Guía de Uso – Modelo VHDL ADC0808

El IC ADC0808 es un componente de adquisición de datos que contiene un conversor analógico-digital de 8 bits, multiplexor de 8 canales y lógica de control compatible con microprocesadores. El convertidor A/D de 8 bits utiliza aproximaciones sucesivas como técnica de conversión, el multiplexor de 8 canales permite acceder a cualquiera de 8 señales análogas de voltaje unipolar de entrada. La arquitectura del modelo de simulación de este IC se puede ver en la Figura 1 y a diferencia del IC físico este admite voltajes bipolares.

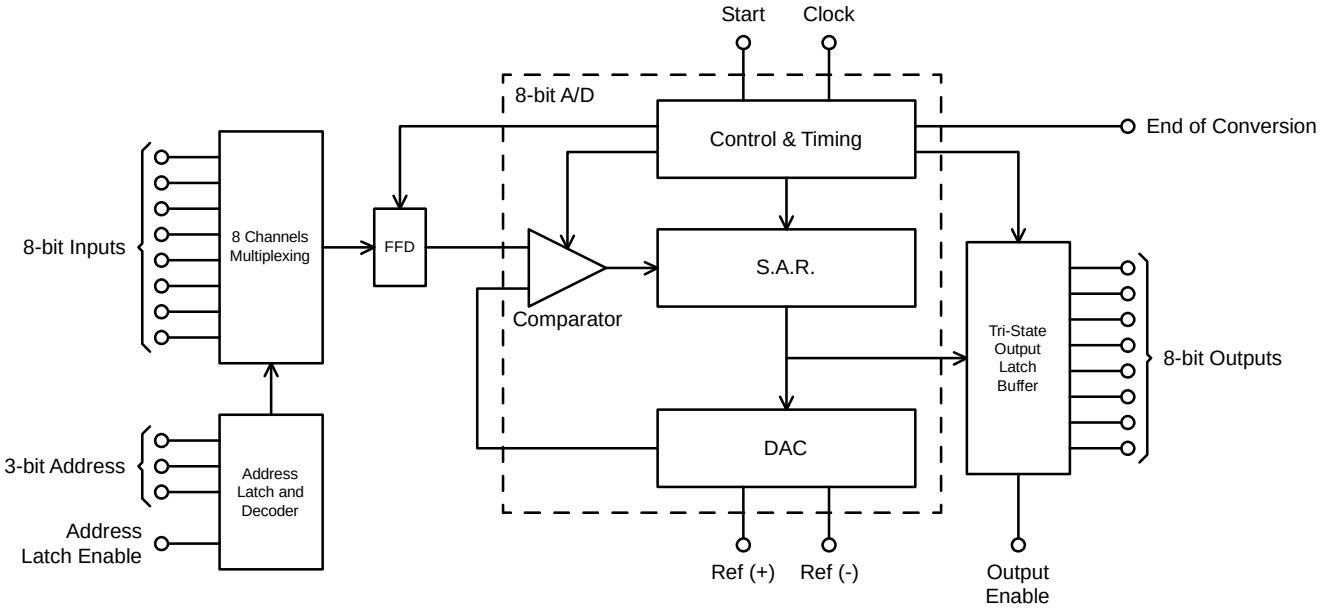

Figura 1. Diagrama de bloques del modelo de simulación del IC ADC0808

Nota: Se puede comparar con la arquitectura del IC físico para ver sus diferencias y similitudes

I. TENSIONES NOMINALES MÁXIMAS

El modelo de simulación del IC ADC0808 puede operar con voltajes tanto positivos como negativos, sin embargo, para mantener la simulación lo más fiel posible a la realidad se recomienda utilizarlo con voltajes proporcionaos por la hoja de especificaciones del fabricante. Los voltajes de operación nominales máximos del IC físico se pueden ver en la Tabla 1.

TABLA 1. VOLTAJES NOMINALES MÁXIMOS DE OPERACIÓN DEL ADC0808

Voltaje de alimentación (Vcc)	6.5V
Voltaje en cualquier pin excepto las entradas de control	−0.3V a (VCC+0.3V)
Voltaje en las entradas de control (START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)	−0.3V a + 15V

Nota: Revisar la hoja de especificaciones del fabricante

Los valores de voltaje de alimentación y de las señales de entrada en el modelo de simulación se tienen que ingresar en base binaria y representación en punto fijo, dentro del archivo “ADC0808.vhd” existen dos parámetros genéricos “bits_int” y “bits_res”, los cuales se usan para indicar la cantidad de bits destinados para expresar los valores de voltaje. Para saber la cantidad de bits que se requieren para representar en binario la parte entera de un número decimal se puede usar la ecuación (1).

$$n = \log_2(x+1) = \frac{\log(x+1)}{\log 2} \tag{1}$$

Nota: Donde n es el número de bits requeridos para representar en binario el número entero x, en caso de que el resultado no sea un número entero se aproxima al siguiente número entero más cercano.

Para la parte fraccionaria se indica la cantidad bits que se quiera usar en el factor de escala para su representación como número entero, para esto se tiene en cuenta la resolución del conversor (ecuación (2)) la cual indica el mínimo cambio de voltaje que puede detectar, con lo cual, se debería poder expresar un voltaje de entrada en punto fijo con esa resolución.

$$resolución = \frac{V_{ref+} - V_{ref-}}{2^n - 1} \tag{2}$$

Nota: Con n = cantidad de bits de entrada del conversor

La elección de cuantos bits se va a usar para representar la parte fraccionaria o cuantas veces se quiera correr la coma a la derecha depende del criterio del diseñador, sin embargo, es importante cumplir con la condición de la expresión (3) para garantizar la resolución de los datos de entrada en su representación.

$$resolución * 2^m > 0 \tag{3}$$

Nota: Donde m es la cantidad de bits para representar la parte fraccionaria

Por ejemplo, para representar los valores de voltaje de entrada al ADC con un voltaje de alimentación positivo de 3.3V y negativo de 0V ¿cuantos bits se requieren?

Usando la ecuación (1) para la parte entera se tiene:

$$n = \frac{\log(3+1)}{\log 2} = 2$$

La resolución del conversor con estos voltajes de referencia se tiene:

$$resolución = \frac{3.3V}{255} = 0{,}012941V$$

Con la expresión (3) se tiene:

$$m = 7$$

Es así que se podría elegir correr la coma desde 7 posiciones en adelante, entre más posiciones se corra la coma, más exacto será el valor representado en punto fijo, pero también se necesitaran más recursos de procesamiento. Para este ejemplo, se optará por correr la coma 8 posiciones, de esta manera, dentro del archivo "ADC0808.vhd" el parámetro genérico bits_int = 2 y el parámetro bits_res = 8, utilizando así 10 bits para la representación de los datos de voltaje del ADC.

Para expresar los voltajes hay que multiplicar su valor por 2 elevado a la cantidad de veces que se corre la coma (8 en este caso) y convertir la parte entera a base binaria.

$$3.3 * 2^8 = 844.8 \tag{4}$$

$$844_{10} = 1101001100_2$$

II. MULTIPLEXOR

El modelo de simulación contiene un multiplexor de señal "analógica" de 8 canales. Se selecciona un canal de entrada particular utilizando el decodificador de dirección. La Tabla 1 muestra los estados de entrada de las líneas de dirección para seleccionar cualquier canal. La dirección se engancha al decodificador en el flanco ascendente de la señal de habilitación de enganche de dirección (Address Latch Enable - ALE).

TABLA 2. SELECCIÓN DEL CANAL ANÁLOGO

Canal analógico seleccionado	Línea de dirección			Canal analógico seleccionado	Línea de dirección		
	2	**1**	**0**		**2**	**1**	**0**
in_0	L	L	L	in_4	H	L	L
in_1	L	L	H	in_5	H	L	H
in_2	L	H	L	in_6	H	H	L
in_3	L	H	H	in_7	H	H	H

III. CARACTERÍSTICAS DE CONVERSIÓN

El modelo de simulación del IC ADC0808, que se muestra en la Figura 1, se puede dividir funcionalmente en tres secciones principales: el conversor DAC, el registro de aproximaciones sucesivas y el comparador.

El conversor DAC (Figura 1) es una implementación con operaciones de un conversor digital a analógico ideal.

El conversor de aproximación sucesivo (SAR), transforma la salida analógica del multiplexor en una palabra digital de 8 bits. La lógica de control del convertidor controla un voltaje particular al comparador, en función del resultado de esta comparación, la lógica de control decide si el voltaje generado debe ser mayor o menor que el presente. Este algoritmo se ejecuta ocho veces por conversión, una vez cada 8 períodos de reloj en el IC físico, produciendo un tiempo de conversión total de 64 períodos de reloj, en el modelo de simulación se usan 66 períodos de reloj.

El registro de aproximaciones sucesivas se restablece en el flanco ascendente del impulso de inicio de conversión de la señal *Start*. La conversión se inicia en el flanco descendente se la señal de reloj (Clock) una vez la señal *Start* vuelva a estar en bajo. Una conversión en proceso se interrumpirá al recibir un nuevo impulso de conversión de inicio se la señal *Start*. La conversión en modo continuo se puede lograr vinculando la salida de fin de conversión *EOC* a la entrada de la señal de inicio *Start*. Si se usa en este modo, se debe aplicar un impulso de conversión de inicio externo al inicio de la simulación.

Cuando se completa el ciclo de conversión, los datos resultantes se cargan en un registro de salida tri-estado. Los datos de salida pueden ser leídos por el sistema host en cualquier momento antes del final de la próxima conversión.

Cuando el EOC se pone en alto, indica a la lógica de la interfaz que los datos resultantes de la conversión están listos para ser leídos.

IV. DIAGRAMA DE TIEMPOS

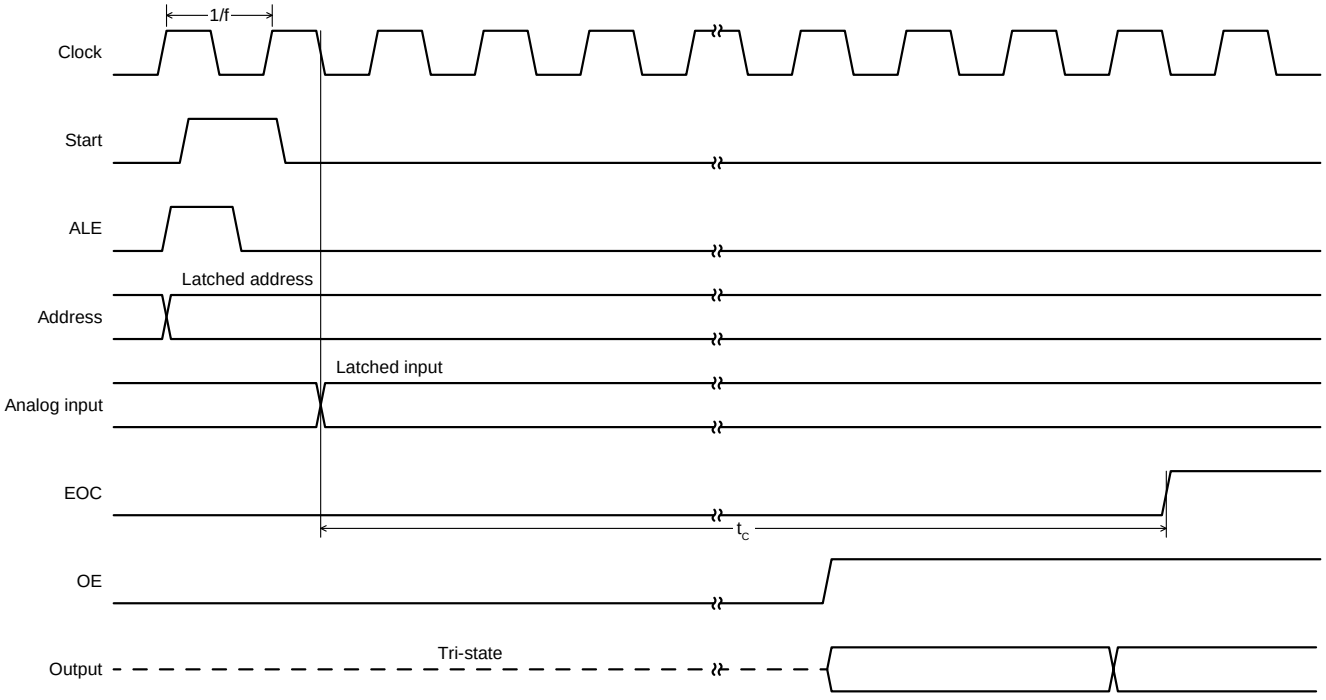

Figura 2. Diagrama de tiempos del modelo de simulación

Donde:

t_C = Tiempo de conversión

V. ECUACIONES DE CONVERSIÓN

El voltaje equivalente por un código binario viene dado por:

$$V_{in} = \frac{N(V_{ref+} - V_{ref-})}{2^n - 1} + V_{ref-} \tag{5}$$

El código de salida N para una entrada arbitraria de voltaje dentro del rango está dado por:

$$N = \frac{(2^n - 1)(V_{in} - V_{ref-})}{V_{ref+} - V_{ref-}} \tag{6}$$

Donde:

V_{in} = Voltaje de entrada al comparador

V_{ref+} = Voltaje de referencia positivo

V_{ref-} = Voltaje de referencia negativo

VI. GENERACIÓN DE DATOS DE ENTRADA PARA SIMULACIONES

Para hacer la simulación en Test Bench del conversor A/D hay que crear formas de onda para sus entradas, esto se puede hacer en el software de Matlab y la información de las señales se pueden guardar en un archivo de texto. Siguiendo con el ejemplo anteriormente trabajado, se va a generar una señal senoidal de 3.3V de amplitud pico a pico y con offset, ya que el IC físico solo admite señales unipolares.

$$x = d + a \cdot \sin(2 * \pi * f * t) \tag{7}$$

Donde:

d = Desfase

a = Amplitud

f = Frecuencia

t = Tiempo

El siguiente código de Matlab genera una señal senoidal la cual está descrita por la ecuación (7), la creación de los datos se puede hacer una sola vez y luego en la simulación variar el tiempo de lectura del archivo de texto para variar su frecuencia. Antes de guardar los datos, se tiene en cuenta el número de bits de resolución que se eligió, multiplicando el dato generado de la señal así como en la expresión (4) y redondeando su valor. En este caso que la señal a muestrear es de 1Hz, la frecuencia de muestreo indica la cantidad de datos que se van a generar, entre más datos se generen (mayor frecuencia de muestreo) más definida será la señal generada.

```
bits_res = 8;                                   % Resolution bits

fs = 1;                                         % Signal frequency
fc = 1000;                                      % Sampling frequency
t = 0:1/fc:(1/fs)-(1/fc);                       % Time vector
vpp = 3.3;                                      % Peak to peak voltage

signal = (vpp/2)+(vpp/2)*sin(2*pi*fs*t);
plot(t,signal)

fid = fopen('sine.txt','w');                    % Open the file in write mode
for i=1:length(signal)
    fprintf(fid,'%d\n',round(signal(i)*(2^bits_res)));
end
fclose(fid);                                    % Close the file
```

VII. SIMULACIÓN – TEST BENCH

1.Resultados de simulación

En la Figura 3 se puede observar una parte de la simulación del archivo “ADC0808_tb”, en esta se puede observar tres de sus entradas con una señal senoidal a diferentes frecuencias (30Hz, 300Hz y 3KHz) y una frecuencia de muestreo de 5KHz, la salida es el dato binario de 8 bits que tiene como salida el modulo, la cual se puede observar gráficamente dando clic derecho sobre el nombre de la señal en el panel izquierdo y cambiando el formato de esta a análogo.

La salida del conversor cambia cuando se selecciona un canal diferente de entrada, allí se pueden observar tres casos a la salida, una señal supermuetreada, una señal muestreada cumpliendo el criterio de Nyquist y por ultimo una que no lo cumple, donde se presenta el efecto de aliasing, la señal muestreada no corresponde con la señal de entrada y hay pérdida importante de información.

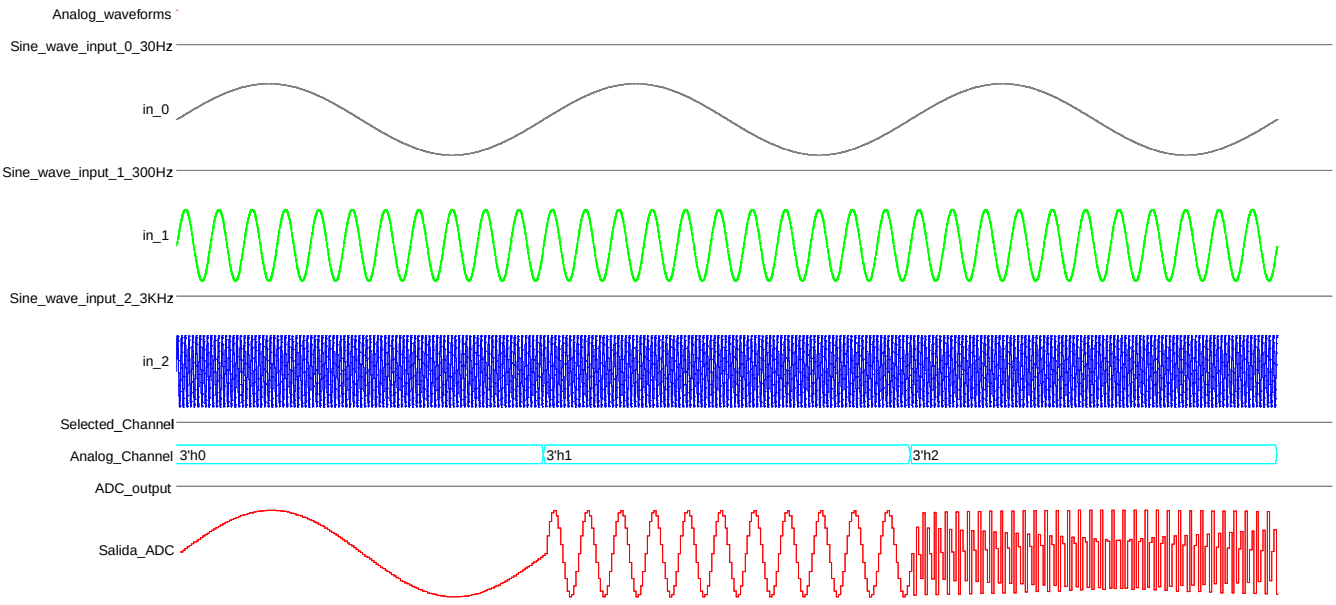

Figura 3. Resultados de simulación del módulo ADC0808

2. Sugerencia de aplicación

El modelo de simulación puede operar en modo continuo, para esto se debe hacer una conexión del pin *EOC* con el pin de *Start*, de otro modo, en el circuito de control debe generar esta señal de control con una frecuencia inferior a la de funcionamiento del conversor que inicie una nueva conversión.

Para simular la señal generada en Matlab a una frecuencia deseada hay que hallar el periodo de esta frecuencia y dividirlo en el número de muestras que contiene el archivo de texto, así se hallara el valor de tiempo de retardo que debe tener la lectura de un dato y otro. Por ejemplo, para simular una señal de entrada al conversor a 50Hz, primero se hallaría su periodo con la ecuación (8).

$$t = \frac{1}{F} \tag{8}$$

Donde se tiene:

$$t = \frac{1}{50} = 0.02s$$

Dividiendo el periodo de la señal, entre el número de muestras que se generaron en Matlab (frecuencia de muestreo), se tiene:

$$t_r = \frac{0.02}{1000} = 20\mu s$$

Con lo cual, el tiempo de retardo en leer las muestras del archivo de texto para generar una señal de 50Hz es de 20µs.

Para expresar los voltajes se tienen que expresar con el bit de signo, por ejemplo, para expresar el voltaje de referencia positivo de 3.3V se debe escribir dentro el archivo de simulación de la siguiente manera:

$$signo - parte\ entera(2\ bits) - parte\ fraccionaria(8\ bits)$$

$$01101001100$$