Proyecto

Introducción a la Microfabricación y las FPGA

Instituto Balseiro

8 de Septiembre 2014

La clase pasada

- Finite State Machine con Datapath. Combina una FSM y uno o varios circuitos regulares secuenciales.
  - *FSM o Control-path*: examina las entradas y el estado, y genera las señales de control para especificar las operaciones de los circuitos regulares secuenciales.
  - *Data path*: circuitos secuenciales regulares que hacen operaciones entre registros.
- Se usa para implementar sistemas descritos con la metodología de Register Transfer. Manipulaciones de datos entre registros.

Ejercicio: Filtro

Vimos como ejercicio el diseño e implementación de un filtro. Vimos:

- La herramienta *fdatool* de matlab para crear el filtro (y tener los coeficientes).
- Aritmética de punto fijo. Multiplicación y Suma. Precisión necesaria.
- Un poco de diseño: full paralelo o full secuencial.
- Cómo testear: testbench con lectura de archivos.
- Tarea: implementar un filtro con POCAS constantes (tres por ejemplo).

Hoy

- Algunos tips para el diseño del filtro.
- Comunicación UART.
- Lectura de módulo del micrófono.

Package y array

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;

package filtro_pkg is

   constant Bits_cte_t : integer := 23;
   constant Bits_x_t : integer := 8;

   type x_t is array(integer range <>) of std_logic_vector(Bits_x_t-1 downto 0);
   type cte_t is array(integer range <> ) of std_logic_vector(Bits_cte_t-1 downto 0);

end filtro_pkg;
```

Entidad

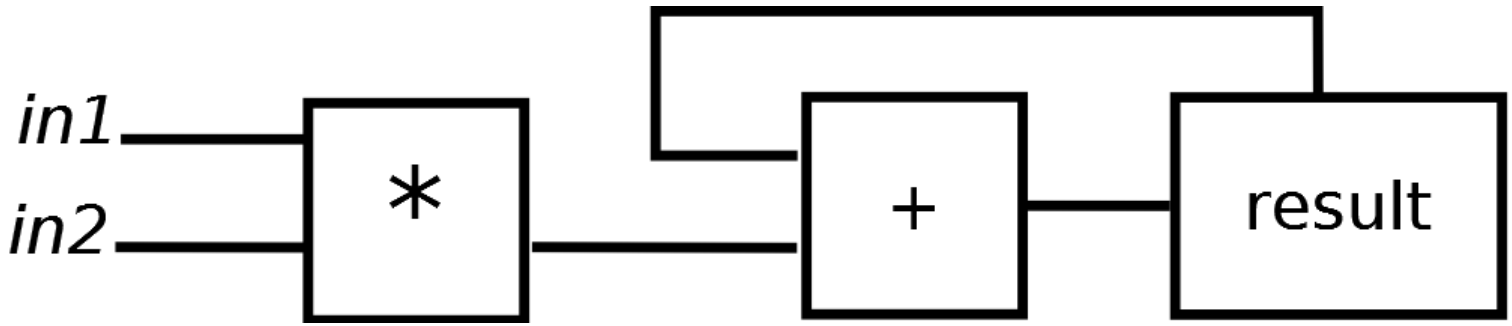
```
entity filtro_fsmd is
   generic (N: natural := 3;
            N_idx : natural := 2
      );
   port(
      clk, reset : std_logic;
      -- entradas del filtro: a son datos, cte son las ctes del filtro.
      a   : in x_t(0 to N);
      cte : in cte_t(0 to N);
      start : in std_logic;
      -- salida
      result : out std_logic_vector(Bits_x_t-1 downto 0);
      done : out std_logic
   );
end filtro_fsmd;
```

Data path

- Una vez que pensamos la entidad, pensar primero en el data path : circuitos secuenciales regulares que hacen las operaciones entre los registros.

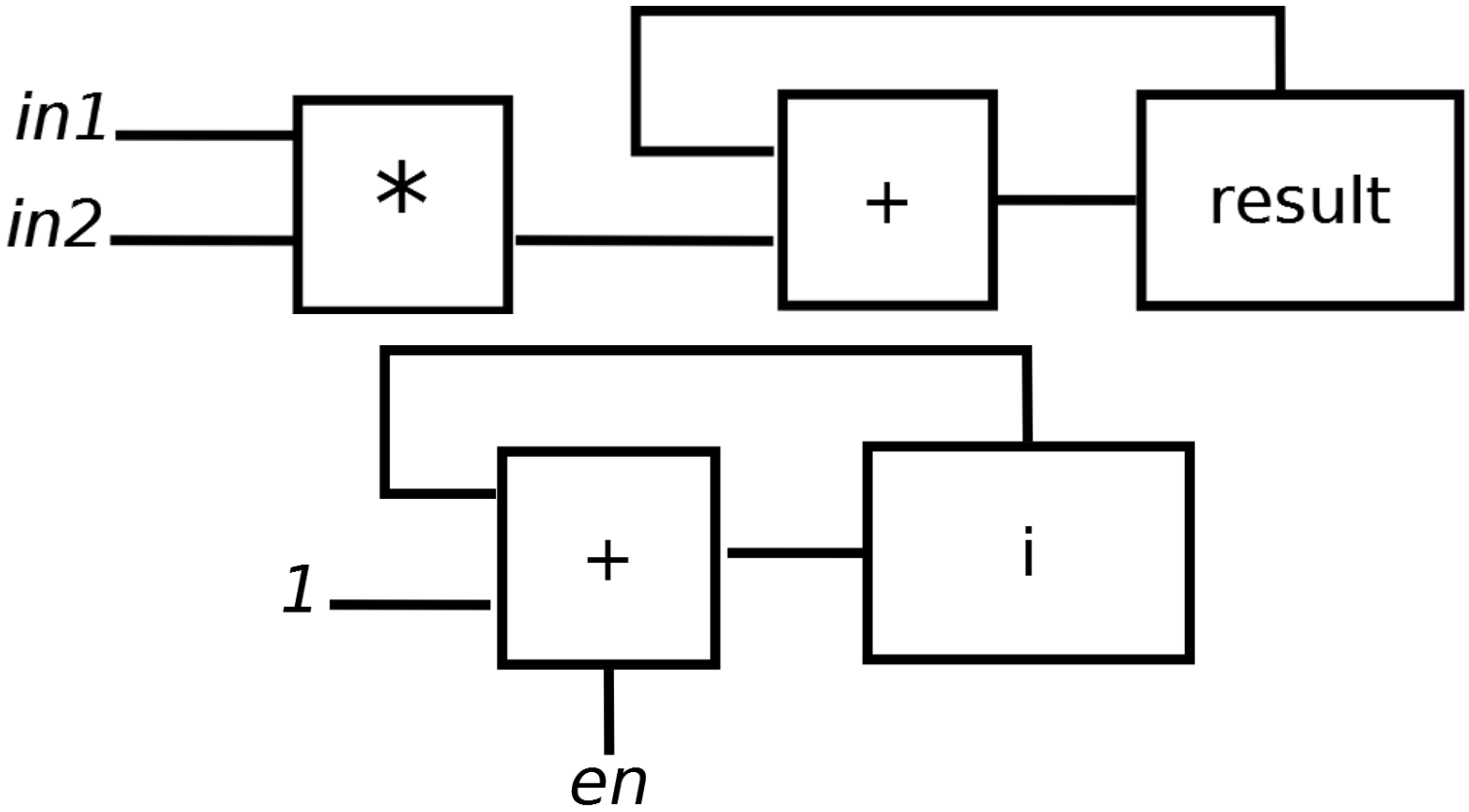
Data path

- Una vez que pensamos la entidad, pensar primero en el data path : circuitos secuenciales regulares que hacen las operaciones entre los registros.



Data path

- Una vez que pensamos la entidad, pensar primero en el data path : circuitos secuenciales regulares que hacen las operaciones entre los registros.



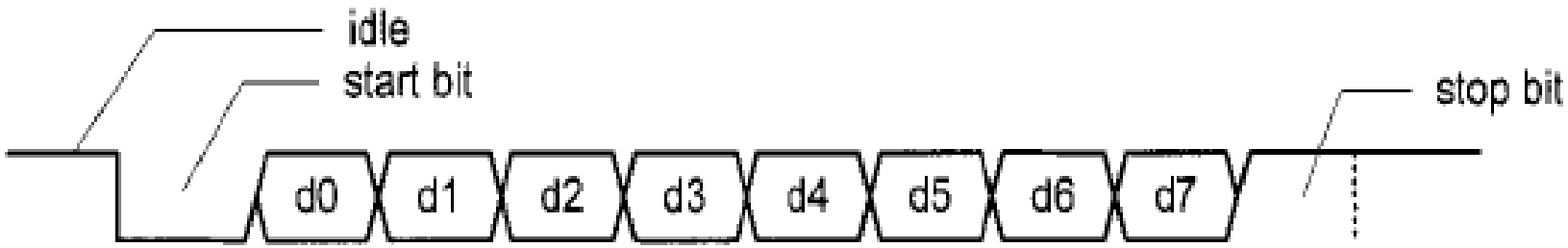
Control path

- Ahora si, pensar entradas y salidas del control path: máquina de estados que controla las operaciones entre los registros.

Control path

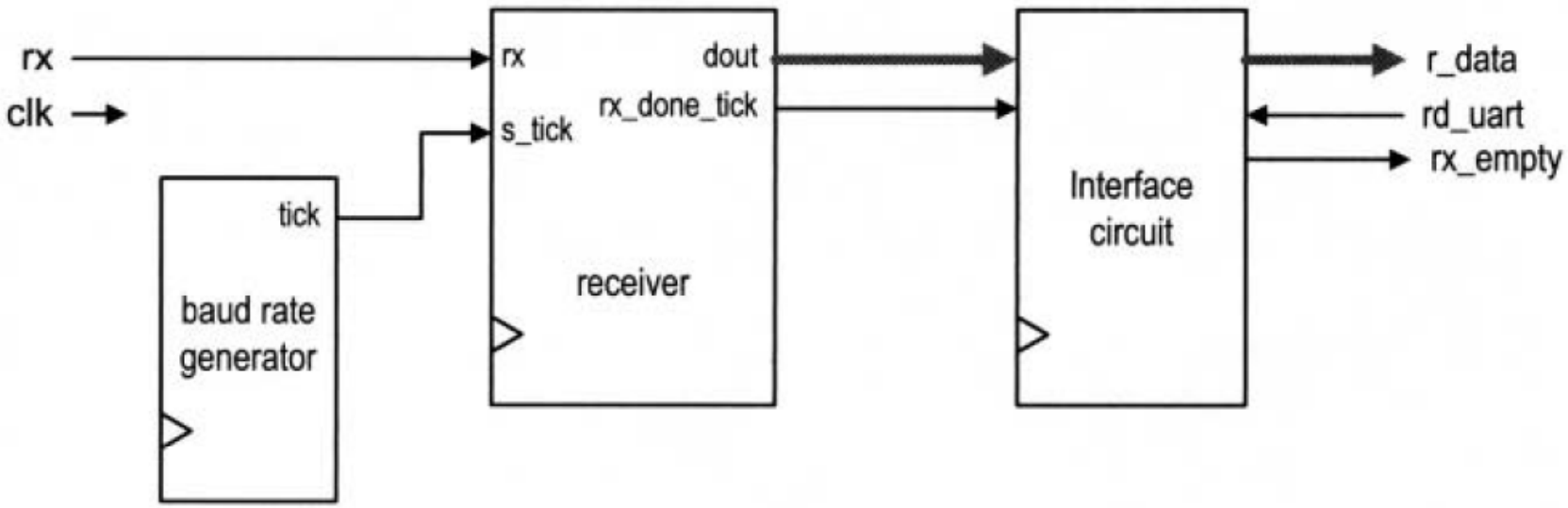
- Ahora si, pensar entradas y salidas del control path: máquina de estados que controla las operaciones entre los registros.
- Entradas: `start`, `max_tick`, `i`.
- Salidas: `in1`, `in2`, `en`, `done`.

UART



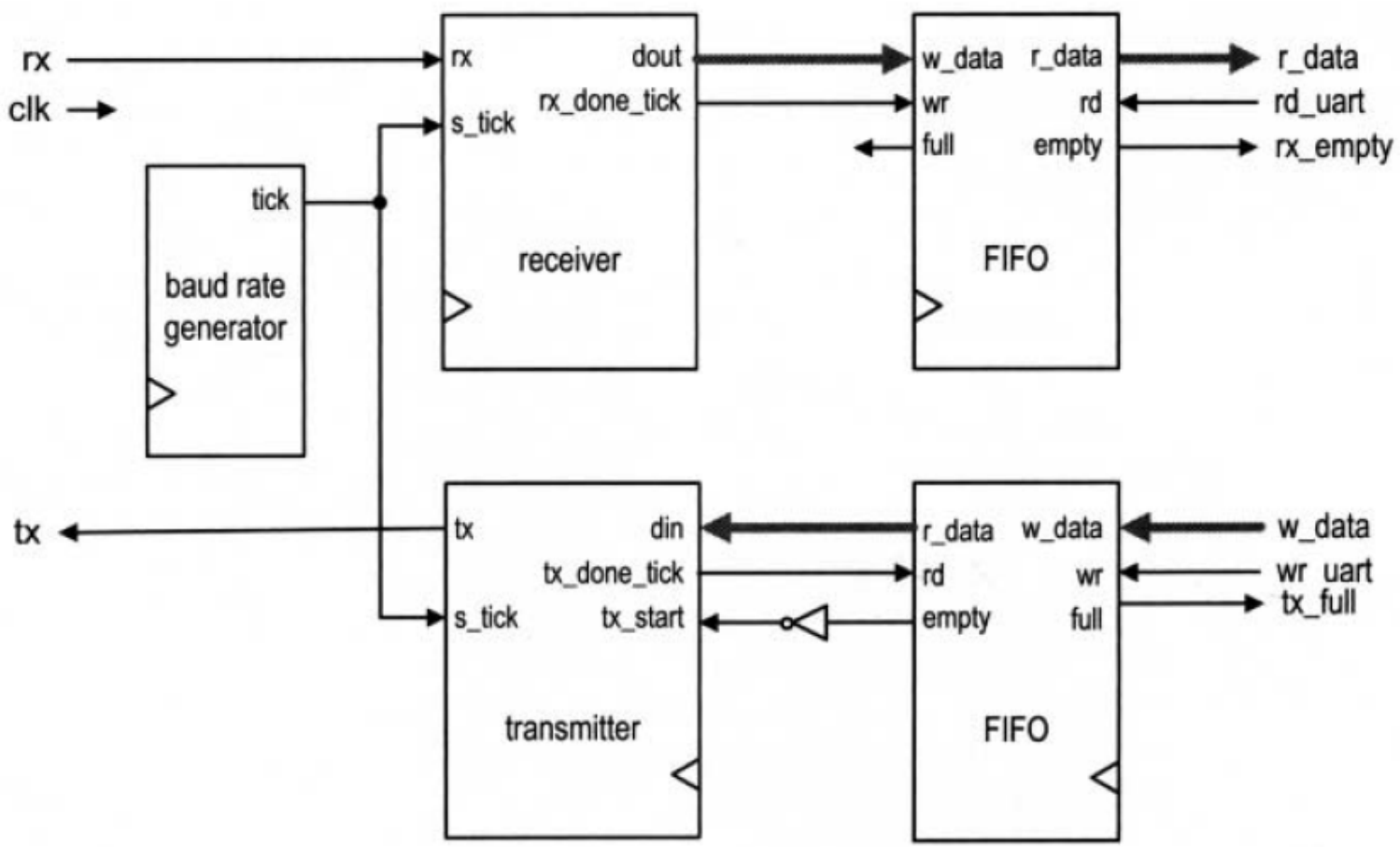
- Previamente, acordar baudrate y cantidad de bits de parada.

UART - Receptor



- Baud rate generator: oversampling.
- Interfaz: FIFO.

UART - Completo

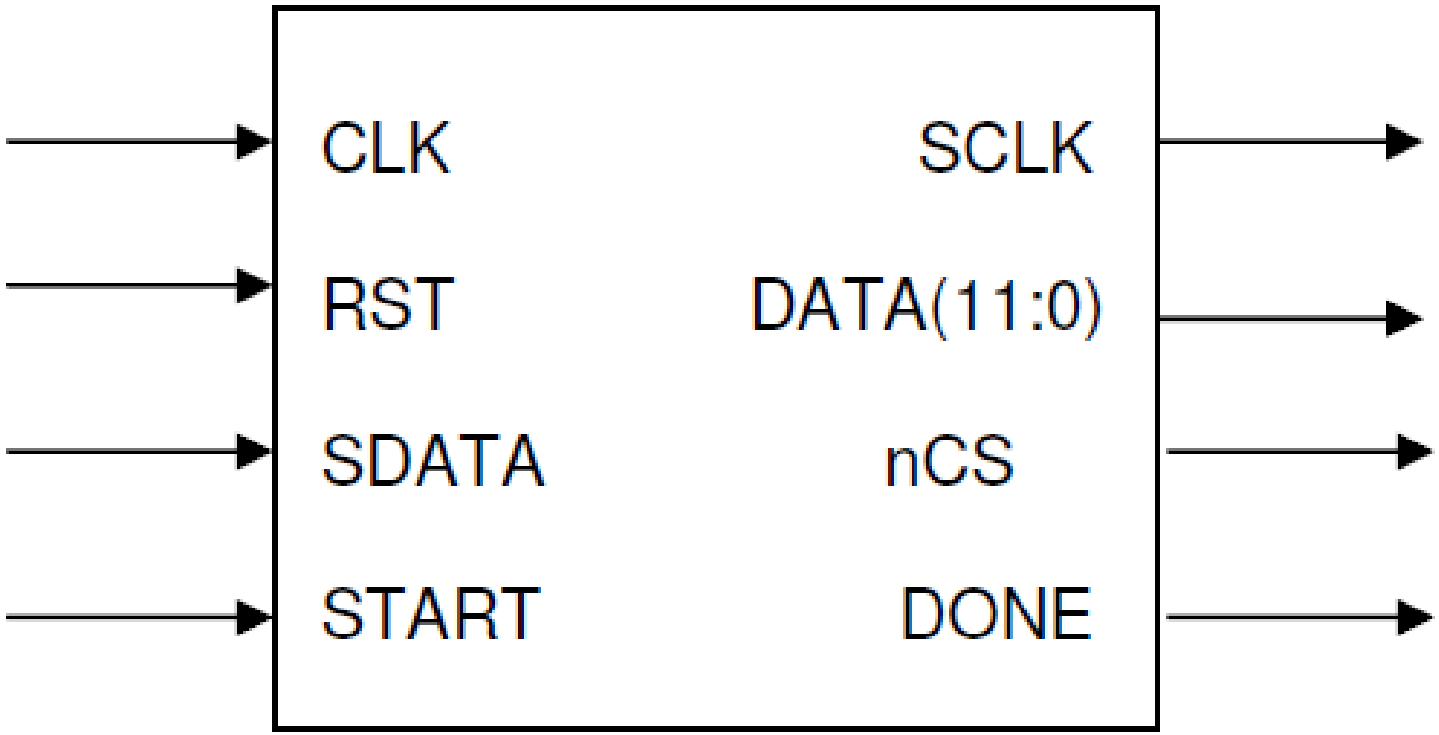


- Baud rate generator: oversampling.

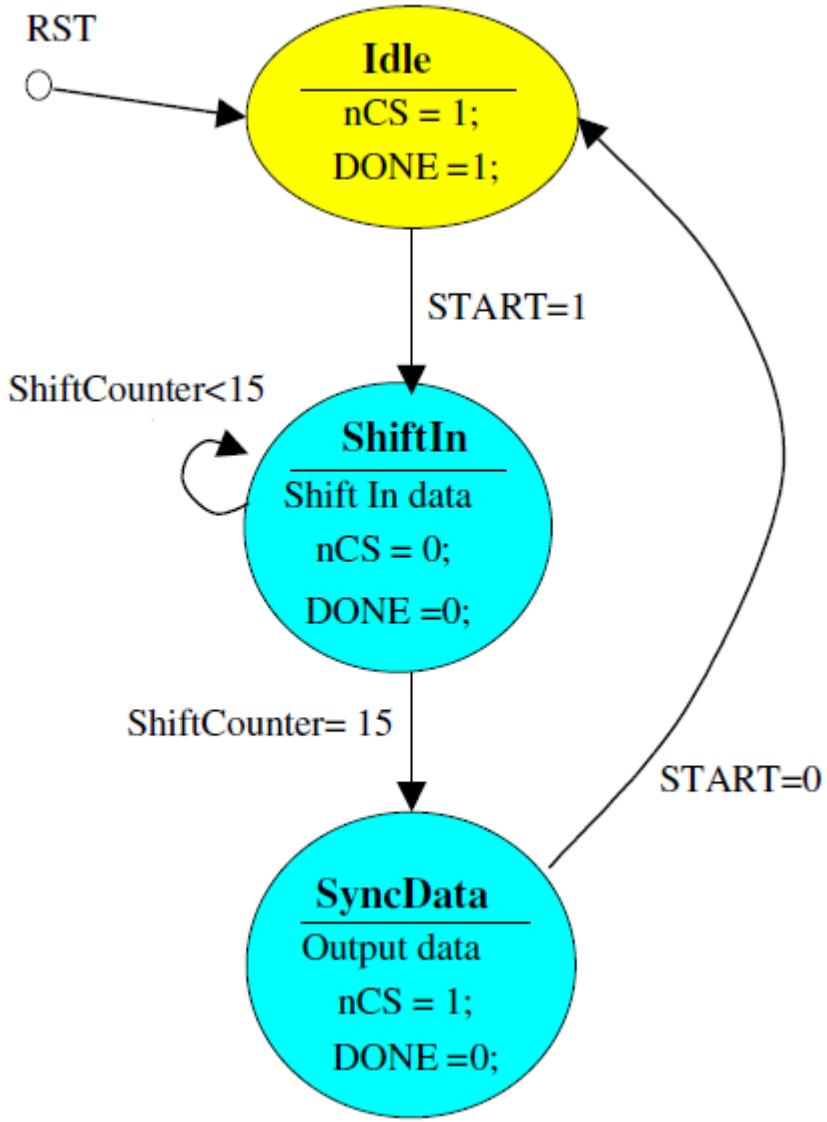
UART - Transmisión Entity

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

entity uart_tx_unit is
  generic(
     -- Default setting:
     -- 38,400 baud, 8 data bis, 1 stop its, 2^2 FIFO
     DBIT: integer:=8;     -- # data bits
      SB_TICK: integer:=16; -- # ticks for stop bits, 16/24/32
                            --   for 1/1.5/2 stop bits
      DVSR: integer:= 163;  -- baud rate divisor
                            -- DVSR = 100M/(16*baud rate)
      DVSR_BIT: integer:=9; -- # bits of DVSR
      FIFO_W: integer:=2    -- # addr bits of FIFO
                            -- # words in FIFO=2^FIFO_W
   );
   port(
      clk, reset: in std_logic;
      -- interfaz hacia usuario
      wr_uart: in std_logic;
      w_data: in std_logic_vector(7 downto 0);
      tx_full: out std_logic;
      -- interfaz hacia compu
      tx: out std_logic
   );

end uart_tx_unit;
```

Block diagram



FSM


RST
Idle
nCS = 1;
DONE =1;
START=1
ShiftCounter<15
ShiftIn
Shift In data
nCS = 0;
DONE =0;
ShiftCounter= 15
START=0
SyncData
Output data
nCS = 1;
DONE =0;

Entonces, ¿qué tengo que hacer?

- Filtro:
  - Terminar filtro pequeño y realizar su testbench.
  - Extenderlo luego a genérico con arrays y probar con los coeficientes del pasa bajos del *fdatool*, y el file I/O en el testbench.
- UART
  - Instanciar el transmisor. Implementar un módulo para enviar datos a 20 Khz, seteando el baud rate a 460.000 baudios, 8 bits de datos, 1 bit de parada. Enviar un contador (de 0 a FF por ejemplo) y testear que lleguen todos los datos.
  - En windows: hyperteminal. Setear control de flujo a ninguno + xvi32 (lector de hexa).

Entonces, ¿qué tengo que hacer?

- PmodMic
  - Instanciar el módulo. Implementar un módulo para samplear el micrófono a 20Khz. Quedarse sólo con los 8 bits mas significativos.
  - Implementar un módulo que lea del pmodMic a 20khz y envie los datos a traves de UART a la compu (crudos).
  - Levantar los datos con matlab y guardar el wav. (fopen, fread, uint8, wavwrite).
- Intregración : integrar el filtro a la lectura del micrófono y envio de datos.