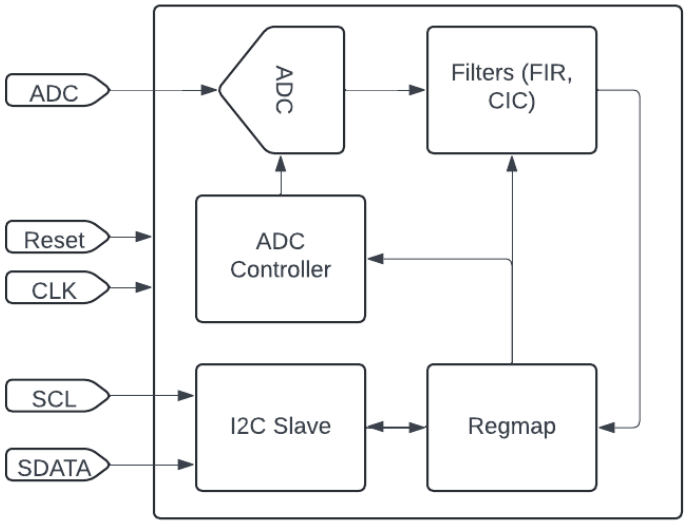
VERIFICACIÓN: DUT

El DUT consiste en un chip que implementa un ADC. El chip realizará conversiones automáticamente. Además, los datos capturados son procesados por dos filtros, un CIC y un FIR.

La configuración se realiza escribiendo a unos registros mediante I2C. La lectura de los datos convertidos y filtrados también se hará a través de registros.

La parte analógica se compone de varios bloques para gestionar la alimentación, que no son relevantes desde el punto de vista de la verificación digital. También consta de un ADC, el cual se modelará digitalmente.

Diagrama de bloques



Pines

- Reset: 1 para resetear
- CLK: reloj del sistema, usado en ADC, filtros y lógica digital. Periodo 2 ns.
- SCL: reloj del interfaz I2C.
- SDATA: datos del interfaz I2C.
- Otros (no relevantes para DV): VCC, GND

Registros
Los registros son todos de 8 bits.
Se accede por medio de I2C.

Dirección	Registro	Permisos	Descripción	Default value	Bits
					7 6 5 4 3 2 1 0
0	FIR_COEF_0	RW	Coeficiente 0 filtro FIR	1	coef0
1	FIR_COEF_1	RW	Coeficiente 0 filtro FIR	0	coef1
2	FIR_COEF_2	RW	Coeficiente 0 filtro FIR	0	coef2
3	FIR_DIV	RW	Divisor. Desplazamiento	0	div
4	CIC_COEF	RW	Decimation rate	1	reserved (6) dr (1:0)
5	CHIP_ID	R	CHIP IP	0xA5	ID
6	CONTROL	W	Control del ADC	-	reserved (7:1) E (0)
7	OUTPUT	R	Señal convertida y filtrada	0	output

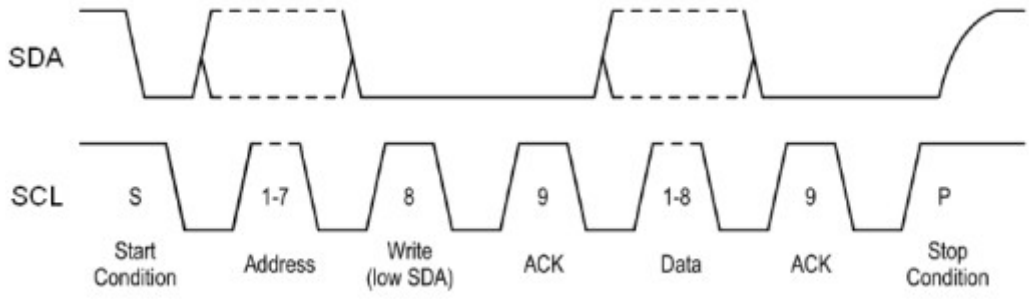
Bitfields:

- coef0, 1 y 2[7:0] son los coeficientes del filtro FIR. Son enteros con signo.
- div[7:0] (signed) es el número de bits por el que se desplaza a la derecha el resultado del filtro FIR. Se debe usar en consonancia con los filtros digitales (escala y número de coeficientes distintos de cero).
- dr[1:0] es el decimation rate del filtro CIC. Puede ir entre 0 y 3. 0 significa que no se decima.
- ID[7:0] es el identificador del chip, de sólo lectura. Su valor es 0xA5.
- E es la señal de enable del ADC. Es de sólo escritura.
- output[7:0] es la señal del ADC convertida y filtrada. Es un entero con signo. Sólo lectura.

Los bits reservados y los registros no implementados se deben leer como 0. Su intento de escritura no tendrá ningún efecto.

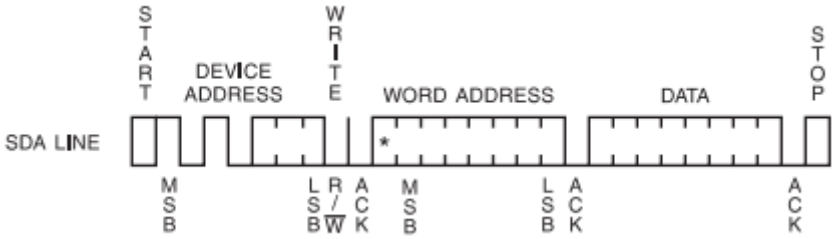
I2C
El sistema usa el estándar I2C. Puede consultarse el estándar en el aula virtual.

La transmisión utiliza dos líneas, una de reloj SCL y otra de datos SDA. El reloj es generado por el master (testbench), y los datos pueden ser enviados tanto por el master como por el slave, según corresponda. La línea de datos se muestrea cuando el reloj tiene un flanco de subida y los datos se cambian en el flanco de bajada. La transmisión se basa en bytes, al final de cada uno habrá un ACK. Además hay dos bits especiales para indicar el comienzo y final de la transacción:



Una transacción I2C a nuestro dispositivo está formada por 3 bytes:

- Dirección del chip y bit de lectura/escritura. La dirección i2c del chip es 1,
- Dirección del registro
- Dato



El bit R/W indicará si la transacción es de lectura o escritura. Si es de lectura, los datos serán escritos por el slave (el chip); si es de escritura los datos son escritos por el master (el testbench).

No se soporta el enviar datos a direcciones consecutivas, ni otra dirección en la misma trama, etc. Siempre hay que enviar la condición de start, una vez el chip id, una vez la dirección del registro y un único dato. Después siempre la condición de stop.

Pines del subsistema del I2C:

- Entradas:
  - clk: reloj del sistema
  - rst_n: reset. 0 resetea el sistema; 1 para funcionar
  - SCLK: pin del I2C de reloj
  - SDA (inout): pin del I2C de datos
  - address: chip address. 1 por defecto
  - reg_rd_data[7:0]: dato leído de un registro cuando la transacción i2c es de lectura. Es el que llega de los registros y se transmitirá por i2c de vuelta al maestro.
- Salidas:
  - reg_addr[7:0]: dirección del registro que se especifica en la transacción i2c
  - reg_wr_data[7:0]: dato a escribir en un registro cuando la transacción i2c es de escritura

- wr1rd0: 1 para indicar que se quiere escribir sobre un registro; 0 para leer. Si se escribe (1) se utilizará reg_wr_data; si se lee (0) se utilizará reg_rd_data
- reg_req: 1 para indicar que se quiere hacer una operación sobre los registros

Nota: la línea SDA puede ser de lectura al DUT o escritura, por lo tanto es bidireccional. Además, también es por la que se transmiten los ACK.

ADC

El ADC se implementará con un modelo DMS. No se tienen en cuenta especificaciones analógicas (INL, DNL, error de offset, de ganancia, etc.) porque el modelo es de alto nivel y no soporta ese nivel de detalle, se considerará un ADC ideal.

El ADC tiene una salida de 8 bits. El nivel de tensión de referencia es fijo y de 5 V.

El ADC tiene los siguientes pines:

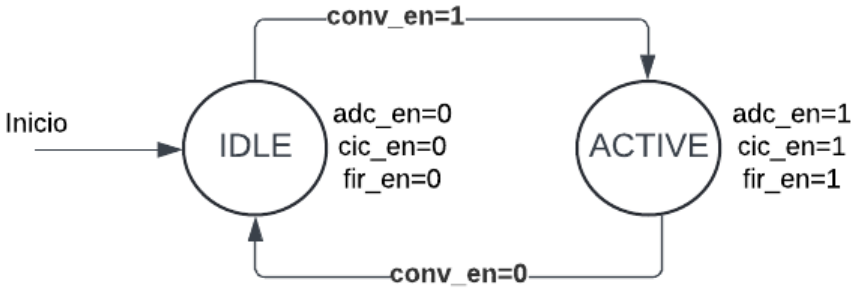
- Entradas:
  - clk: reloj del sistema
  - rst_n: reset. 0 resetea el sistema; 1 para funcionar
  - enable: 1 para habilitar el ADC; 0 para deshabilitarlo
  - V_in (real): tensión de entrada a convertir
  - V_ref (real): tensión de referencia, 5 V
- Salidas:
  - data_ready: el valor de adc_q es válido cuando data_ready es 1
  - adc_q[7:0]: código convertido

El ADC puede tardar varios ciclos en realizar la conversión. Para ello se utilizará el plusarg: “ADC_DELAY”.

El ADC hará conversiones consecutiva a la máxima tasa que puede siempre y cuando esté habilitado.

FSM
El chip cuenta con una pequeña FSM para controlar las habilitaciones del ADC y los dos filtros.

El diagrama de estados es:



Los estados son:
- IDLE: deshabilita ADC, CIC y FIR
- ACTIVE: habilita ADC, CIC y FIR

La FSM transiciona de un estado a otro por medio de la señal conv_en, que viene del registro CONTROL.

Los pines de este módulo son:
- Entradas:
  - clk: reloj del sistema
  - rst_n: reset. 0 resetea el sistema; 1 para funcionar
  - conv_en: 1 para habilitar el ADC, 0 para deshabilitarlo. Esta señal se controla desde el registro
- Salidas
  - adc_en: habilita el ADC para convertir. Si no se deshabilita, continuará convirtiendo a su máxima tasa de conversión.
  - cic_en: habilitación del filtro CIC
  - cic_clr: clear del filtro CIC para llevar sus salidas a valor bajo.
  - fir_en: habilitación del filtro FIR

Filtros

DATAPATH

El datapath es el camino que recorren los datos y a través del cual sufren ciertas transformaciones. En este caso, los datos pasarán por dos filtros, un CIC y un FIR de orden 2.

Ambos filtros están contenidos en el mismo módulo, cuyos pines son:

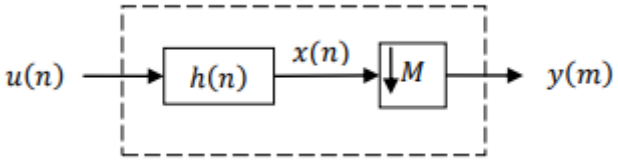
- Entradas
  - clk: reloj del sistema
  - rst_n: reset. 0 resetea el sistema; 1 para funcionar
  - data_ready: el dato del ADC es válido
  - data_in[7:0]: dato del ADC, sólo válido cuando es indicado por data_ready
  - cic_en: habilitación del filtro CIC. Controlado por la FSM.
  - cic_clr: clear del filtro CIC para llevar sus salidas a valor bajo.
  - fir_en: habilitación del filtro FIR.
- Salidas
  - data_filtered[7:0]: dato filtrado por el CIC y el FIR
  - filtered_ready: señal que indica que el dato en data_filtered es válido

CIC

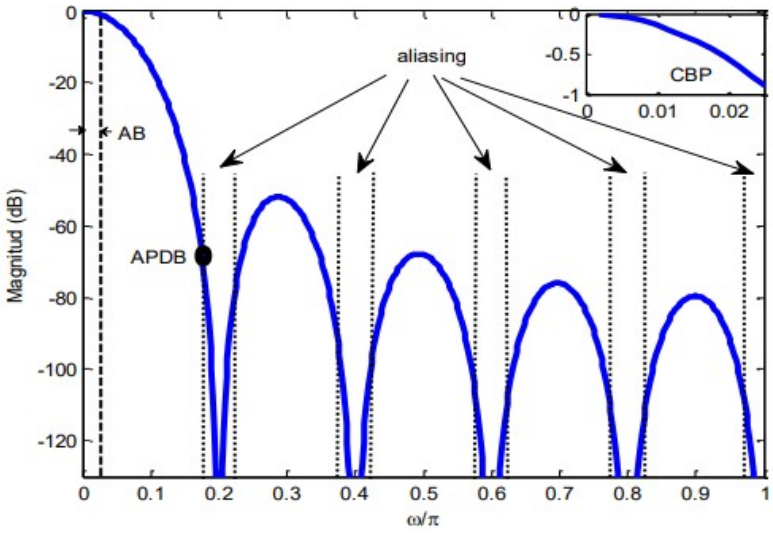
El filtro CIC se encarga de compensar el sobremuestreo del ADC, es decir, baja la frecuencia de muestreo de la señal. Se compone de un decimador y un filtrado.

Un decimador de valor M reduce la frecuencia de muestreo, o lo que es equivalente, el número de muestras en un factor M, siendo este número un entero. Es decir, se queda con 1 muestra de cada M.

El decimador puede introducir distorsión debido al efecto de la aliasing. Para evitarlo se utiliza un filtro digital pasa bajo previo al decimador.



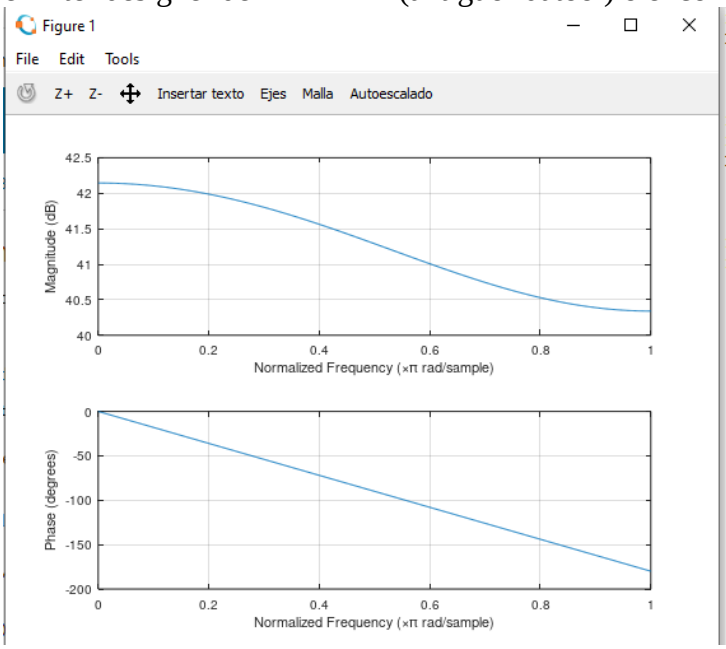
Un filtro CIC está formado por una etapa integradora seguido por un filtro comb. El filtro comb se puede ajustar para que la máxima atenuación coincida con las frecuencias a las que se produce la aliasing:



En el diseño, el factor de decimación se establece en el registro CIC_COEF.

FIR
El filtro FIR tendrá tres coeficientes, los cuales pueden ser configurados a través de los tres primeros registros del chip.

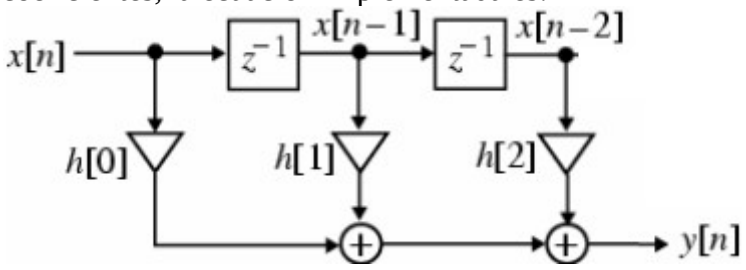
Para desactivar el efecto del filtro se puede asignar 1 al primer coeficiente y 0 a los demás. De otra manera, los coeficientes determinarán la respuesta del filtro. Se pueden calcular los coeficientes con el filter designer de MATLAB (antiguo fdatool) o el comando fir1():




```
>> b = fir1(2, 0.5)    %coeficientes del fir
b =

   0.046303   0.907393   0.046303

>> L=floor(log2((2^(N-1)-1)/max(b))); %redondeo
>> b2=round(b*2^L)
b2 =

     6   116     6

>> freqz(b2, 1)
>>
```

Nota: el filtro usa números enteros con signo de 8 bits. Al convertir de coma flotante a enteros con el método anterior se puede apreciar una ganancia distinta a la unidad.

Una vez fijados los coeficientes, la ecuación implementada es:



y[n] = coef0 * x[n] + coef1 * x[n-1] + coef2 * x[n-2]

Y se puede ver su efecto con:

```
>> input=[1 2 3 4 5 0 1 2 3];   %entrada
>> output = filter(b2, 1, input) %filtrar
output =

     6   128   256   384   512   604    36   128   256
```

Nota: téngase en cuenta el tamaño de datos de 8 bits en el chip (signed).

Como las operaciones matemáticas requieren ampliar el número de bits y, además, los coeficientes pueden requerir compensar la escalada a enteros, se requiere una etapa final de desplazamiento a la

derecha. El desplazamiento puede implementar una división por potencias de 2, por ejemplo para dividir por 128 se desplazarán 7 bits a la derecha.

Los valores por defecto serán tales que el filtro esté desactivado:

- coef0 = 1
- coef1 = 0
- coef2 = 0
- div = 0