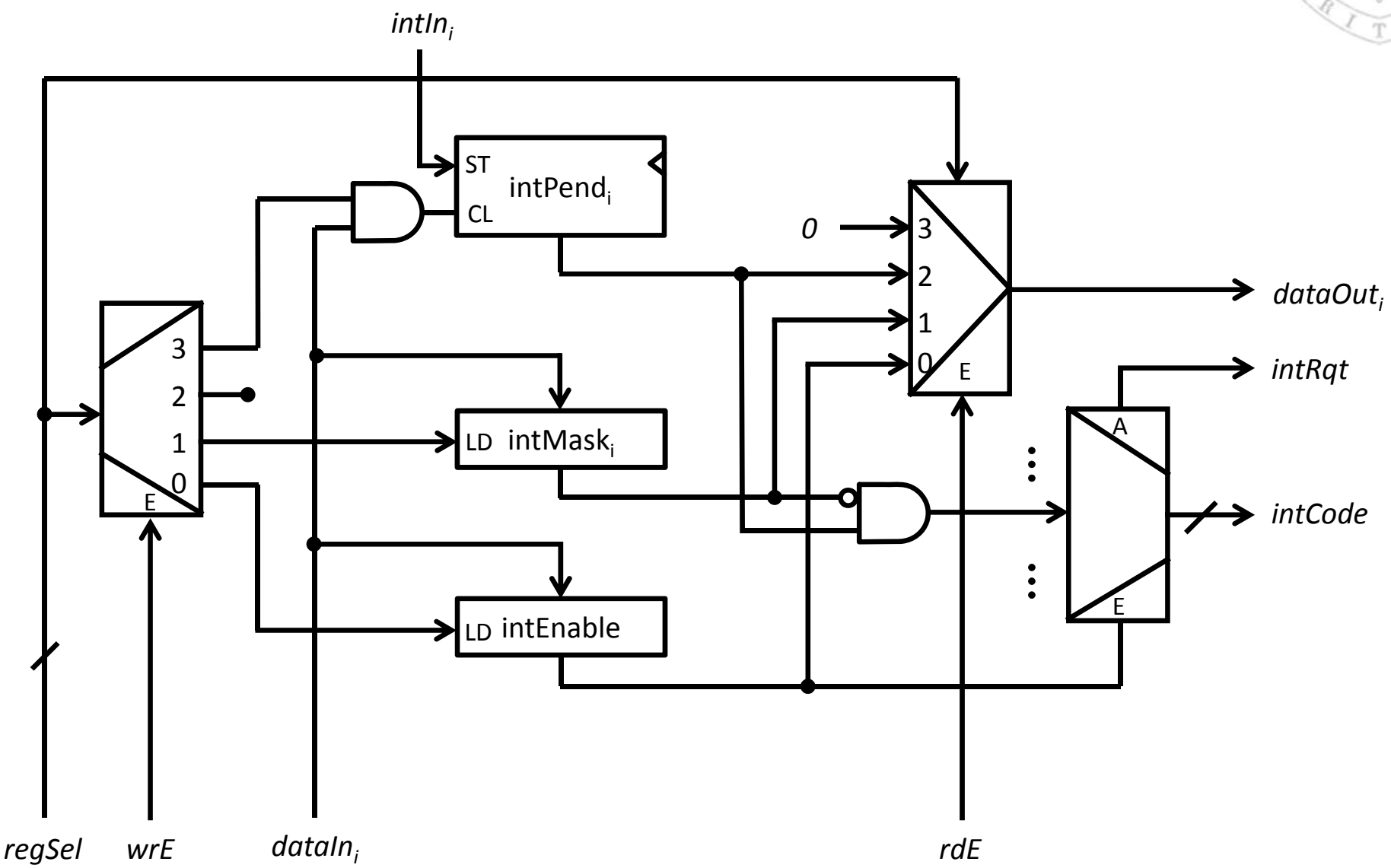
Controlador de interrupciones

diagrama RTL



Controlador de interrupciones

diagrama RTL



Diseño principal

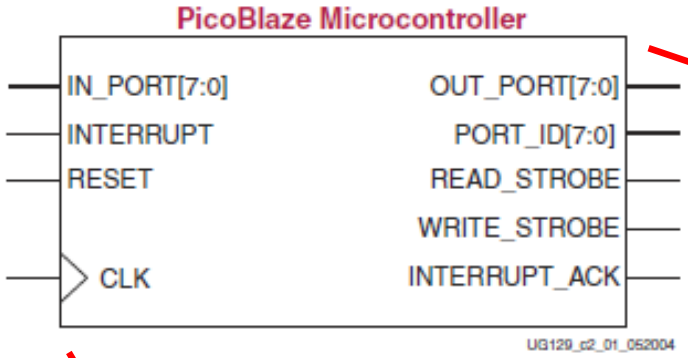
- Para testar el esquema de interconexión conectaremos 3 GPIO con un microcontrolador de 8 bits:
  - Un GPIO estará conectado a los switches y mapeado a partir de la dirección $(fc)_{16}$
  - Un GPIO estará conectado a los leds y mapeado a partir de la dirección $(fe)_{16}$
  - Un GPIO estará conectado a los 7-segs y mapeado a partir de la dirección $(fa)_{16}$

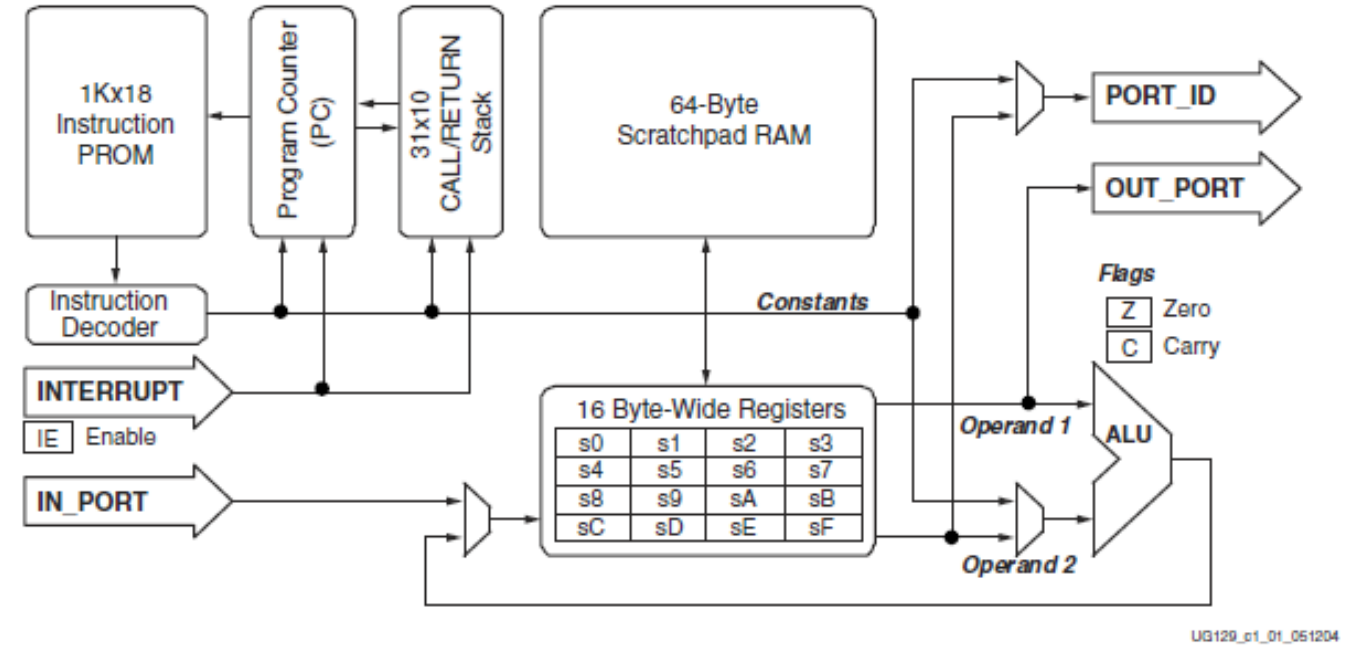
- Como microcontrolador usaremos PicoBlace (versión KCPMS3) de Xilinx.
  - Soft core escrito en VHDL que ocupa 96 slices + 1 blockRAM en una Spartan3
  - Rendimiento predecible: 2 ciclos/instrucción
  - Arquitectura Harvard
  - 16 registros de 8 bits
  - Memoria de datos (Scratchpad RAM) para 64 datos de 8 bits
  - Memoria de programa (PROM) para 1024 instrucciones de 18 bits
  - Pila para 31 direcciones de 10 bits
  - Bus de periféricos de 8 bits con señalización por strobe de 1 ciclo
  - Una única línea de INT/ACK
  - Dispone de ensamblador

J.M. Mendías
2017

Controlador de interrupciones

PicoBlace de Xilinx





J.M. Mendías
2017

Diseño principal

lab10.vhd

```
architecture syn of lab10 is

  ...
begin

 gpioWrapper1 : busWrapper
    generic map ( NUMREG => 2, DWIDTH => 8, AWIDTH => 8, BASEADDR => 16#fe# )
    port map (
      wrCE => wrCE, rdCE => rdCE, aBus => aBus, dBus => dBus,
      regSel => regSel1, wrE => wrE1, dataIn => dataIn1,
      rdE => rdE1, dataOut => dataOut1
    );

  gpio1 : gpio
    generic map ( DWIDTH => 8, PWIDTH => 8 )
    port map (
      rst_n => rst_n, clk => clk,
      regSel => regSel1(0), wrE => wrE1, dataIn => dataIn1,
      rdE => rdE1, dataOut => dataOut1, int => int1,
      io => leds
    );

  ...
```

J.M. Mendías
2017

Diseño principal

lab10.vhd

```
...
 microcontroller: kcpsm3
   port map(
     reset => not rst_n, clk => clk,
     interrupt => interrupt, interrupt_ack => interrupt_ack,
     address => apBus,
     instruction => dpBus,
     port_id => aBus,
     write_strobe => wrCE, out_port => outPort,
     read_strobe => rdCE, in_port => dBus
   );

 dBus <= outPort when wrCE='1' else (others => 'Z');

 programMemory : RAMB16_S18
   generic map (
     INIT_00  => "01FF80016F3FCFFC...",
     ...
   )
   port map (
     clk => clk, en => '1', ssr => '0', we => '0',
     addr => apBus,
     di => X"0000", dip => "00",
     do => dpBus(15 downto 0), dop => dpBus(17 downto 16)
   );

end syn;
```

Diseño principal

lab10.psm

```
CONSTANT GPIODAT1, fe     ;Define las direcciones de dispositivos
CONSTANT GPIOCON1, ff
...
NAMEREG s0, arg           ;Renombra registros
NAMEREG sF, aux
...

ADDRESS 000
main:
  DISABLE INTERRUPT
  CALL gpio_init          ;Inicializa dispositivos
  CALL segs_init
  ENABLE INTERRUPT
  LOAD arg, 00
loop:
  CALL segs_putchar       ;Indefinidamente muestra los numeros 0-f en el display
  CALL delay
  ADD  arg, 01
  AND  arg, 0f
  JUMP loop

isr:
  STORE   aux, 3f         ;Guarda el valor del registro usado
  INPUT   aux, GPIODAT1   ;Lee dato de GPIO-1
  OUTPUT  aux, GPIODAT2   ;Escribe el dato leido en GPIO-2
  FETCH   aux, 3f         ;Restaura el valor del registro usado
  RETURNI ENABLE
...
```

J.M. Mendías
2017

Diseño principal

lab10.psm

```
...
gpio_init:
  LOAD    aux, ff          ;Programa GPIO-1 como entrada
  OUTPUT  aux, GPIOCON1
  LOAD    aux, 00          ;Programa GPIO-2 como salida
  OUTPUT  aux, GPIOCON2
  LOAD    aux, 00          ;Programa GPIO-3 como salida
  OUTPUT  aux, GPIOCON3
  RETURN

segs_init:
  LOAD    aux, 00
  OUTPUT  aux, GPIODAT3    ;Apaga el display
  LOAD    aux, 7e          ;Crea en la scratchad una tabla de codigos 7-segmentos
  STORE   aux, 00
  LOAD    aux, 30
  STORE   aux, 01
  ...
  RETURN

segs_putchar:
  FETCH   aux, (arg)       ;Visualiza un numero en el display indexando la tabla
  OUTPUT  aux, GPIODAT3
  RETURN

ADDRESS 3ff                ;Instala la ISR en la tabla del vector de interrupcion
  JUMP isr
```

Tareas

1. Crear el proyecto **`lab10`** en el directorio **`DAS`**
2. Descargar de la Web en el directorio **`common`** los ficheros
   - **`busWrapper.vhd`**, **`kcpsm3.vhd`** y **`gpio.vhd`**
3. Descargar de la Web en el directorio **`lab10`** los ficheros:
   - **`lab10.vhd`** y **`lab10.ucf`**
4. Completar el fichero **`common.vhd`** con la declaración de los nuevos componentes reusables.
5. Completar el código omitido en el ficheros:
   - **`busWrapper.vhd`** y **`gpio.vhd`**
6. Añadir al proyecto los ficheros:
   - **`common.vhd`**, **`synchronizer.vhd`**, **`busWrapper.vhd`**, **`kcpsm3.vhd`**, **`gpio.vhd`**, **`lab10.vhd`** y **`lab10.ucf`**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Conectar la placa y encenderla.
9. Descargar el fichero **`lab10.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/