CAPÍTULO 7

Librerías de simconverter

Índice:

LIBRERÍAS DE SIMCONVERTER

1. Introducción 37
2. Librería de Sigma Delta en Tiempo Discreto 37
   - 2.1. Bloques ideales 39
   - 2.2. Bloques no ideales 42
   - 2.3. Ejemplos 59
3. Librería de Pipeline.....31
   - 3.1. Bloques ideales 32
   - 3.2. Bloques no ideales 41
   - 3.3. Ejemplos 48
4. Librería de Flash.....85
   - 4.1. Bloques ideales 86
   - 4.2. Bloques no ideales 91
   - 4.3. Ejemplos 92

1. Introducción

En este capítulo nos vamos a centrar en visualizar el contenido de la librería de convertidores analógico-digitales que disponemos. Para ello vamos a ir presentando cada uno de los convertidores en estudio, mostrando una visión esquematizada de cada uno.

Los bloques están realizados mediante la herramienta Simulink de MATLAB 7.0. Se ha desarrollado una máscara para cada uno de ellos, de forma que el usuario no necesite conocer la estructura desarrollada para cada modelo, tan sólo las variables necesarias para su simulación.

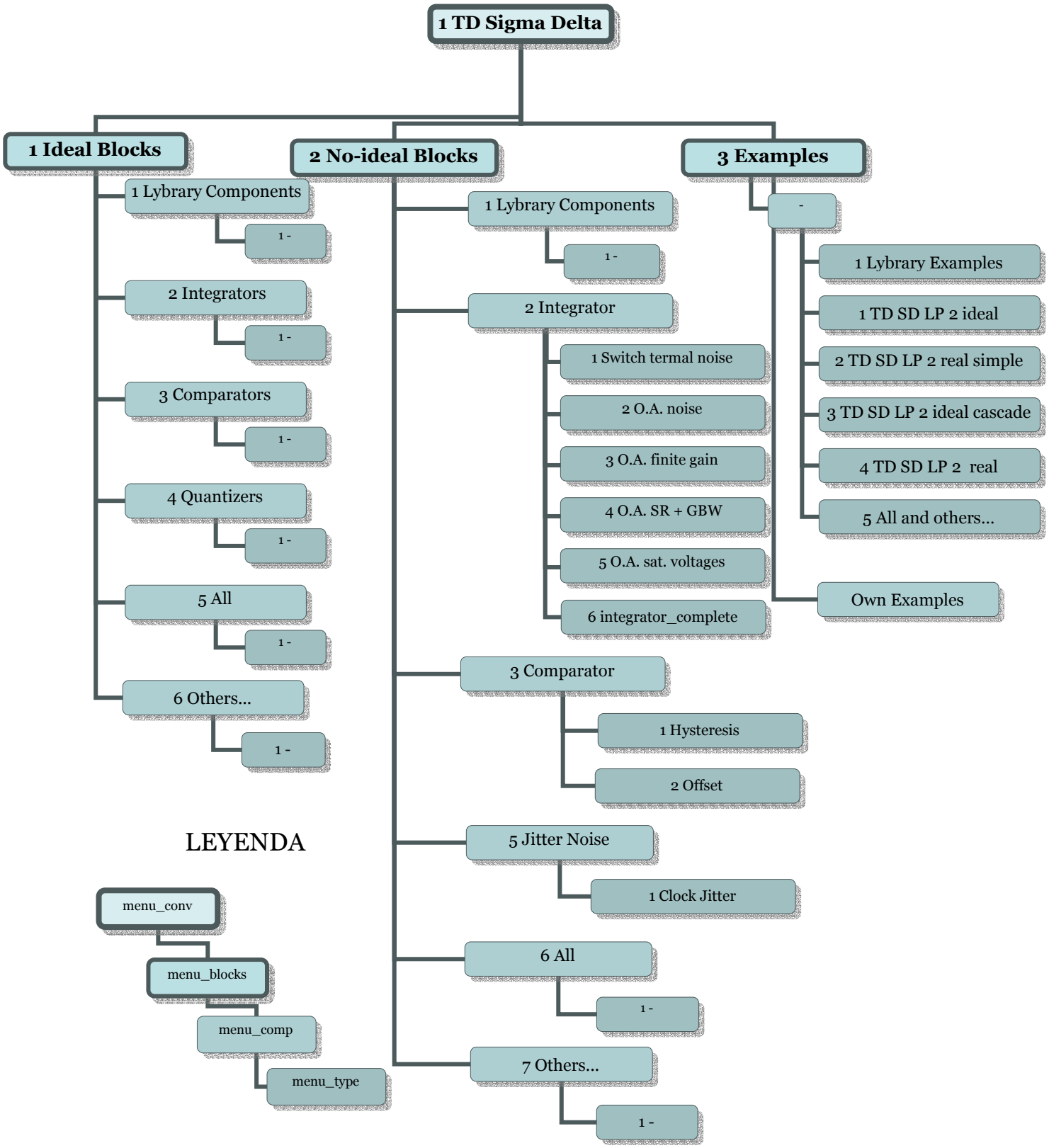
Cada uno de los convertidores estará formado por:

- **Bloques ideales**: Estos bloques no contienen ninguna no-idealidad, y presentaremos varios tipos según el modelo. Para cada convertidor dispondremos de todos los elementos que lo forman, de manera que el usuario disponga de todos los elementos para construir su propio diseño.
- **Bloques no-ideales**: Éstos simularán las no-idealidades a las que pueden estar sometidos los elementos de nuestro diseño. Existe una amplia gama de no-idealidades para cada uno de los bloques de cada convertidor, resultado de un estudio previo de los mismos.
- **Ejemplos**: En este apartado de la librería presentamos algunos diseños ya montados para cada uno de los convertidores. Con ellos pretendemos ayudar al usuario a la comprensión de los convertidores presentando modelos completos en algunos casos particulares. Éstos se podrán utilizar modificando sus especificaciones o sirviendo como base para otros.

Pasemos ahora a mostrar cada una de las partes de la librería desarrollada.

2. Librería de Sigma Delta en Tiempo Discreto

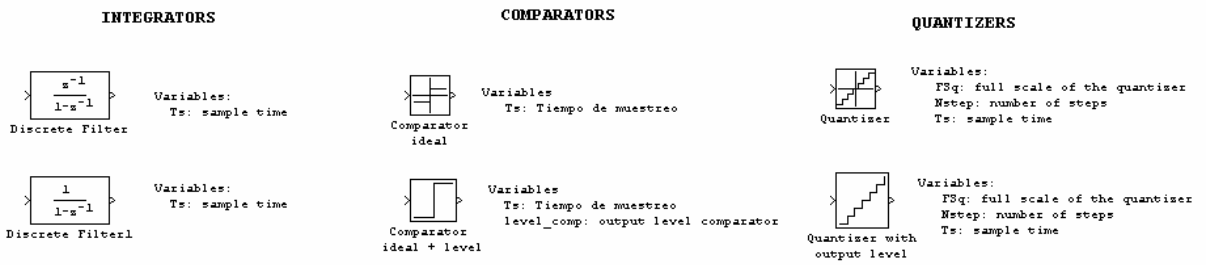
Para tener una visión global de la librería del convertidor Sigma Delta en Tiempo Discreto, presento el siguiente esquema en forma de árbol:


1 TD Sigma Delta
1 Ideal Blocks
1 Lybrary Components
1 -
2 Integrators
1 -
3 Comparators
1 -
4 Quantizers
1 -
5 All
1 -
6 Others...
1 -
2 No-ideal Blocks
1 Lybrary Components
1 -
2 Integrator
1 Switch termal noise
2 O.A. noise
3 O.A. finite gain
4 O.A. SR + GBW
5 O.A. sat. voltages
6 integrator_complete
3 Comparator
1 Hysteresis
2 Offset
5 Jitter Noise
1 Clock Jitter
6 All
1 -
7 Others...
1 -
3 Examples
-
1 Lybrary Examples
1 TD SD LP 2 ideal
2 TD SD LP 2 real simple
3 TD SD LP 2 ideal cascade
4 TD SD LP 2 real
5 All and others...
Own Examples
LEYENDA
menu_conv
menu_blocks
menu_comp
menu_type

2.1. Bloques ideales

Tras el estudio realizado de convertidores Sigma Delta en Tiempo Discreto, llegamos a la conclusión que necesitamos incluir tres tipos de bloques: integradores, comparadores y cuantizadores[1].

A modo de vista global veamos el fichero que contiene todos los bloques diseñados:



Y a continuación los modelos desarrollados por componentes.

2.1.1. Integradores

Los modelos propuestos son:

Fig.7.1. Bloques Ideales: Integradores.

Se tratan de dos modelos propuestos de integradores en tiempo discreto cada uno con una función de transferencia visible en el símbolo de cada bloque.

Para este caso no presento máscaras de los bloques, ya que el modelo está

[1] Bloques básicos como sumas, restas y productos, presentes en cualquier modelo de simulación, no los vamos a considerar para esta librería, de forma que el usuario tendrá que acudir a la librería propia de Simulink para hacer uso de ellos.

formado por ese único elemento.

La variable utilizada la vemos en la Tabla 7.1:

Variable simulador	Variable física	Comentario
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.1. Variables: Integradores.

2.1.2. Comparadores

Los modelos propuestos son:

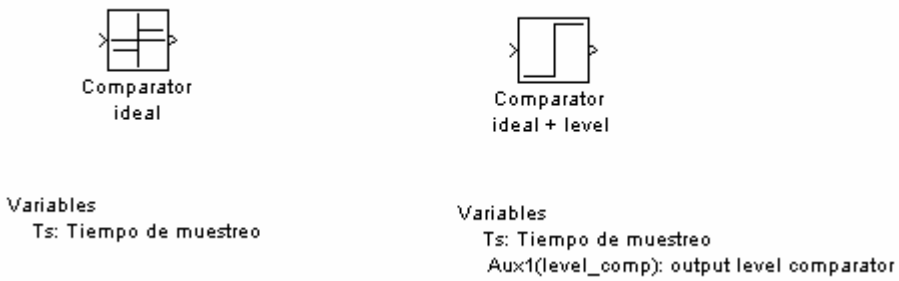


Fig. 7.2. Bloques Ideales: Comparadores.

El primer modelo se trata también un bloque básico tomado de la librería de Simulink. Su salida está dada entre [-1,1]. El segundo de ellos ha sido desarrollado para tener un comparador cuya salida de un cierto nivel distinto del anterior (level_comp).

La máscara para este bloque está formada tan sólo por dos variables, Ts que es el tiempo de muestreo y level_comp, nivel de salida del comparador. Los valores los introduciremos en la interfaz según la Tabla 7.1:

Variable simulador	Variable física	Comentario
Aux1	level_comp	Nivel de salida del comparador
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.2. Variables: Comparadores.

Su diseño viene dado en la Fig. 7.3.

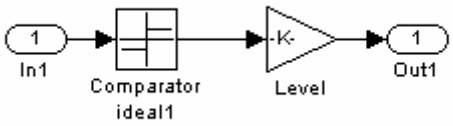
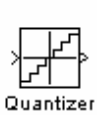
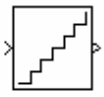


Fig. 7.3. Diseño comparador ideal + level

2.1.3. Cuantizadores

Los modelos propuestos son:





Variables:
Aux1(FSq): full scale of the quantizer
Aux2(Nstep): number of steps
Ts: sample time

Variables:
Aux1(FSq): full scale of the quantizer
Aux2(Nstep): number of steps
Ts: sample time

El primer modelo es un bloque básico de cuantizador de la librería de Simulink donde la salida viene dada entre (-∞,∞). El segundo es un cuantizador desarrollado para obtener a la salida la característica propia de un cuantizador, es decir, cuya salida es equivalente al fondo de escala del cuantizador (FSq). Estos límites de la salida viene dados por las fórmulas:

$$\text{Límite superior: } \frac{FSq}{2} - \left(\frac{FSq}{2Nstep}\right) \tag{7.1}$$

$$\text{Límite inferior: } -\frac{FSq}{2} + \left(\frac{FSq}{2Nstep}\right) \tag{7.2}$$

Los valores que introduciremos en la interfaz vienen dados en la Tabla 7.3:

Variable simulador	Variable física	Comentario
Aux 1	FSq	Fondo de escala del cuantizador
Aux2	Nstep	Niveles del cuantizador
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.3. Variables: Cuantizadores.

El diseño de este segundo cuantizador viene dado en la Fig. 7.4:

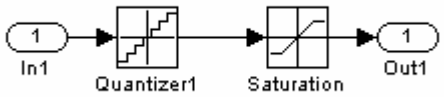


Fig. 7.4. Diseño cuantizador con nivel de salida

2.2. Bloques no ideales

Tal y como comentamos en el Capítulo 2 de convertidores A/D Sigma Delta, éstos presentan una topología para digitalización de señales analógicas con una alta resolución y caracterizadas por tener un ancho de banda (BW) mucho menor que la frecuencia de muestreo (fs).

Centrándonos en convertidores de tiempo discreto que es el punto que nos ocupa, la implementación más común utilizada es mediante capacidades conmutadas (SC, del inglés switched-capacitor). El diagrama de un convertidor sigma delta de primer orden es mostrado en la Fig.7.5:

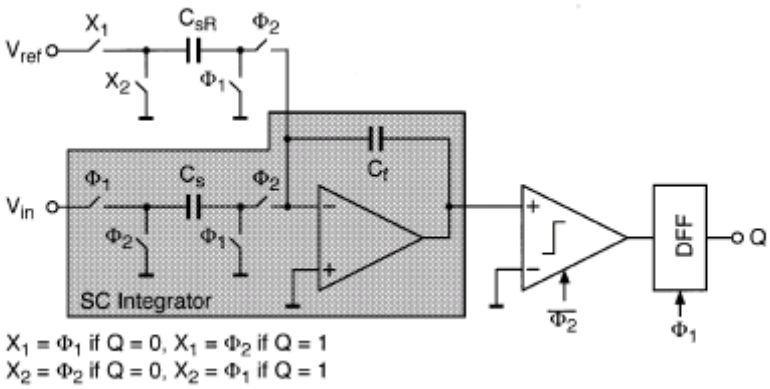


Fig. 7.5. Esquema de un modulador ∑Δ SC de primer orden

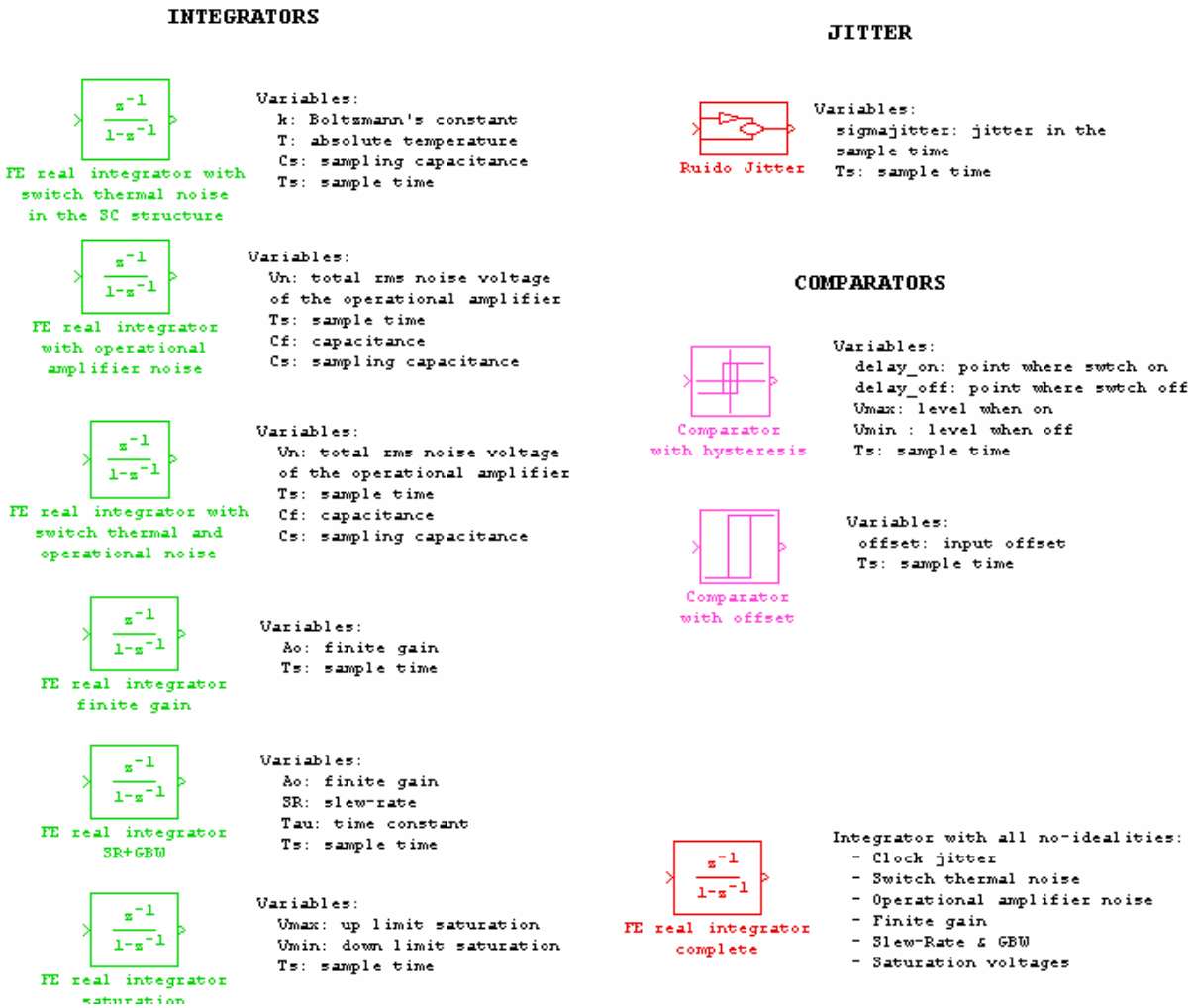
El modulador consiste en una entrada muestreada, un integrador con capacidades conmutadas, un cuantizador y un cuantizador digital-analógico.

Las principales no idealidades de estos circuitos son las siguientes:

1. Jitter en el reloj de muestreo de la entrada
2. Ruido térmico en la estructura SC
3. Ruido en el amplificador operacional
4. Ganancia finita del amplificador operacional

5. Ancho de banda del amplificador operacional
6. Slew-rate del amplificador operacional
7. Tesión de saturación del amplificador operacional
8. Histéresis en el comparador
9. Offset en el comparador

Una visión global de los diseños realizados:



Veamos el estudio de cada uno de ellos, y el modelo de simulación encontrado para cada uno.

2.2.1. Integradores

2.2.1.1. Ruido térmico en la estructura SC

El modelo propuesto es el siguiente:

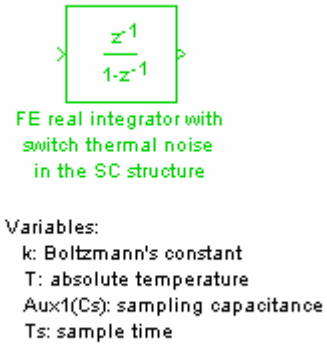


Fig. 7.6. Bloques no ideales: Integradores. Ruido térmico SC.

Los valores que necesitaremos introducir en la interfaz para su simulación son:

Variable simulador	Variable física	Comentario
Aux1	Cs	Capacidad de muestreo
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.4. Variables: Integrador. Ruido térmico SC.

La máscara que nos encontramos al pulsar sobre este tipo de no idealidad es el siguiente:

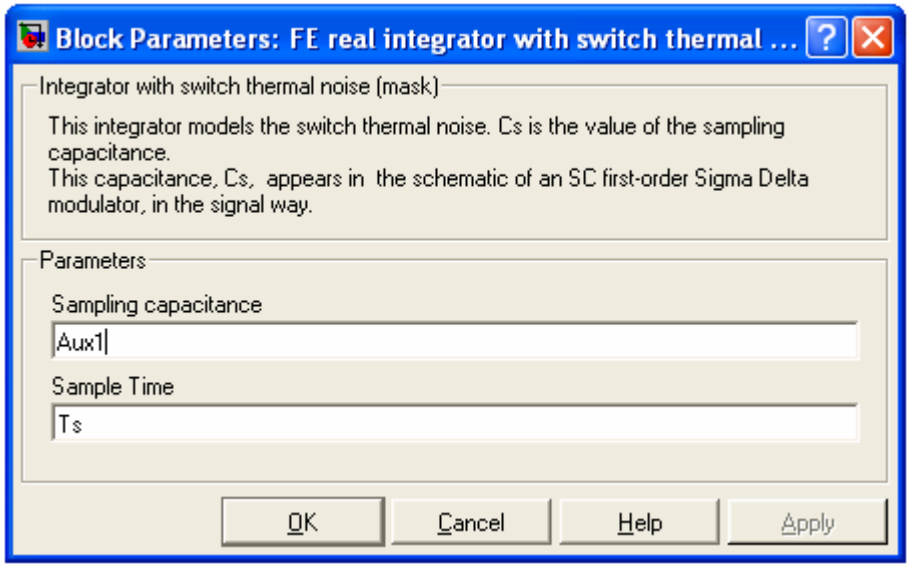


Fig. 7.7. Bloques no ideales: Máscara. Ruido térmico SC.

El diseño correspondiente para esta no idealidad ha sido diseñada bajo la siguiente estructura de bloques:

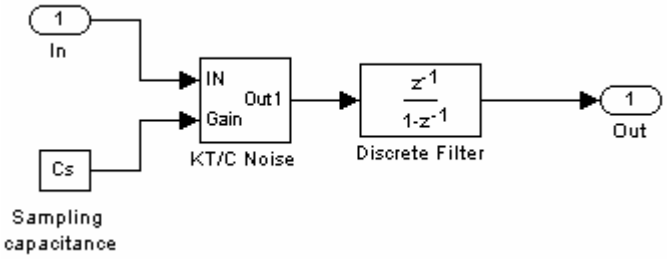


Fig. 7.8. Bloques no ideales: Diseño. Ruido térmico SC.

El bloque KT/C Noise de la Fig.7.8 está representado en la Fig.7.9

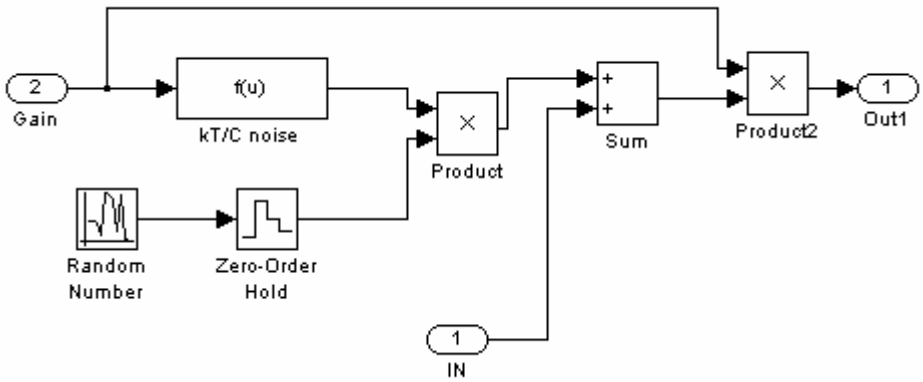


Fig. 7.9. Bloques no ideales: Bloque KT/C Noise.

2.2.1.2. Ruido en el amplificador operacional

El modelo para esta no idealidad lo representamos con el bloque:

Variables:
Aux3(Vn): total rms noise voltage of the operational amplifier
Aux2(Cf): capacitance
Aux1(Cs): sampling capacitance
Ts: sample time

Fig. 7.10. Bloques no ideales: Integradores. Ruido amplificador operacional.

Los valores que necesitaremos introducir en la interfaz para su simulación son tres, y cada uno de ellos lo hacemos corresponder con una variable del simulador. Esta equivalencia la mostramos en la siguiente Tabla 7.5, donde además podemos ver el significado físico de cada uno, así como una breve explicación de a que se refiere cada variable en nuestro modelo.

Variable simulador	Variable física	Comentario
Aux1	Cs	Capacidad de muestreo
Aux2	Cf	Capacidad del amplificador
Aux3	Vn	Ruido total rms del A.O.
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.4. Variables: Integrador. Ruido amplificador operacional.

El cociente de los valores Cs/Cf representa el coeficiente del integrador. Para ello, a partir de la Fig.x.5 podemos obtener la función de transferencia en dominio z, dada por:

$$H_I(z) = \frac{Cs}{Cf}\frac{z^{-1}}{1-z^{-1}} = b\frac{z^{-1}}{1-z^{-1}} \qquad (x.3)$$

siendo b el coeficiente del integrador antes mencionado.

La máscara que nos encontramos al pulsar sobre este tipo de no idealidad es el siguiente:

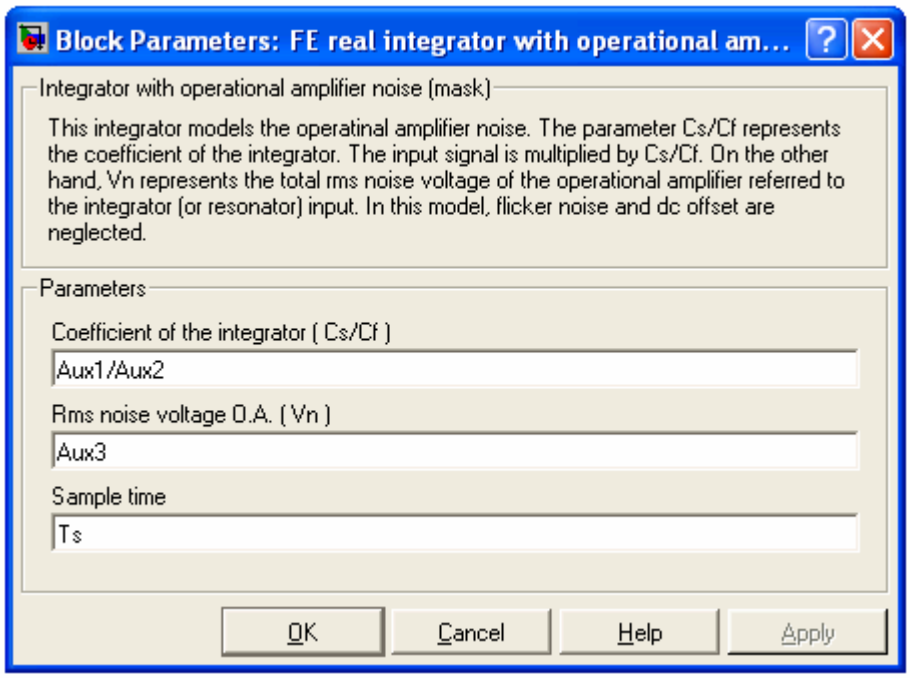


Fig. 7.11. Bloques no ideales: Máscara. Ruido amplificador operacional.

El diagrama de bloques al que corresponde es al siguiente:

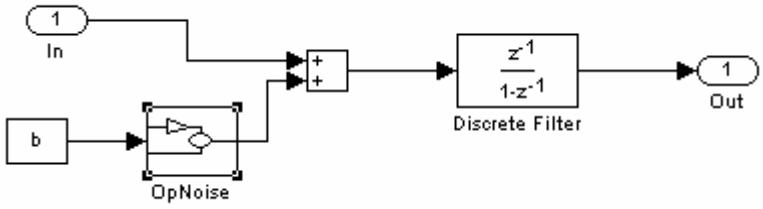


Fig. 7.12. Bloques no ideales: Diseño. Ruido amplificador operacional.

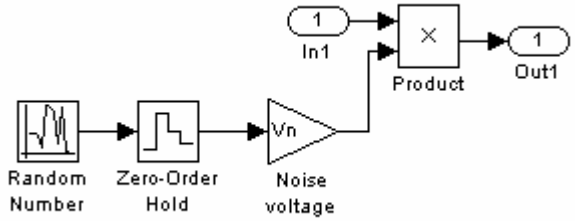


Fig. 7.13. Bloques no ideales: Bloque OpNoise.

2.2.1.3. Ganancia finita del amplificador operacional

El bloque que presentamos para este caso es el representado en la Fig.7.14:

Fig. 7.14. Bloques no ideales: Integradores. Ganancia finita.

Para este caso ambas variables coinciden con entradas propias del simulador, ya que Ts es la inversa de la entrada Sampling Freq, y Ao es la entrada de Gain en la zona de no idealidades.

La máscara realizada para este caso contiene por tanto las dos variables Ao y Ts, sin ser necesaria ninguna variable auxiliar:

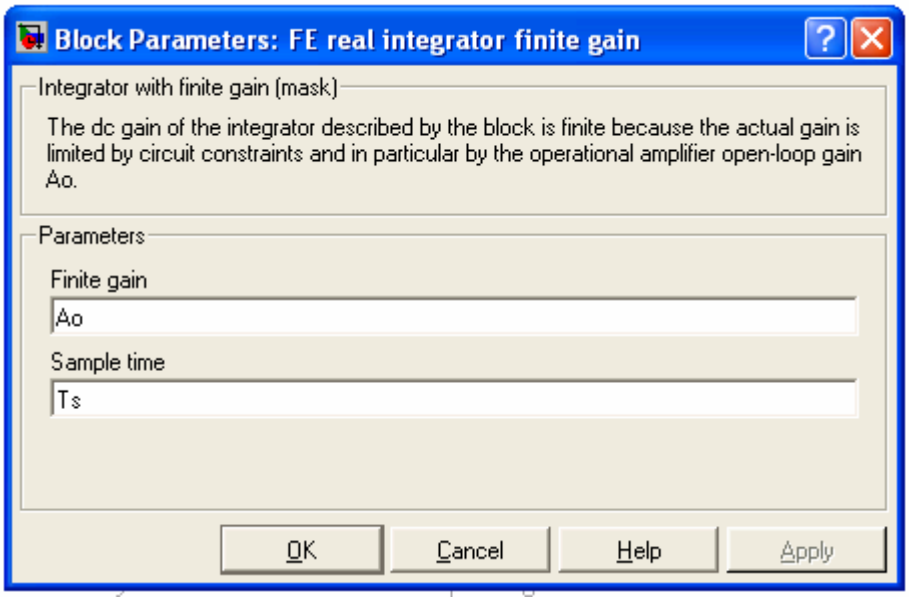


Fig. 7.15. Bloques no ideales: Máscara. Ganancia finita.

El diagrama de bloques diseñado para el caso de simular la ganancia finita procede de:

$$H(z)=\frac{z^{-1}}{1-\alpha\cdot z^{-1}}\xrightarrow{dcGain} Ao=H(1)=\frac{1}{1-\alpha}\xrightarrow{despejo_\alpha}\alpha=1-\frac{1}{Ao} \quad (7.4)$$

Por tanto consiste en añadir α al lazo de realimentación del integrador de la siguiente forma:

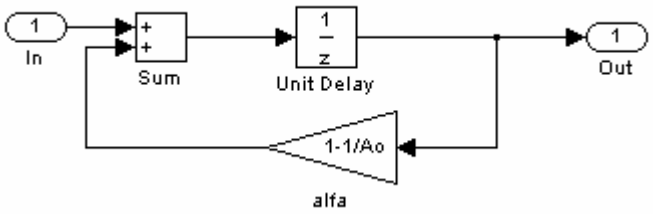


Fig. 7.16. Bloques no ideales: Diseño. Ganancia finita.

2.2.1.4. Ancho de banda del amplificador operacional y Slew-Rate

El modelo viene dado por:

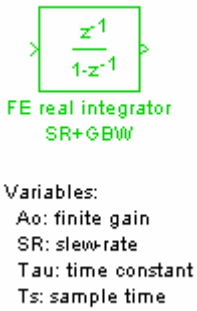


Fig. 7.17. Bloques no ideales: Integradores. SR+GBW.

Todas las variables forman parte de entradas de la interfaz del simulador. Ao (Gain), SR y Tau están en la zona de no idealidades y Ts (la inversa de Sampling Freq) en la zona de entradas.

La máscara hecha para este caso queda de la forma:

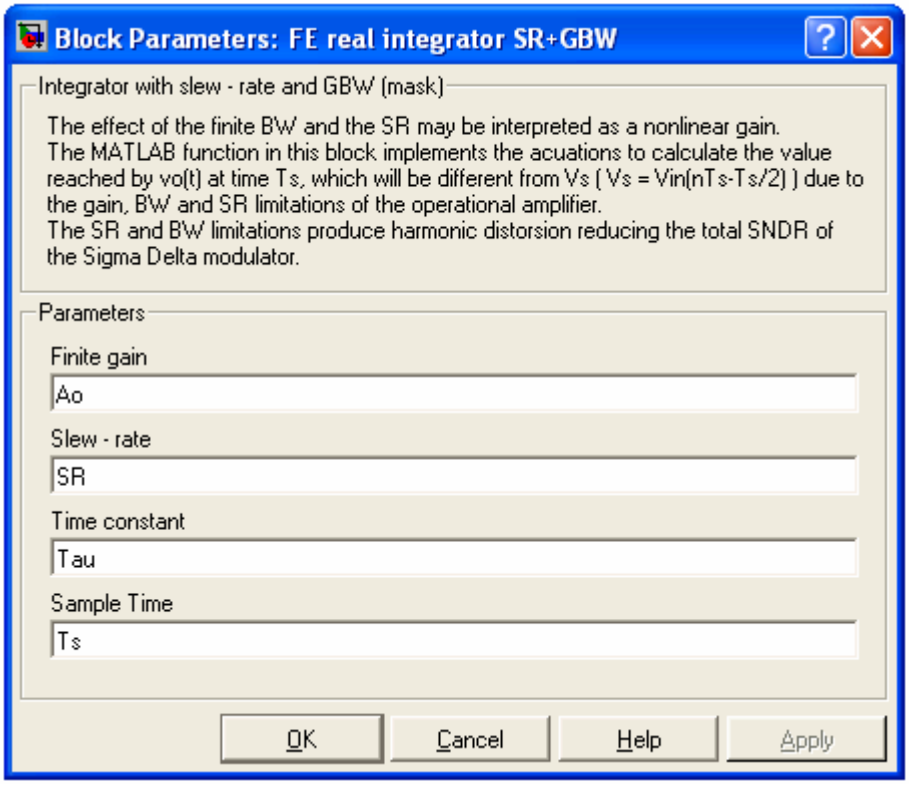


Fig. 7.18. Bloques no ideales: Máscara. SR+GBW.

El efecto del ancho de banda finito (BW) y del slew-rate (SR) puede ser interpretado como una no idealidad en la ganancia. Refiriéndonos al modulador de primer orden mostrado en la Fig.7.5, la evolución de la salida durante el periodo n de integración (cuando φ_2 está encendido) viene dada por:

$$vo(t) = vo(nTs - Ts) + \alpha Vs(1 - e^{-\frac{t}{\tau}}) \text{ en } 0<t<\frac{Ts}{2} \qquad (7.5)$$

donde $Vs = Vin(nTs - \frac{Ts}{2})$, α proviene de la ganancia finita del amplificador operacional, τ es la constante de tiempo del integrador e igual a $\tau = \frac{1}{2\pi GBW}$ y GBW es el producto ganancia ancho de banda.

El valor de esta curva toma el máximo cuando t=0, dando:

$$\frac{d}{dt}vo(t) = \alpha\frac{Vs}{\tau} \qquad (7.6)$$

Consideramos dos casos:

1. El valor de (7.6) es más pequeño que el SR del amplificador operacional. En

este caso el SR no limita la evolución de vo(t) durante todo un periodo de reloj (hasta t=Ts/2).

2. El valor de (7.6) es mayor que el SR, en ese caso:

$$\text{t} \le \text{to: } vo(t) = vo(nTs - Ts) + SRt$$

$$\text{t} > \text{to: } vo(t) = vo(to) + (\alpha Vs - SRto)\left(1 - e^{\frac{t-to}{\tau}}\right) \text{ siendo } to = \frac{\alpha Vs}{SR} - \tau \qquad (7.7)$$

Según todo esto, el modelo que tomamos la simular el efecto del SR y del ancho de banda, es el siguiente:

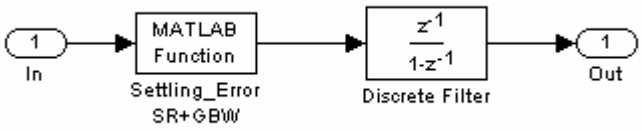


Fig. 7.19. Bloques no ideales: Diseño. SR+GBW.

A continuación mostramos la función matlab que cumple el desarrollo matemático expuesto anteriormente que modela los efectos de SR y del GBW en amplificadores operacionales:

```
% Calculates the gain error due to settling and slew-rate

function y = settlinggain(x,Ao,SR,Tau,Ts)

% output = settlinggain(input,Av,SR,Tau,Ts)

alfa=1-1/Ao;
to=alfa*x/SR-Tau;

if x<(Tau*SR/alfa)
    y=alfa*x*(1-exp(-Ts/2/Tau));
end

if (x>=(Tau*SR/alfa))&(to>=ts/2)
    y=SR*Ts/2;
end

if (x>(Tau*SR/alfa))&(to<Ts/2)
    y=SR*to+(alfa*x-SR*to)*(1-exp(-1*(Ts/2-to)/Tau));
end
```

2.2.1.5. Tensión de saturación del amplificador operacional

El modelo utilizado es el siguiente:

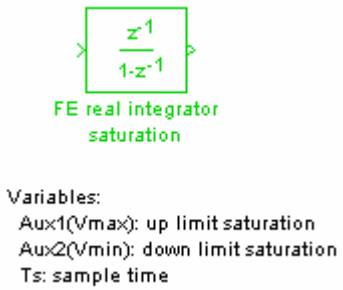


Fig. 7.20. Bloques no ideales: Integradores. Saturación.

Las variables que tomaremos para su simulación y que deberemos introducir en la interfaz de simconverter son las siguientes:

Variable simulador	Variable física	Comentario
Aux1	Vmax	Limite de saturación superior
Aux 2	Vmin	Limite de saturación inferior
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.5. Variables: Integrador. Saturación.

La máscara estará formada por estas variables:

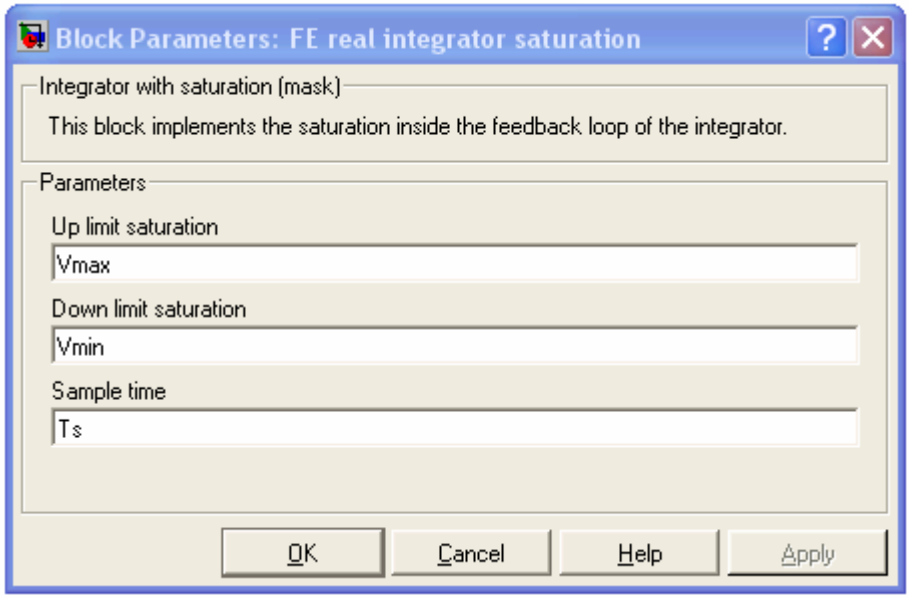


Fig. 7.21. Bloques no ideales: Máscara. Saturación.

El modelo de bloques desarrollado para este caso es muy sencillo, y consiste en saturar la salida del integrador entre unos ciertos niveles (Vmax y Vmin).

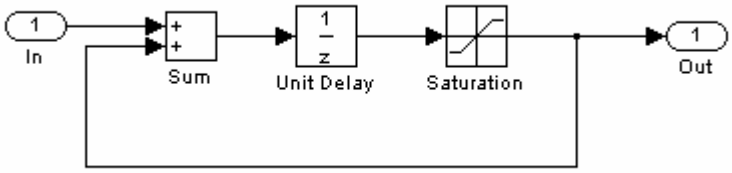


Fig. 7.22. Bloques no ideales: Diseño. Saturación.

2.2.1.6. Integrador con todas las no idealidades

Presentamos un modelo de integrador en el que hemos unido todas las no idealidades vistas hasta ahora para integradores de convertidores analógicos digitales Sigma Delta. El modelo en cuestión, que se encuentra dentro de la librería de integradores de los bloques no ideales, es el siguiente:

Fig. 7.23. Bloques no ideales: Integrador no ideal completo.

En este caso en el que el número de no idealidades es superior[2] al valor de

[2] Que el número de no idealidades sea superior al de variables auxiliares no hace que nuestra interfaz sea limitada, ya que si recordamos, las variables auxiliares nos permiten dar valores a no idealidades pero tienen como objetivo realizar un barrido de los valores de éstas respecto a otras. En caso de tener más no idealidades que variables auxiliares, una opción es que el usuario se cree un archivo con extensión .m con las variables que quiere modelar y sus valores, haciendo coincidir el nombre de las variables con el de la máscara. Otra forma también válida es sustituir directamente el valor en la máscara del bloque del que vamos a estudiar la no idealidad de aquellas

variables auxiliares que dispone por ello hay que tener especial cuidado con dar valor a todas las variables que no correspondan con las variables auxiliares y no sean entradas de nuestra interfaz, por ejemplo para el caso que dispongo el valor b, que es el coeficiente del integrador.

Para indicar cada una de estas variables que visualizaremos más adelante en la máscara, detengámonos primero en la Tabla 7.6 que nos muestra cada una de ellas, tanto su nombre físico como el que debemos buscar el la interfaz del simulador para sustituir su valor.

Variable simulador	Variable física	Comentario
Aux1	Vmax	Límite de saturación superior
Aux2	Vmin	Límite de saturación inferior
Aux3	Cs	Capacidad de muestreo
Aux4	Vn	Ruido total rms del A.O.
-	b	Coeficiente del integrador
Gain	Ao	Ganancia finita
SR	SR	Slew-rate
Tau	Tau	Constante de tiempo
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.6. Variables: Integrador no ideal completo.

La máscara realizada es la siguiente:

variables en las que no vamos a necesitar hacer un barrido de las mismas.

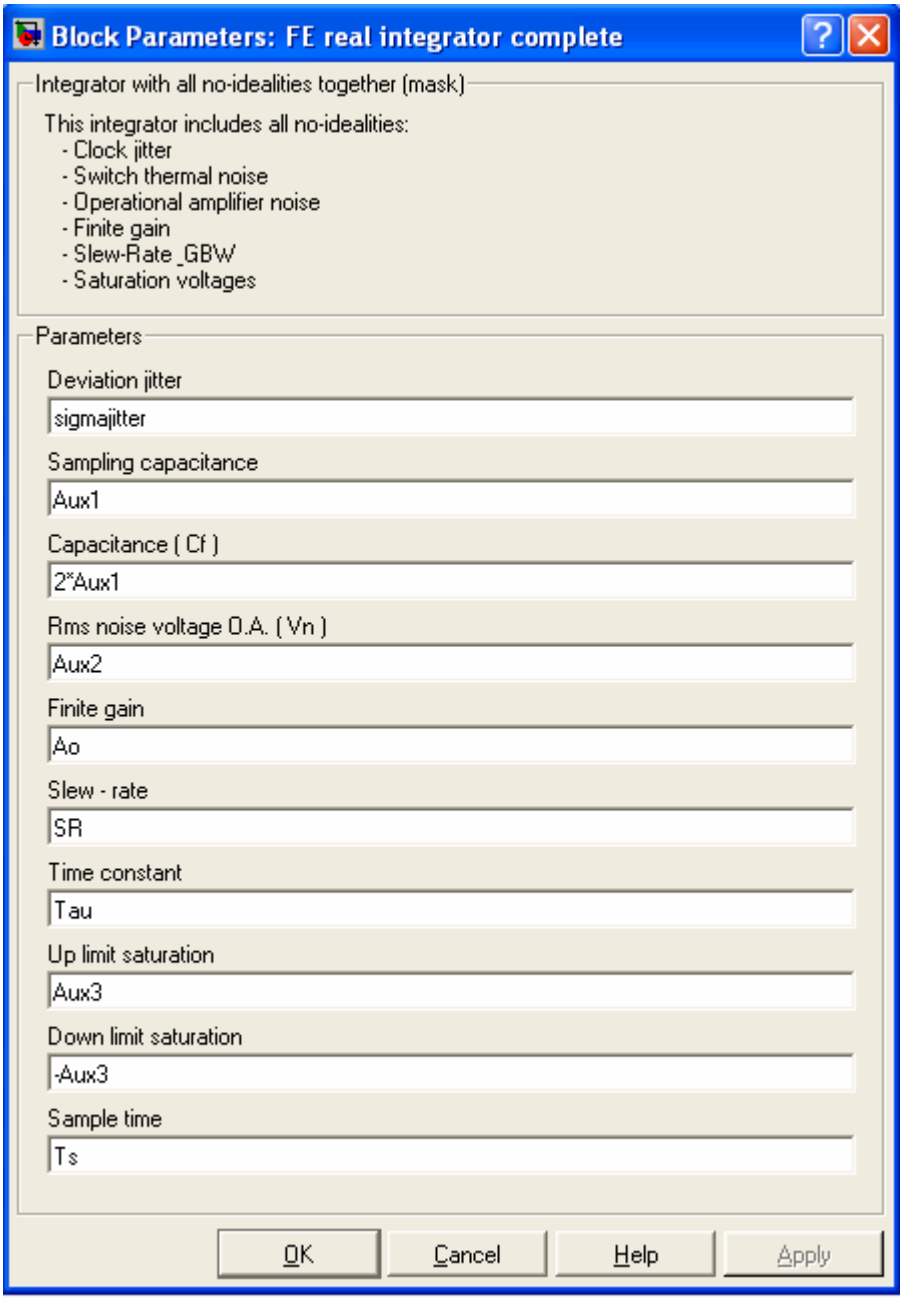


Fig. 7.24. Máscara. Integrador no ideal completo.

Consideramos que Cf va a ser siempre el doble que Cs para que el coeficiente de integración sea ½. Por simplificación también consideramos que Vmax y Vmin tienen los mismos valores absolutos y por ello tomamos la misma variable pero negada.

2.2.2. Comparadores

2.2.2.1. Histéresis

El modelo que proponemos para el caso de la histéresis de los comparadores:

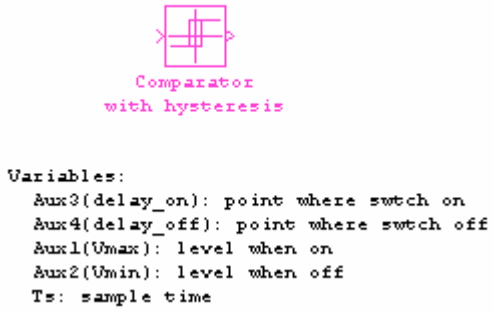


Fig. 7.25. Bloques no ideales: Comparadores. Histéresis.

Este modelo pertenece a la librería de simulink, por tanto tan sólo debemos adaptar las variables a las entradas de la interfaz.

2.2.2.2. Offset

El modelo que hemos diseñado es el siguiente:

Fig. 7.26. Bloques no ideales: Comparadores. Offset.

Las variables que utilizaremos para su situación, y que deberemos introducir en la interfaz de simconverter son los mostrados en la Tabla 7.6

Variable simulador	Variable física	Comentario
Aux1	offset	Offset del comparador
Sampling Freq	Ts	Tiempo de muestreo

Tabla 7.7. Variables: Comparador. Offset.

Según esto, la máscara realizada para este bloque queda:

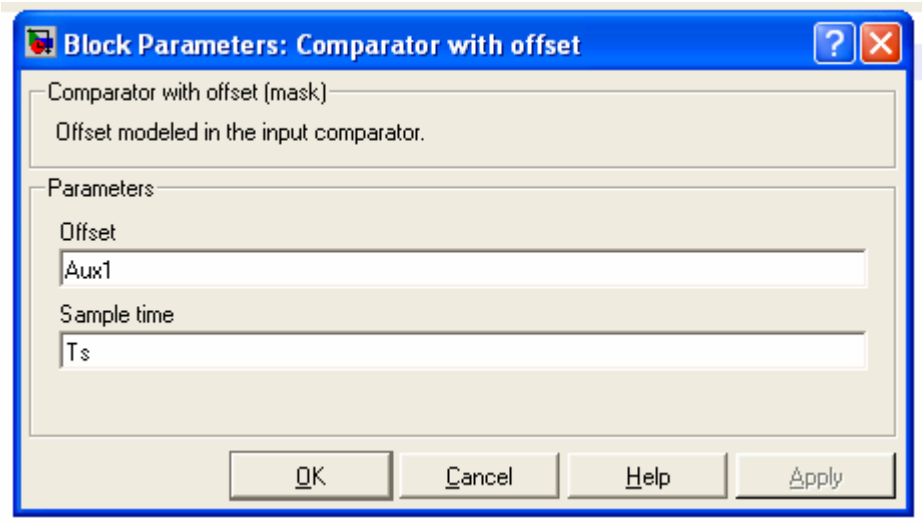


Fig. 7.27. Bloques no ideales: Máscara. Offset.

El diseño implementado para la simulación del offset, consiste sencillamente en sumarle una constante antes de realizar la comparación.

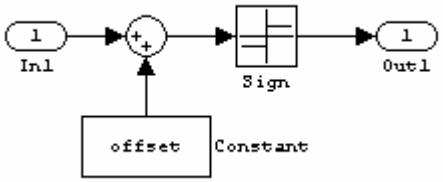
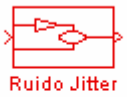


Fig. 7.28. Bloques no ideales: Diseño. Offset.

2.2.3. Ruido Jitter

El modelo que hemos diseñado viene dado por el siguiente bloque:



Variable:
sigmajitter: jitter in the sample time
Ts: sample time

Fig. 7.29. Bloques no ideales: Ruido Jitter.

Ambas variables, sigmajitter (desviación del ruido jitter) y Ts (tiempo de muestreo), son entradas de la interfaz del simulador, la primera dentro de la zona de no idealidades y la segunda dentro de la zona de entradas.

La máscara para este caso aparecería una ventana como la de la Fig.7.30:

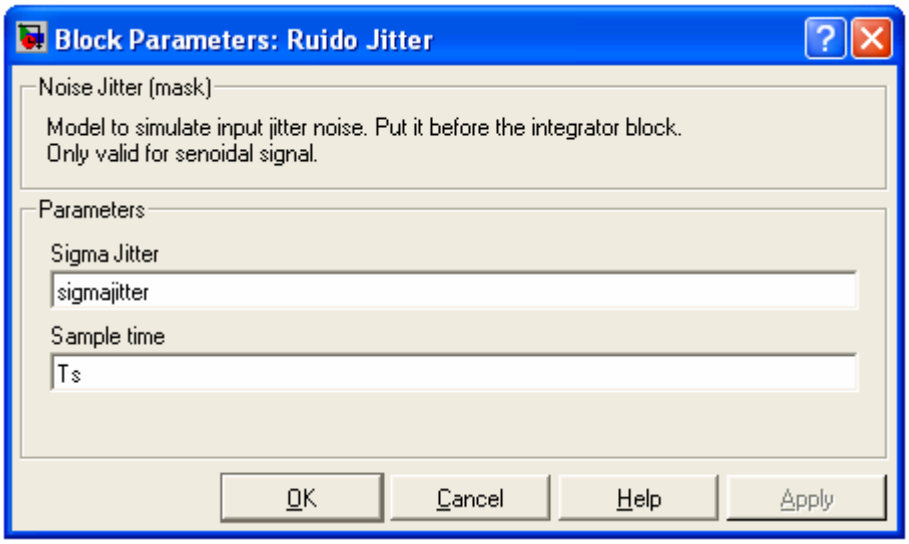


Fig. 7.30. Bloques no ideales: Ruido Jitter.

El efecto del ruido jitter en los moduladores Sigma Delta es independiente de la estructura u orden del modulador. El resultado es un muestreo no uniforme en la secuencia temporal y produce un error, el cual incrementa la potencia del ruido a la salida del cuantizador.

El error introducido cuando una señal sinusoidal a la entrada x(t) con

amplitud A y frecuencia fsin es muestreada en un instante con error δ está dado por:

$$x(t+\delta) - x(t) = 2\pi f_{\sin} \delta A \cos(2\pi f_{\sin} t) = \delta \frac{d}{dt} x(t) \qquad (7.8)$$

El diagrama de bloques diseñado para emular el ruido jitter a la entrada siguiendo la fórmula (1), vendría dado por:

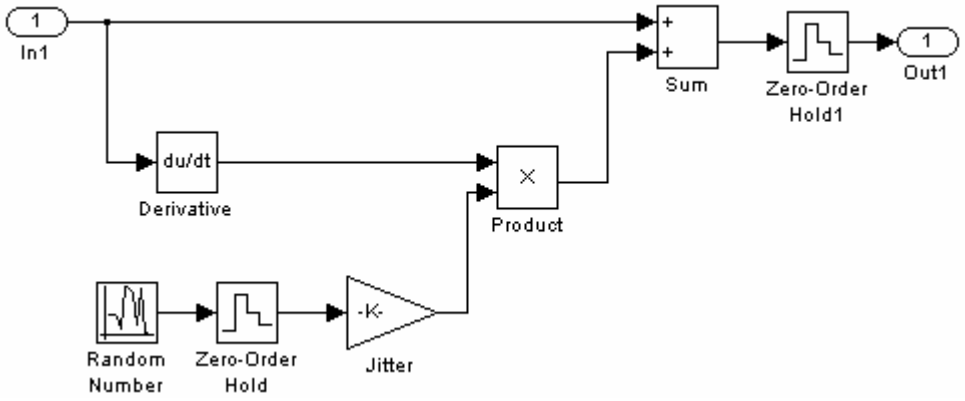


Fig. 7.31. Bloques no ideales: Diseño. Ruido Jitter.

2.3. Ejemplos

En este apartado vamos a mostrar algunos ejemplos de convertidores Sigma Delta en Tiempo Discreto. En concreto hemos diseñado cuatro que cito a continuación:

1. ∑Δ de 2º orden, paso de baja, lazo simple y componentes ideales.
2. ∑Δ de 2º orden, paso de baja, lazo simple y componentes no ideales.
3. ∑Δ de 2º orden, paso de baja, cascada y componentes ideales.
4. ∑Δ de 2º orden, paso de baja, cascada y componentes no ideales.

Estos ejemplos nos ayudarán a la compresión de este tipo de convertidores o nos podrán servir de base para nuestro modelo.

Pasamos a realizar una descripción a nivel de bloques de estos ejemplos diseñados, y en el Capítulo 8 mostraremos los resultados obtenidos tras la simulación de algunos de estos ejemplos. Pero antes, mostremos todos los ejemplos realizados para sigma delta, y así tener una visión general de esta parte de la librería.

Sampling Freq: 1/Ts =
FS: FS =
Nbits: N =
Amplit: A =
Frec: fsin =
sigmajitter: sigmajitter =
Values in the interface of the simulator:
Inl
Outl
TD Sigma Delta LP 2° order simple ideal
TD Sigma Delta LP 2° order - simple real
Cs (sampling capacitance): Aux1=
Gain: AodB =
TD Sigma Delta LP 2° order simple - real - switch thermal
TD Sigma Delta LP 2° order simple - real - gain
Un(total rms noise voltage of the operational amplifier): Aux3 =
Cs/Cf (coefficient of the integrator): Aux1/Aux2 =
Gain: AodB =
SR: SR =
Tau: Tau =
TD Sigma Delta LP 2° order simple - real - op noise
TD Sigma Delta LP 2° order simple - real - SR & GBW
Cs (sampling capacitance): Aux1=
Cf (capacitance): Aux2=
Un(total rms noise voltage of the operational amplifier): Aux3 =
Up limit saturation: Aux1=
Down limit saturation: Aux2 =
TD Sigma Delta LP 2° order simple - real - op noise & switch thermal
TD Sigma Delta LP 2° order simple - real - sat
Input
Output
ADC to ADSL+ aplication: 14 bits and BW 2.2 MHz ideal
ADC to ADSL+ aplication: 14 bits and BW 2.2 MHz real
Sampling Freq: 1/Ts =
FS: FS =
Nbits: N =
Amplit: A =
Frec: fsin =
sigmajitter: sigmajitter =
Values in the interface of the simulator:

2.3.1. ∑Δ de 2º orden, paso de baja, lazo simple y componentes ideales

El modelo en estudio viene definido por el bloque:

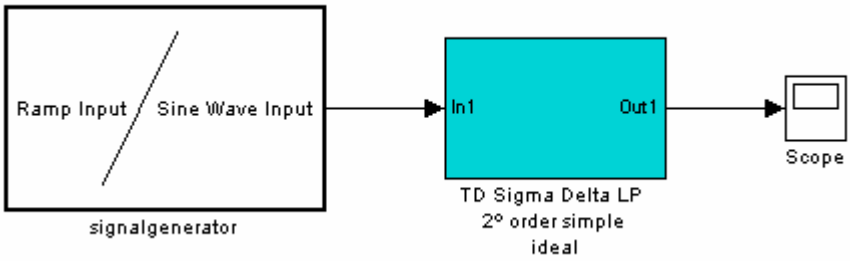


Fig. 7.32. Ejemplo: ∑Δ de 2º orden, paso de baja, lazo simple y componentes ideales.

Los valores de simulación[3] de la entrada para este ejemplo son:

- Sampling Freq: 1/Ts = 1600 Hz
- FS: FS = 2 V
- Nbits: N = 8
- Amplit: A = 0.5 V
- Frec: fsin = 70 Hz

O también otra forma de cargar estos valores es mediante la opción Load del menú Environment de la interfaz de simconverter. Para ello cargamos el archivo que contenga el mismo nombre que el ejemplo.

El diseño considerado para este ejemplo lo mostramos a continuación:

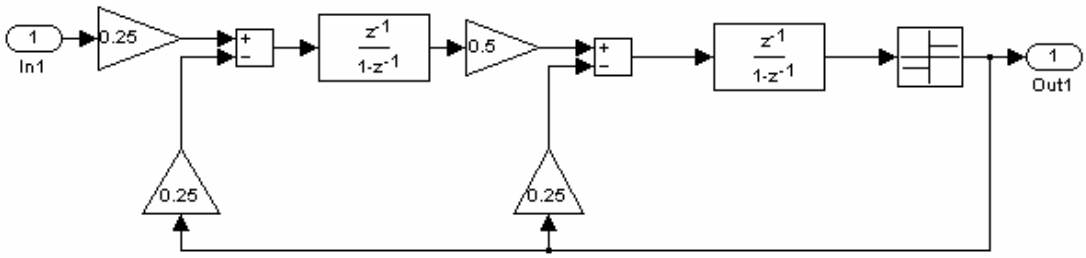


Fig. 7.33. Diseño: ∑Δ de 2º orden, paso de baja, lazo simple y componentes ideales.

[3] Estos valores de simulación son orientativos y pueden ser modificado por el usuario en la interfaz.

2.3.2. ∑Δ de 2º orden, paso de baja, lazo simple y componentes no ideales

Este modelo es similar al anterior, con la salvedad que hemos introducido componente no ideales al diseño. El bloque correspondiente a este ejemplo:

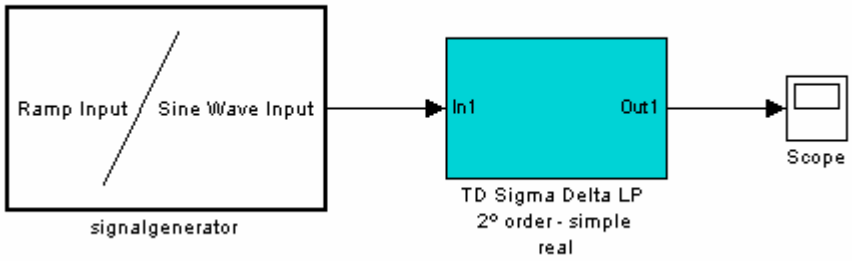


Fig. 7.34. Ejemplo: ∑Δ de 2º orden, paso de baja, lazo simple y componentes no ideales.

Para este diseño los valores de simulación[4] son:

- Sampling Freq: 1/Ts = 1600 Hz
- FS: FS = 2 V
- Nbits: N = 8
- Amplit: A = 0.5 V
- Frec: fsin = 70 Hz
- Gain: AodB = 22 dB
- Cs[5]: Aux1 = 1e-12
- Vn: Aux2 = 10e-3
- Vmax[6]: Aux3 = 0.65
- Sigmajitter = 3e-5

O también otra forma, al igual que el caso anterior, es cargar estos valores es mediante la opción Load del menú Environment de la interfaz de simconverter.

El diseño considerado para este ejemplo, es a partir de modelo real, añadir las no idealidades de jitter a la entrada de la señal y ruido en los amplificadores[7].

[4] Estos valores de simulación son orientativos y se cargan automáticamente. El usuario podría modificarlos por otros en la interfaz.
[5] El valor de Cf es el doble de Cs para tener un coeficiente de integración igual a ½.
[6] Tomaremos el valor de Vmin igual al negado del valor absoluto de Vmax.
[7] Los modelos de jitter noise y del integrador real ya han sido mostrados en el apartado 2.2.4 y 2.2.1.2 de este mismo capítulo.

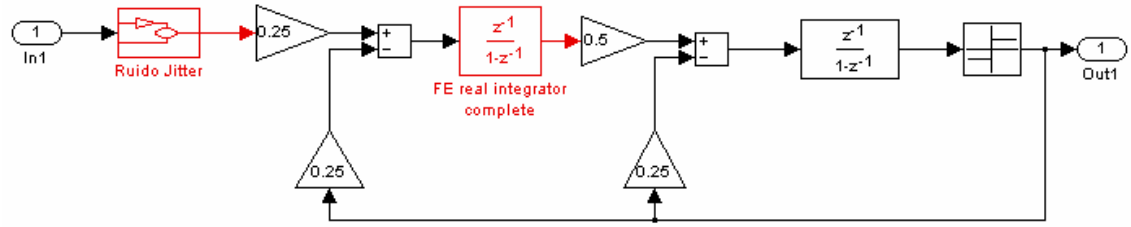


Fig. 7.35. Diseño: ∑Δ de 2º orden, paso de baja, lazo simple y componentes no ideales.

2.3.3. ∑Δ de 2º orden, paso de baja, cascada y componentes ideales

Presentamos un bloque que nos va a simular en mismo caso anterior, pero para ello vamos a utilizar un modelo en cascada. El bloque que nos representará este ejemplo es el siguiente:

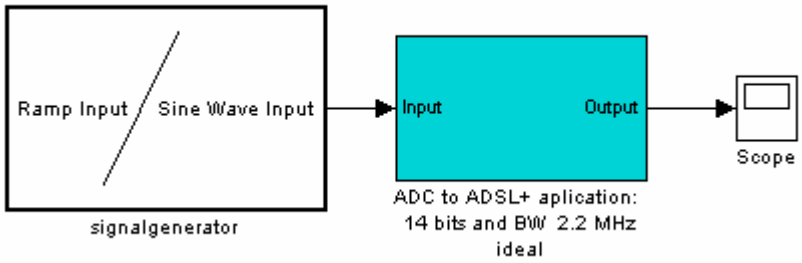


Fig. 7.36. Ejemplo: ∑Δ de 2º orden, paso de baja, cascada y componentes ideales.

Los valores de simulación que tomaremos para el caso de una aplicación ADSL+ son los siguientes:

- Sampling Freq: 1/Ts = 704e6 Hz
- FS: FS = 2 V
- Nbits: N = 14
- Amplit: A = 0.5 V
- Frec: fsin = 550 Hz

Estos valores también podremos cargarlos a partir de la opción Load del menú Environment de la interfaz.

El diseño desarrollado queda representado en la siguiente ilustración:

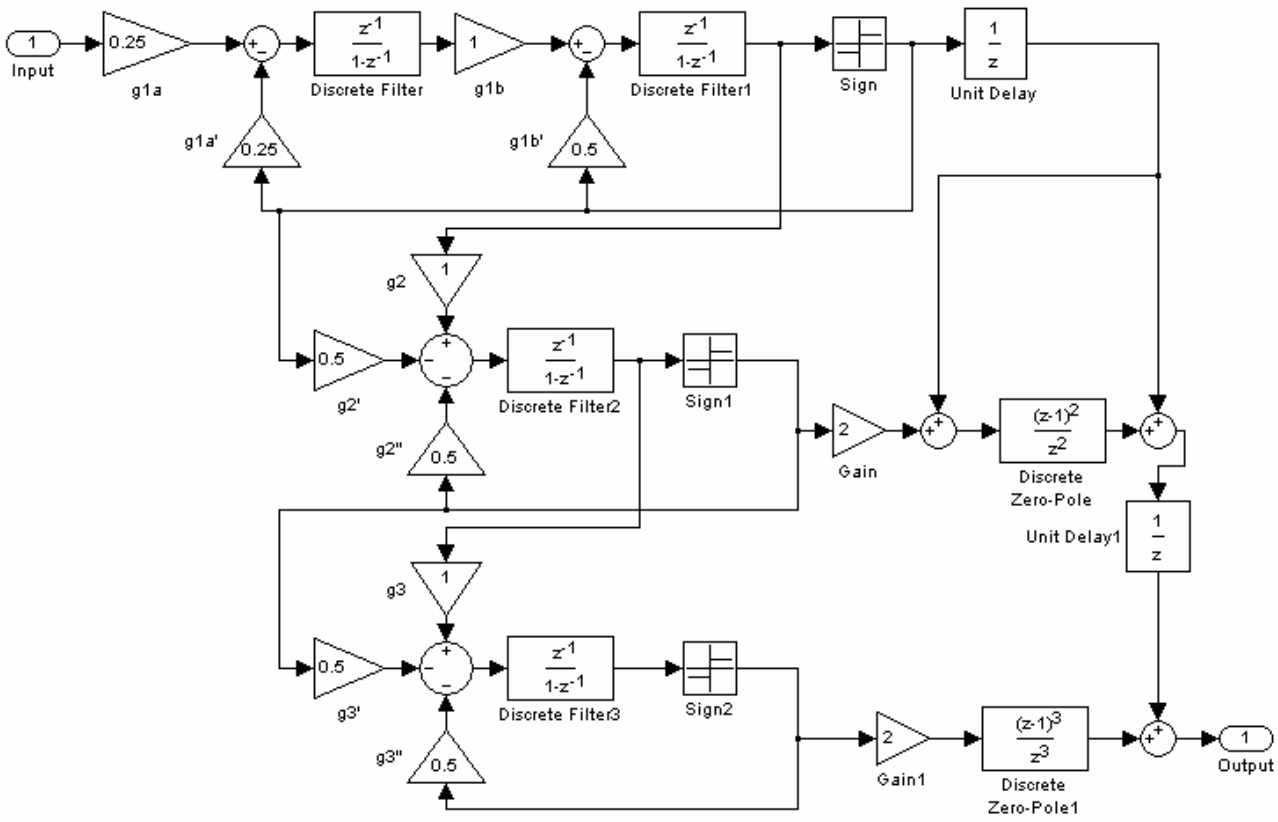


Fig. 7.37. Diseño: ∑Δ de 2º orden, paso de baja, cascada y componentes ideales.

2.3.4. ∑Δ de 2º orden, paso de baja, cascada y componentes no ideales

El bloque que mostramos en este apartado es similar al anterior, con la salvedad que algunos elementos del mismo no son ideales.

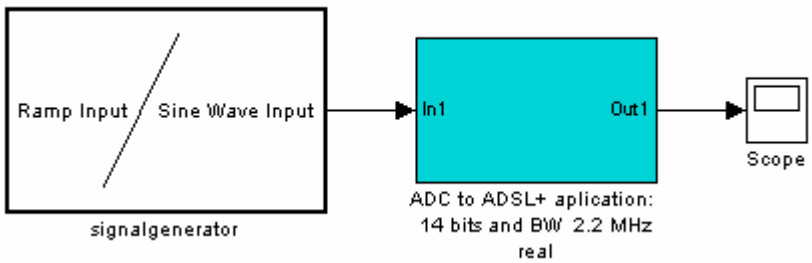


Fig. 7.38. Ejemplo: ∑Δ de 2º orden, paso de baja, cascada y componentes no ideales.

Los valores que tomaremos para la simulación serán los correspondientes a la señal sinusoidal de entrada, así como a las no idealidades de ruido jitter a la entrada y de ruido de los amplificadores operacionales:

- Sampling Freq: 1/Ts = 704e6 Hz
- FS: FS = 2 V
- Nbits: N = 14
- Amplit: A = 0.5 V
- Frec: fsin = 550 Hz
- Gain: AodB = 30 dB
- Cs: Aux1 = 6pF
- Vn: Aux2 = 1.7e-5
- Vmax: Aux3 = 0.7
- Sigmajitter = 40 ps

Estos valores también podremos cargarlos por medio del menú Environment de la interfaz.

El diseño para este caso, es similar al de el caso ideal, sustituyendo los elementos reales:

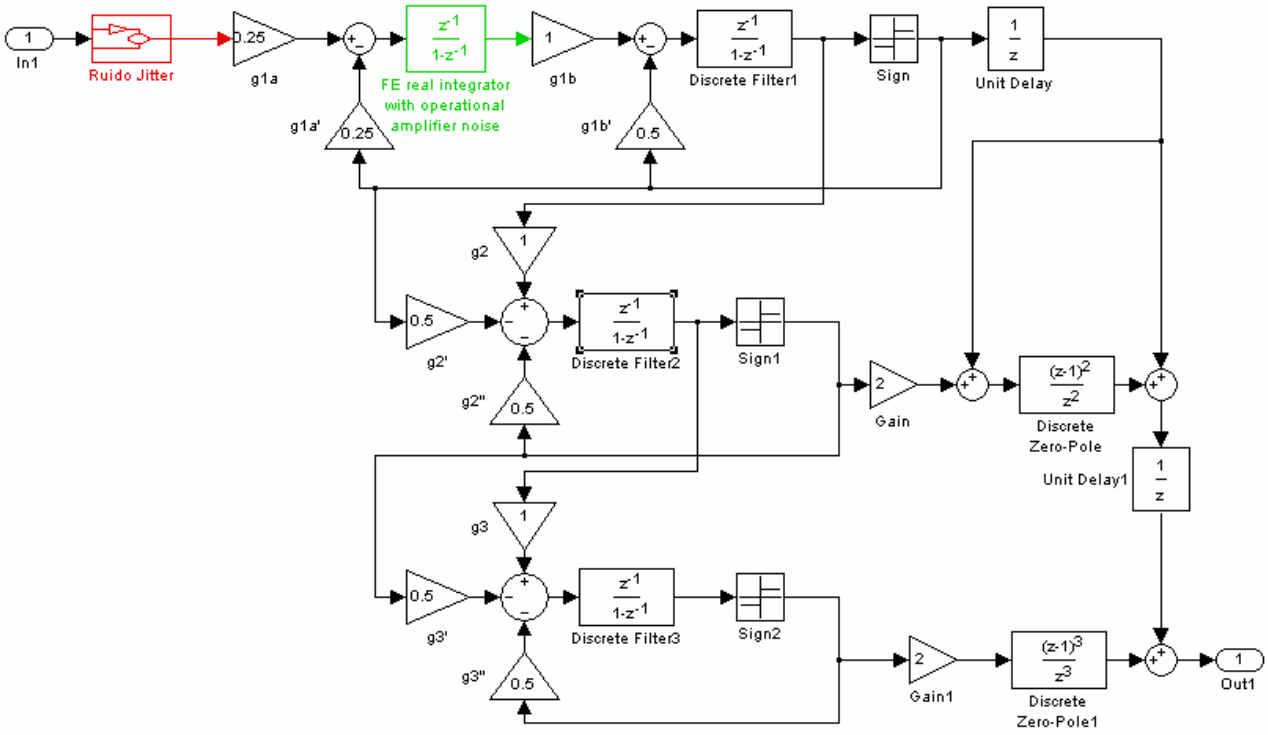
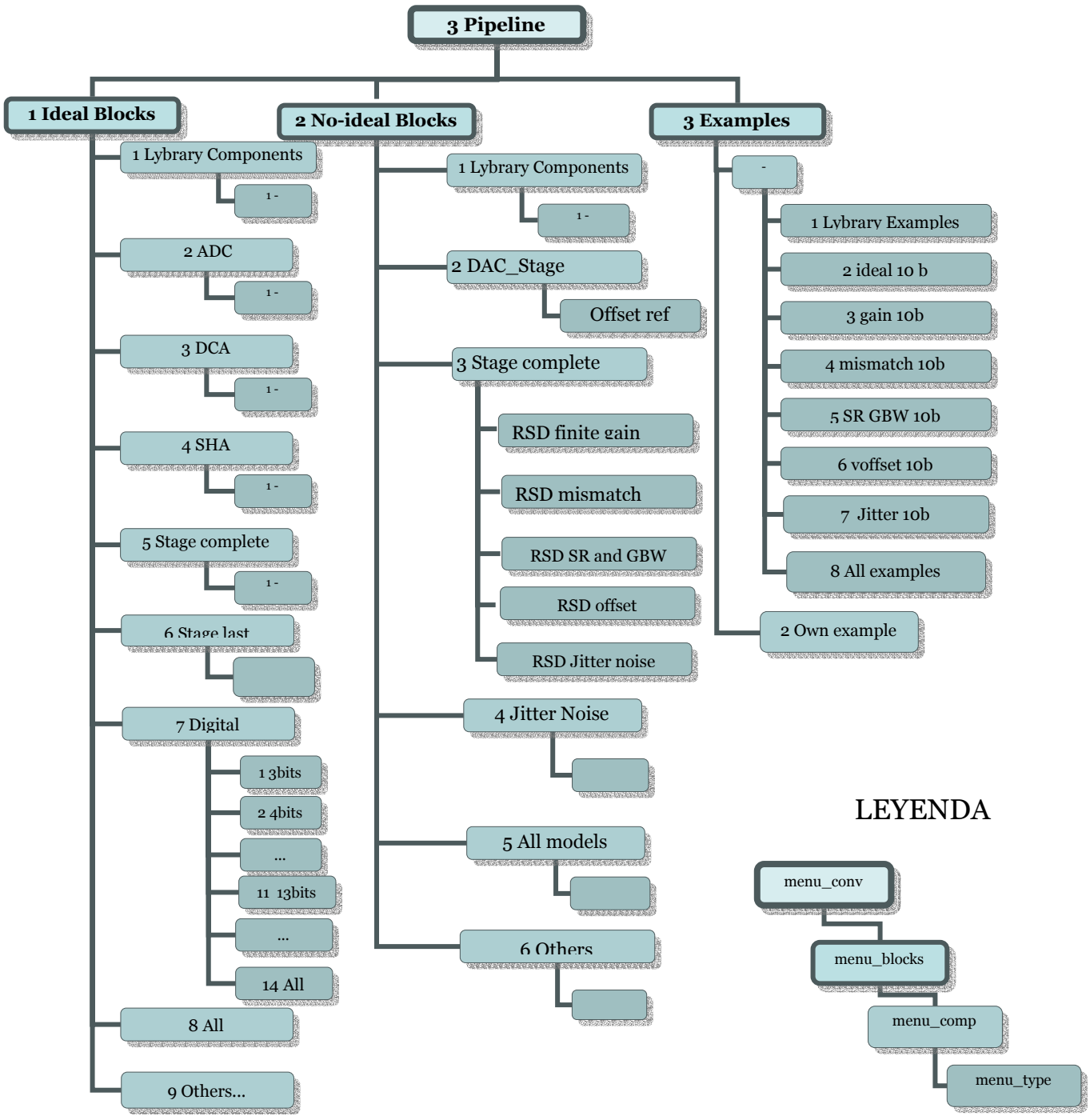


Fig. 7.39. Diseño: ∑Δ de 2º orden, paso de baja, cascada y componentes no ideales.

3. Librería de Pipeline

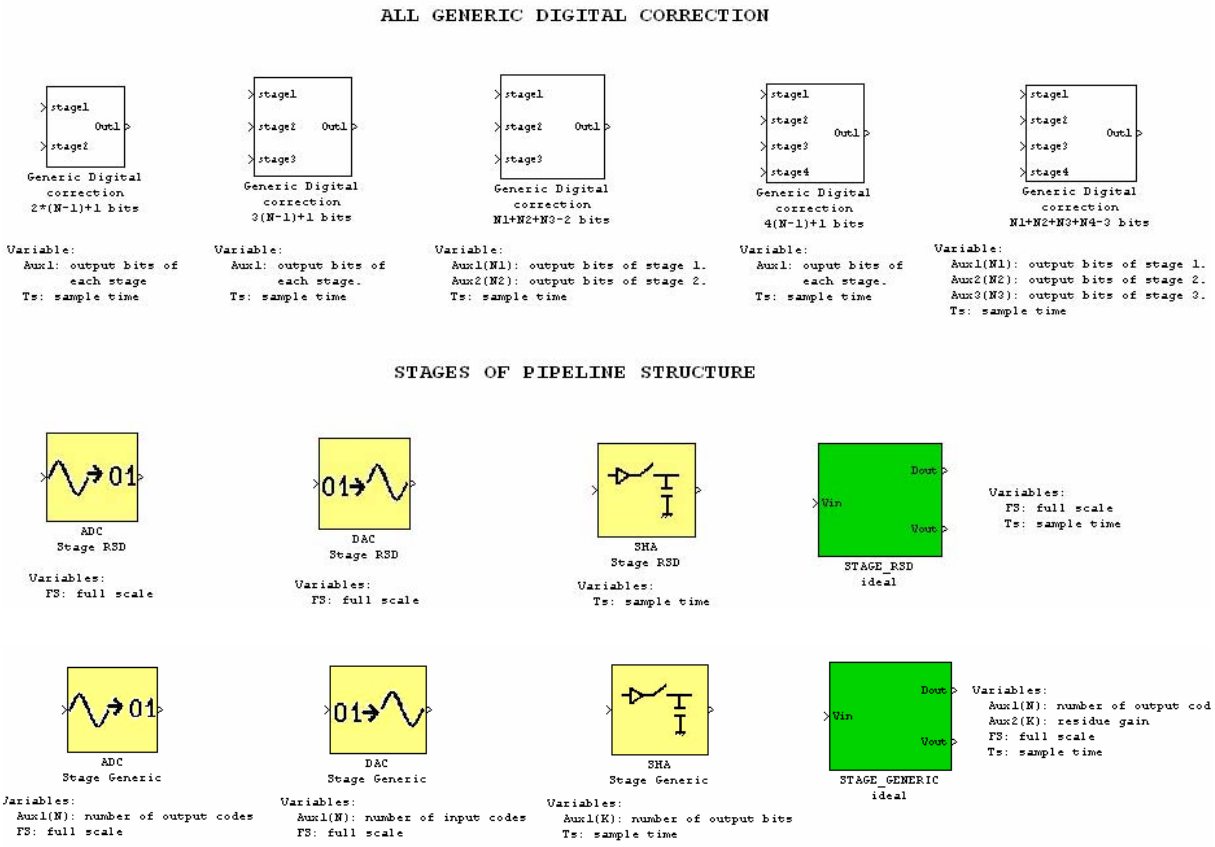
Para tener una visión completa de la librería, muestro a continuación el siguiente esquema en forma de árbol que incluye a grande rasgos los modelos diseñados para convertidores pipeline:

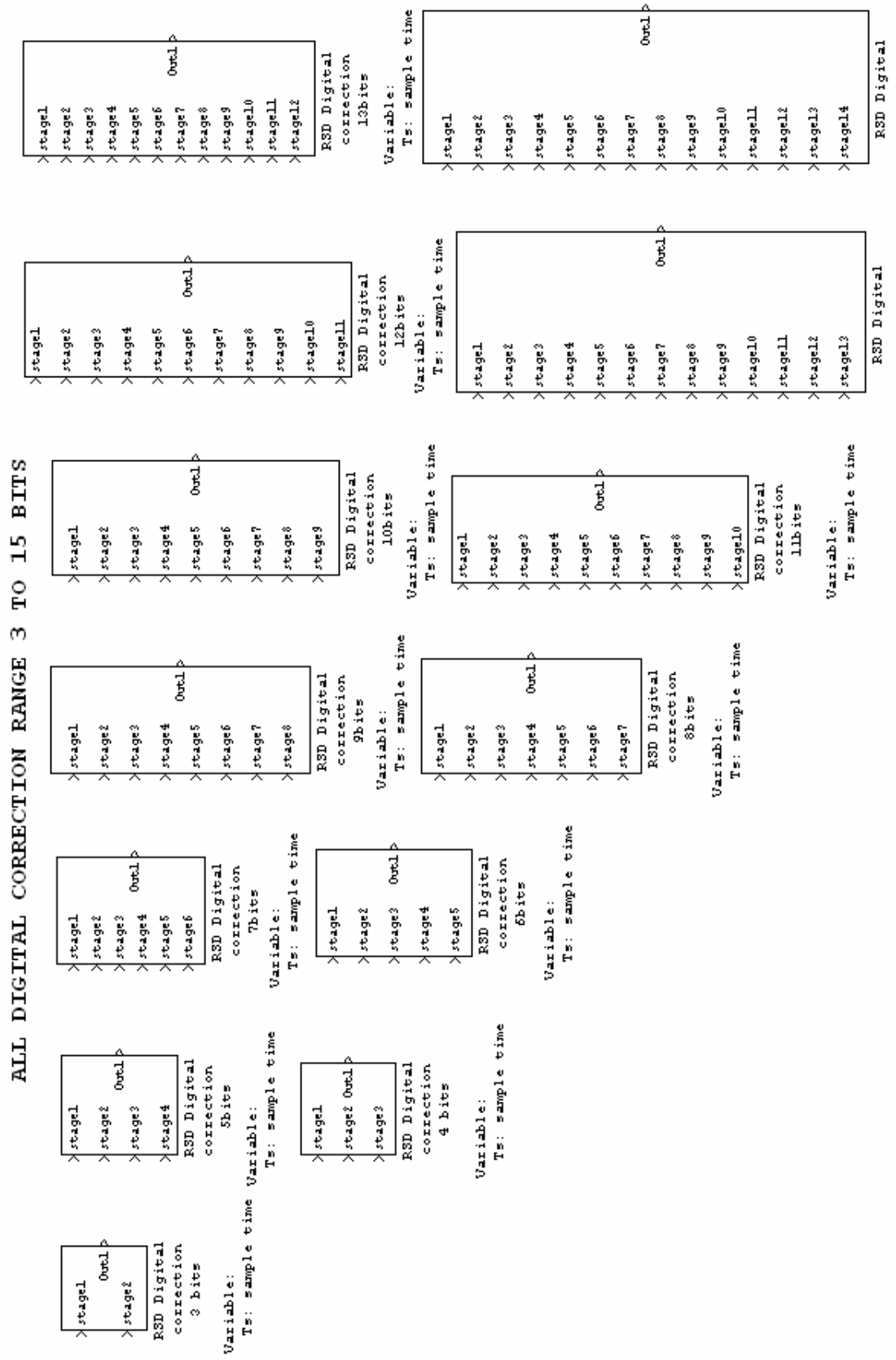


3.1. Bloques ideales

Tras el estudio realizado de convertidores Pipeline, llegamos a la conclusión que necesitamos incluir tres tipos de bloques: ADC, DAC, SHA, Stage completo y último de la cadena, corrección digital. Para cada uno de estos bloques veremos el caso RSD, explicado en el capítulo 3, y el caso genérico.

Una visión global de los bloques generados es la siguiente:




ALL DIGITAL CORRECTION RANGE 3 TO 15 BITS
stage1
stage2
Out1
RSD Digital correction 3 bits
Variable:
Ts: sample time
stage3
stage4
RSD Digital correction 5bits
stage5
stage6
RSD Digital correction 7bits
stage7
stage8
RSD Digital correction 9bits
stage9
RSD Digital correction 10bits
stage10
RSD Digital correction 11bits
stage11
RSD Digital correction 12bits
stage12
RSD Digital correction 13bits
stage13
stage14
RSD Digital

Veamos a continuación los modelos desarrollados para cada uno de ellos.

3.1.1. ADC_Stage

Los modelos desarrollados son:

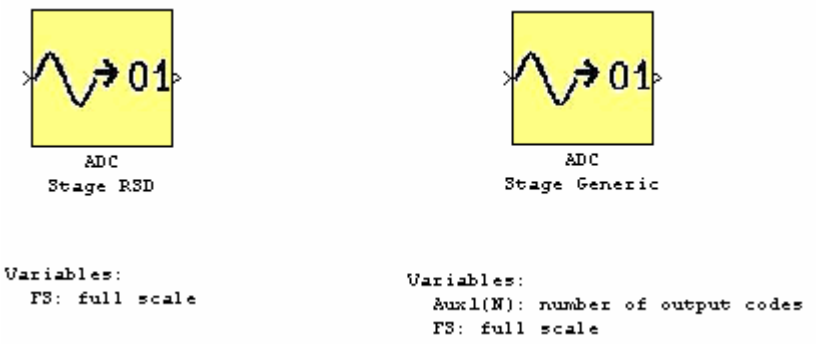


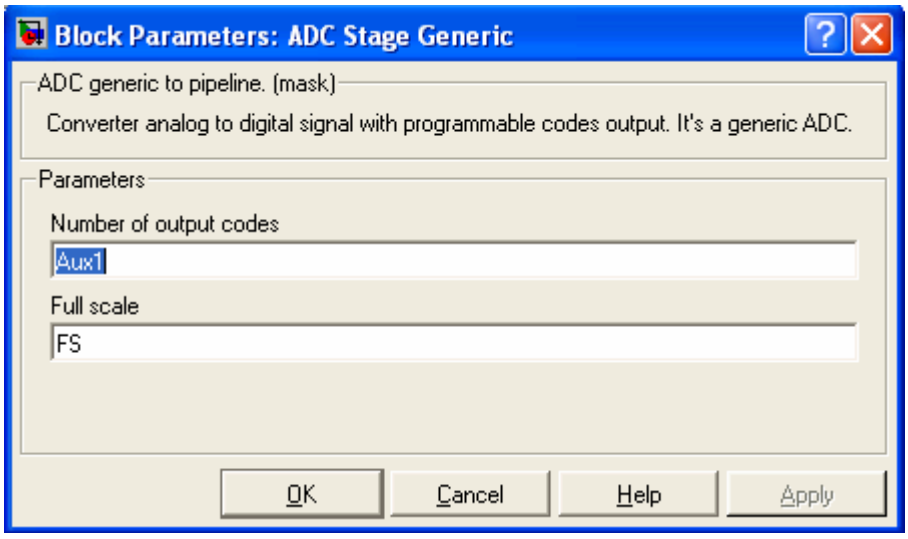
Fig.7.40. Bloques ideales. ADC_Stage.

El primer bloque modela un ADC cuya salida tiene 2 códigos de salida. El segundo, esta variable es programable.

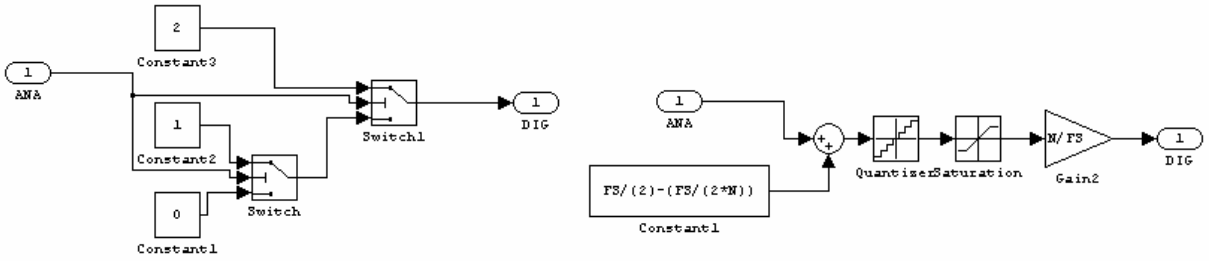
Variable simulador	Variable física	Comentario
Aux1	N	Número de códigos de salida
FS	FS	Fondo de escala

Tabla.7.8. Variables: Bloques ideales. ADC_Stage.

Al pinchar sobre ellos nos aparece la máscara creada. Por ejemplo la del ADC genérico es:



Los diseños de cada uno son los siguientes:



(a) **(b)**

Fig.7.41. (a) Diseño ADC_Stage RSD (b) Diseño ADC_Stage genérico

3.1.2. DAC_Stage

Para el caso del DAC, los modelos son:

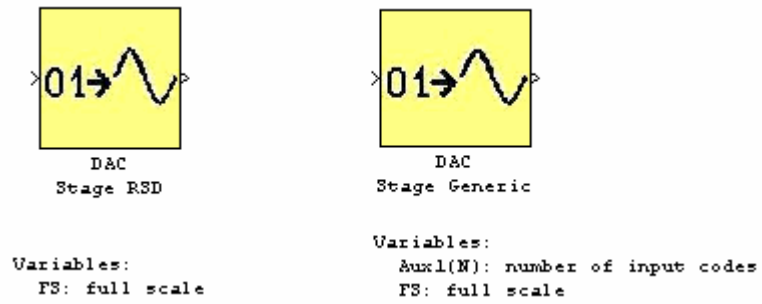


Fig.7.42. Bloques ideales. DAC_Stage.

Las variables son como las del ADC pero con la conversión contraria, FS para el fondo de escale, y ahora N, como el número de códigos de entrada (igual a la N del DAC).

El diseño en bloques Simulink desarrollado es el siguiente:

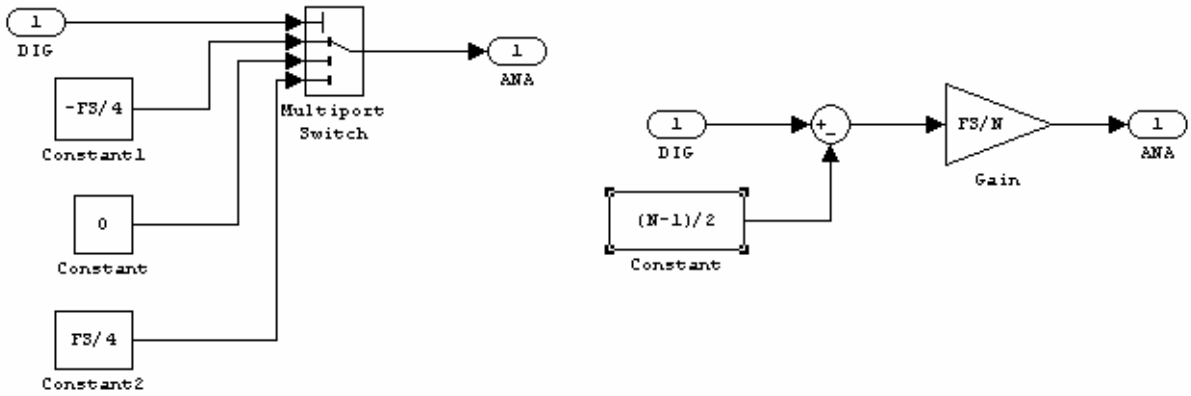


Fig.7.43. (a) Diseño DAC_Stage RSD (b) Diseño DAC_Stage genérico

3.1.3. SHA_Stage

Los modelos para la función de sample and hold con ganancia son:

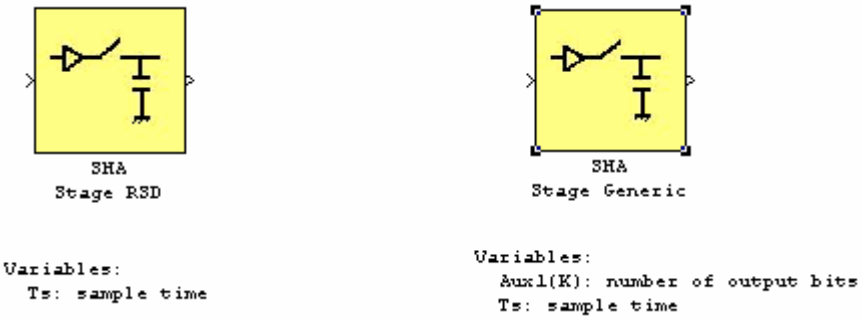


Fig.7.44. Bloques ideales. SHA_Stage.

Las variables son el tiempo de muestro y la ganancia que coincide con el número de códigos de salida, N, que vimos en el apartado del ADC.

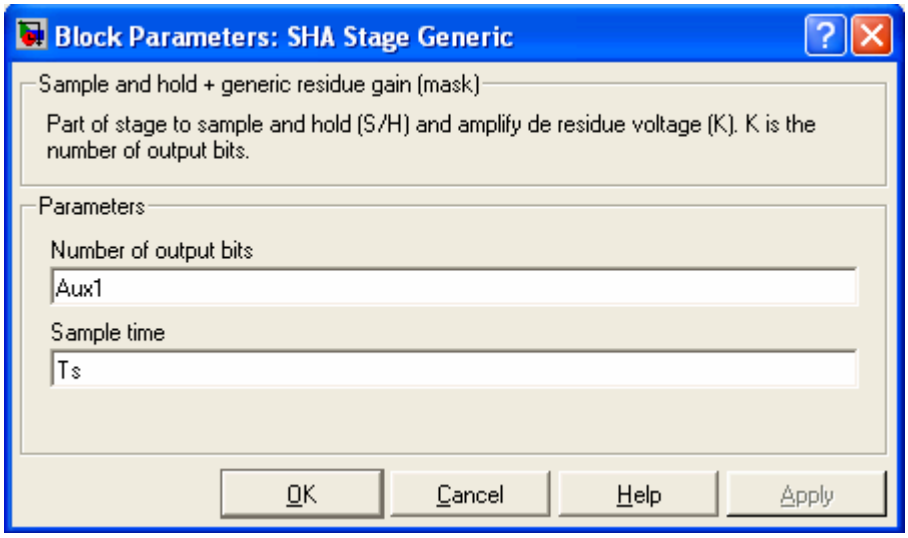
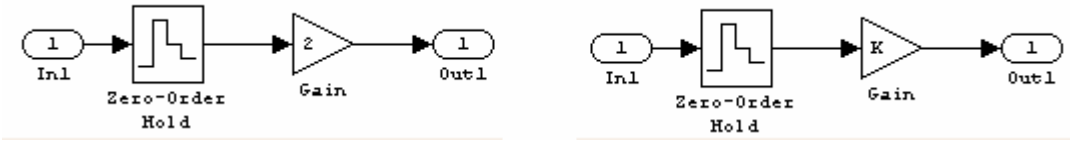


Fig.7.45. Variables: Bloques ideales. SHA_Stage.

Los diseños lo mostramos en la Fig. 7.45:



(a) **(b)**

Fig.7.46. (a) Diseño SHA_Stage RSD (b) Diseño SHA_Stage genérico

3.1.4. Stage_complete

Los modelos para RSD y genérico son:

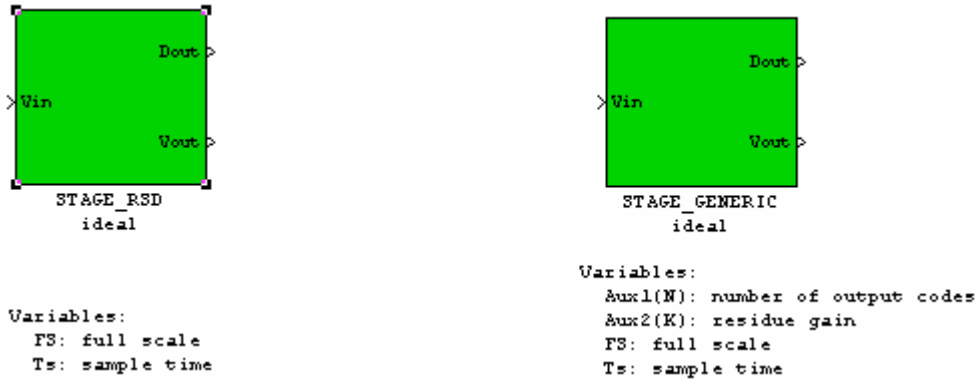


Fig.7.47. Bloques ideales. Stage_complete.

En la leyenda de cada bloque podemos ver las variables que necesitan. Pasemos ahora a mostrar los diseños a nivel e bloques de cada uno de ellos:

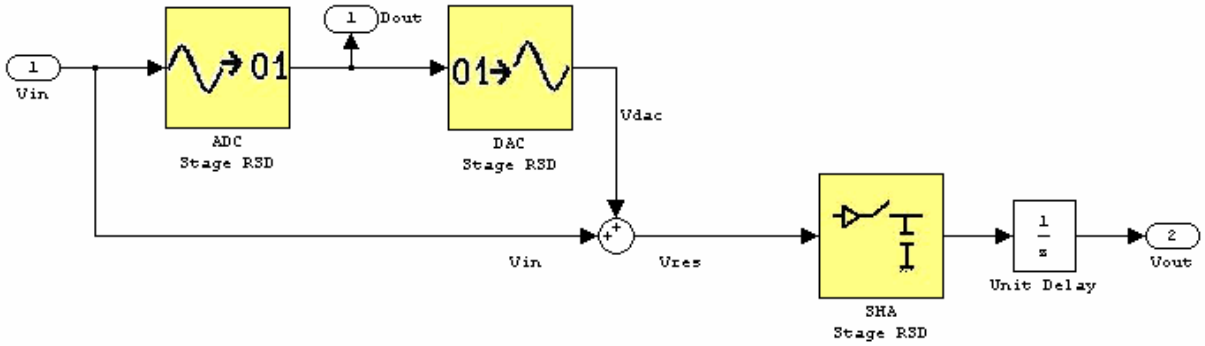


Fig.7.48. Diseño. Stage_complete. RSD.

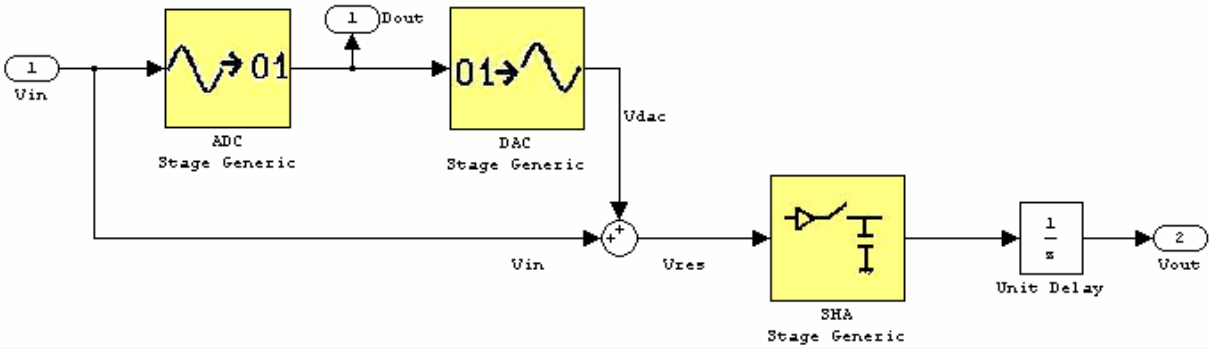


Fig.7.49. Diseño. Stage_complete. Generic.

El esquema en ambos es similar, tan sólo hay que cambiar bloques RSD por genéricos.

3.1.5. Stage last

Se ha desarrollado un modelo distinto para la última etapa, ya que en convertidores analógico-digitales pipeline, la última etapa de la cadena es un SDC genérico.

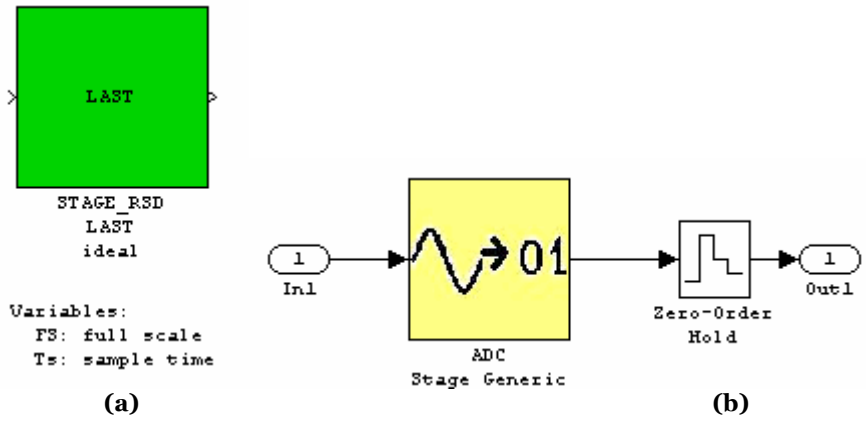


Fig.7.50. (a) Bloques ideales. Last_stage. (b) Diseño

3.1.6. Corrección digital

La corrección digital, posterior a la conversión, puede ser también para el caso de genérico o para RSD. Veamos por separado estos modelos.

3.1.6.1. Corrección digital RSD

Para no extendernos mostrando cada uno de los bloques realizados para la corrección digital RSD, mostraremos uno de ellos como ejemplo.

Se han realizado bloques de corrección de 3 a 15 bits. Cada uno de ellos tiene tan sólo como parámetro el tiempo de muestreo Ts. Estos bloques toman de entrada la salida de cada etapa RSD. La retrasan tantas veces como posiciones se encuentren en la cadena de pipeline para no haya retraso entre ellos y se multiplica por una ganancia. Esta ganancia es potencia de dos, 2^p , siendo p el número de retrasos.

Para el ejemplo de corrección digital de 12 bits, el modelo de diseño es el siguiente:

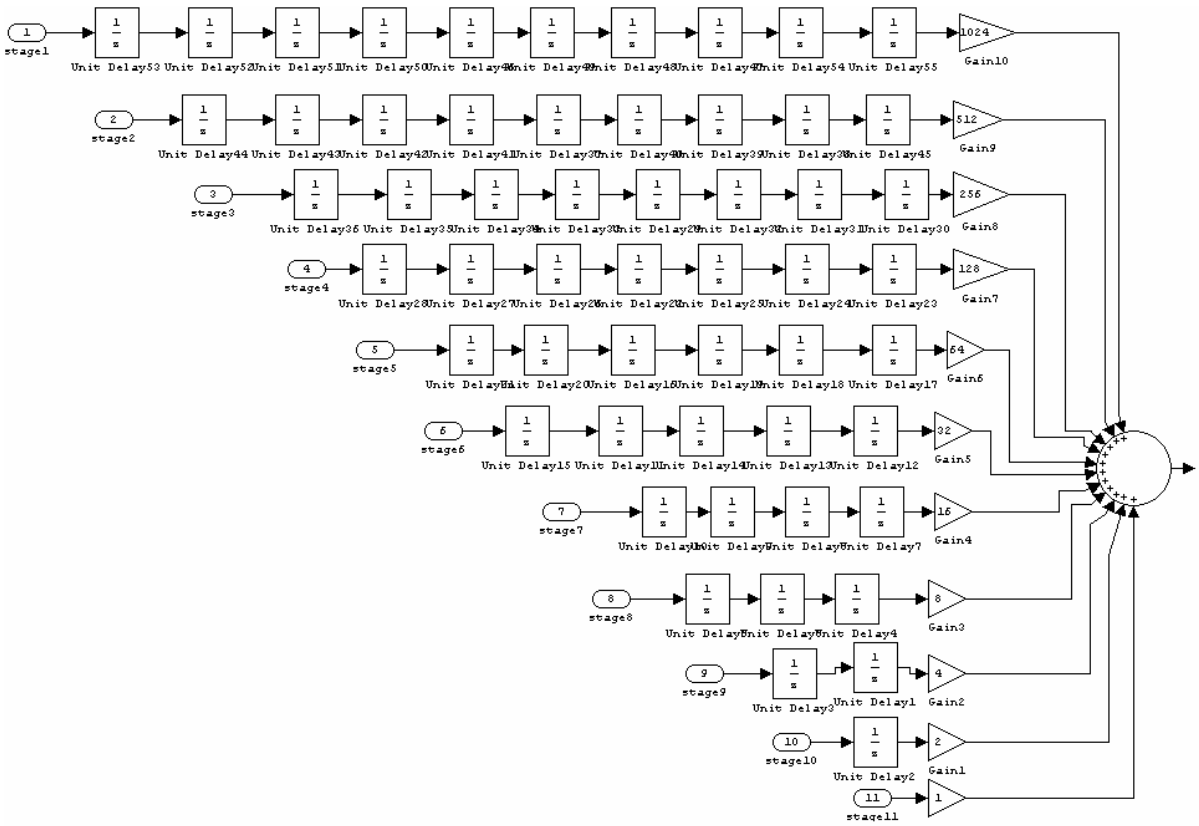


Fig.7.51. Bloques ideales. Corrección Digital 12 bits.

3.1.6.2. Corrección digital genérica

La forma de operación es igual a la anterior, pero en este caso la ganancia depende del número de códigos a la salida de cada etapa. Vamos a distinguir dos casos:

- todos los bloques tengan igual numero de códigos de salida
- los bloques tengan códigos distintos de salida

Para el primer caso, si tuviéramos 3 bloques con salida N códigos cada uno, las ganancias serían $2^{N-1}, 2^{N}, 2^{N+1}$. El número total de bits a la salida sería Nb·(N-1)+1, siendo N el número de códigos a la salida y Nb el número de bloques.

Para el segundo caso, suponiendo también 3 bloques con N_1, N_2 y N_3 códigos de salida, las ganancias serían $2^{N1-1}, 2^{N2-1}, 2^{N3-1}$. El número total de bits a la salida serán $\sum_{i=1}^{Nb} Ni - Nb + 1$, siendo Ni el número de códigos a la salida de la etapa i, y Nb el número de bloques.

Sabiendo esto, presentamos los siguientes bloques de corrección digital genérica.

Modelos para bloques con el mismo número de códigos de salida:

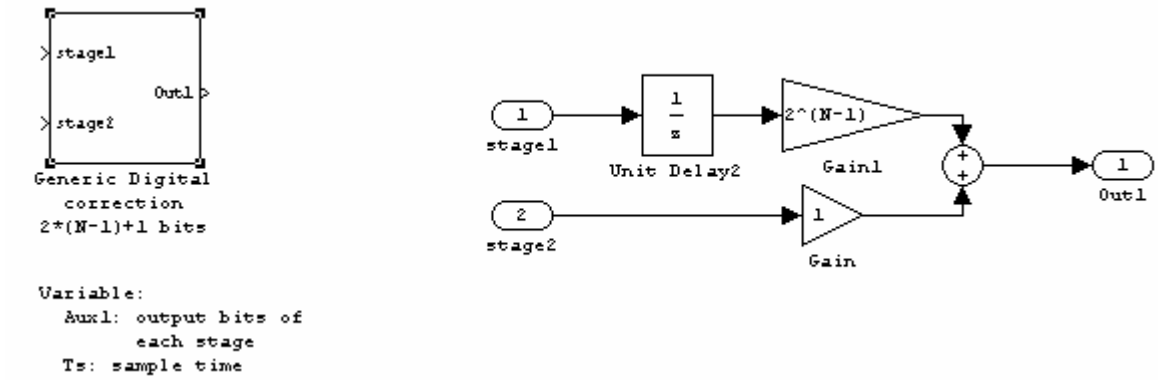


Fig.7.52. Bloques ideales. Corrección Digital 2 stage.

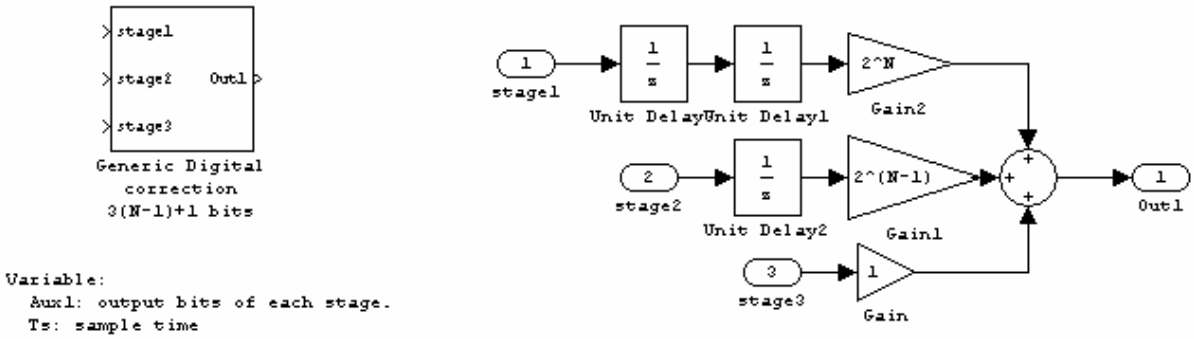


Fig.7.53. Bloques ideales. Corrección Digital 3 stage.

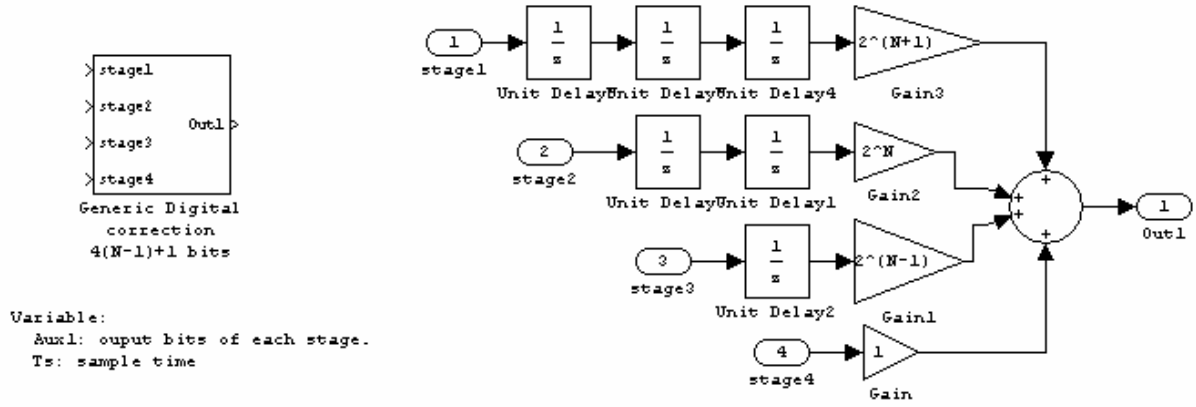


Fig.7.54. Bloques ideales. Corrección Digital 4 stage.

Y para el caso de que el número de código de salida de los bloques sea distinto:

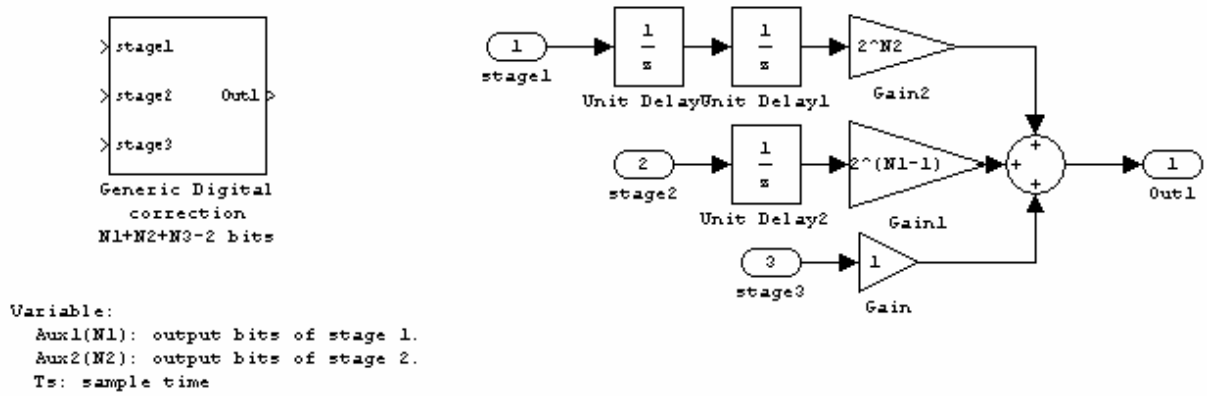


Fig.7.55. Bloques ideales. Corrección Digital 3 stage.

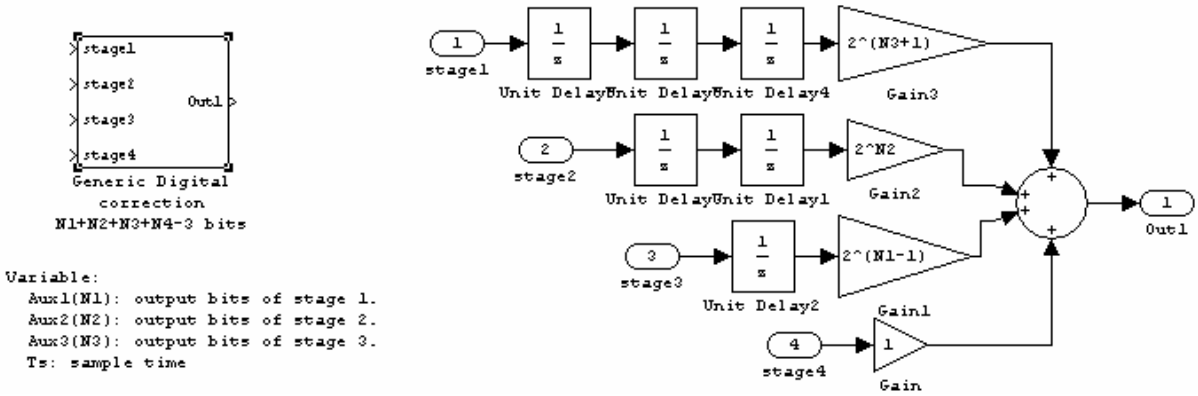


Fig.7.56. Bloques ideales. Corrección Digital 4 stage.

3.2. Bloques no ideales

Ya vimos en el capítulo 3 los diferentes tipos de no idealidades que podían a afectar a un convertidor analógico digital pipeline. Pues ahora, en este apartado mostraremos los modelos de diseño que hemos realizado para simularlas.

3.2.1. DAC Stage

El modelo diseñado para el caso del DAC lo mostramos en la Fig.7.57. Este muestra como no idealidad el offset el la tensión de referencia.

3.2.1.1. Offset en la tensión de referencia

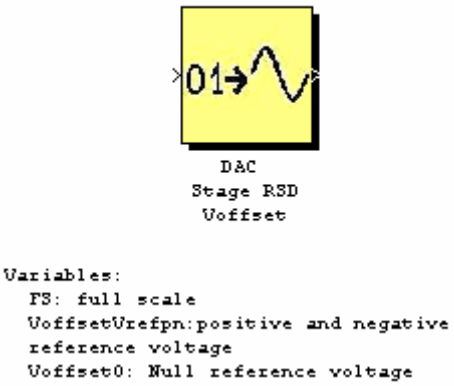


Fig.7.57. Bloques no ideales. DAC.

Este bloque presenta los siguientes parámetros:

Variable simulador	Comentario
VoffsetVrefpn	Offset en la referencia positiva y negativa
Voffset0	Offset en la referencia nula
FS	Fondo de escala

Tabla.7.9. Variables: Bloques no ideales. DAC.

El diseño:

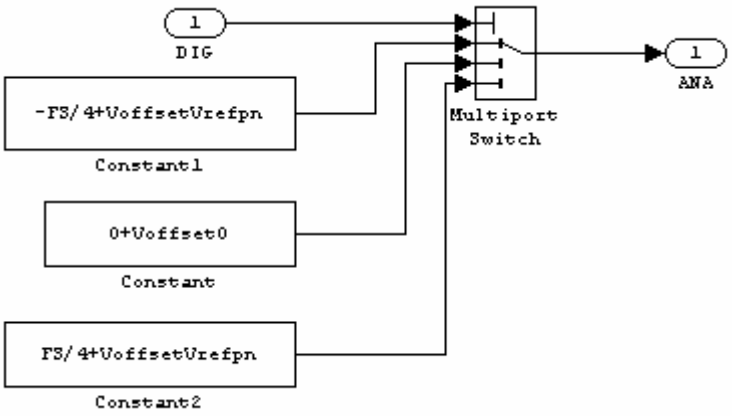


Fig.7.58. Diseño: Bloques no ideales. DAC.

3.2.2. Stage Complete

Veamos a continuación diferentes no idealidades para una etapa concreta del convertidor pipeline.

3.2.2.1. Stage RSD. Ganacia finita.

El modelo para este caso:

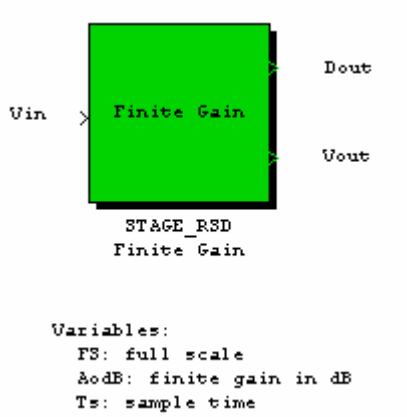


Fig.7.59. Bloques no ideales. Stage. Gain Finite.

El diseño lo mostramos en la siguiente gráfica:

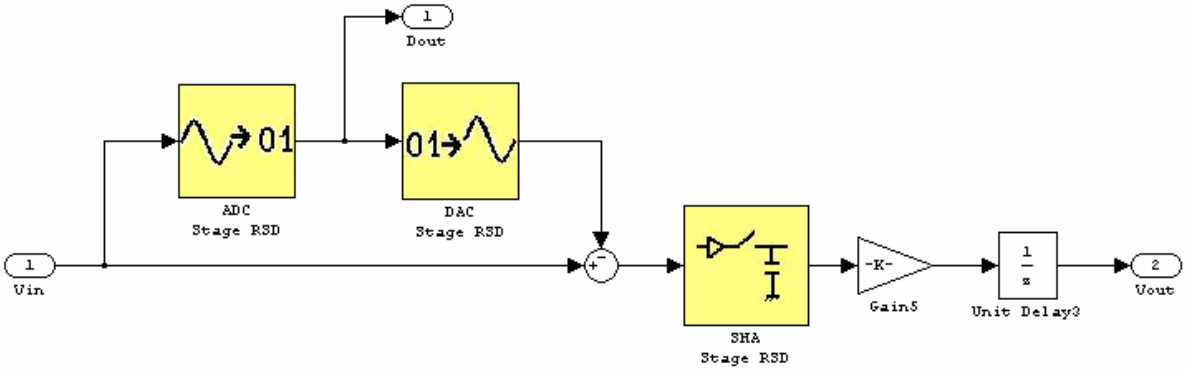


Fig.7.60. Diseño: Bloques no ideales. Stage. Gain Finite.

El valor de la ganancia finita viene dado por la constante K en el modelo, que tiene el valor: 1/(1+(2/(10^(AodB/20)))).

Para el caso genérico bastaría con sustituir los bloques anteriores por bloques genéricos, tal y como mostramos en el apartado 3.1.4.

3.2.2.2. Stage RSD. Mismatch.

El bloque que representa la no idealidad de mismatch[8] lo representamos en la figura siguiente:

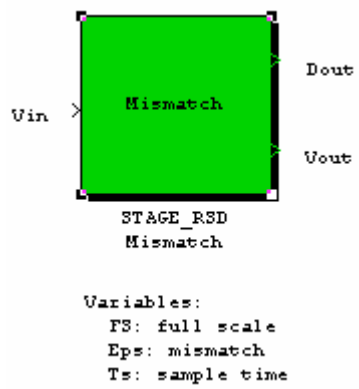


Fig.7.61. Bloques no ideales. Stage. Mismatch.

Las variables que utiliza el modelo las podemos observar en la máscara[9]:

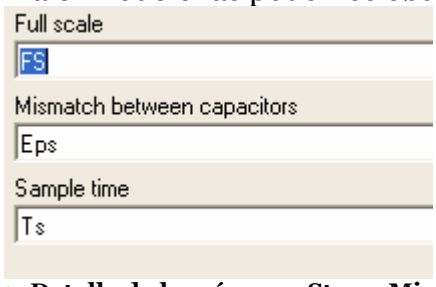


Fig.7.62. Detalle de la máscara. Stage. Mismatch.

El diseño para este bloque:

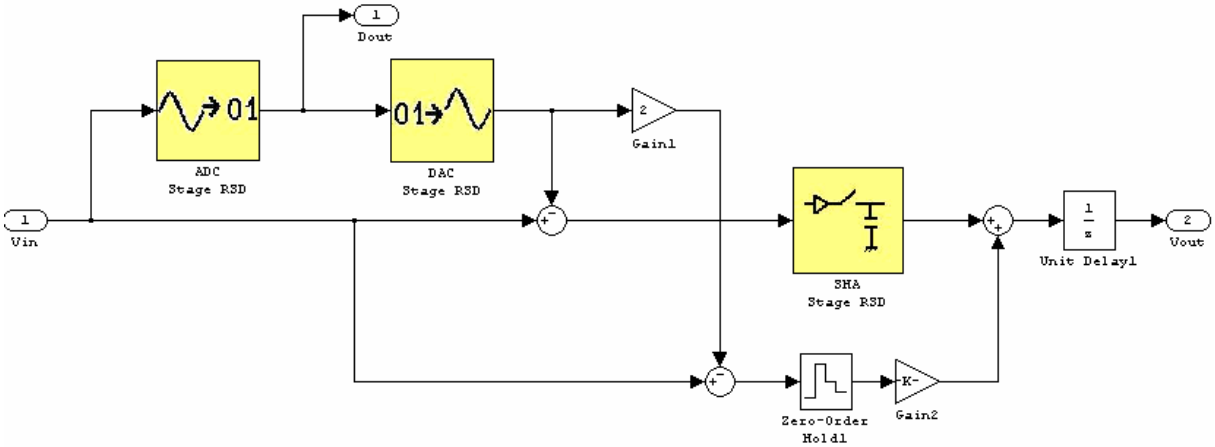


Fig.7.63. Diseño: Bloque no ideal. Stage. Mismatch.

[8] Para el caso genérico sería igual cambiando los bloques del diseño RSD por genéricos.

[9] Es bueno recordar, que para cada modelo se ha diseñado una máscara que contiene una descripción del mismo y las variables utilizadas.

3.2.2.3. Stage RSD. SR y GBW.

En este caso, el bloque diseñado:

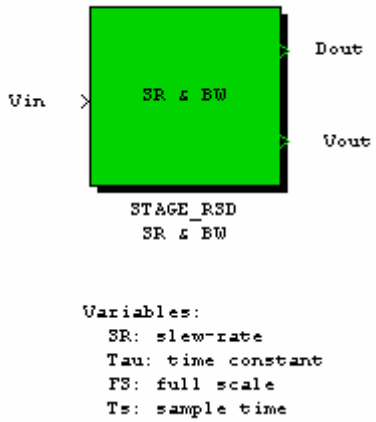


Fig.7.64. Bloque no ideal. Stage. SR y GBW.

Las variables utilizadas las podemos ver debajo del bloque: SR, Tau, FS y Ts. El diseño realizado:

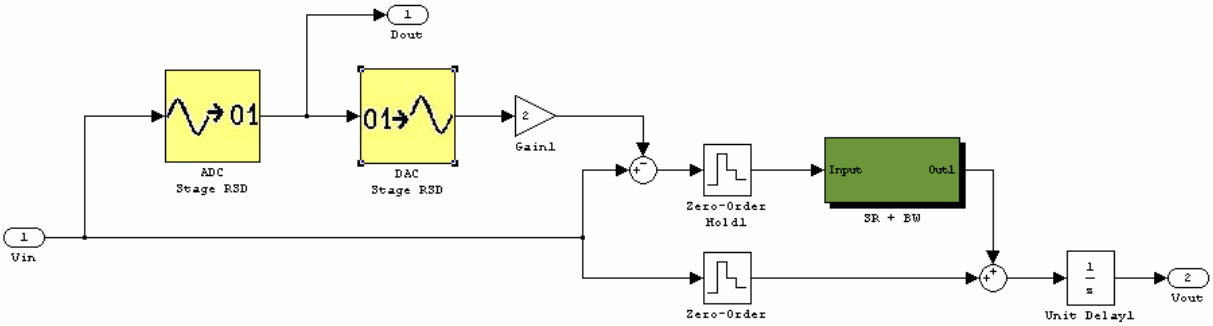


Fig.7.65. Bloque no ideal. Stage. SR y GBW.

Donde el bloque SR + BW tiene como diseño:

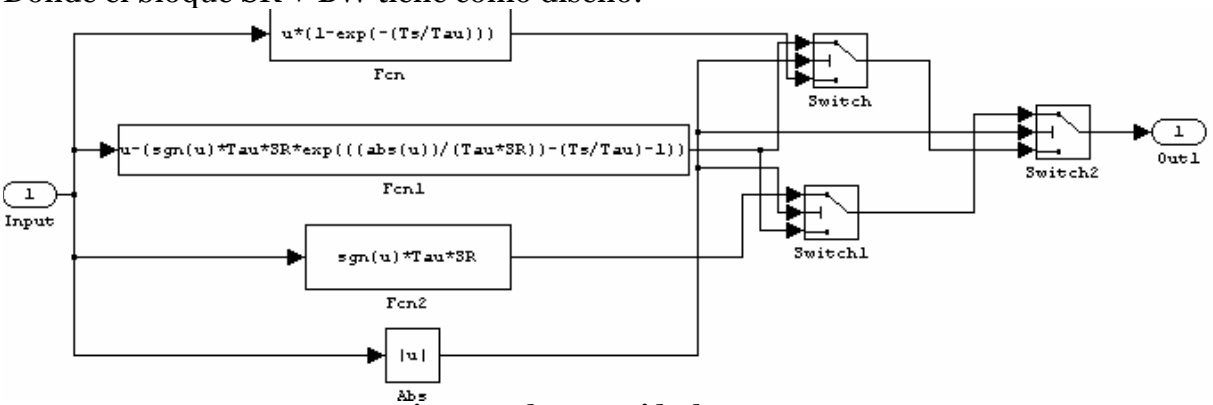


Fig.7.66. Bloque no ideal. SR+BW.

3.2.2.4. Stage RSD. Offset.

En este caso el modelo:

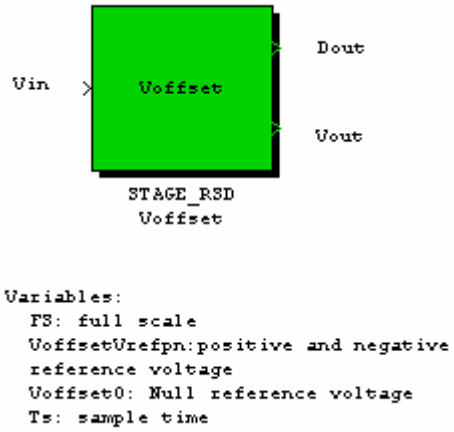


Fig.7.67. Bloque no ideal. Offset.

El diseño consiste en añadir al caso ideal el DAC con características no ideales de offset. Nos queda la siguiente composición:

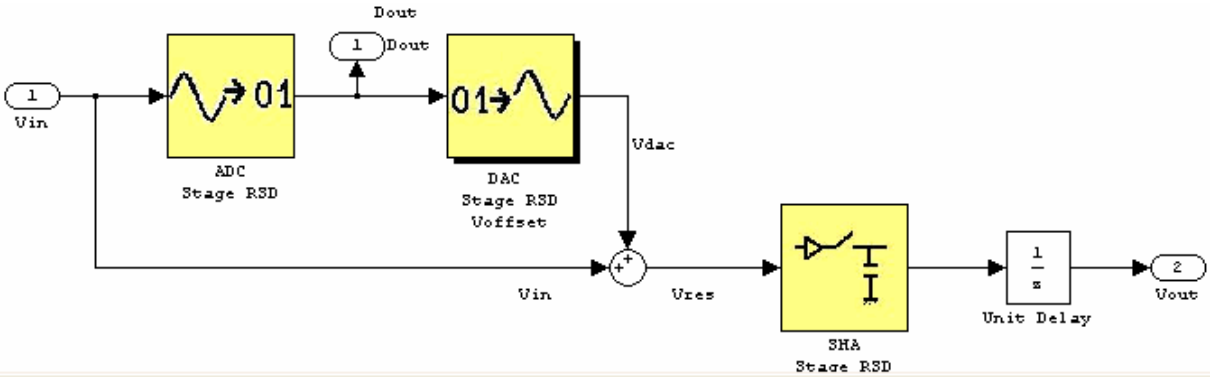


Fig.7.68. Diseño: Bloque no ideal. Offset.

3.2.2.5. Stage RSD. Jitter noise.

En este bloque lo que hacemos es añadir al bloque ideal el bloque de ruido jitter a la entrada. Este bloque deberá ir al comienzo.

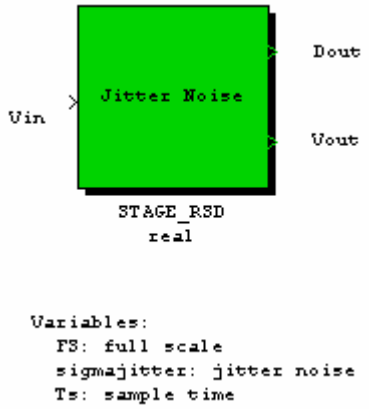


Fig.7.69. Bloque no ideal. Stage. Jitter.

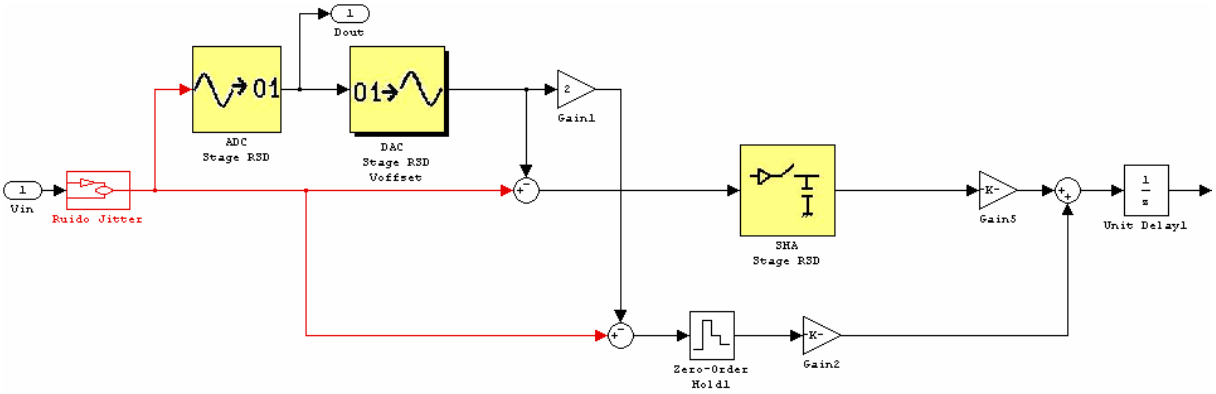


Fig.7.70. Diseño: Bloque no ideal. Stage. Jitter.

El bloque del Jitter lo vemos en el apartado 3.2.3.

3.2.3. Jitter noise

El modelo del Jitter, visto ya en otro tipo de convertidores es el siguiente:

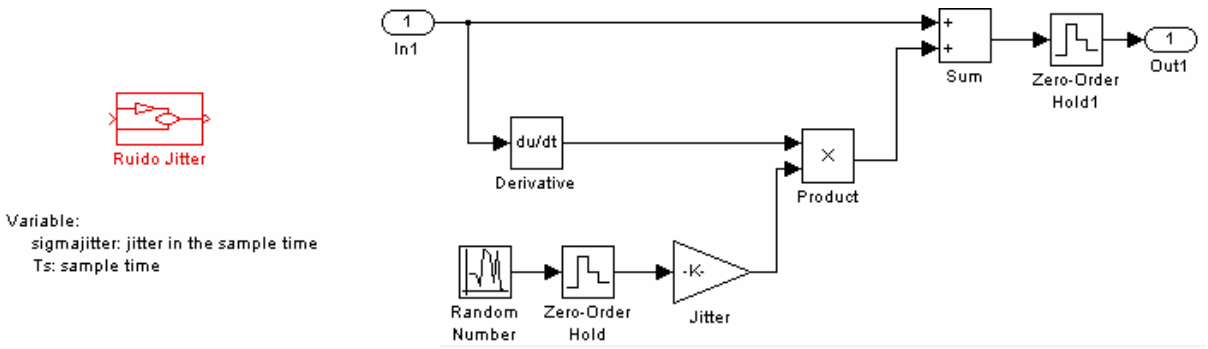


Fig.7.71. Diseño: Bloque no ideal. Jitter.

A modo de resumen, presentamos todos los modelos de bloques no ideales desarrollados en el apartado 3.2.

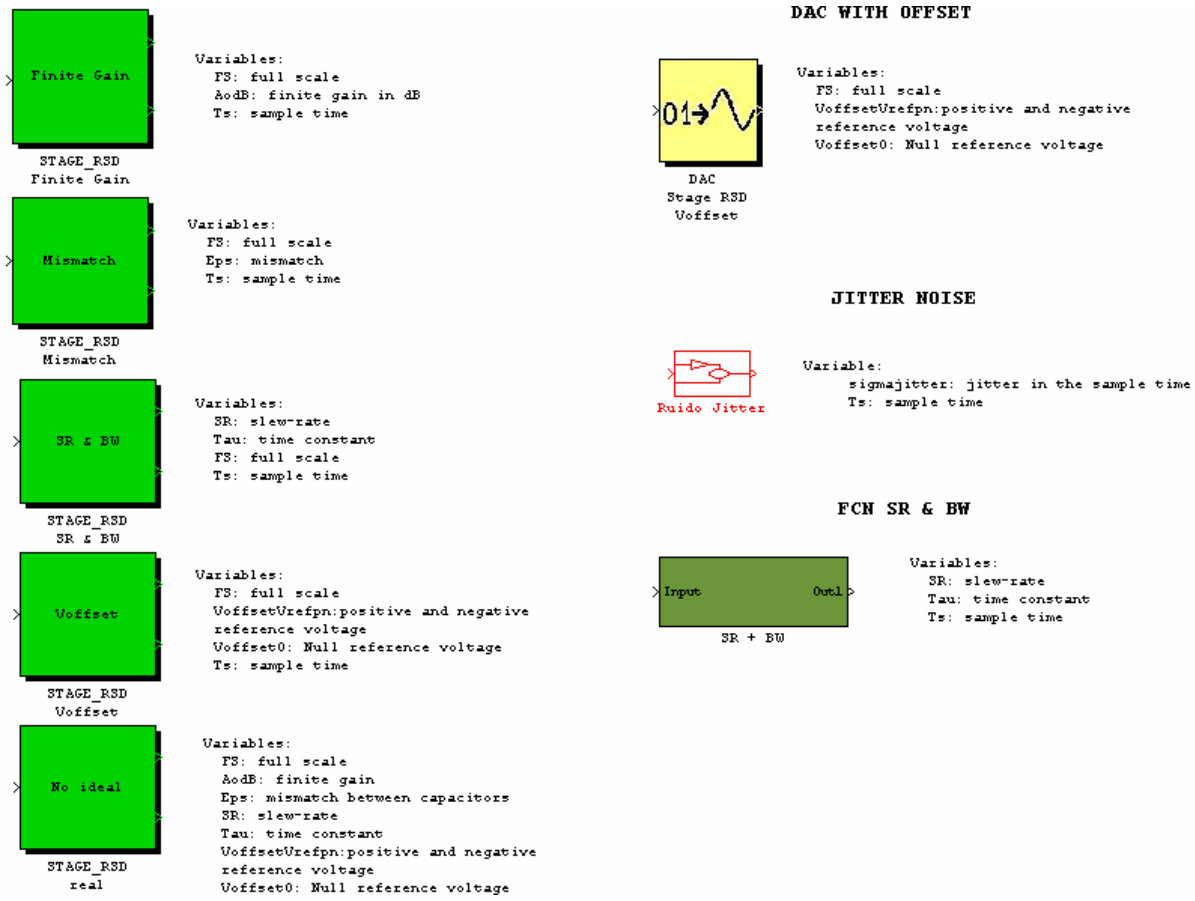


Fig.7.72. Bloques no ideales.

3.3. Ejemplos

En este apartado nos centramos en presentar los ejemplos desarrollados. Cada uno de los bloques de los que están compuestos ya los hemos descrito en apartados anteriores. Mostraremos el esquema general para cada uno de ellos.

En el capítulo 8 mostraremos la simulación de estos ejemplos, de forma que este apartado nos limitamos a mostrarlos estructuralmente.

Para tener una visión global de todos los ejemplos desarrollados para el

pipeline, les mostramos a continuación la agrupación de todos ellos:

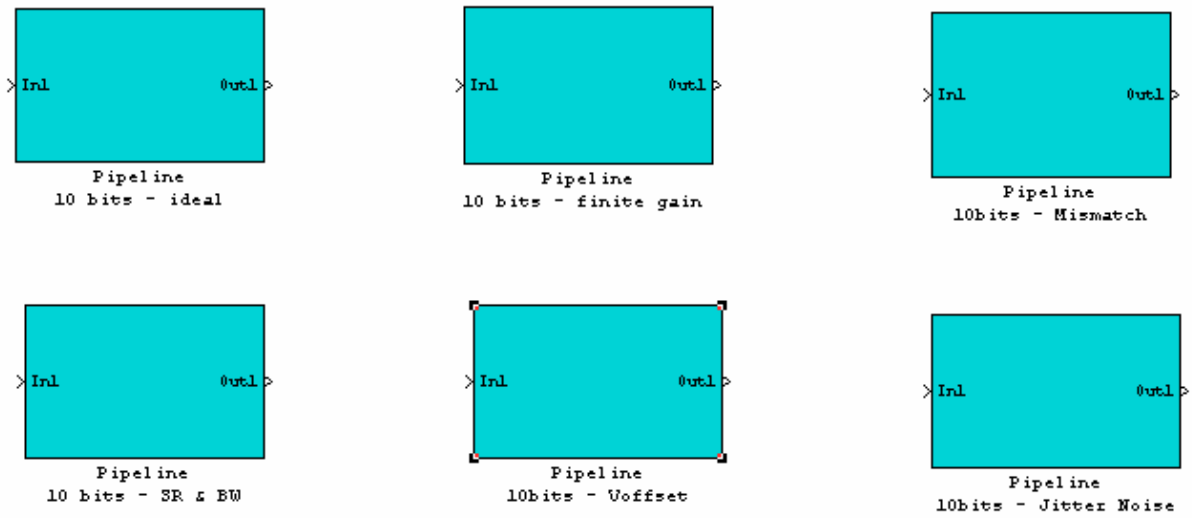


Fig.7.72. Ejemplos pipline.

Todos los ejemplos siguen la estructura que mostramos en la Fig.7.74. Lo que varía es el primer bloque que dependiendo de la no idealidad que queramos simular, tendremos el bloque correspondiente a eso no idealidad. En la Fig. 7.74 presentamos como ejemplo el caso del mismatch. El resto de ejemplo aunque están montados no lo mostramos por nos extendernos, en caso de querer consultarlos pueden acceder al CD de instalación del programa.

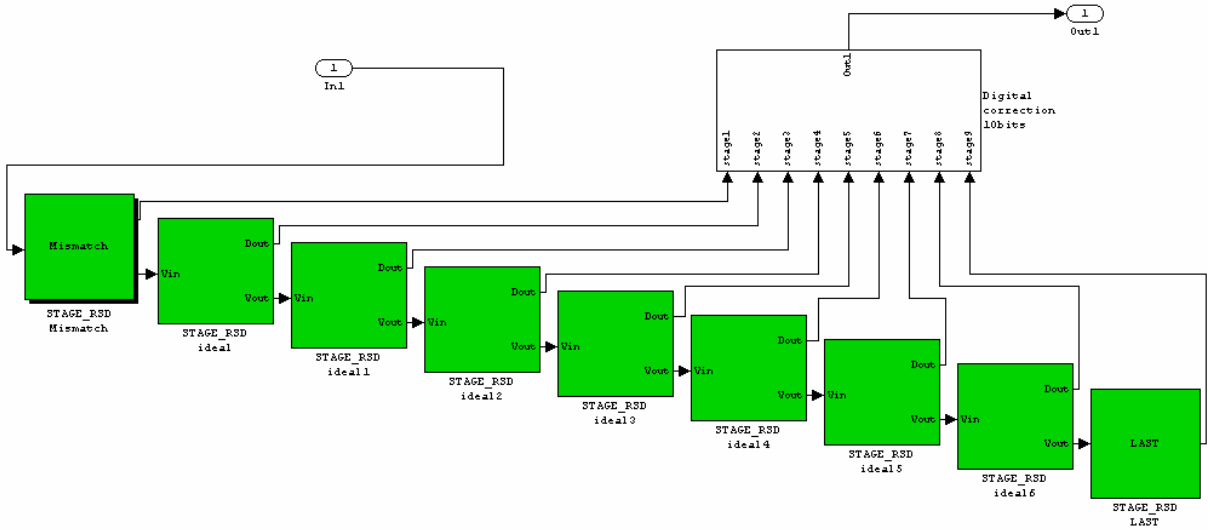
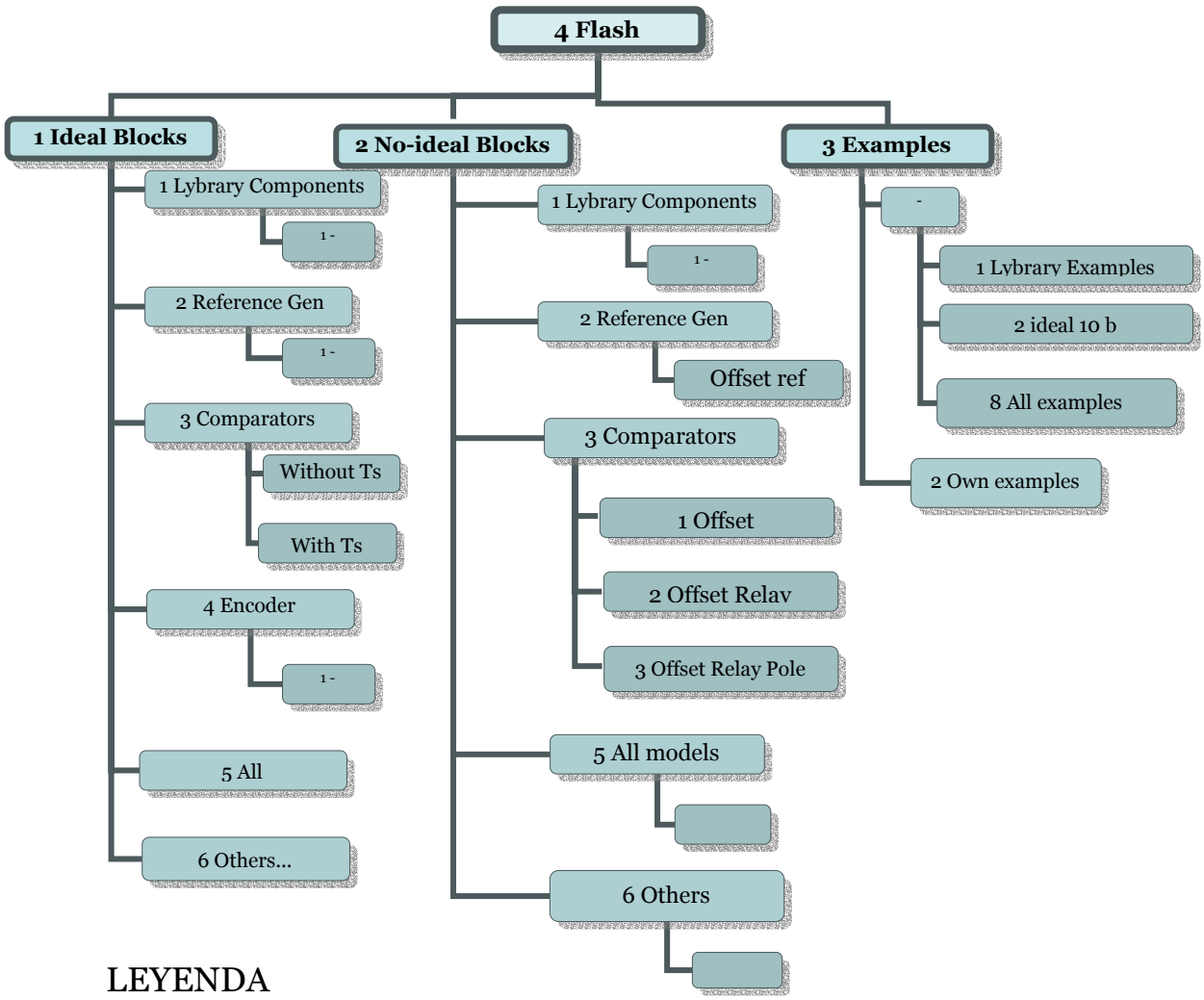


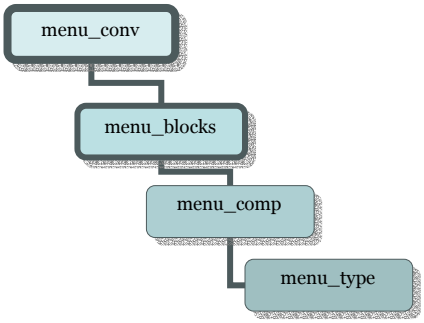
Fig.7.74. Ejemplos pipeline. Mismatch.

4. Librería del Flash

Para tener una visión completa de la librería, muestro a continuación el siguiente esquema en forma de árbol que incluye a grande rasgos los modelos diseñados para convertidores flash:



LEYENDA



4.1. Bloques ideales

Tras el estudio realizado de convertidores Flash, llegamos a la conclusión que necesitamos incluir bloques como comparadores, generadores de tensión de referencia para la entrada a los comparadores, encoger...

Una visión global de los bloques generados es la siguiente:

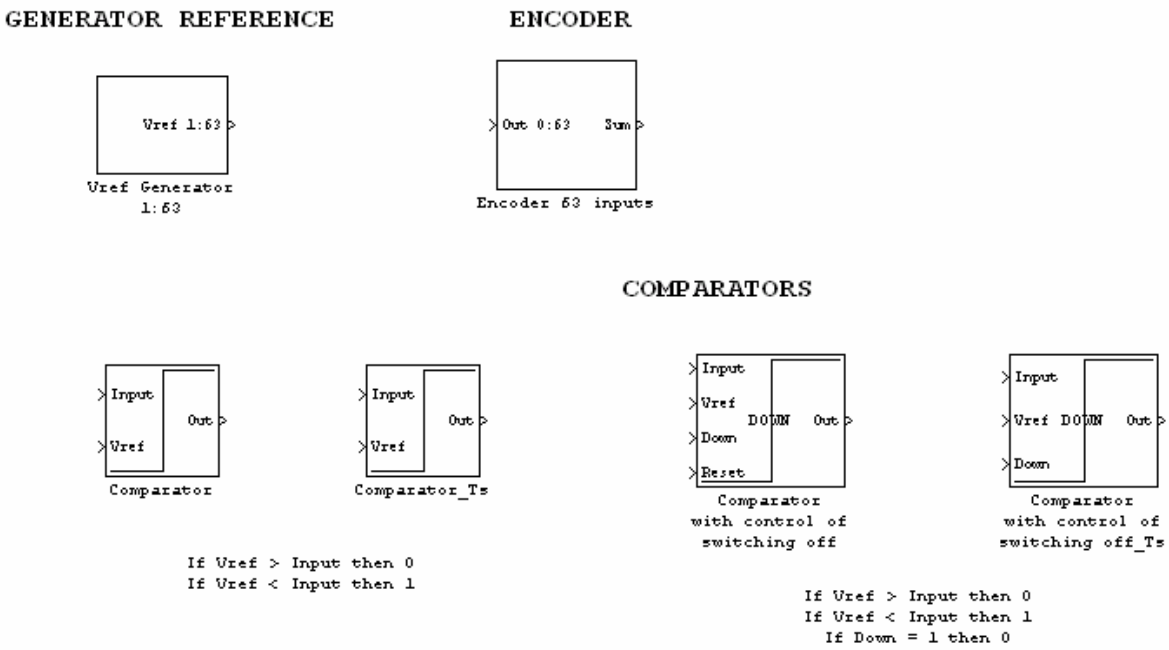


Fig.7.75. Bloques ideales Flash.

4.1.1. Generador de referencia

Este bloque viene modelado por:

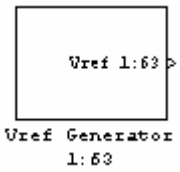


Fig.7.76. Bloques ideales Flash. Gen_ref.

Este bloque dará la entrada de la tensión de referencia a un banco de 63 comparadores. Como estamos en el caso ideal, el valor de la variable offset es nulo. Una visión parcial de este bloque lo vemos en la siguiente figura:

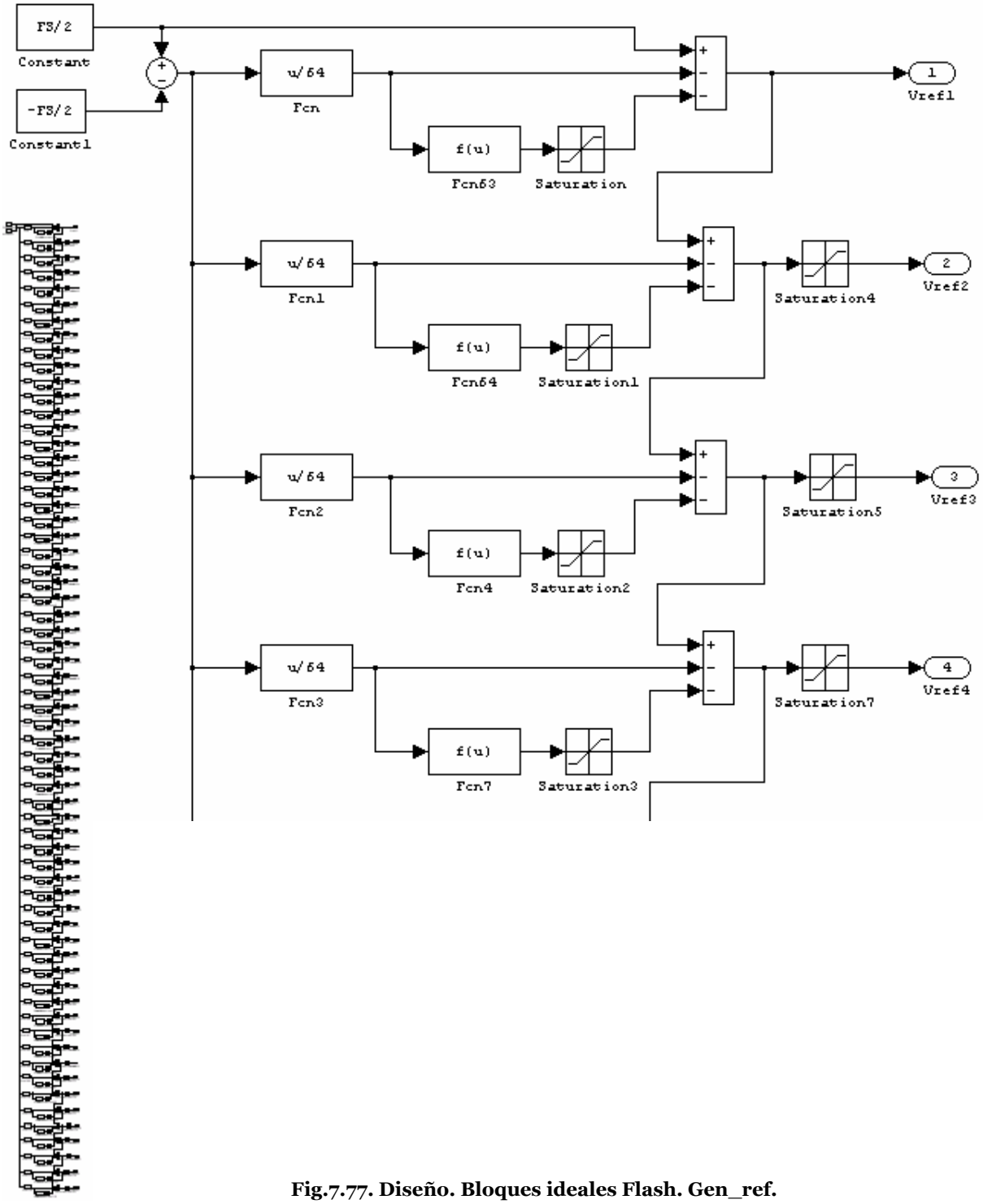


Fig.7.77. Diseño. Bloques ideales Flash. Gen_ref.

4.1.2. Comparadores

En este apartado incluido varios tipos de comparadores. Los presentamos a continuación:

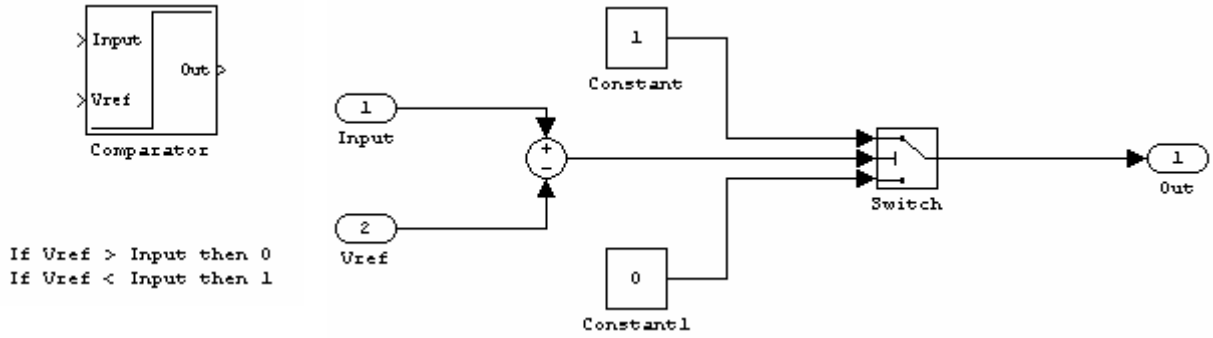


Fig.7.78. Diseño. Bloques ideales Flash. Comp.

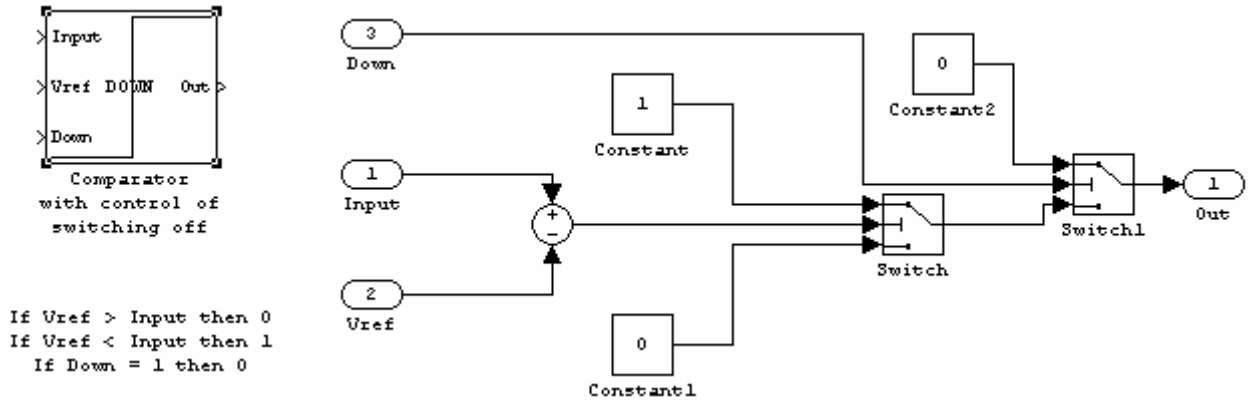


Fig.7.79. Diseño. Bloques ideales Flash. Comp_down.

La diferencia entre el comparador de la Fig.7.78 y el de la Fig.7.79, es que en el segundo caso se incluye al comparador una señal de power down para apagar el comparador en caso de que no queramos que funcione. Esto puede servirnos de utilidad en métodos de calibración.

Los otros modelos que presentamos, son análogos a éstos con la salvedad que la señal de entrada incluye un sample and hold. Estos dos nuesvos modelos de comparadores los incluimos en las Fig. 7.80 y 7.81.

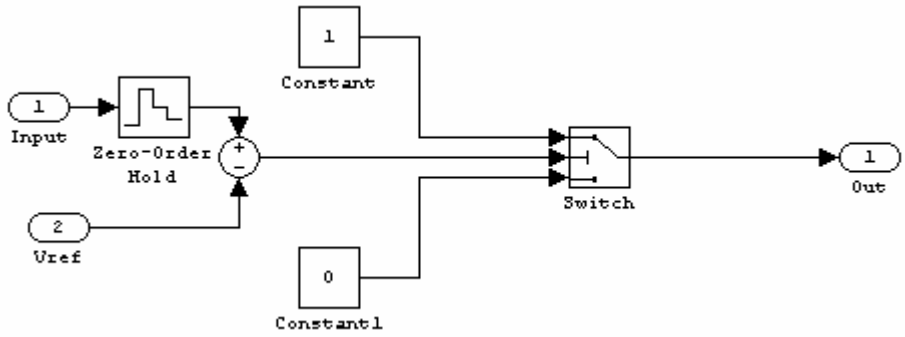


Fig.7.80. Diseño. Bloques ideales Flash. Comp con Ts.

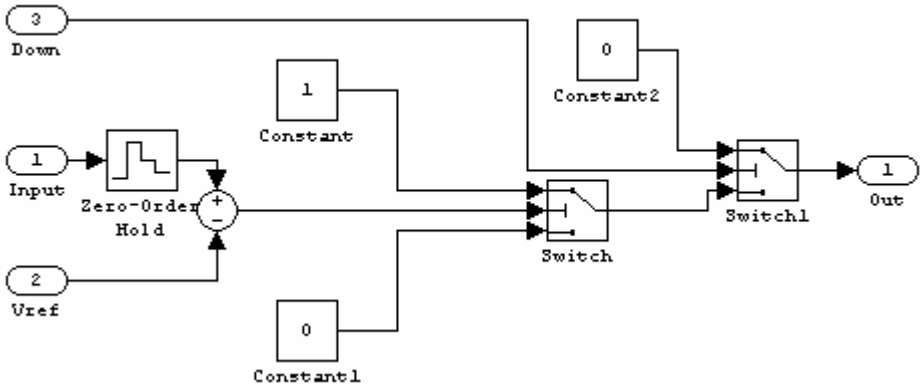


Fig.7.81. Diseño. Bloques ideales Flash. Comp_down con Ts.

4.1.3. Encoder

Se encarga de tomar todas las salidas del banco de comparadores y sumarlas. El modelo con el que realizaremos esta función:

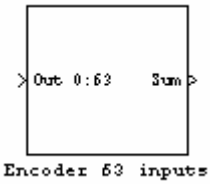


Fig.7.81. Bloques ideales Flash. Encoder.

Veamos a continuación el diseño para en caso de un encoger que tome 63 entradas.

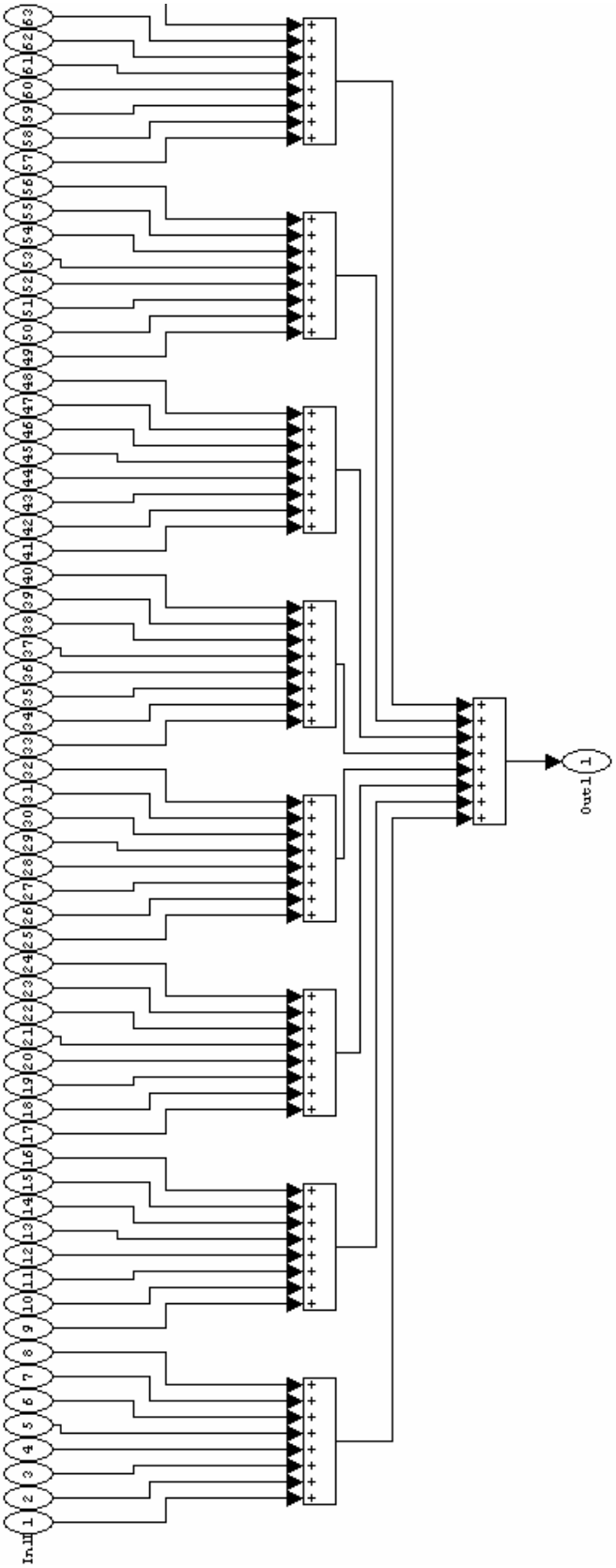


Fig.7.82. Diseño. Bloques ideales Flash. Encoder.

4.2. Bloques no ideales

En este apartado mostraremos los diferentes diseños de bloques para convertidores analógico-digitales flash con no idealidades. Presentaremos el bloque y el diagrama de bloques que implementa su funcionalidad.

4.2.1. Generador de referencia

Este modelo, representado en la Fig.7.83. tiene el mismo diseño que el comentado en el apartado 4.1.1. La única diferencia es que debemos dar un valor al offset, lo que modelaría la tolerancia a la que está sometida la cadena de resistencias en una estructura flash .

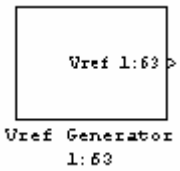


Fig.7.83. Bloque no ideal Flash. Gen_ref_offset.

4.2.1. Comparadores

En este caso vamos a presenta tres modelos: uno con offset, otro con offset y relay, y por último, uno con offset, relay y efecto del polo. Cada uno de estos modelos los representamos en las Fig. 7.84, 7.85 y 7.86 respectivamente.

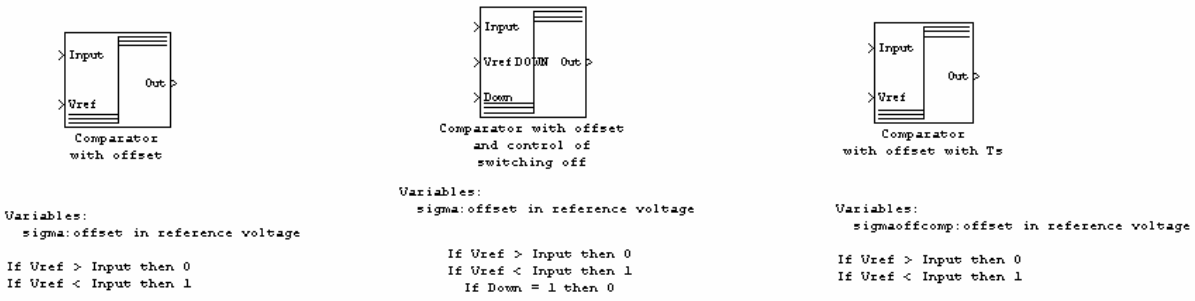


Fig.7.84. Bloques no ideales Flash. Com_offset.

Su diseño ya fue presentado en el apartado 4.1.2.

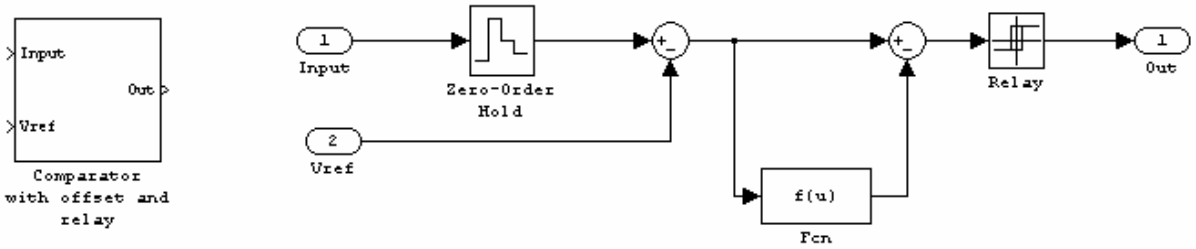


Fig.7.85. Bloque no ideales Flash. Com_offset_relay.

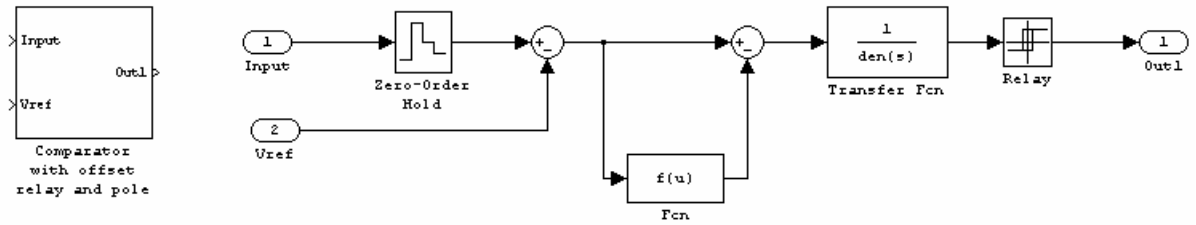


Fig.7.86. Bloque no ideales Flash. Com_offset_relay_polo.

4.3. Ejemplos

En este apartado vamos a presentar un ejemplo completo de un flash.

En este modelo se representa un convertidor flash de 6 bits, considerándose las siguientes no idealidades: offset en las tensiones de referencia, offset en los comparadores, histéresis en los comparadores. Contiene los bloques GenVref6bits, Comparator Ts Offset Relay Bank 6bits, DigToThermo, Binary Logic Converter, y ConvBitstoNumber. Estos bloques se han diseñado en exclusiva para este ejemplo.

El diagrama de bloques diseñado se muestra en la siguiente figura:

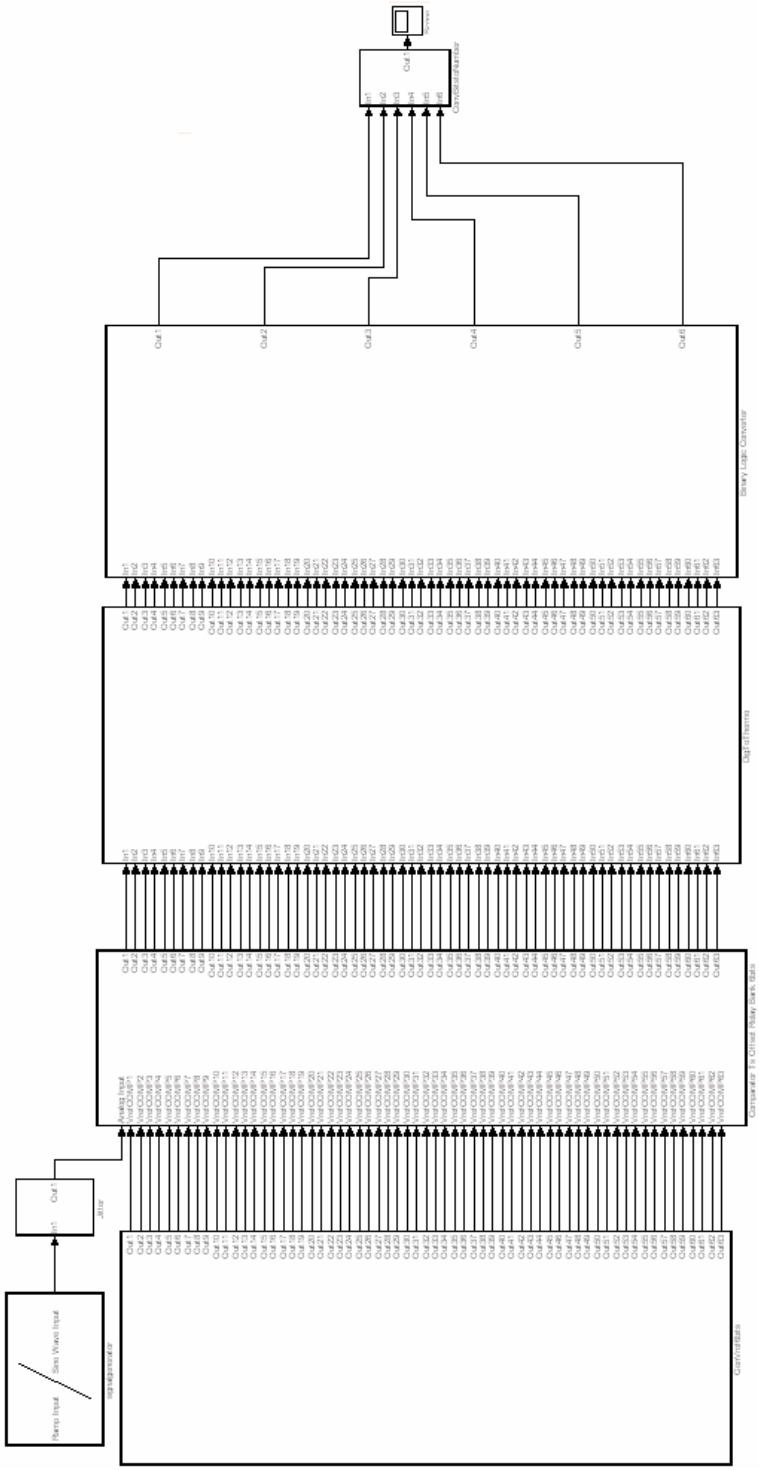


Fig.7.87. Ejemplo Flash de 6 bits.