Guía de Uso

Modelo de Simulación MCP3008

I. Introducción

El dispositivo MPC3008 es un conversor analógico-digital de 10 bits de aproximaciones sucesivas. La comunicación con el IC se logra mediante una interfaz serie compatible con el protocolo SPI, y puede alcanzar tasas de conversión de hasta 200 ksps. El rango de voltaje de operación es de 2.7 V a 5.5 V. El objetivo de esta guía es poder realizar experimentos virtuales en donde el circuito físico del MCP3008 se reemplace por una descripción VHDL correspondiente y con esto, poder verificar el comportamiento del controlador que se diseñe para este IC.

II. Modelo de simulación

1. Descripción de pines

A continuación, se presenta la tabla de pines implementados en el modelo de simulación:

Tabla 1. Funciones de los pines

Símbolo	Descripción
$CH0$	Entrada análoga
$CH1$	Entrada análoga
$CH2$	Entrada análoga
$CH3$	Entrada análoga
$CH4$	Entrada análoga
$CH5$	Entrada análoga
$CH6$	Entrada análoga
$CH7$	Entrada análoga
D_{GND}	Tierra digital
$\overline{CS}/SHDN$	Selección de chip
D_{IN}	Entrada de datos serie
D_{OUT}	Salida de datos serie
CLK	Reloj serial
A_{GND}	Tierra análoga
V_{REF}	Entrada de voltaje de referencia
V_{DD}	Fuente de alimentación de + 2.7 V a 5.5 V
NC	Sin conexión

Nota: Las señales que no están en gris no están implementadas

a. Entradas análogas ($CH0$ – $CH7$)

El modelo de simulación no tiene la opción de utilizar los canales en modo de entradas pseudo-diferenciales, solo se pueden usar como entradas de un solo extremo. Revisar la sección "*a. Entradas* análogas" y la sección "*3. Comunicación serial*", para obtener más información sobre la programación de la configuración del canal.

b. Reloj serial (CLK)

El pin de reloj SPI se utiliza para iniciar una conversión y cambiar el bit de la conversión en D_{OUT} con cada flanco descendente.

c. Entrada de datos en serie (D_{IN})

El pin de entrada de datos en serie del puerto SPI se utiliza para cargar los datos de configuración del canal en el modelo de simulación.

d. Salida de datos en serie (D_{OUT})

El pin de salida de datos en serie SPI se utiliza para desplazar los resultados de la conversión A/D. Los datos siempre cambiarán en el flanco descendente de cada reloj.

e. Selección/apagado de chip ($\overline{CH}/SHDN$)

El pin $\overline{CS}/SHDN$ se utiliza para iniciar la comunicación con el modelo de simulación cuando se baja (0). Cuando se pone en alto (1), finalizará una conversión. El pin $\overline{CS}/SHDN$ debe colocarse en alto (1) entre conversiones.

2. Operación del modelo de simulación

a. Entradas análogas

La configuración se realiza como parte del comando en serie que se envía antes de que comience cada conversión. Las entradas análogas incluyendo el voltaje de referencia se tienen que expresar en punto fijo, la cantidad de bits que se use para la parte fraccionaria se puede obtener con la siguiente expresión:

$$resolución * 2^m > 1 \quad (1)$$

Nota: m es la cantidad de bits para representar la parte fraccionaria, si su valor es fraccionario se aproxima al siguiente entero

b. Referencia de entrada

La resolución del conversor está determinada por el voltaje de referencia (V_{REF}), en consecuencia, a medida que este se reduce, la resolución se hace más pequeña.

$$resolución = \frac{V_{REF}}{1023} \quad (2)$$

El código de salida digital teórico producido por el convertidor A/D es una función de la señal de entrada analógica y el voltaje de referencia, como se muestra a continuación:

$$N = \frac{1023 \cdot V_{IN}}{V_{REF}} \quad (3)$$

3. Comunicación serial

La comunicación se logra mediante una interfaz serie compatible con el protocolo SPI. El inicio de la comunicación se realiza bajando la línea $\overline{CS}/SHDN$ (Fig. 1). El primer ciclo de reloj recibido con $\overline{CS}/SHDN$ en bajo y D_{in} en alto, constituirá el bit de inicio. El bit $SGL/\overline{DIFF}$ sigue al bit de inicio y determinará si la conversión se realizará utilizando el modo de entrada diferencial o de un solo extremo (solo está implementado el modo de un solo extremo). Los siguientes tres bits ($D0$, $D1$ y $D2$) se utilizan para seleccionar la configuración del canal de entrada. La Tabla 2 muestra los bits de configuración.

TABLA 2. BITS DE CONFIGURACIÓN

Bits de control				Configuración	Canal
$Single/\overline{Diff}$	$D2$	$D1$	$D0$		
1	0	0	0	single-ended	0
1	0	0	1	single-ended	1
1	0	1	0	single-ended	2
1	0	1	1	single-ended	3
1	1	0	0	single-ended	4
1	1	0	1	single-ended	5
1	1	1	0	single-ended	6
1	1	1	1	single-ended	7
0	X	X	X	no implemented	

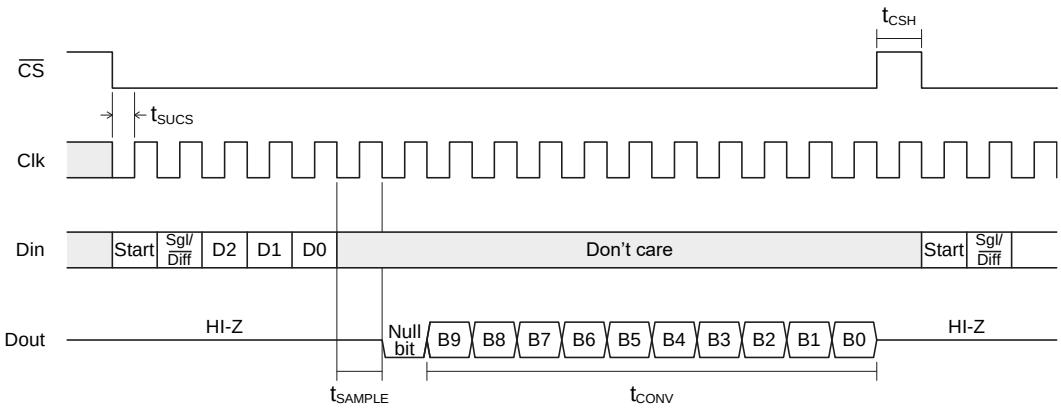

Fig. 1. Comunicación con el modelo de simulación

Una vez que se ingresa el bit $D0$, se usa un ciclo de reloj para hacer el muestreo de la señal de entrada. En el flanco descendente del siguiente reloj, se generará un bit nulo bajo. Los siguientes 10 ciclos de reloj generarán el resultado de la conversión con el MSB primero, como se muestra en la Fig. 1. Los datos siempre salen en el flanco descendente del reloj. Si se han transmitido los 10 bits de datos y el modelo de simulación continúa recibiendo ciclos de reloj mientras la señal $\overline{CS}/SHDN$ se mantiene bajo, se emitirá el último bit transmitido de forma indefinida.

TABLA 3. PARÁMETROS DE TIEMPO

Parameter	Sym	Min	Typ	Max	Units
Conversion Time	t_{CONV}		**10**		**clock cycles**
Analog Input Sample Time	t_{SAMPLE}		**1**		**clock cycles**
Throughput Rate	f_{SAMPLE}	**-**	**-**	**200**	**ksps**
Clock Frequency	f_{CLK}	**-**	**-**	**3.6**	**MHz**
CS Fall To First Rising CLK Edge	t_{SUCS}	**100**	**-**	**-**	**ns**
CS Disable Time	t_{CSH}	**270**	**-**	**-**	**ns**

III. SIMULACIÓN – TEST BENCH

A continuación, se presenta el resultado de dos simulaciones realizadas con el modelo de simulación donde se puede ver el comportamiento de las formas de ondas del IC. El voltaje de referencia que se uso es de 5V, usando la ecuación (2) se tiene que la resolución que tendría el conversor es:

$$resolución = \frac{5}{1023} = 4.8mV$$

Usando la expresión (1) y despejando la cantidad de bits m necesarios para expresar la parte fraccionaria de los voltajes de entrada, se encuentra que se requieren más de 7.6 bits, por lo que se eligen 8 bits.

$$m > \log_2\left(\frac{1}{resolución}\right) \tag{4}$$

$$m > \log_2\left(\frac{1}{4.8m}\right)$$

$$m > 7.676$$

1. Simulación 1

En esta primera simulación, se puede ver el comportamiento del modelo de simulación seleccionando dos canales de entrada diferente, el canal 4 y 2, los cuales tienen los voltajes de entrada 3.3V y 2V respectivamente, expresados en punto fijo con 8 bits para la parte fraccionaria. La representación en punto fijo de los voltajes se obtiene multiplicando su valor por el factor de escala 2^m, así:

$$V_{3.3} = 3.3 \cdot 2^8 \cong 853$$

$$V_2 = 2 \cdot 2^8 = 512$$

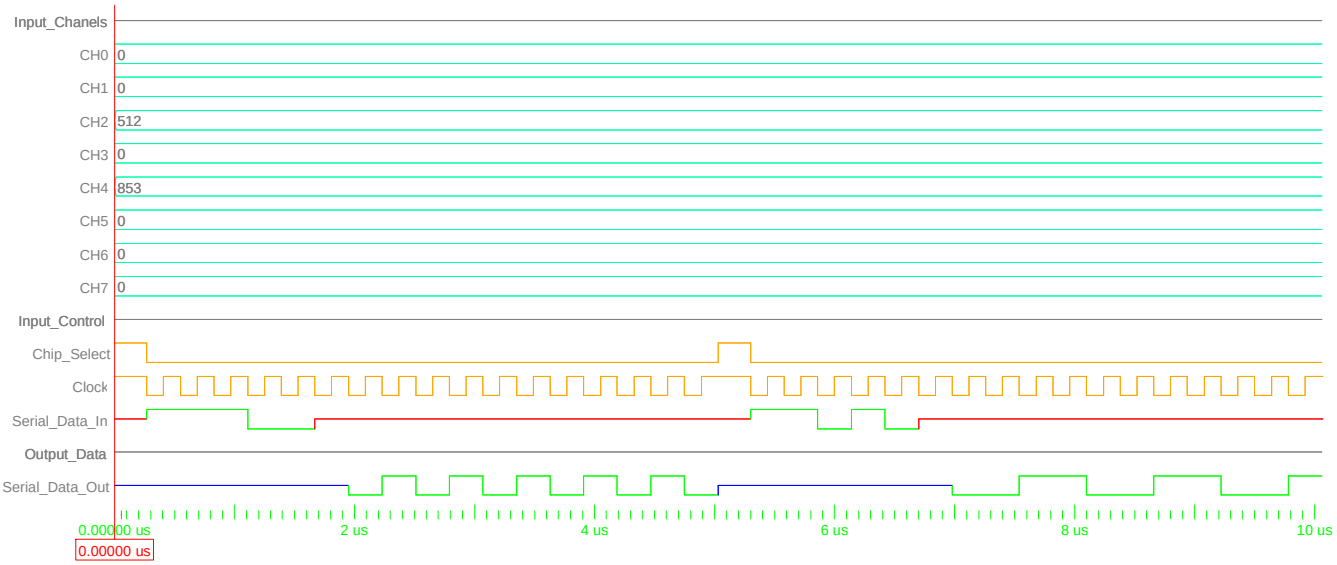

Fig. 2. Respuesta de la simulación número uno

2. Simulación 2

En la siguiente simulación se está muestreando una señal sinusoidal de 20KHz a una frecuencia aproximada de 200ksps, con lo cual se pueden observar 10 muestras por periodo de la señal.

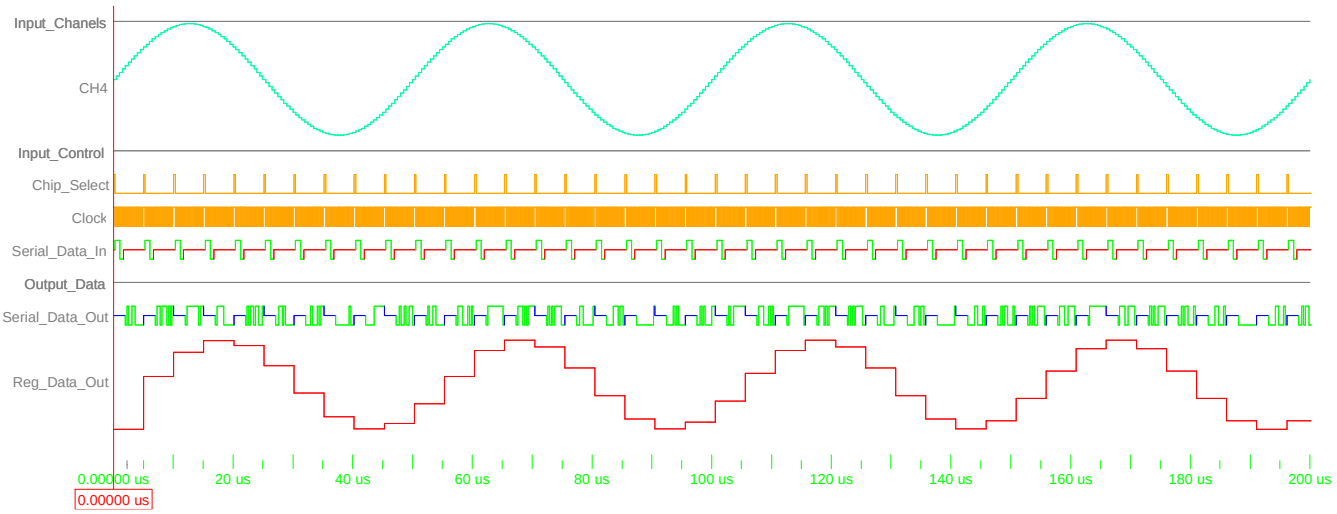

Fig. 3. Respuesta de la simulación número dos

Los datos de estímulo para esta simulación se generaron usando el software de Matlab haciendo uso del código encontrado en la ruta "*MCP3008/Matlab/signal.m".* En este código no es necesario cambiar la frecuencia de la señal que se quiere generar (esto se hará en el banco de pruebas), solamente el número de muestras por periodo que se quiera tener, en este caso se eligieron 100 muestras, un número más elevado generara una señal más suave en la simulación, pero puede tardar más tiempo en ejecutarse.

El archivo de texto que se genera con el código de Matlab se agrega al banco de pruebas, la velocidad con la que se leen los datos determina la frecuencia de la señal, para este ejemplo se quiere generar una señal de 20KHz, con lo cual su periodo es de 50us, al tener 100 muestras se divide el periodo de la señal en la cantidad de muestras, con lo cual, el tiempo de duración de cada una es de 500ns, como se puede ver en la implementación que se muestra a continuación:

```
architecture behav of MCP3008_2_tb is
  .
  .
  .
  file vector_data_00    :text;
begin
  .
  .
  .
  ch_4: process
          variable vect_line_data:line;
          variable data:integer;
  begin
          file_open(vector_data_00, "sine.txt", read_mode);    -- Open the file
          while not endfile(vector_data_00) loop       -- Read while file is not finished
                  readline(vector_data_00, vect_line_data);    -- Read a line
                  read(vect_line_data, data);    -- Read the value
                  ch(4) <= to_integer(to_unsigned(data, 1 + bits_int + bits_res)); -- Conversion
                  wait for 500 ns;        -- Signal at 20KHz
          end loop;
          file_close(vector_data_00);   -- Close the file
  end process;
end architecture;
```