ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

PROYECTO FIN DE CARRERA

DIGITAL DATA FILTERING IN A HALL EFFECT BASED ANGULAR POSITION SENSOR

Autor: Jorge Solana Muñoz
Director: Pavel Kejik (EPFL)
Codirector: Vicente Garcerán Hernández

Cartagena, octubre 2009

Autor	Jorge Solana Muñoz
e-mail del autor	Jorge.solanamunoz@gmail.com
Director	Vicente Garcerán Hernández
e-mail del director	Vicente.garceran@upct.es
Codirector	Pavel Kejik (EPFL)
e-mail del codirector	pavel.kejik@epfl.ch
Título del PFC	Digital data filtering in a Hall effect based angular position sensor
Descriptores	Diseño de circuitos, sistemas electrónicos digitales, VHDL
Resumen	Este proyecto empieza con el análisis de un sensor de efecto Hall como los estudiados en la asignatura *Instrumentación Electrónica*. Durante su estudio se pondrán en práctica algunos de los conocimientos adquiridos en ella. Posteriormente se entra en una fase de diseño de un sistema electrónico digital que procesará la señal procedente del sensor. En esta parte entran en juego conceptos de *Sistemas Electrónicos Digitales* necesarios para la elección de los bloques y las funciones lógicas oportunas. Los distintos bloques que formarán el diseño final se implementan en VHDL, con lo que también resulta de gran utilidad lo aprendido en *Arquitectura de Computadoras*. La última parte del proyecto consiste en ir siguiendo los pasos típicos de un flujo de diseño de arriba abajo (*top-down*) hasta obtener el layout final del circuito. Los pasos que hay que seguir en esta fase del proyecto se estudiaron en la asignatura *Diseño de Circuitos y Sistemas Electrónicos*. En definitiva, el objetivo de este PFC es obtener un circuito electrónico capaz de aumentar la relación señal-a-ruido de una señal con ancho de banda variable. Es decir, se busca un filtro adaptativo, que funcione con las peculiaridades de una medida angular como la que ofrece el sensor de efecto Hall que se estudia en la primera parte del proyecto.
Titulación	Ingeniería de Telecomunicación
Intensificación	Sistemas y Redes de Telecomunicación
Departamento	Electrónica, Tecnología de Computadoras y Proyectos
Fecha de presentación	Octubre 2009

“Divide et impera”, célebre frase atribuida a Julio Cesar que significa “divide y vencerás”.

Esta ha sido y es mi máxima en los momentos difíciles, en la vida, en la carrera y más aún en el desarrollo de este proyecto fin de carrera.

Agradecimientos

Hasta la culminación de este proyecto, el ciclo universitario ha ocupado algo más del 20% de mi vida. Un ciclo largo en el que hubo momentos de gran esfuerzo y de gran disfrute. Momentos de superación personal y que dieron lugar al planteamiento de nuevas metas. Pero para mí, la universidad ha sido mucho más que aulas en las que dar clases, laboratorios en los que manejar compleja instrumentación o una bonita biblioteca. La universidad es un entorno social en el que se comparten los conocimientos mediante la interacción con las personas que forman parte de ella.

Por supuesto, existe la finalidad de obtener un reconocimiento académico, pero creo que son mucho más interesantes los momentos de discusión científica o incluso política que, de vez en cuando, se producen en pequeños círculos, en los que queda muy lejos la necesidad del reconocimiento. Esas discusiones en las que se participa por el mero hecho de enriquecer la conversación, o por buscar una solución a un problema común.

No puedo olvidar la representación estudiantil, de la que he formado parte activa. Este es un ejemplo de cómo se puede dedicar tiempo a algo que no produce beneficios y que, en cuanto al citado reconocimiento… digamos que a veces puede llegar a producir un efecto negativo. Pero no deja de ser otro de esos círculos en los que se discute de forma abierta para solucionar problemas.

Buscar soluciones a los problemas que nos rodean o idear formas de de simplificar y acomodar nuestro entorno y nuestras interacciones con él, no deja de ser la tarea principal de cualquier ingeniero. Por tanto, creo que estos pequeños círculos de discusión son tan útiles como los conocimientos que se imparten en las aulas. A fin de cuentas, todo son herramientas y ensayos que esperamos nos sirvan en los ciclos venideros.

Por estos motivos no puedo dejar de agradecer su aportación a todos aquellos que han participado conmigo en alguna de estas discusiones.

Pero una universidad no debe aislarse ni conformarse con sus propios círculos de discusión, sino que debe relacionarse con su entorno, la sociedad, y con los de su especie, otras universidades. Pretendo ahora, agradecer la existencia de programas de intercambio de estudiantes entre universidades a las personas que lo promueven. Porque si en algún momento un estudiante puede sentirse aislado del resto del mundo en su universidad, solo tiene que abrir los ojos a la multitud de posibilidades que esta entidad le facilita. Gracias al programa Erasmus he vivido una de las mejores experiencias de mi vida. He podido conocer otro entorno universitario y otros círculos de discusión.

Quiero agradecer a Pavel Kejik (mi director de proyecto) y a Serge Reymond (mi *assistant* y compañero de despacho) toda la ayuda que me han facilitado durante mi estancia en la EPFL (Suiza). Sin ellos habría sobrevivido, pero con ellos lo he supervivido.

Y a mi tutor aquí en la UPCT, Vicente Garcerán Hernández, quiero agradecerle su aportación a este proyecto y su continua disponibilidad desde el primer día.

Por último, agradecer a mi familia su apoyo incondicional, que nunca he olvidado ni olvidaré.

Índice General

Índice de Figuras

Índice de Tablas

1.Introducción

El rápido crecimiento de la automatización en la producción industrial demanda gran variedad de sensores de movimiento y, entre ellos, una solución sencilla y precisa para la medida de ángulos absolutos con dispositivos sin contacto (*contactless*). La elección de los sensores magnéticos está a menudo motivada por la robustez que caracteriza a estos sensores, su insensibilidad a los factores externos y su bajo coste e fabricación. Una gran parte de los codificadores magnéticos de ángulos están basados en el efecto Hall.

Los codificadores basados en efecto Hall pueden ser una simple combinación de dos sondas Hall ortogonales, pero las mejores configuraciones se obtienen con tecnologías más elaboradas como VertX o IMC [1,2,3]. Estas configuraciones, sin embargo, tienen que pagar el precio de no ser tecnologías de fabricación estándar: su fabricación es incompatible con la tecnología CMOS o, cuando se incluye un proceso CMOS, este requiere un paso extra de post-procesado. De hecho algunas compañías (AMS, RLS e IC-Haus) han desarrollado un microchip completamente CMOS, pero su sensor está limitado a un imán con una geometría magnética bien definida (para una alta exactitud del ángulo necesita un alineamiento preciso) y, como muchas otras técnicas, requiere un procesador de señal digital on-chip.

Aquí se presenta un sensor de posición angular (APS) que demuestra el potencial de un nuevo principio de medidas de ángulos: el dispositivo Hall vertical. El APS se compone de una versión industrial del microchip que contiene al dispositivo Hall vertical y un elemento giratorio lleva el imán permanente. Comparado con las versiones anteriores, la versión industrial integra todas las funciones necesarias para dar el ángulo tanto en forma digital como en forma analógica.

Este sensor ha sido desarrollado recientemente por el LMSI3[1] de la EPFL[2]. Tras su estudio y comprensión, en el presente Proyecto Fin de Carrera se plantea la elaboración de un circuito que conectado al sensor, sea capaz de aumentar la precisión de su señal de salida. Dicha señal representa medidas codificadas en formato digital binario. El objetivo principal es aumentar la precisión de esas medidas únicamente mediante el tratamiento digital de las mismas. Para ello se pretende desarrollar el correspondiente circuito electrónico que lo realice y un algoritmo adaptativo, que sea capaz de regular el aumento de precisión en función de la velocidad de giro del propio sensor.

Se planteará el problema como un filtrado que se busca reducir la relación señal-a-ruido de la medida realizada por el sensor, adaptando dicho filtrado a los cambios de ancho de banda debidos a los cambios velocidad angular del sensor.

[1] LMSI3 – Laboratorio de Microsistemas Integrados 3.
[2] EPFL – Escuela Politécnica Federal de Lausana, Suiza.

2. Estado del Arte

2.1. Estudio previo del sensor

Un sensor magnético integrado y adecuado para la medición de ángulos absolutos sin contacto es una composición de un chip de silicio de estado sólido e imanes permanentes unidos al eje de giro. El dispositivo integrado detecta la dirección del campo magnético producido por el imán y por tanto, proporciona el ángulo del eje de rotación con respecto a una referencia fija. Más concretamente, el campo magnético producido por el imán permanente se encuentra en el plano del chip. Así, para medir la dirección de este vector, el chip de silicio debe contener un sensor magnético que sea sensible en dos ejes y cierta electrónica para procesar la señal de medida. El ángulo de salida esta referenciado a una dirección fija del plano del chip.

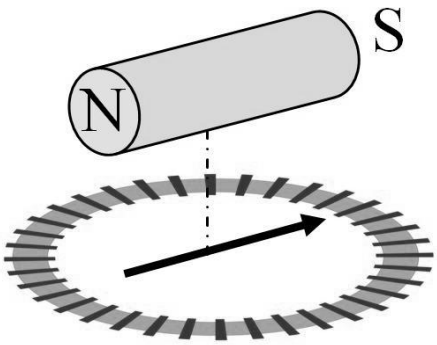


Figura 2.1-1. Dispositivo Hall circular combinado con un imán permanente.

El objetivo es obtener una precisión absoluta de ±0.1º con las siguientes condiciones:

Temperatura:	entre -50º y +80ºC.
Ancho de banda:	10Hz.
Tiempo de vida:	15 años.

Para conseguir estos resultados, se supone que la mínima resolución del sensor debe ser 0.01º.

2.1.1. Fuente del campo magnético

Requerimientos

- Dirección: La dirección del campo magnético debe estar comprendida en el plano del chip. Aunque una pequeña desviación no tiene por qué ser un problema ya que una pequeña componente constante del campo perpendicular al chip no afecta al ángulo medido.

- Intensidad: Un campo magnético con una intensidad grande tiene dos efectos positivos:

    1. Aumenta la relación señal-a-ruido, lo que permite reducir la corriente a través del sensor. Esto supone una ventaja importante, tanto a la hora de reducir el consumo de potencia, como de minimizar los efectos no lineales de la región sensible del sensor que provocan un offset en la medida. Gracias a esto, la corriente en el sensor puede establecerse en un valor relativamente bajo, menor de 0.4mA. Consecuentemente la potencia suministrada al dispositivo Hall queda en unos 0.5mW. Potencia que permite al sensor detectar campos con intensidades superiores a 10 μT, en un ancho de banda de 1kHz. De forma que para conseguir una resolución de 0.01^{o}, el campo magnético debe estar por encima de los 50mT.

    2. Disminuye los errores debidos a los campos parásitos externos. Si el campo exterior es del orden del campo magnético terrestre, el requerimiento de precisión de 0.05^{o} obliga a establecer una magnitud mínima del campo magnético de 60mT.

- Homogeneidad: Es fácil suponer que el campo magnético debe ser homogéneo en el área ocupada por el sensor, que es de 50μm. Pero, además la homogeneidad del campo también es necesaria en todo el volumen en el que se espera que se mueva el sensor. Se define el radio de homogeneidad RH como la menor distancia, desde el centro, donde la proyección del campo magnético en el plano del sensor gira unos 0.05^{o} (ver Figura 2.1-2). RH varía con la distancia a lo largo del eje z. La revolución de la función RH(z) alrededor del eje z define el volumen de homogeneidad.

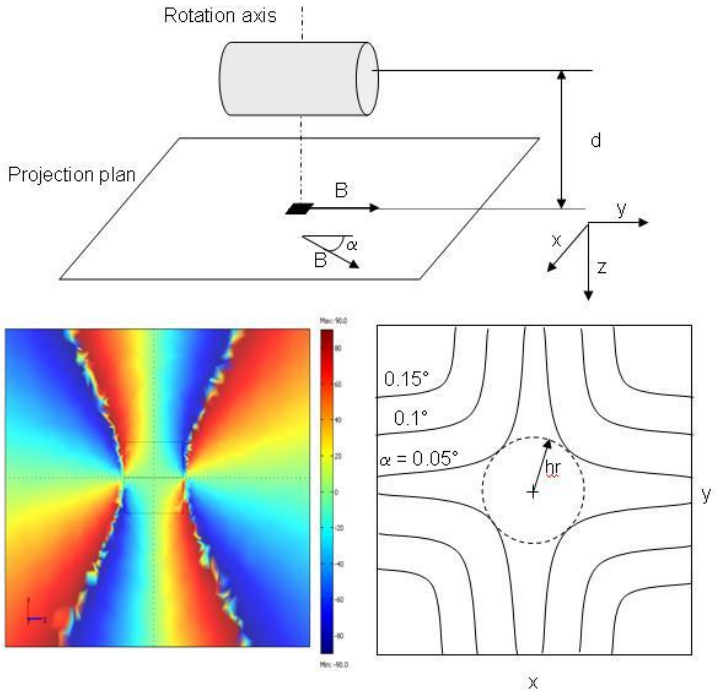


Figura 2.1-2. Definición del radio de homogeneidad RH.

Según la bibliografía consultada, resulta relativamente sencillo obtener mecánicamente una excentricidad de varias decenas de milímetros, pero tolerancias menores necesitan algunos mecanismos de centrado en las proximidades del sensor. Finalmente y para flexibilizar el resultado, se justifica que el volumen de homogeneidad debe contener una espera de 0.5mm de radio.

- Distancia de campo disperso: Es esencial reducir el campo que se escapa fuera de la región del sensor. Un material magnético (con alta permeabilidad) localizado en la región de dispersión modifica la distribución del campo magnético a su alrededor y esto puede causar errores angulares. Además una pequeña región de dispersión conlleva una mayor compatibilidad con los dispositivos cercanos. Se define la distancia de campo disperso como la mayor distancia desde el centro del sensor angular (APS) donde la intensidad de campo sobrepasa el 0.1% del campo remanente.

- Resistencia al entorno: La dirección del campo magnético no debe variar más de 0.05º en un rango de temperatura entre los -50º y +80ºC. Pequeños cambios en la intensidad del campo no afectan directamente al ángulo medido, pero en un bloque magnético real, algunas regiones suelen estar muy próximas a la desmagnetización (bordes y esquinas), y el problema puede aumentar cuando la temperatura induce una desmagnetización no uniforme. Además es preferible un imán con el menor coeficiente de temperatura posible. De igual forma, las radiaciones que inducen la desmagnetización deben limitarse. El material del material también debe ser resistente a la humedad.

- Peso: La masa no excederá los 50g.

- Tamaño: El diámetro exterior no excederá los 25mm.

Configuración del imán

Debido al limitado volumen que se exige en los requerimientos listados, se descartan las configuraciones que incluyen una parte ferromagnética débil (yugo magnético[3]). La razón principal es que para producir el máximo campo es preferible ocupar el volumen útil con imanes en vez de hierro. Otra razón viene de los requerimientos de homogeneidad. En general, un campo homogéneo se obtiene cuando la distancia entre los polos (el gap) es menor que el diámetro de los polos. Dado que el gap debe ser al menos de 7mm, son necesarios polos relativamente grandes. Superficies de polo que no sean planas pueden reducir el tamaño pero al precio de hacer más compleja la fabricación. Por tanto, se limita este diseño para configuraciones sin yugo magnético.

En aras de la simplicidad de magnetización y producción, el ensamblaje del imán debe consistir en simples bloques básicos (paralelepípedos, cilindros, etc.) con la orientación del momento magnético alineado con uno de los bordes.

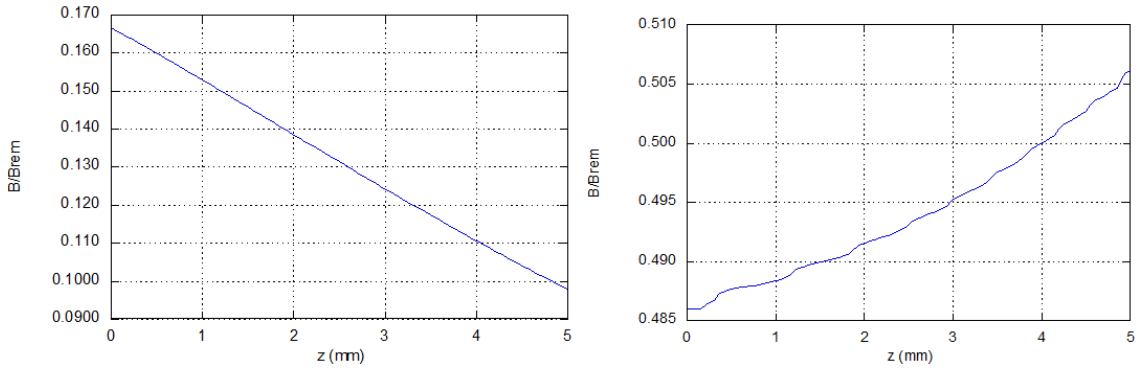


Figura 2.1-3. Intensidad de campo (izquierda) y radio de homogeneidad de un imán de bloque simple de 20x20x10mm.

A modo de ejemplo, se muestran los resultados obtenidos con un bloque de paralelepípedos simples (20x20x10mm, x-y-z, magnetizado a lo largo del eje x). La Figura 2.1-3 muestra la intensidad de campo (en unidades del campo remanente, que es próximo a 1 Tesla para un imán formado por SmCo [4]) y el HR[4] como una función de la distancia z desde la superficie del imán.

A 2mm de la superficie del imán (una posición razonable para el sensor) la intensidad del campo tiene un valor aproximado de 140mT y el radio de homogeneidad es de 0.5mm. Estos valores cumplen absolutamente con los requerimientos de homogeneidad e intensidad. Pero la distancia de campo disperso es relativamente grande: 85mm.

[3] Una configuración en yugo magnético o *magnetic yoke* consiste en un imán con forma de herradura y un material ferromagnético que se coloca en contacto con las dos puntas del imán, cerrando un circuito magnético.
[4] Radio de Homogeneidad.

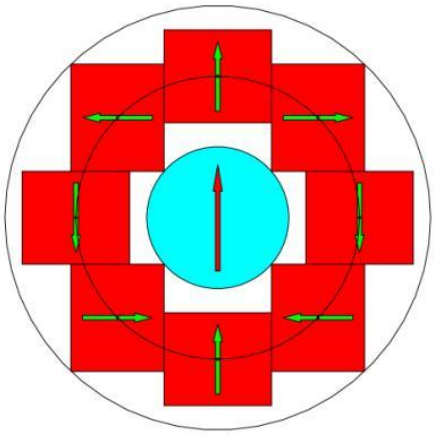


Figura 2.1-4. Corona Halbach hecha con 8 paralelepípedos en contacto.

Resultados mejores en configuraciones más compactas pueden ser conseguidas con configuraciones magnéticas más complejas. La idea principal es superponer constructivamente el campo procedente de diferentes imanes en la localización del sensor, de forma que esta superposición de campos sea negativa fuera del sensor. Un acuerdo conocido para hacerlo es el array "Halbach" [5]. Un array Halbach cilíndrico produce un intenso campo en una región limitada y tiene una pequeña distancia de campo disperso porque el momento magnético total es nulo. Para la presente aplicación la configuración adecuada consiste en una corona Halbach hecha con 8 cubos, como se muestra en la Figura 2.1-4. Con menos bloques, se pierde volumen y con más bloques, el cubo elemental tiene que ser demasiado pequeño y la intensidad de campo disminuye (es directamente proporcional al tamaño del cubo) [6].

El apilamiento de dos coronas mejora la homogeneidad del campo y proporciona cierta libertad de desplazamiento a lo largo del eje z (eje de rotación). Los requerimientos de tamaño imponen que el bloque básico tenga 4mm de lado. Para mejorar la estabilidad a expensas de la intensidad de campo, se reduce el tamaño del cubo, y al hacer esto se incrementa la distancia entre cubos. Para optimizar la homogeneidad del campo, la posición del cubo se elije de tal forma que cancele los fuertes armónicos esféricos. La configuración final se describe en la Figura 2.1-5 y en la tabla 2.1-1.

Outer diameter	Inner diameter	Height (along z)	Cube side	Cube inter-distance	Cube center radius
19	7.7	9.7	4	0.7	6.7

Tabla 2.1-1. Dimensiones de ensamblaje del imán.

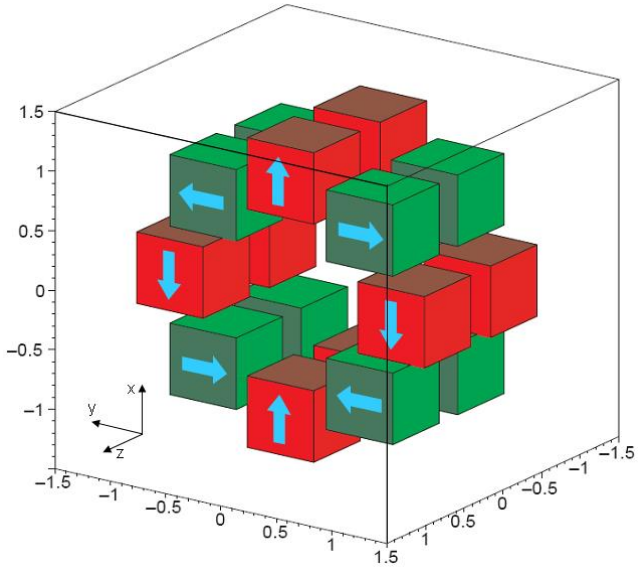


Figura 2.1-5. Configuración final: apilamiento de dos coronas Halbach. Las distancias están expresadas en unidades del radio principal.

El radio de homogeneidad es de aproximadamente 1mm. Si nos alejamos del centro, el radio de homogeneidad aumenta (ver Figura 2.1-6). La homogeneidad angular producida por el ensamblado en el plano central (z=0) se muestra en la Figura 2.1-7. Como resultado la esfera de homogeneidad debe tener aproximadamente 1mm de diámetro. Los efectos de desplazamientos, los errores de posicionamiento y de tamaño y las imperfecciones en la magnetización se estudiarán en un montaje real. De acuerdo con estos resultados, las posibilidades de ajustes mecánicos serán exploradas. Los cálculos preliminares muestran que un desplazamiento del orden de 0.1mm afecta al radio de homogeneidad en un valor de la misma magnitud.

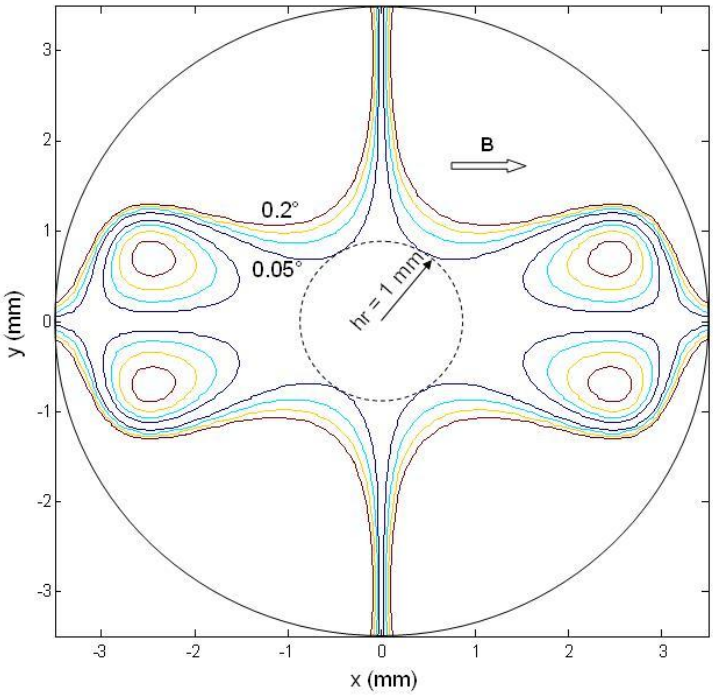


Figura 2.1-6. Homogeneidad angular en el plano central del imán (z=0).

La intensidad de campo producida por el montaje del imán esta en torno a los 270mT. Este valor relativamente alto permite limitar la corriente de alimentación del sensor y esto disminuye los *offsets*. La distancia de campo disperso es de unos 22mm, es decir, aproximadamente 10mm fuera del diámetro exterior. El valor está muy por debajo de los valores obtenidos para montajes de momento magnético no-nulo (solución de bloque único).

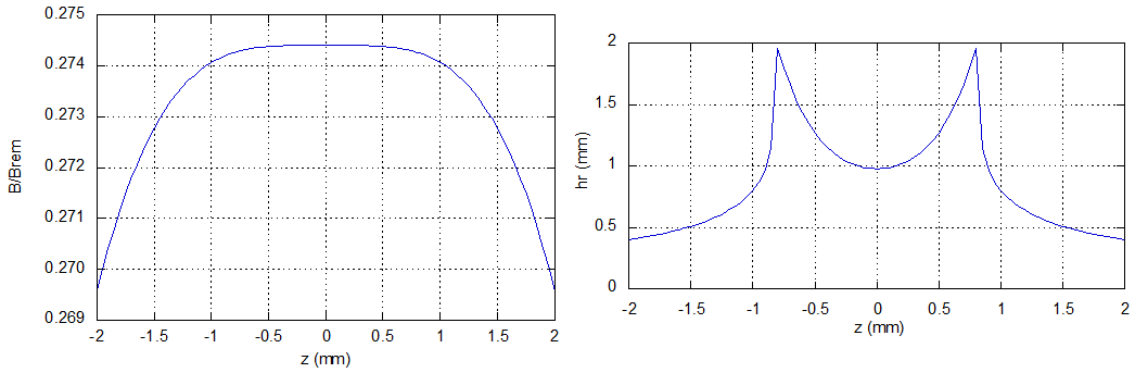


Figura 2.1-7. Intensidad de campo (izquierda) y radio de homogeneidad (derecha) a lo largo del eje z (eje de rotación) del imán.

2.1.2. Elemento sensor

El dispositivo consiste en una estructura sensora, circuitos de acondicionamiento de la señal y la interfaz de salida de datos.

El elemento sensor es un anillo dopado con impurezas tipo n, con N contactos distribuidos equidistantemente en la superficie del anillo [7] (ver Figura 2.1-8). Mediante el uso de *switches* NMOS, cada contacto puede ser conectado a la fuente de corriente, a tierra, al voltaje del circuito sensor o mantenerse flotante. Existen muchas combinaciones diferentes de realizar las conexiones que pueden dar información sobre el campo magnético. Se presenta a continuación una secuencia simple de medida con la sensibilidad asociada y el offset de campo nulo. Más adelante se propone otra secuencia con una cancelación de offset más eficiente.

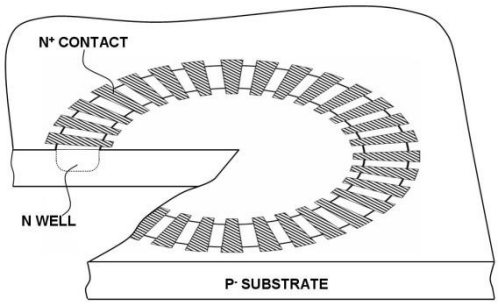


Figura 2.1-8. Estructura del dispositivo Hall circular.

Secuencia básica

La secuencia está basada en el elemento Hall vertical de 5 contactos (5CVH). Este elemento está compuesto por 5 contactos sucesivos alineados. La corriente se inyecta a través de los contactos exteriores y se recoge en el conector central. La diferencia de potencial entre los dos contactos restantes es proporcional la componente radial coplanaria del campo magnético:

$$V_H = \mu\, \theta\, V_{app}\ \boldsymbol{B} \cdot \boldsymbol{n} \qquad (1)$$

Donde μ es la movilidad de los electrones, que es aproximadamente de 0.1 T^{-1} para capas con dopado tipo n usual, ϑ es un factor geométrico adimensional y $\boldsymbol{n}$ es un vector unitario comprendido en el plano de la superficie y normal a la línea de contactos. En un instante de tiempo *t*, una serie de 5 contactos vecinos están conectados en la configuración 5CHV.

En un dispositivo perfecto, la salida del voltaje es proporcional a $Bcos(\alpha)$, donde α es el ángulo entre $\boldsymbol{n}$ y $\boldsymbol{B}$. En el instante $t+1/f_{CLK}$, donde f_{CLK} es la frecuencia de reloj, el elemento 5CVH se desplaza un contacto, y el voltaje Hall pasa a ser $B\cos(\alpha + 2\pi/N)$. Para una vuelta completa, el voltaje Hall toma la forma de una función sinusoidal, cuya amplitud es proporcional la componente coplanaria del campo y la fase es igual a la dirección del campo. Esto significa que la geometría circular genera una modulación natural de la señal, necesaria para suprimir el *offset* y el ruido de baja frecuencia de la fase de amplificación. La información sobre el campo queda comprendida en el primer armónico en un periodo de un giro completo.

En una estructura real, debido a las imperfecciones, el voltaje esta superpuesto a un voltaje de offset. La clave está en que un desplazamiento uno-a-uno del elemento 5CVH permite compensar este offset. Esto se puede entender fácilmente si modelamos un anillo Hall como un array circular de resistencias r_i no idénticas (ver Figura 2.1-9). En el instante *t* la medida del

voltaje es *I/2 (r_i-r_{i-1})* y en el instante *t+1/f_{CLK}* el mismo voltaje es *I/2 (r_i+r_{i-1})*, con lo que la contribución de la resistencia *i*-ésima se cancela. En otras palabras, si un defecto estático induce un exceso del voltaje en la salida en el instante *t*, el mismo defecto causará un voltaje opuesto en el instante *t+1/f_{CLK}*. En este modelo óhmico, el offset dc, o mejor dicho su suma al dar una vuelta completa es exactamente cero.

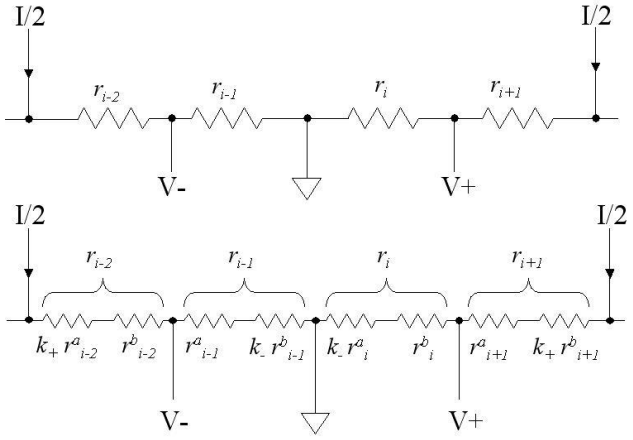


Figura 2.1-9. Modelo "array de resistencias". Arriba: Modelo óhmico. Abajo: modelo que incluye el efecto de unión de campo.

Aunque esta secuencia básica de medidas no suprime completamente el offset de campo nulo (cuando no hay campo magnético). Incluso si el offset desaparece, un array circular de resistencias, aleatoriamente distribuidas, crea un primer armónico no nulo. Puede demostrarse que para resistencias independientemente distribuidas con una variación media δr, la variación media del voltaje δV es $\sqrt{2}\,\delta r\ {}^{I}/_{2}$ y la amplitud media del primer armónico V_1 es

$$V_1 = \frac{\sqrt{2}\pi}{N^{3/2}}\delta V \qquad (2)$$

En dispositivos reales, hay efectos adicionales no lineales que contribuyen a incrementar el offset. El más importante es el efecto de unión de campo que mediante estrangulamiento del canal (en función del voltaje aplicado) causa una modificación dinámica del canal de conducción. Como resultado, la suma de los voltajes en un ciclo no es nula. Este efecto puede ser modelado por la sustitución de cada resistencia r_i por dos resistencias r_i^a y r_i^b en serie (Figura 2.1-9). Sus valores están modificados por un factor que depende del voltaje que hay en el contacto más cercano. Por ejemplo, una resistencia cercana al contacto de la corriente de polarización aumentaría por un factor *k+*, digamos de 1.3, debido a la gran diferencia de voltaje entre el canal n y el sustrato. Se simuló un array de 64 resistencias cuyos valores tienen una distribución gaussiana y toman factores de unión de campo de los experimentos realizados con el 5CVH simple. Sabiendo que el offset individual δV corresponde a 45mT, este cálculo produce un offset residual de 1.5mT. Por lo que el offset se reduce solo en un factor 1/30, mientras que para el modelo óhmico, de acuerdo con (2), el factor de reducción es 1/115.

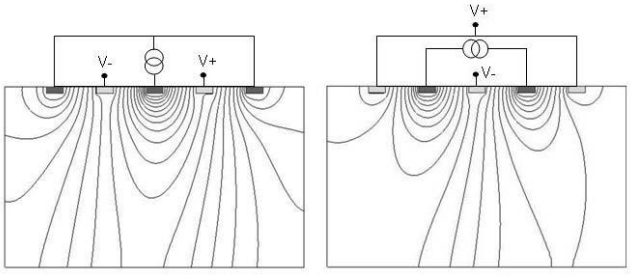


Figura 2.1-10. Cálculo del potencial con elementos finitos para dos pasos de la secuencia de *subspinning*. Los otros dos pasos se obtienen conmutando los contactos sensores por los contactos de polarización.

Secuencias de medidas alternativas

Dependiendo de la aplicación, otras secuencias pueden ser implementadas. Por ejemplo, se puede obtener una mayor relación señal-a-ruido sin utilizar más espacio en el chip de silicio. Esto se haría con un segundo 5CVH, diametralmente opuesto, en el anillo. Si el offset es un problema, se propone una secuencia especial llamada *subspinning*[5], que cancela la suma de voltajes de salida en un periodo mucho menor que el de un ciclo completo. El objetivo es suprimir el primer armónico residual dado por (2). Esto no ha sido probado hasta ahora, pero a continuación se da una secuencia *subspinning* particular. El elemento básico sigue siendo un 5CVH, en el que los dos contactos se acortan y se conectan a una única fuente de corriente para crear, así, un dispositivo de cuatro terminales. En cada posición del 5CVH, las cuatro conexiones se permutan circularmente, igual que en el método *spinning current* [8, 9] normal (ver Figura 2.1-10). Con una disposición perfecta de las resistencias estos cuatro pasos hacen que la suma se anule. Como consecuencia, el primer armónico se reduce fuertemente y quedaría como:

$$V_1 = \frac{\pi^2}{\sqrt{2}N^{5/2}}\delta V \qquad (3)$$

Con N=64, el factor de supresión del offset es 1/4695. Simulaciones numéricas muestran que la secuencia *subspinning* también suprime el offset debido al efecto de unión de campo: el modelo presentado anteriormente dejaba un offset residual de 40µT.

[5] Subspinning: se trata de una alternancia los papeles de los conectores que forman el dispositivo. Tomando una medida por cada configuración posible de las conexiones del 5CVH. Así se cancela el *offset* residual.

2.1.3. Diseño del sensor

La tecnología seleccionada para la realización del sensor es la tecnología CMOS de 0.35µm convencional de alto voltaje de "austriamicrosystems" (H35) [10]. En esta tecnología la tensión de alimentación es de 3.3V.

La parte sensora está compuesta por un pozo-n con forma de anillo y 64 contactos realizados con dopaje tipo n. El anillo está formado por una capa-n con 6µm de profundidad. El diámetro exterior del anillo es de 50µm. La anchura del anillo, así como el número, tamaño y forma de los contactos se determinó por optimización de la sensibilidad mediante cálculos de elementos finitos. Un circuito lógico asegura la correcta secuencia de conexiones. El segmento 5C se transforma en una estructura Hall vertical 4C más corriente de polarización. En cada segmento 4C se realiza una técnica de *spinning current* de cuatro fases, de forma que en total se obtienen 256 (4x64) medidas por cada secuencia completa de medida.

Diagrama de bloques básico del microsistema y formas de onda de la señal principal

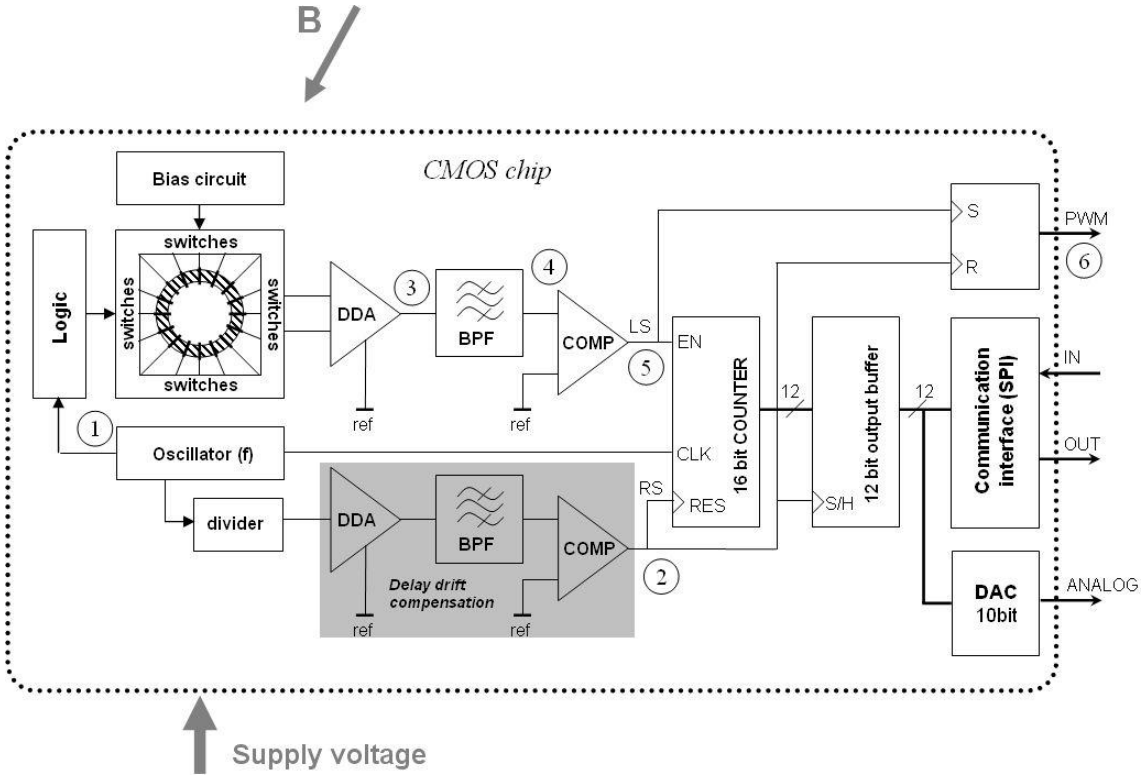


Figura 2.1-11. Diagrama de bloques del APS (Sensor de Posición Angular).

El principio de medida básico está basado en la medida del desfase entre la señal detectada (5) y la señal de referencia (2). Los conmutadores realizados con puertas de transmisión NMOS conectan la estructura sensora a la corriente de polarización, mediante un dispositivo que llamaremos *Differential Difference Amplifier* (DDA). La corriente de polarización se genera en

un circuito tipo *bandgap*[6]. El oscilador, así como los bloques *Logic* y *Divider* se construyen con celdas estándar digitales disponibles en fábrica. El DDA, el Filtro Pasa-Banda (BPF) y el Comparador (COMP) están optimizados para características de bajo ruido y un amplio rango de temperatura. La señal de salida está disponible en tres formatos diferentes: Modulación de Anchura de Pulso (PWM), Interfaz para Periféricos en Serie (SPI) y salida analógica con 10 bits de resolución. Más adelante se realiza una descripción detallada del procesado de la señal (*Procesado de la señal*).

En la Figura 2.1-12 se muestran las principales señales y se pueden asociar al diagrama de bloques de la Figura 2.1-11 gracias a las referencias numéricas.

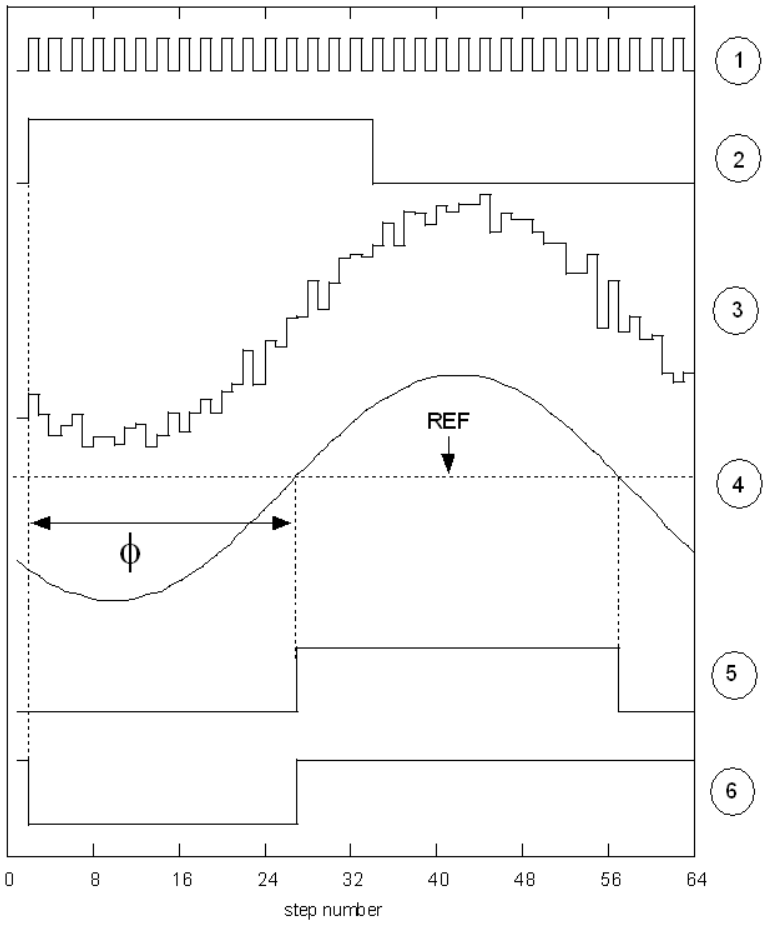


Figura 2.1-12. Señales principales.

[6] Se trata de un circuito formado por varios transistores CMOS. El nombre *bandgap* se debe a que la tensión de salida del circuito es aproximadamente igual a dos veces la tensión de la banda prohibida del silicio [11,12].

Procesado de la Señal

El voltaje Hall proveniente de la estructura es captado y amplificado por el DDA y posteriormente encaminado. La ganancia de amplificación se fija de forma que la oscilación de la señal amplificada es aproximadamente la mitad del rango completo de la tensión de alimentación (3.3V). La seña es posteriormente filtrada a través de un filtro pasa-banda de capacidad conmutada (SC) de orden 4, para suprimir cualquier señal parásita de alta frecuencia que se pueda inducir en los elementos Hall, así como los *offsets* y el ruido *1/f* causado por el DDA. La elección del filtro SC se debe a su excelente estabilidad en el filtrado de frecuencias y retardo en función de la temperatura. A continuación, la señal se filtra con un filtro pasa-baja pasivo RC con una frecuencia de corte mucho mayor (al menos 10 veces la frecuencia del ciclo de medida del anillo), lo que se hace para eliminar el rizado que se acopla en la señal al pasar por el filtro SC. La señal filtrada del 1^{er} armónico (señal 4 en la Figura 2.1-12) se transforma después en una onda cuadrada mediante un comparador (se obtiene la señal 5 de la Figura 2.1-12). De este modo, se puede comparar la fase de la señal cuadrada medida con la señal de referencia gracias a un circuito detector de fase lógica COMP. De esta forma, la señal analógica se convierte en una señal con formato PWM. Así, una señal una señal puede ser fácilmente convertida en una palabra digital utilizando después un contador. La señal digitalizada se puede leer a través de la interfaces PWM o SPI, o ser convertida a analógica mediante un Conversor Digital a Analógico.

Con el fin de minimizar la influencia de las derivas de la fase con la temperatura causadas a la electrónica de la fase de medición, la señal de referencia se trata de la misma forma que la señal detectada. En otras palabras, la señal de referencia, con una magnitud y nivel de señal adecuados, se hace pasar por un circuito duplicado en paralelo compuesto por el DDA, el filtro SC y el COMP. De esta manera ambas señales están expuestas al mismo retraso y las derivas se minimizan.

Un sensor de temperatura *on-chip*, basado en un circuito de banda prohibida ayudará a un mejor análisis de los efectos de la temperatura en el microsistema.

Acondicionamiento de la Señal

El circuito de acondicionamiento del se realiza con un circuito de banda prohibida con el fin de hacerlo insensible a las variaciones de tensión y mejorar el control de las derivas de temperatura de la corriente de polarización en la parte sensora (elemento Hall). Dado que la parte sensora es polarizada con una corriente relativamente baja para mantener débiles los efectos no lineales, la caída de tensión en la parte sensora no debe sobrepasar el rango de trabajo de un transistor MOS tipo N. Por tanto, la parte sensora está conectada al circuito de acondicionamiento y al circuito de procesado de la señal por medio de puertas transmisoras compuestas únicamente por transistores NMOS con tamaños optimizados. Esto permite disminuir al mínimo la inyección de carga al camino de procesado de la señal.

La frecuencia de reloj del microsistema puede ser también controlada por un oscilador *on-chip* de de baja potencia de 4MHz o mediante una fuente exterior.

Estimación del Layout del Sensor

- El tamaño estimado del chip es de 1.9 x 1.9 mm aproximadamente.
- El sistema tiene 32 patillas de conexión exterior de 120µm.
- La parte sensora está centrada con respecto a las líneas de salida del chip.

Las conexiones de alimentación de la parte digital y analógica están divididas en dos partes, con el fin de aislar la parte analógica de la parte digital y evitar el *feedthrough*[7] en la señal de reloj.

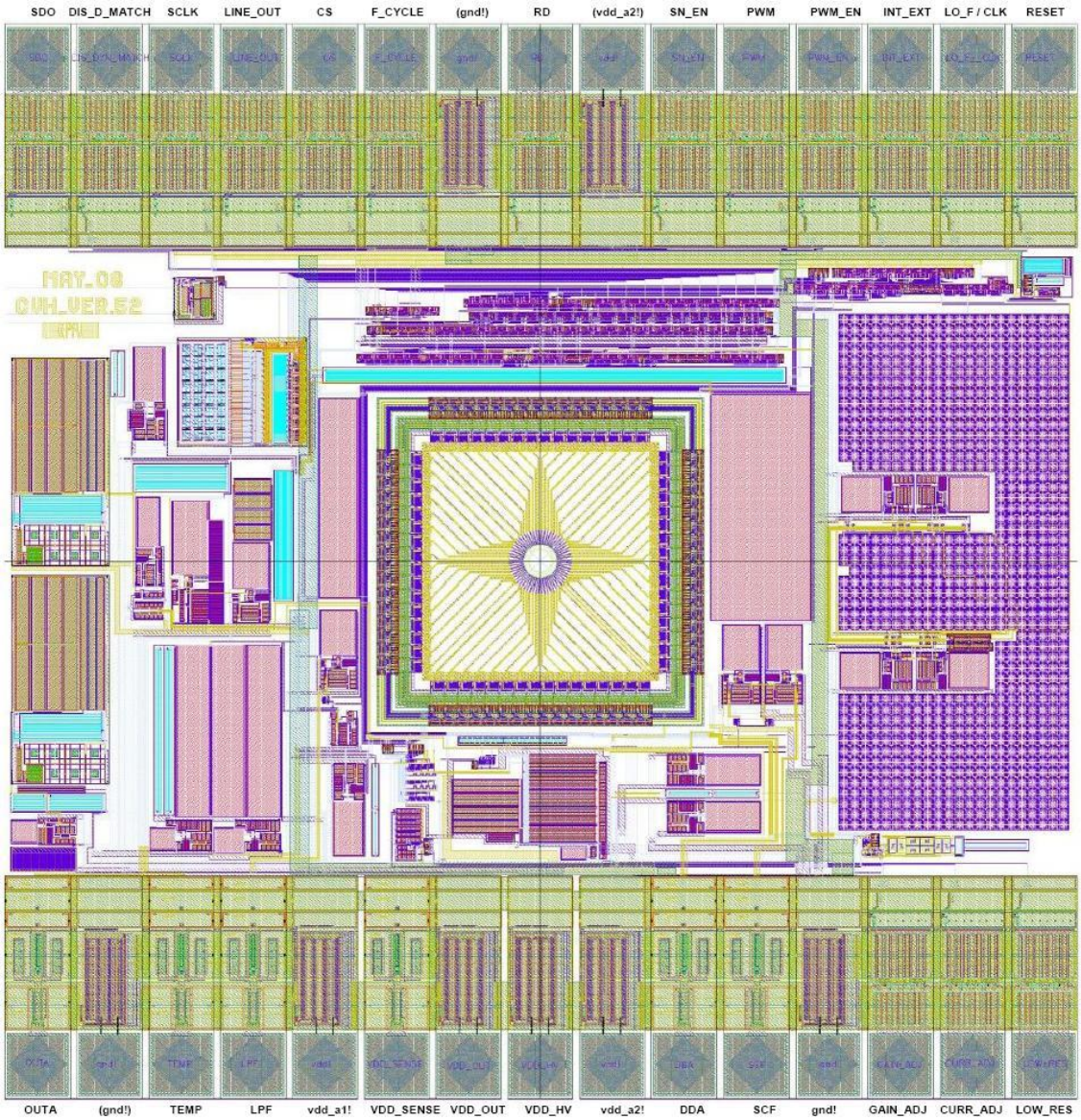


Figura 2.1-13. Borrador del layout en chip de silicio del APS.

[7]El reloj suele generar cierta autoinducción en los conversores analógico-digitales, que se conoce como *feedthrough*. Existen diferentes tipos de técnicas y circuitos para evitar este problema [13, 14, 15]. Algunos ejemplos se puede ver en sitios como:

Lista de Conexiones (pines)

Pad name	*Type / direction*	*Description*	*Default value*
OUTA	analog / out	Analog output proportional to an in plain magnetic field direction (0 – 360 ° => 0 – 2.4 V)	max value @ master reset
		Ground	
TEMP	analog / out	Analog output proportional to a chip temperature with PTC = ~4.191mV / °C	~1.16 V @ 0 °C
LPF	analog / out	Internal node monitoring – low-pass filter output, the output is activated by SN_EN signal	high impedance, disabled
vdd_a1!	power	Analog positive supply voltage 3.3 V (analog part 1)	3.3 V supplied
VDD_SENSE	analog / in	5.3 V module, sensing input connected to 3.3 V power supply track on baseboard	high impedance, 3.3 V supplied
VDD_OUT	power	5.3 V module, power output delivering 3.3 V to the power supply track on baseboard	3.3 V
VDD_HV	power	5.3 V module, positive supply voltage 5.3 V	5.0 – 5.5 V, supplied
vdd_a2!	power	Analog positive supply voltage 3.3 V (analog part 2, SC module)	0 V, supplied
DDA	analog / out	Internal node monitoring – differential difference ampl. output, the output is activated by SN_EN signal	high impedance, disabled
SCF	analog / out	internal node monitoring – switched capacitor filter output, the output is activated by SN_EN signal	high impedance, disabled
gnd!	power	Ground 0 V	3.3 V, supplied
GAIN_ADJ	digital / in	Gain adjustment, log 0 = 180, log 1 = 360	log 0
CURR_ADJ	digital / in	Hall bias curr. adj., log 0 = 0.15 mA, log 1 = 0.25 mA	log 0
LOW_RES	digital / in	Data stream out 5 Hz / 30 Hz => default 5 Hz	log 0
RESET	digital / in	Master reset, system initialization, active on log 1	log 0
LO_F / CLK	digital / in	Internal clock frequency divider by 2, active on log 1; external clock input – activated with INT_EXT signal	log 0
INT_EXT	digital / in	Mode change of LO_F / CLK, log 0 => internal clock generator & LO_F mode, log 1 => external clk mode	log 0
PWM_EN	digital / in	PWM output enable, log 0 => disable, log 1 => enable	log 0
PWM	digital / out	Pulse-Width-Modulation output	log 0, disabled
SN_EN	digital / in	Service mode enable, monitoring outputs activation, log 0 => service mode disabled, log 1 => enabled	log 0
	power	voltage 3.3 V	
RD	digital / in	Reverse measurement direction, clockwise – counterclockwise	log 0
	power	Digital ground 0 V (digital part)	0 V, supplied
F_CYCLE	digital / out	Internal node monitoring – reference frequency output, the output is activated by SN_EN signal	log 0, disabled
CS	digital / out	SPI module, Chip Select	log 0
LINE_OUT	digital / out	Internal node monitoring – comparator output, the output is activated by SN_EN signal	log 0, disabled
SCLK	digital / out	SPI module, Serial Clock	log 0
DIS_DYN_MATCH	digital / in	Dynamic cancellation of the temperature effect on analog electronics, log 1 => cancellation disabled	log 0
SDO	digital / out	SPI module, Serial Data Out	log 1

Tabla 2.1-2. Lista de conexiones del chip que contiene el sensor APS.

Montaje Mecánico

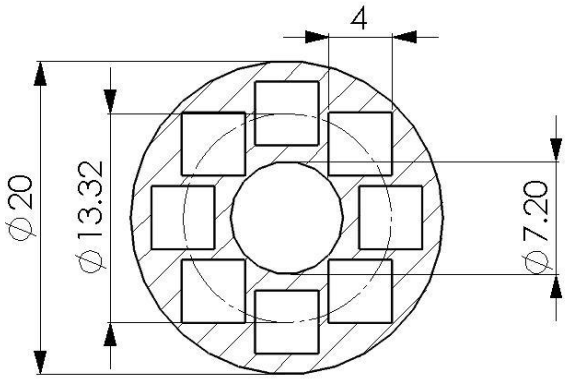


Figura 2.1-14. Matriz de aluminio con algunos parámetros ajustables.

Robustez del Sensor

La robustez y la precisión del sistema dependen de las siguientes características:

- La fuerza del campo magnético (270mT), la homogeneidad (volumen de homogeneidad: 1mm) y la distancia de dispersión de campo (22mm).
- Minimizaciones del offset inducido al nivel del sensor.
- Cancelación dinámica de la dependencia de las derivas de temperaturas. Dado que el esquema de medida es insensible a la magnitud del campo magnético medido.

Con estas características el sistema debería cumplir con el requisito de 0.05º.

2.1.4. Medidas realizadas sobre el APS versión 5.2.

La primera fase del desarrollo del proyecto fue la realización de medidas sobre el sensor APS versión 5.2. Las medidas se realizaron en un laboratorio convenientemente equipado, del Departamento de Microsistemas de la EPFL. El equipo principal del laboratorio estaba formado por un gran electroimán (ver Figura 2.1-15), una gran fuente de alimentación para el electroimán y un circuito de agua para la refrigeración del imán, cuyo motor y depósito se encuentran situado en una habitación contigua. En la misma sala se encuentran algunos equipos informáticos, así como varios osciloscopios, generadores de señales, fuentes de alimentación… además de un equipo de resonancia magnética (RM) para el análisis del campo magnético generado y una máquina para alterar las condiciones de temperatura de la medida (*Temptronic*).

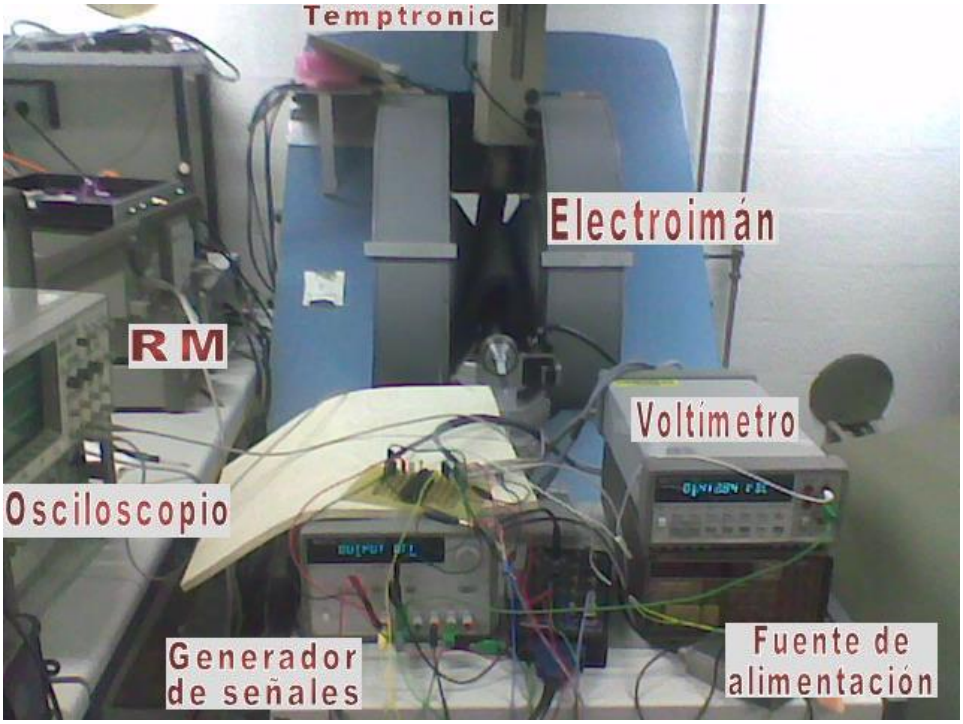


Figura 2.1-15. Fotografía del equipo del laboratorio.

En el centro de la Figura 2.1-15 puede verse el rotor que sujeta un extremo de una varilla metálica. En el otro extremo de la varilla se encuentra el sensor, situado así en el centro del imán, justo donde se cumplen las especificaciones de homogeneidad de campo que necesita el sensor para una buena medida. El rotor tiene un motor eléctrico muy preciso que se controla con un programa informático (ver Figura 2.1-16) que, al mismo tiempo, que hace girar al rotor y consecuentemente al sensor, va tomando medidas.

Campo magnético en el electroimán

En las ilustraciones 2.1-16 y 2.1-17 se muestra la posición de las bobinas en el electroimán. Entre ellas es se concentra el campo magnético con la homogeneidad deseada para el correcto funcionamiento del chip. Rodeando a las bobinas se puede ver un material ferromagnético de grandes dimensiones que sirve para confinar el campo magnético y evitar interferencias con los equipos de su entorno. En estas ilustraciones también se representa mediante flechas la dirección que sigue el flujo magnético a lo largo del material ferromagnético, así como en la parte de vacio (aire) que hay entre las bobinas (donde se sitúa el chip).

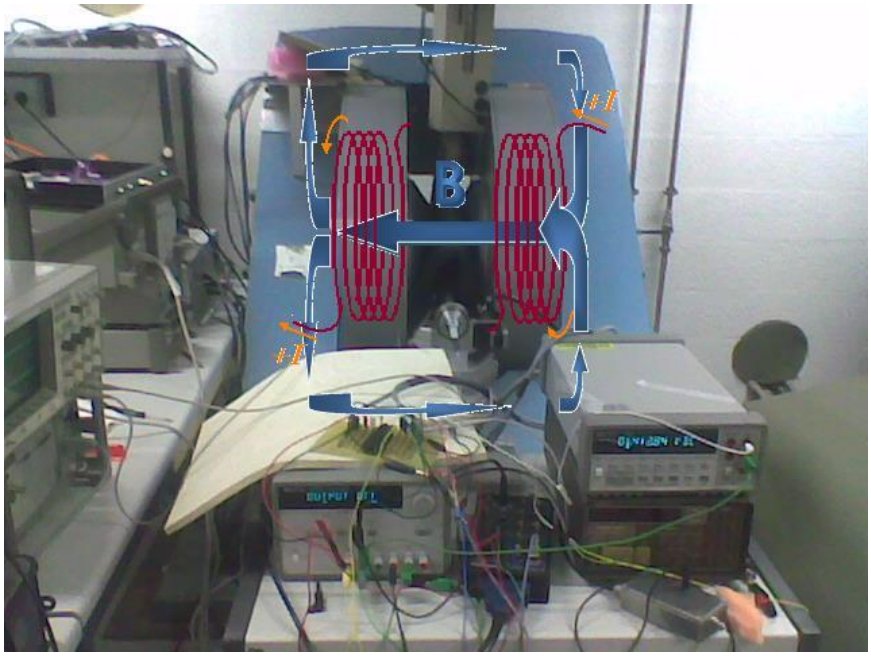


Figura 2.1-16. Representación del flujo de campo magnético en el imán cuando la intensidad de corriente es positiva.

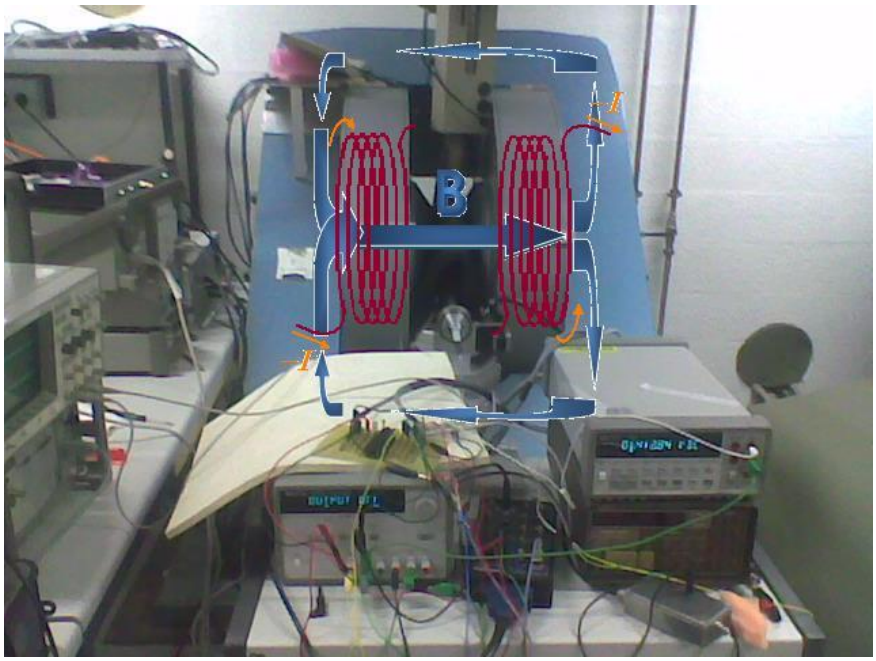


Figura 2.1-17. Representación del flujo de campo magnético en el imán cuando la intensidad de corriente es negativa.

Notas sobre la puesta en funcionamiento

De todas las conexiones que se muestran en la tabla 2.1-2, sólo es necesario conectar las que se especifican en la tabla 2.1-3.

Pad		*Type / direction*	*Description*	*Default value*
num	*name*			
1	**OUTA**	analog / out	10 bits resolution analog output proportional to the in-plane magnetic field direction (0 – 360 ° => 0 – 2.4 V, approx. values)	-
3	**TEMP**	analog / out	Analog output proportional to a chip temperature with PTC = ~ 4 mV / °C	~ 1.16 V @ 0 °C
5	**vdda!**	power	Analog positive supply voltage 3.3 V (analog part)	3.3 V supplied
9	**vdd!**	power	Analog positive supply voltage 3.3 V (digital part)	3.3 V, supplied
12	**gnd!**	power	Ground 0 V	0 V, supplied
13	**RANGE A**	digital / in	Magnetic field range adjustment	log 0
14	**RANGE B**	digital / in	Magnetic field range adjustment	log 0
15	**LOW_RES**	digital / in	Data stream-out rate : 5 Hz or 30 Hz	log 0
23	**RD**	digital / in	Reverse measurement direction, clockwise – counterclockwise	log 0
26	**CS**	digital / out	SPI module, Chip Select	log 0
28	**SCLK**	digital / out	SPI module, Serial Clock	log 0
30	**MOSI**	digital / out	SPI module, Serial Data Out	log 1

Tabla 2.1-3. Resumen de conexiones necesarias para la puesta en funcionamiento del chip.

- Campo magnético aplicado.

Las medidas se realizaron sobre una versión del chip de entrenamiento más compleja que las versiones comerciales. Se definen los 4 rangos de funcionamiento mostrados en la tabla 2.1-4.

Applied field (mT)	**RANGE A**	**RANGE B**
70mT to 110mT	1	1
110mT to 160mT	1	0
160mT to 210mT	0	1
210mT to 310mT (default)	0	0

Tabla 2.1-4. Rangos de funcionamiento recomendados.

Las variables ***RANGE A*** y ***RANGE B*** son señales de entrada al chip que se utilizan para establecer el rango de funcionamiento del sensor. Se corresponden con ajustes de la ganancia del bloque DDA (ver Figura 2.1-11) y de la corriente de polarización del dispositivo Hall. Estos ajustes son necesarios para mantener la precisión del sensor en los distintos rangos de funcionamiento.

Gain adjustment (RANGE A)	0	Low gain (default value) ⇒ DDA gain is 180
	1	High gain ⇒ DDA gain is 360
Current adjustment (RANGE B)	0	Low current (default value) ⇒ 150µA
	1	High current ⇒ 250µA

Tabla 2.1-5. Ajustes de ganancia y corriente.

Aunque el chip funciona correctamente en los cuatro rangos especificados, existe una intensidad de campo ideal para cada uno de ellos (tabla 2.1-6).

RANGE A	RANGE B	Optimum applied field (mT)
1	1	108 mT
1	0	162 mT
0	1	209 mT
0	0	313mT

Tabla 2.1-6. Campo óptimo para obtener la mejor estabilidad con la temperatura.

Los resultados empeorarán fuera de estos rangos, pero el sensor puede funcionar hasta los valores límite que se establecen en la tabla 2.1-7.

		RANGE A	RANGE B
Max field	Tested up to 2T	0	0
Min field	25 mT	1	1

Tabla 2.1-7. Intensidades de campo límites de funcionamiento del sensor.

- Tasa de salida de datos.

En esta versión del chip existe, también, la posibilidad de escoger entre dos grados de resolución. Por tanto, se debe escoger un valor para la señal de entrada ***LOW_RES*** según la información facilitada por la tabla 2.1-8.

LOW_RES	Stream-out rate
0 (default)	5 Hz
1	30 Hz

Tabla 2.1-8. Tasas de salida de datos.

- Alimentación.

El voltaje de alimentación del sensor (vdda! y vdd! en la tabla 2.1-2) debe tener un valor nominal no menor de 3.3V, pudiendo ampliarse hasta 3.6V con una fluctuación menor del 0.5%. El consumo del sensor es de unos 5mA.

- Interfaz SPI *(Serial Peripheral Interface)*.

El sensor se conecta a un equipo informático mediante una interfaz SPI. A esta interfaz se conectan las siguientes señales del sensor:

- CS: señal *chip select* generada por el sensor.
- SCLK: señal de reloj generada por el sensor. Frecuencia aproximada: 2.5 kHz.
- MOSI: señal *master output – slave input* generada por el sensor. 16bits de datos en serie validos en el flanco de subida de la señal de reloj.

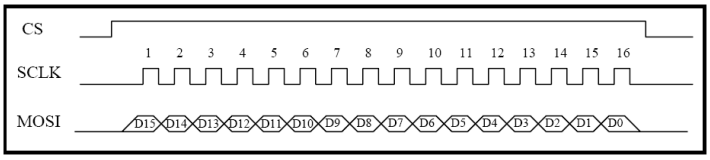


Figura 2.1-18. Formato de las señales CS, SCLK y MOSI del sensor.

En una medida típica se debe especificar:

- La excursión angular de la medida, es decir, los ángulos de inicio y finalización de medidas y el intervalo entre ángulos intermedios.
- El número de muestras con las que se desea obtener la media que se tomará como medida de cada ángulo.
- Y el tiempo entre movimientos.

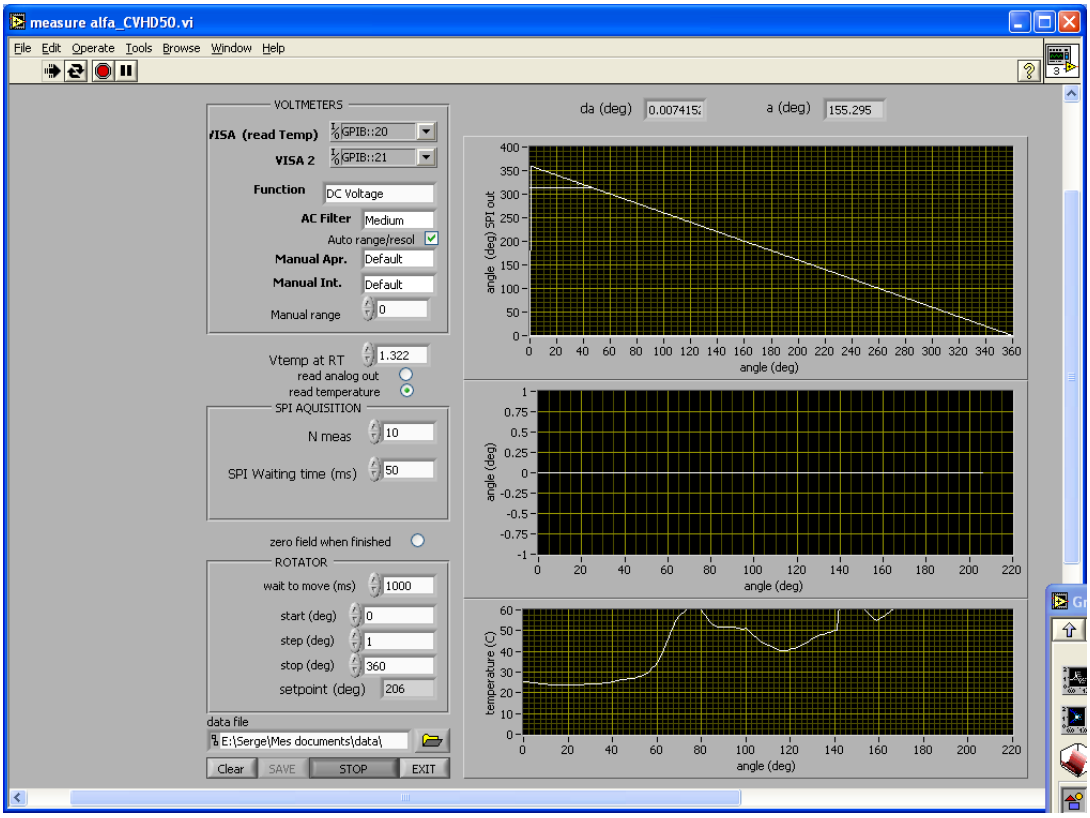


Figura 2.1-19. Programa de medidas "*mesure_alfa_CVHD50*", de *LabVIEW 7.1*.

En la Figura 2.1-19 se puede ver que el programa de medidas utilizado ofrece tres representaciones gráficas. La primera de ellas es la única válida en las medidas realizadas, ya que las otras dos corresponden a salidas opcionales de ciertas versiones del chip, pero que no se han considerado interesantes en estas medidas. En la gráfica superior se observa el ángulo medido por el sensor (en realidad una media de 10 muestras por cada ángulo) en función del ángulo dado por el rotor, que es siempre relativo a la posición inicial. Concretamente en esta medida se realizó una excusión de 360º a 0º, de ahí que la pendiente de la gráfica sea negativa.

Los resultados obtenidos con este tipo de medidas se guardaban en archivos de texto para poder ser analizados posteriormente y cuyos resultados se exponen a continuación.

Medidas de caracterización del sensor

La primera medida realizada consiste en una excursión de una vuelta completa del sensor en el sentido de las agujas del reloj, es decir, de 360º a 0º. El programa informático va deteniendo al rotor en intervalos de un grado, para obtener y guardar la media y la desviación típica de una sucesión de muestras (10 en este caso). Esta primera medida tiene el objetivo de comprobar las características básicas del sensor. Para ello se aplican dos campos magnéticos de distintas intensidades, es decir, primero se realiza una excursión angular con una intensidad de campo determinada y, después, se cambia la intensidad magnética para realizar otra serie de medidas. Los campos aplicados son el ideal de 0.3 Teslas y otro menor de 0.16T. La intensidad de campo magnético de 0.3T se considera ideal porque genera en el sensor una respuesta de la misma amplitud que la señal de referencia (ver "*Procesado de la señal*" en el apartado 2.1.3) y, por tanto, un funcionando en condiciones ideales. Con el otro campo aplicado, al ser menor, se genera en el sensor una respuesta de menor amplitud que la de referencia, produciendo un pequeño offset en la precisión del dispositivo.

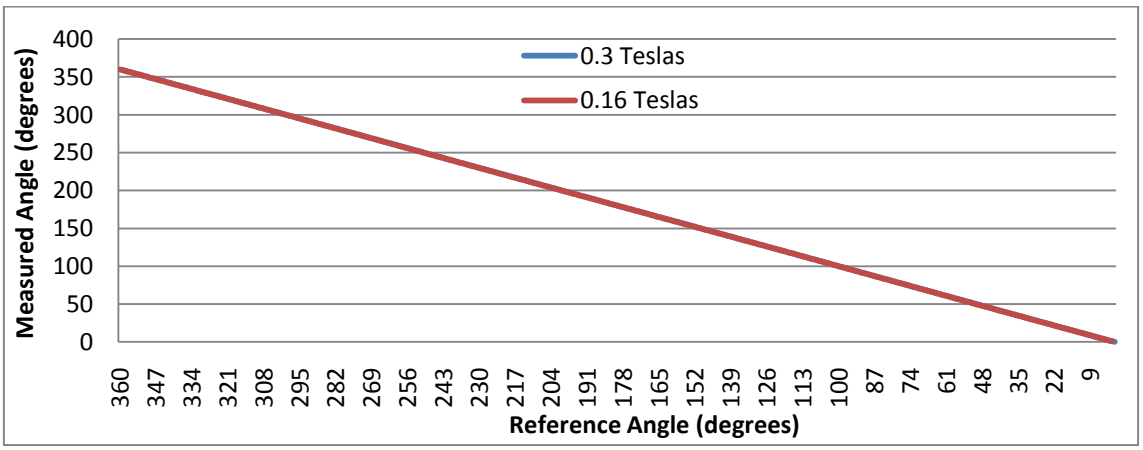


Figura 2.1-20. Representación de la ángulo medido con B=0.3T y B=0.16T.

En la Figura 2.1-20 se representan las dos medidas angulares realizadas en los 360º con los dos campos aplicados. La pendiente de la curva es negativa porque en el eje horizontal se muestran los ángulos relativos a la posición inicial, sin tener en cuenta la dirección del giro. A la vista del ángulo medido, el resultado es prácticamente idéntico para los dos campos aplicados.

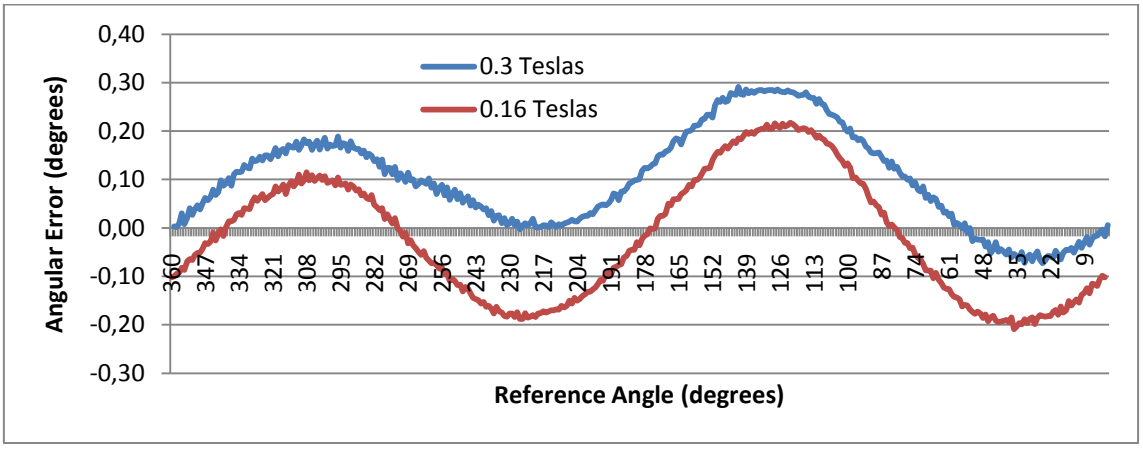


Figura 2.1-21. Representación de la error con B=0.3T y B=0.16T.

Sin embargo en la Figura 2.1-21 se aprecia un offset entre las medidas de los dos campos aplicados. Esto se debe a que la señal generada por el sensor tiene menor amplitud que la de

referencia, lo que genera problemas en el comparador, porque se desplaza la linea de cruces por cero (ver apartado 2.1.3).

En la Figura 2.1-21 se muestra el error de la medida respecto al angulo del rotor que se considera como ángulo de referencia (antes de realizar cualquier medida se busca la posición de 0º dada por el sensor para establecer los 0º del rotor). Esta representación del error sirve, también, para establecer la precisión del dispositivo, unos 0.3° en condiciones de campo ideal (0.3T). Este error que depende de la posición del dispositivo no se puede corregir. No se debe a ningun offset del dispositivo, puesto que estos se eliminan gracias a la técnica de spinning-current y el subspinning que realiza el propio chip. Se cree que este error se crea en el proceso de fabricación, tal vez debido a pequeñas anisotropias en el material. Esto explicaría la existencia de una dirección óptima en el chip (donde se produce un error mínimo). Además se ha comprobado que con distintos chips esta dirección cambia, con lo que la hipotesis del error de la fabricación cobra sentido. Quedaría por comprobar si todos los chips de una misma oblea de silicio adquieren la misma dirección de anisotropía. Esto, sin embargo, no es tan fácil como parece porque los chips llegan ya cortados y se sabe que los chips contiguos en una misma oblea no estan alineados, sino que se fabrican de forma perpendicular.

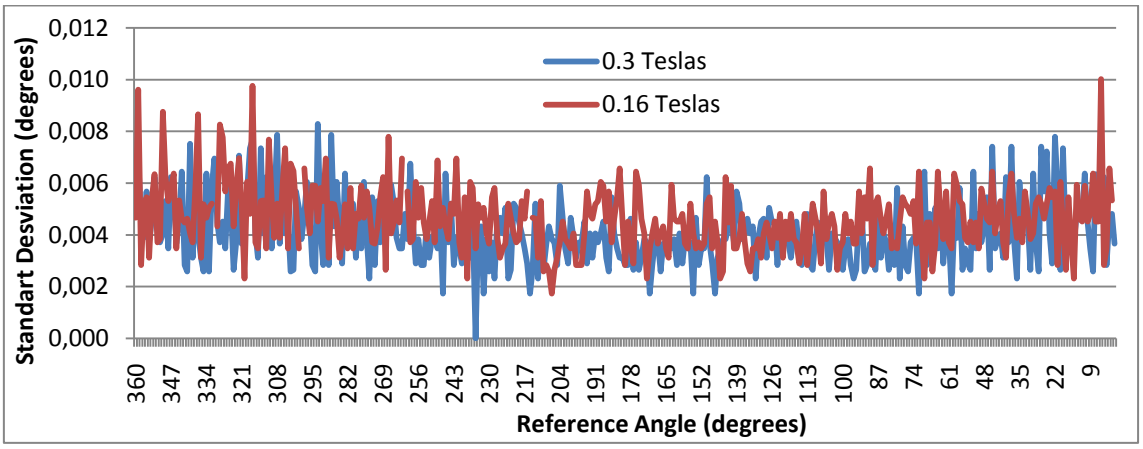


Figura 2.1-22. Representación de la desviación estándar con B=0.3T y B=0.16T.

Como último paso de esta medida, la Figura 2.1-22 muestra la desviación estándar de las medidas realizadas con las dos intensidades de campo. Esta desviación estándar, al realizarse pon sólo 10 muestras por ángulo, no puede tomarse como un dato relevante por sí solo. Pero tras observar resultados similares en las próximas medidas, podrá comprobarse la precisión del sensor.

Medidas del efecto de la temperatura

Al principio de este capítulo se especifica que se busca una precisión absoluta de ±0.05º en un rango de temperaturas comprendido entre -50ºC y 80ºC. Con las medidas que se muestran a continuación se pretende mostrar cuán cerca está el sensor de conseguir este objetivo.

Para ver el comportamiento general del sensor respecto a la temperatura se realizan algunas excursiones de giro completo (360º), realizando medidas como antes, a intervalos de un grado y tomando 10 muestras por ángulo. Pero ahora, además de cambiar el campo aplicado entre cada giro completo, se cambiará la temperatura del chip.

Para alterar la temperatura del chip disponemos del equipo llamado Temptronic. Se trata de un calentador-refrigerador de aire al que hay que conectar un compresor de aire. El Temptronic tiene un tubo de salida de aire que se puede ver en el centro de la parte superior de la Figura 2.1-15. El dispositivo permite regular la temperatura del aire de forma bastante precisa gracias a un sensor que se coloca en el interior del tubo, pero cerca de la salida final del aire. Pero por muy preciso que sea el Temptronic, no podemos situar el APS dentro del tubo de aire. Por lo tanto, habrá que analizar en qué grado se transmite la temperatura del tubo del Temptronic al sensor APS.

La parte final del tubo de aire es de goma, de forma que se puede introducir en el imán hasta casi tocar el sensor sin alterar en gran medida el campo magnético. La disposición física del tubo, así como la de los aislantes que se añaden alrededor del sensor APS, obliga a no moverlos entre las realizaciones de medidas que se pretendan comparar. En caso contrario se estarían alterando sustancialmente las condiciones del experimento y este perdería su validez.

Bajo estas condiciones se toman medidas con los siguientes parámetros:

Try	**Field Intensity**	**Air Temperature**	**RANGE A**	**RANGE B**	**Notes**
1	0.3 T	80ºC	0	0	
2	0.15 T	80ºC	0	1	Wrong range for this field
3	0.3 T	120ºC	0	0	
4	0.3 T	40ºC	0	0	
5	0.15 T	60ºC	1	0	
6	0.15 T	120º	1	0	

Tabla 2.1-9. Parámetros de las medidas del efecto de la temperatura.

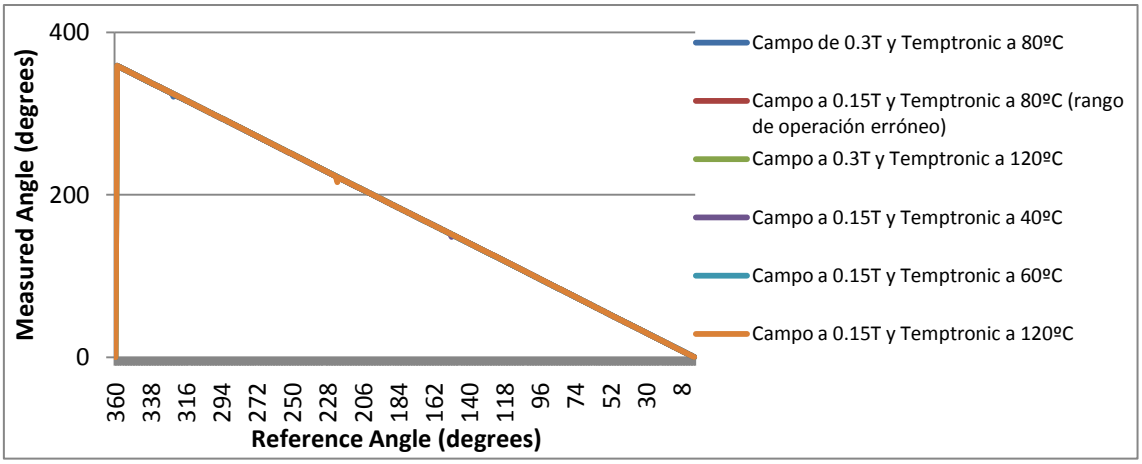


Figura 2.1-23. Representación del ángulo medido con diferentes campos y T^{as}.

En la Figura 2.1-23 se superponen las representaciones del ángulo medido para cada uno de los experimentos realizados (ver tabla 2.1-9). Con esta representación se comprueba que el sensor funciona correctamente con bajo las distintas condiciones de funcionamiento que se han experimentado. Pero, realmente no resulta muy representativo, porque en una escala de 360º no se pueden apreciar las pequeñas diferencias de la medida. Más útil será la representación de la desviación estándar que se puede ver en la Figura 2.1-24. Lo primero que se puede apreciar en esta gráfica es el desplazamiento que sufre la desviación típica del experimento 2 (en rojo, ver también tabla 2.1-9). Este desplazamiento se debe a una mala configuración del rango de campo aplicado. El experimento se configuró con los valores de RANGE A y RANGE B contrarios a los indicados en la tabla 2.1-4 para observar el efecto producido. Con ello se ha podido comprobar la importancia de corregir la ganancia del elemento DDA y la corriente de polarización del sensor, si se aplica un campo de menor intensidad.

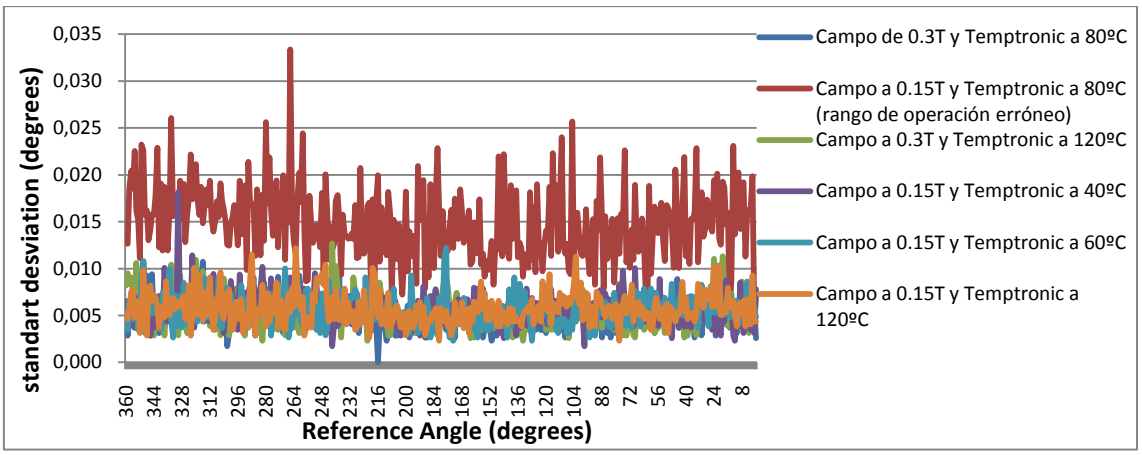


Figura 2.1-24. Desviación estándar con variación de campos y temperaturas.

Par poder apreciar correctamente los efectos de la temperatura en la desviación estándar de las medidas, en la Figura 2.1-25 se descarta el experimento 2. De esta forma, la gráfica muestra que no existen cambios aparentes entre las medidas de cada experimento, por lo que se puede afirmar que los cambios realizados en la temperatura y el campo aplicado no afectan a la precisión del sensor. O, para ser más exacto, los cambios de temperatura probados, no afectan a la precisión del APS, y los efectos producidos por los cambios de campo aplicado se

pueden corregir de forma bastante satisfactoria con las modificaciones de la ganancia del elemento DDA y la corriente de polarización.

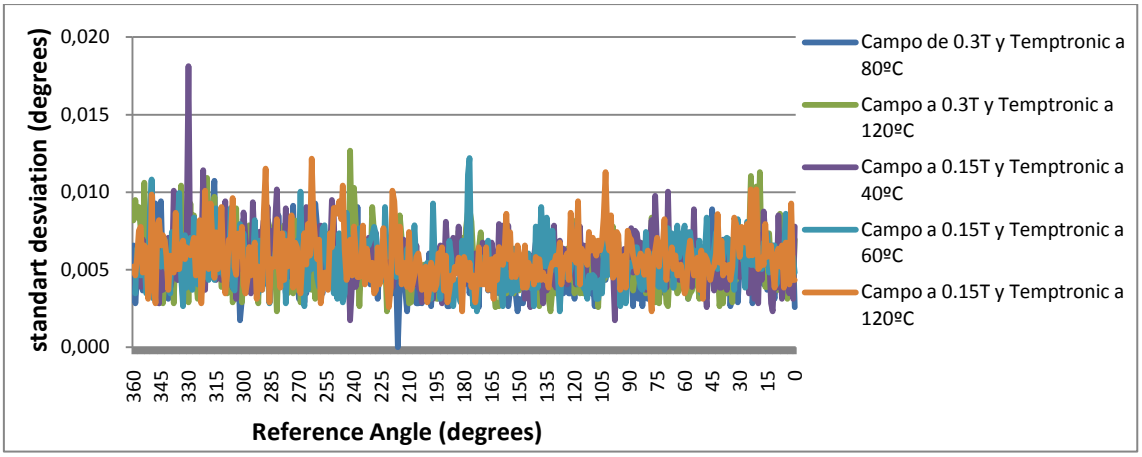


Figura 2.1-25. Desviación estándar con variación de campos y temperaturas (medidas correctas).

Pero si se analiza detalladamente el error de la medida angular para cada experimento (tomando como error angular la diferencia entre el ángulo medido y el de referencia, dado por el rotor[8]) se puede detectar una diferencia entre los experimentos realizados.

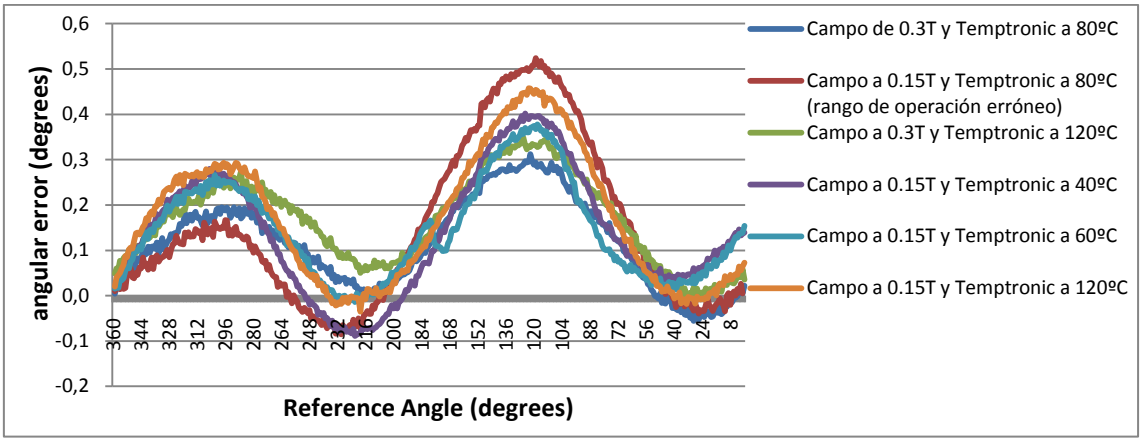


Figura 2.1-26. Representación del error de la medida respecto al ángulo de referencia del rotor.

En la Figura 2.1-26 se puede ver que el margen de error aumenta al alejarnos de las condiciones ideales. Se puede ver que el menor error se produce en el experimento 1, en el que se aplica el campo ideal (0.3T) y la temperatura está en el límite del rango aconsejado (80ºC). Si comparamos este experimento con el realizado a temperatura ambiente (representado en la Figura 2.1-21 en azul), veremos que los resultados respecto al error son similares. Esto confirma la fiabilidad del sensor en una parte del rango de temperaturas aconsejado [$T^{a}_{ambiente}$, 80ºC]. Aunque debe tenerse en cuenta que los 80ºC se refieren a la temperatura del aire que se hace circular alrededor del chip, que además están medidos en el interior del tubo. Además el chip gira para tomar cada medida, quedando solo 1 segundo detenido en cada ángulo. Por lo tanto, es de suponer que el chip se encuentre a una temperatura inferior a los 80ºC.

[8] Aunque el ángulo del rotor no esté libre de cierta imprecisión, es lógico pensar, que la desviación sobre el ángulo real sea igual para todos los ángulos. De forma que fijando una posición inicial como origen de ángulos puede suponerse suprimida dicha desviación para considerar el ángulo del rotor como referencia válida.

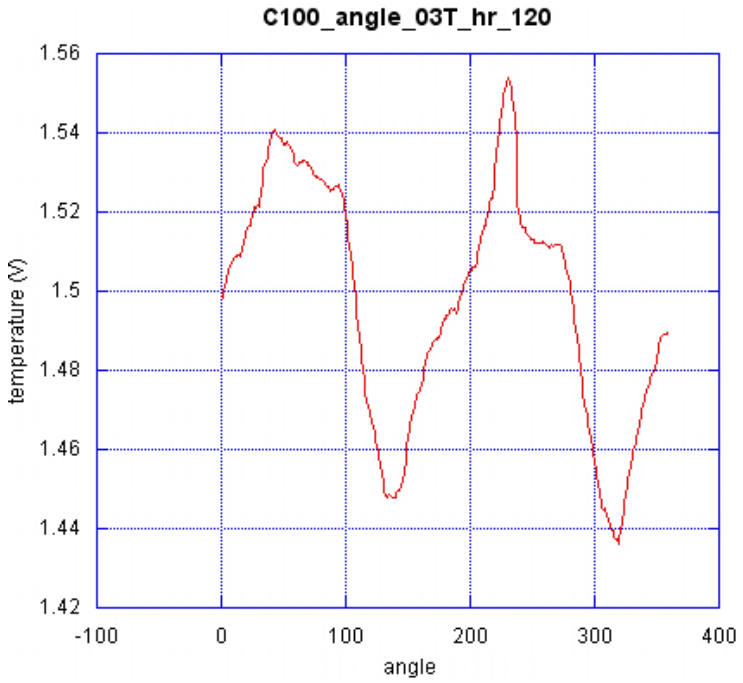


Figura 2.1-27. Transferencia de calor al chip en función del ángulo de exposición.

Para comprender mejor como afecta la temperatura exterior (el chorro de aire) al chip observemos la Figura 2.1-27. En ella se muestra (en Voltios) la salida del sensor de temperatura que tiene integrado el chip. De la forma de la curva se deduce que el chorro de aire caliente tiene una influencia distinta según el ángulo de incidencia al chip[9]. Esto se puede explicar con la asimetría del relieve final del chip. Hay que tener en cuenta que la versión actual del chip necesita que se le conecten dos grandes condensadores[10] para eliminar algunos acoplos. Estos condensadores producen, sobre el sensor, una alteración del flujo de aire que cambia en función de su posición relativa con la dirección incidencia dl aire. Por tanto, para caracterizar el comportamiento del sensor en función de los cambios de temperatura, sería mejor fijar un ángulo sobre el que realizar las distintas medidas.

Atendiendo a la posición de los condensadores exteriores, se realizaron algunas pruebas para determinar una posición ideal. Finalmente se decide fijar el sensor en un ángulo de 100º, posición en la que se comprueba una alta relación entre las variaciones de temperaturas de aire y la temperaturas (en V) leídas en el sensor interno del chip. En este ángulo se realizan cuatro experimentos de cambio de temperaturas, con cuatro intensidades de campo distintas (una por cada rango de funcionamiento), tal y como se muestra en la tabla 2.1-10.

Try	**Field Intensity**	**Air Temperature**	**RANGE A**	**RANGE B**	Field range (T)
1	0.08 T	22ºC to 130ºC	1	1	0.07T to 0.11T
2	0.12 T	22ºC to 130ºC	1	0	0.11T to 0.16T
3	0.2 T	22ºC to 130ºC	0	1	0.16T to 0.21T
4	0.3 T	22ºC to 125ºC	0	0	0.21T to 0.31T

Tabla 2.1-10. Parámetros de los experimentos de caracterización del efecto de la Tª.

[9] En la Figura se representa el ángulo de referencia (respecto a la dirección del campo magnético) dado por el rotor. El ángulo de incidencia del aire sobre el sensor es perpendicular al flujo magnético.
[10] Condensadores de 10µF y 100µF, de un tamaño mayor al del sensor.

Field of 0.08T (RA=1, RB=1)			
Air T (°C)	Chip T(V)	angle	error abs
22	1.3210	99.880	0.120
30	1.3352	99.887	0.113
40	1.3550	99.880	0.120
55	1.3850	99.850	0.150
70	1.4150	99.830	0.170
85	1.4460	99.814	0.186
100	1.4765	99.805	0.195
115	1.5120	99.799	0.201
130	1.5430	99.785	0.215

Field of 0.12T (RA=1, RB=0)			
Air T (°C)	Chip T(V)	angle	error abs
22	1.329	99.77	0.23
30	1.3387	99.761	0.239
40	1.358	99.756	0.244
55	1.387	99.74	0.26
70	1.417	99.733	0.267
85	1.45	99.734	0.2661
100	1.481	99.728	0.272
115	1.516	99.709	0.291
130	1.55	99.692	0.308

Field of 0.2T (RA=0, RB=1)			
Air T (°C)	Chip T(V)	angle	error abs
22	1.328	100.07	0.069
30	1.3387	100.04	0.042
40	1.357	100.05	0.046997
55	1.388	100.02	0.019997
70	1.4225	100.01	0.0090027
85	1.453	99.987	0.013
100	1.487	99.99	0.010002
115	1.517	99.98	0.019997
130	1.5526	99.964	0.036003

Field of 0.3T (RA=0, RB=0)			
Air T (°C)	Chip T(V)	angle	error abs
22	1.3185	100	0
25	1.326	99.997	0.0029984
40	1.353	99.98	0.019997
55	1.3827	99.983	0.016998
70	1.412	99.965	0.035004
85	1.44	99.953	0.046997
100	1.466	99.926	0.073997
120	1.502	99.92	0.080002
125	1.515	99.931	0.069

Tabla 2.1-11. Comparativa del efecto de la temperatura en la medida del APS.

Los resultados de estos experimentos se muestran en la tabla 2.1-11. Y a partir de estos datos, la Figura 2.1-28 representa la variación de la medida angular en función de la temperatura, para cada intensidad de campo aplicado. El sensor muestra líneas de tendencia prácticamente paralelas para los distintos campos aplicados. Existe también un desplazamiento debido al campo aplicado, pero que en el peor de los casos es de unos 0.3º, similar al máximo error cometido en una excursión de giro completo en condiciones ideales (ver Figura 2.1-21).

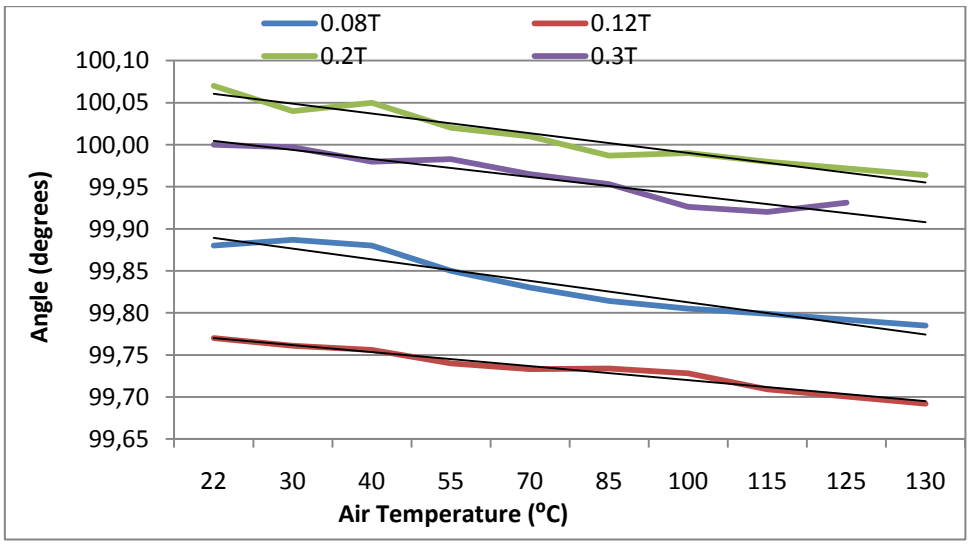


Figura 2.1-28. Variación del ángulo medido en función de la temperatura y del campo aplicado.

Lo mismo se puede deducir de la observación del error angular (Figura 2.1-29). Donde el cambio de tendencia que se observa en el error del experimento 3 (0.2T) se debe únicamente a que se está representando el error absoluto.

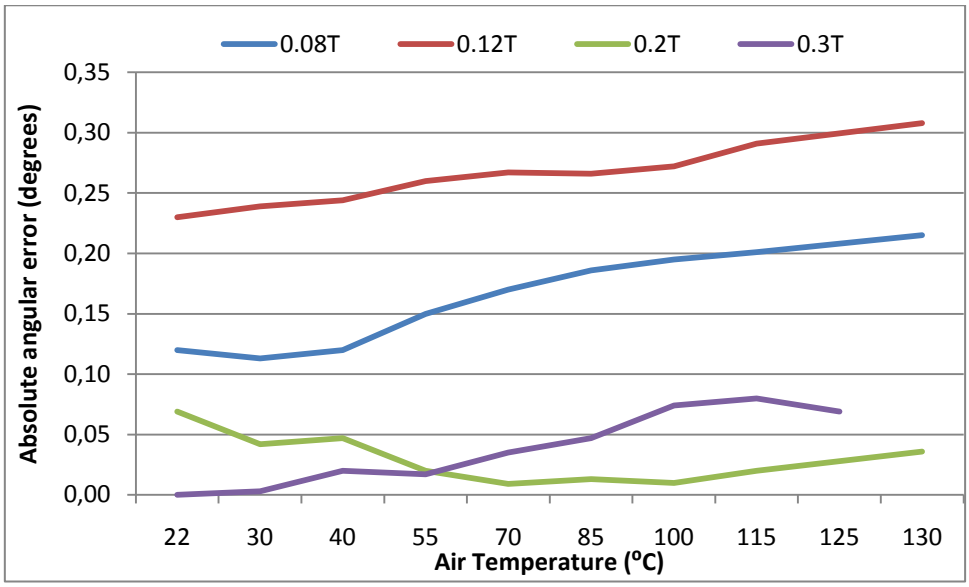


Figura 2.1-29. Error debido a cambios de temperatura en distintos rangos de campo.

Para comprobar definitivamente la relación de las variaciones observadas en la medida del ángulo con la variación de la temperatura, la Figura 2.1-30 muestra la respuesta del sensor de temperatura del chip respecto a los cambios en la temperatura del aire. Al ver que las respuestas son similares, se deduce la influencia de la temperatura es independiente del campo aplicado. Demostrada esta independencia se obtener una característica Temperatura de Aire (ºC) – Temperatura del Chip (V) más exacta, a través de la media de los datos de cada experimento (Figura 2.1-31).

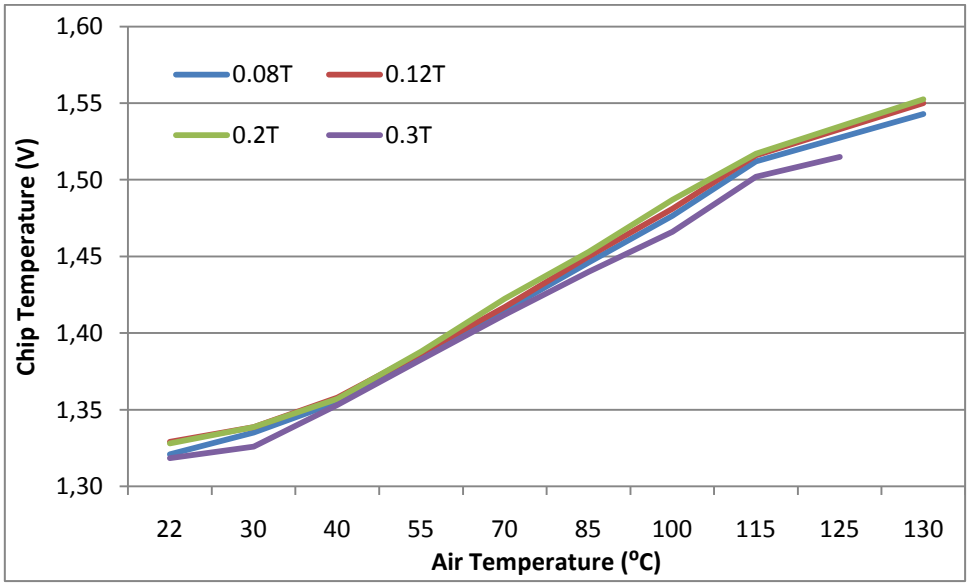


Figura 2.1-30. Sensibilidad del chip a temperatura ambiente para cada campo aplicado.

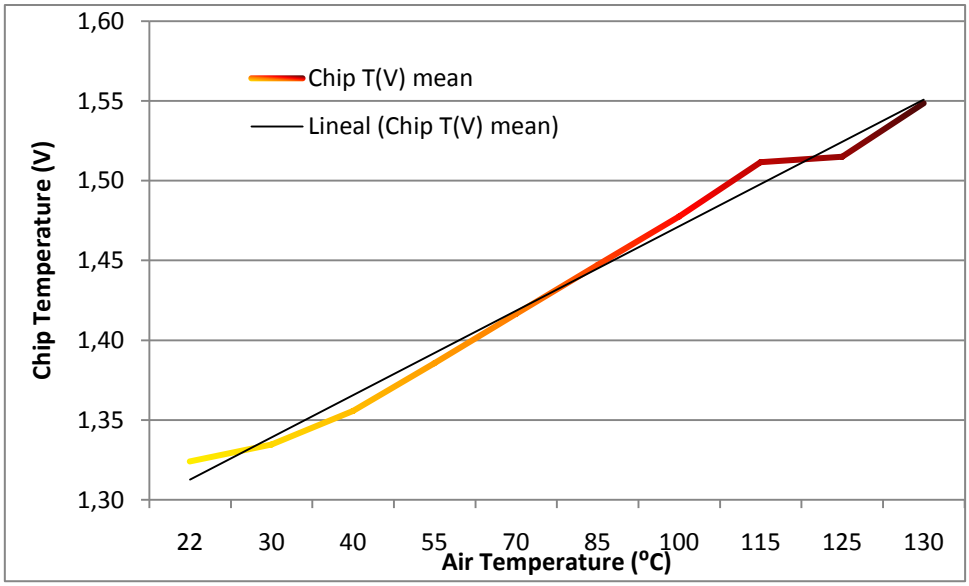


Figura 2.1-31. Sensibilidad media del chip a la temperatura ambiente.

Hasta el momento se ha caracterizado el comportamiento del chip con la temperatura pero solo con valores positivos de la misma. El dispositivo Temptronic de este laboratorio no permite obtener temperaturas de aire negativas, pero se dispone de otro equipo similar que si lo permite. Así que tras sustituir el Temptronic por otro que permite llegar hasta 60ºC bajo cero, se realiza una última medida. En este experimento se vuelve a fijar el sensor APS en un ángulo de 100º y ahora se fija la intensidad de campo a su valor ideal de 0.3 Teslas. Con estos parámetros se obtienen los resultados que se muestran en la tabla 2.1-12 y que se representan gráficamente en las ilustraciones 2.1-32, 2.1-33 y 2.1-34.

Air T (ºC)	Chip T(V)	Angle	Abs error
-60	1.111	100.04	0.036003
-45	1.1445	100.085	0.0855025
-30	1.176	100.08	0.0810015
-15	1.219	100.05	0.050503
0	1.25875	100.03	0.026001
22	1.3185	100	0
22.7	1.32	100.01	0.01
30	1.326	99.997	0.0029984
40	1.353	99.98	0.019997
55	1.3827	99.983	0.016998
70	1.412	99.965	0.035004
85	1.44	99.953	0.046997
100	1.466	99.926	0.073997
115	1.502	99.92	0.080002
125	1.515	99.931	0.069

Tabla 2.1-12. Efecto de la temperatura entre -60º y 125º, aplicando campo ideal, 0.3 Teslas.

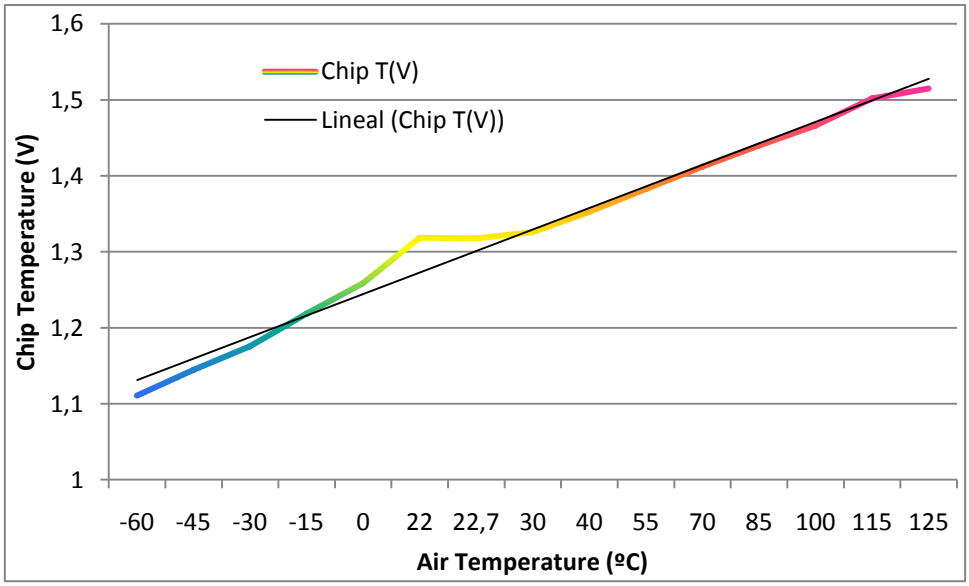


Figura 2.1-32. Sensibilidad del chip a la temperatura con campo ideal (0.3T).

La Figura 2.1-32 muestra la respuesta del sensor de temperatura del chip respecto a la temperatura del aire. Se puede ver que tiene una forma similar a la característica obtenida anteriormente (Figura 2.1.31), incluso en la extensión de temperaturas negativas.

Respecto a la medida angular que muestra la Figura 2.1-33, se puede decir que mantiene la tendencia observada en las medidas anteriores con temperaturas positivas. Además se puede observar una variación aproximada de 0.1º grados por cada 100ºC, lo que concuerda con las especificaciones del sensor, donde se establecía una precisión absoluta del sensor de ±0.05º en un rango de -50ºC a 80ºC.

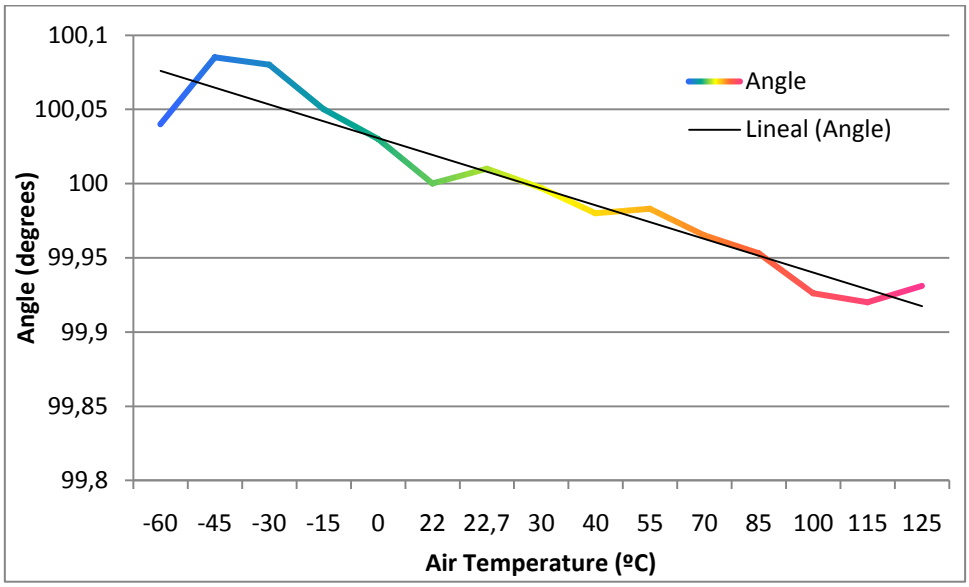


Figura 2.1-33. Representación del ángulo medido en función de la temperatura.

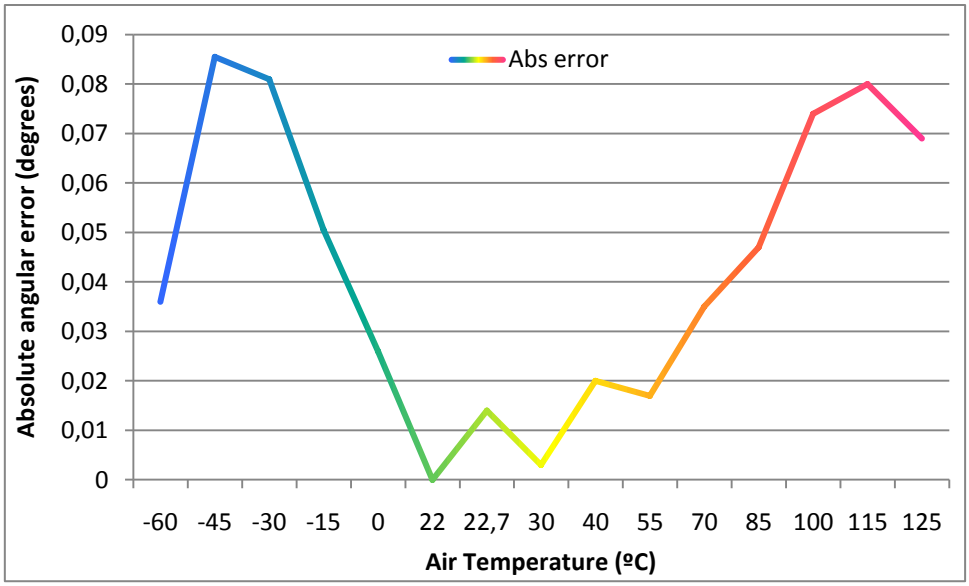


Figura 2.1-34. Error producido por la variación de temperatura.

Por último, se muestra una representación del error absoluto del ángulo medido. En este caso no es necesario utilizar la referencia del ángulo del rotor mecánico. Si no que se fija una posición a temperatura ambiente y se realiza una primera medida que servirá como referencia. Después basta con observar la variación de la medida respecto a esa primera medida. De esta forma se consigue la gráfica de la Figura 2.1-34, en la que se puede ver como el error absoluto crece al alejarse de la medida de referencia (la tomada a Tª ambiente 22ºC), pero sin llegar a superar los 0.1º.

2.2. Otros sensores parecidos en el mercado

En este apartado se muestran un par de sensores de posición angular que utilizan un principio de funcionamiento parecido. Estos sensores se pueden encontrar en la web de Direct Industry *www.directindustry.es*.

Figura 2.2-1. Sensor alternativo (1).

Figura 2.2-2 Sensor alternativo (2).

3. Objetivos del PFC

Durante el desarrollo del presente proyecto final de carrera se han valorado distintos objetivos. Es decir, el proyecto ha pasado por etapas en las que ha sido necesario el cambio de algunos de estos objetivos o, al menos, un reajuste en la ambición de los mismos. Estos cambios se justifican unas veces por la aparente imposibilidad de alcanzarlos de la forma prevista, otras porque se ha estimado que la consecución de ciertos objetivos podría complicar la de algunos otros o incluso porque el coste en cuestión de retardos o de tamaño del circuito resultante implicarían la imposibilidad de realizarlo. A continuación se comentan estos cambios que ayudarán a comprender el establecimiento de los objetivos finales.

Objetivos Iniciales (primera idea)

a. Estudiar la precisión del sensor de posición angular basado en una célula de efecto Hall vertical, desarrollado por *Professor Dr. Radivoje Popovic, Dr. Pavel Kejic y Serge C. Reymond* del Laboratorio de Microsistemas 3 (LMSI3) de la Universidad Politécnica Federal de Lausana (EPFL), en Suiza.
b. Diseñar un circuito digital capaz de utilizar las salidas del sensor anteriormente estudiado para, primero, aumentar su precisión y después calcular la velocidad angular de rotación del propio sensor, respecto a un campo magnético de dirección fija.
c. Implementar el diseño utilizando VHDL y realizar simulaciones que muestren su correcto funcionamiento.
d. Realizar los pasos necesarios para transformar el código VHDL en esquemáticos y posteriormente llevar el diseño hasta la fase de *layout*.
e. Fabricar el circuito en un chip de diseño específico ASIC y realizar los test pertinentes para comprobar su funcionamiento final y en caso exitoso, la nueva precisión de la medida de la posición angular.

Primeros cambios, prioridades y concreciones

El primer objetivo queda realizado en las primeras semanas. Este objetivo es especialmente útil para una buena familiarización con el equipo de medida y para conocer el funcionamiento interno y externo del sensor del que proceden las señales que se pretenden procesar con el diseño electrónico desarrollado en este PFC.

Se decide separar el proceso en dos partes: por un lado la parte que debe aumentar la precisión de la medida de la posición angular; por otro de la etapa de cálculo de la velocidad de rotación. Además se establece que la primera parte tiene prioridad sobre la segunda. Así se establece la funcionalidad de la primera parte, es decir, el bloque encargado de aumentar la precisión:

a. Procesado en tiempo real: deben generarse salidas a una tasa similar a la de llegada de medidas.
b. La salida tendrá forma de señal binaria de 16 bits, aunque aún no se especifica si debe ser paralelo, en serie, o en ambos formatos.
c. La entrada será una señal binaria de 12 bits (más tarde se cambio a 16 bits), de la que un número aún no especificado de los bits menos significativos serán inestables, pudiendo caracterizarse dicha inestabilidad como un ruido aleatorio blanco y gaussiano (AWGN).
d. La salida del sensor es una señal serie, por lo que si se quiere trabajar con bits en paralelo será necesario implementar un bloque de conversión serie/paralelo.
e. Cuando el sensor gira hay una relación inversa entre velocidad de giro y máxima precisión en la salida del nuevo diseño. Se pretende, por tanto, que el diseño sea capaz de detectar si el sensor está girando muy despacio, para poder ofrecer el máximo de precisión y cuando la velocidad sea muy elevada no se produzca aumento alguno de dicha precisión para distorsionar la medida. Se deberán establecer casos intermedios, es decir grados de precisión en función de la velocidad de giro del sensor. Además se especifica que este automatismo deberá buscar la situación de máxima precisión de forma gradual. Pero si se detecta algún incremento de velocidad angular, el sistema debe ser capaz de volver rápidamente a la situación estable que ofrezca mayor precisión.

Respecto al bloque encargado de calcular la velocidad de giro del sensor. Este debe tomar como entrada, la medida con precisión aumentada que ofrece el bloque anterior. Y debe procesarla para calcular la velocidad angular a la que gira el sensor. Esta velocidad debe ofrecerse en una señal de salida binaria de 16 bits. Además, este bloque debe disponer de una entrada que indique un rango de velocidad angular del propio sensor.

Con estos objetivos comenzó la fase de diseño del circuito, es decir, se empezaron a diseñar diagramas de bloque e incluso a implementarlos y simularlos mediante programación VHDL.

Objetivos finales (últimos cambios)

Durante el proceso de diseño pronto surge la certeza de que el bloque incrementador de precisión supone la parte más grande, complicada y, por tanto, más importante del presente PFC. Por lo que se decide suprimir el bloque que calcula la velocidad angular. Se estima que, aunque en esta etapa se dispone ya de un diseño del mismo, su implementación no queda justificada. Porque supone un aumento importante en las dimensiones finales del circuito, (debido a la necesidad de incluir multitud de registros), mientras que resulta mucho más simple hacer la misma operación con el software encargado de interpretar las salidas del circuito.

De esta forma, el diseño del circuito queda reducido al bloque incrementador de la precisión de la medida de la posición angular.

Por otra parte, es interesante comentar, que aunque inicialmente la finalidad del proceso incluye la fabricación del circuito como celda independiente, durante el desarrollo del proyecto, las condiciones económicas del laboratorio se vieron alteradas, por lo que se decidió incluir el circuito como una parte complementaria de un circuito mayor. De forma que este forme parte de la nueva versión del sensor que el grupo de investigación del laboratorio LMSI3 desarrolla. Así, las especificaciones finales del circuito son las que se detallan en el siguiente apartado.

a. Procesado en tiempo real: la tasa de salida de datos debe ser igual a la tasa de llegada de muestras. Si bien se admite un pequeño desfase o retraso debido al proceso.
b. La salida del sensor es una señal serie, pero el bloque de conversión serie/paralelo queda fuera de este bloque.
c. La entrada es un bus de 16 bits.
d. La salida tendrá forma de bus de 16 bits en paralelo.
e. Cuando el sensor gira hay una relación inversa entre velocidad de giro y máxima precisión en la salida del nuevo diseño. Se pretende, por tanto, que el diseño sea capaz de detectar si el sensor está girando muy despacio, para poder ofrecer el máximo de precisión y cuando la velocidad sea muy elevada no se produzca aumento alguno de dicha precisión para distorsionar la medida. Se deberán establecer casos intermedios, es decir, grados o niveles de precisión en función de la velocidad de giro del sensor. Además se especifica que este automatismo deberá buscar la situación de máxima precisión de forma gradual. Pero si se detecta algún incremento de velocidad angular, el sistema debe ser capaz de volver rápidamente a la situación estable que ofrezca mayor precisión.
f. En la versión actual del señor, se sabe que los 5 bits menos significativos son inestables, pudiendo caracterizarse dicha inestabilidad como un ruido aleatorio blanco y gaussiano (AWGN). Para hacer el diseño más reutilizable se debe diseñar el circuito de forma que pueda funcionar con otro sensor en el que el número de bits inestables pueda verse aumentado hasta 8 (siempre entre los menos significativos).
g. Se desea poder comprobar el funcionamiento del circuito de forma que las señales internas más importantes sean fácilmente observables, al menos durante la fase de test del mismo.

4. Memoria – Trabajo Realizado

Este es el capítulo más importante del presente documento. En él se exponen los trabajos realizados durante la realización de este Proyecto Final de Carrera. Para facilitar la comprensión de la tarea realizada, se explicarán los fundamentos teóricos y las ideas que han llevado a la construcción del diseño final del circuito. Y se plantearán cada una de las etapas de diseño que transformarán los bocetos iníciales del diseño en el *layout* final del circuito.

4.1. Fundamento teórico del aumento de la precisión

Del estudio previo del sensor se sabe que el sistema recibe un elevado número de muestras correspondientes a un mismo ángulo. También se sabe que las sucesivas medidas de un mismo ángulo tienen asociado un ruido con una función de distribución gaussiana de media cero y varianza acotada por el número de bits inestables. Entonces se tiene lo siguiente:

$$s \triangleq medida\ exacta\ del\ ángulo$$

$$n \triangleq ruido\ gaussiano\ \rightarrow N(0,\sigma) \qquad \Rightarrow x = s + n\ \rightarrow N(s,\sigma)$$

$$x \triangleq muestra\ con\ ruido$$

Donde ***s*** es la media muestral cuando todas las muestras utilizadas para calcularla corresponden a una medida de un mismo ángulo, o en otras palabras, si la señal de entrada es estacionaria (ver definición de señal estacionaria en el capítulo 3. Objetivos del PFC). Se busca, por tanto, una estimación de ***s*** tal que:

$$\hat{s} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (4)$$

Dada la importancia de esta operación, se decide crear un bloque independiente (una caja negra) con la única función de acumular muestras y calcular la estima de su media. Surge entonces la necesidad de calcular el número óptimo de muestras a acumular.

El número de muestras determinará el aumento de precisión que se consigue con cada bloque acumulador y consecuentemente el número de bloques o fases necesarias para alcanzar la máxima precisión (o máximo nivel de filtrado). Por eso se debe tener en cuenta el requisito que se establece en el punto e) de los objetivos[11]. Así, se decide establecer 5 niveles de precisión, cada uno de ellos con un bit estable más que el inmediatamente anterior. Por lo tanto, cada bloque deberá acumular el número de muestras necesarias para obtener un bit estable más. Por tanto, ahora el problema consiste en calcular el número de muestras necesarias para conseguir estabilizar un bit más.

[11] En el capítulo 3, apartado 3.1, bajo el subtitulo *Objetivos finales – Últimos cambios* se enumeran los requisitos del sistema y, entre ellos, el punto e) que habla de la necesidad de crear niveles de precisión.

Por otra parte, se puede establecer una equivalencia entre incrementar la precisión de una señal digital y aumentar su relación señal-a-ruido, cuya expresión es la siguiente:

$$SNR = \frac{P_S}{P_N} = \frac{{A_S}^2}{{A_N}^2} \qquad (5)$$

donde $\boldsymbol{P_S}$ y $\boldsymbol{A_S}$ son la potencia y la amplitud de la señal respectivamente, y $\boldsymbol{P_N}$ y $\boldsymbol{A_N}$ son la potencia y la amplitud del ruido respectivamente.

Al tratarse de señales digitales, si denotamos por $\boldsymbol{L}$ el número de bits de cada muestra, $\boldsymbol{L_e}$ el número de bits estables y $\boldsymbol{L-L_e}$ el número de bits ruidosos (no estables), podemos escribir:

$A_S = 2^{L_e}$ y $A_N = 2^{L-L_e}$ para la señal de entrada.

$A_S^{'} = 2^{L_e{}'}$ y $A_N^{'} = 2^{L-L_e{}'}$ para la señal de salida.

Como se ha dicho antes, este sistema va a realizar una estima de la media muestral, por lo que se sumarán N-muestras de amplitudes semejantes y ruidos también iguales. Por tanto la amplitud de la señal resultante será N-veces A_S. Y como la relación señal-a-ruido depende cuadráticamente de dicha amplitud, se puede asegurar que la acumulación de N-muestras produce un incremento de la relación señal-a-ruido igual a N^2:

$$SNR^{'} = N^2 SNR \qquad (6)$$

$$SNR^{'} = \frac{{P_S}'}{{P_N}'} = \frac{({A_S}')^2}{({A_N}')^2} = \left(\frac{2^{L_e{}'}}{2^{L-L_e{}'}}\right)^2 = N^2 SNR = N^2 \frac{{A_S}^2}{{A_N}^2} = N^2 \left(\frac{2^{L_e}}{2^{L-L_e}}\right)^2$$

$$\frac{2^{L_e{}'}}{2^{L-L_e{}'}} = N \frac{2^{L_e}}{2^{L-L_e}}$$

$$N = \frac{2^{L_e{}'} 2^{L-L_e}}{2^{L-L_e{}'} 2^{L_e}} = \frac{2^{L+L_e{}'-L_e}}{2^{L-L_e{}'+L_e}} = 2^{2(L_e{}'-L_e)} \qquad (7)$$

De forma que si pretendemos incrementar la precisión en un solo bit, necesitamos

$$L_e{}' - L_e = 1 \qquad \Rightarrow \qquad N = 2^2 = 4\ muestras.$$

Y para obtener una precisión máxima, en la que los 16 bits sean estables (después del último bloque acumulador), tendremos que acumular:

$$N = 2^{2(L_e{}'-L_e)} = 2^{2(16-11)} = 2^{10} = 1024\ \ muestras$$

Tras este estudio, se puede hacer una primera representación esquemática del circuito.

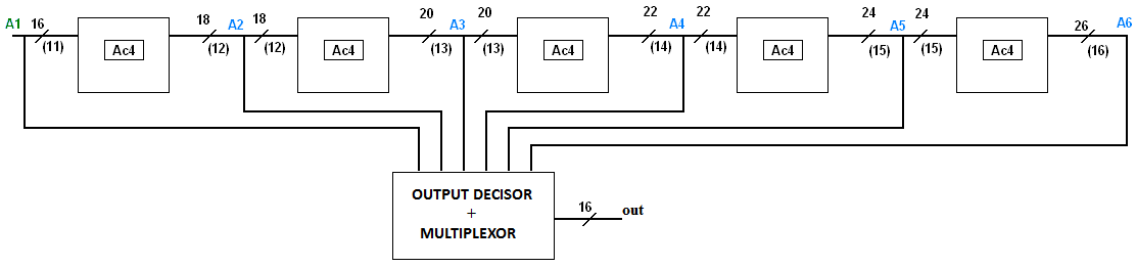


Figura 4.1-1. A1, A2, A3, A4, A5 y A6 son las salidas posibles para cada nivel de precisión. Entre paréntesis se encuentra el número de bits estables y por encima das líneas el número total de bits de casa bus de datos.

La Figura 4.1-1 muestra 5 bloques acumuladores (uno por cada bit inestable que se debe estabilizar) de N=4 muestras cada uno, conectados en serie. Así, la salida de cada bloque establece un nuevo nivel de filtrado[12]. El sexto bloque será el encargado de evaluar la estabilidad de cada nivel de filtrado y de seleccionar el apropiado para ofrecerlo como salida del sistema.

Queda así determinada la arquitectura básica del sistema. La siguiente cuestión a estudiar debe ser el método de acumulación de muestras, para que sea lo más efectivo posible y, a continuación, se definirá un algoritmo para seleccionar la salida deseada. Pero estas cuestiones se resolverán en el siguiente apartado.

[12] Durante todo el capítulo se hablará indistintamente de niveles de filtrado o niveles de precisión.

4.2. Resultados esperados

El objetivo principal de este PFC es conseguir un sistema electrónico digital que ofrezca una señal digital de 16 bits con mayor precisión de la que el sistema toma a su entrada. Es decir, que el procesado debe conseguir que la señal resultante tenga más bits de estabilidad que los que tendría si no pasara por este sistema digital.

En el mejor de los casos (señal estacionaria y con el mínimo ruido) la señal llega con 11 bits estables y 5 inestables:

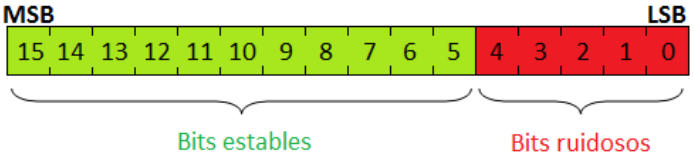


Figura 4.2-1. Señal de entrada al circuito.

Así se puede calcular la incertidumbre en grados de la señal que llega al sistema:

$$U_{before} = \pm\frac{360°}{2^{11bits}} = \pm 0.17578°$$

Para este caso se espera que el sistema consiga un incremento máximo de la precisión, es decir, estabilizar los 5 bits menos significativos para obtener una señal sin ruido.

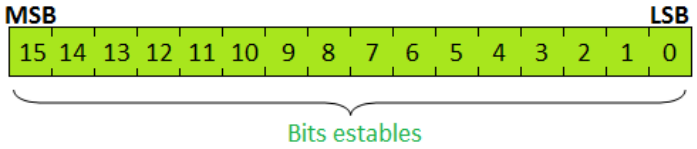


Figura 4.2-2. Señal de salida del circuito.

Esto equivale a una incertidumbre sobre el ángulo medido de:

$$U_{after} = \pm\frac{360°}{2^{16bits}} = \pm 0.005493°$$

Por tanto, para el caso de una entrada estable y con el mínimo ruido, la máxima mejora esperada en la incertidumbre sobre el ángulo medido de ±0.17578° a ±0.005493°. Es decir, una reducción del 96.875% en la incertidumbre. Este porcentaje también puede interpretarse como el máximo aumento de recisión esperado, ya que aumentar la precisión implica disminuir la incertidumbre y a la inversa.

4.3. Discusión sobre las posibilidades de implementación

Este apartado comienza con una discusión sobre los problemas que se plantean como consecuencia de la naturaleza angular de la magnitud medida, así como de su resolución para la consecución de los objetivos del proyecto. Después se entra en una fase de estudio de tres métodos distintos para acumular muestras. Además se justificará la arquitectura y el diseño final del bloque acumulador. Por último se explicará el algoritmo que selecciona las salidas deseadas y también se mostrará su diseño y arquitectura final.

4.3.1. Problema del Zero-Pass

La medida de posiciones angulares presenta una particularidad que dificulta el procesado de las señales digitales que se utilizan para codificarla. Esta particularidad reside en la naturaleza cíclica de las posiciones angulares. Mientras que una medida de posición lineal se puede determinar de forma unívoca respecto a una referencia absoluta, las medidas angulares se repiten cada vez que se completa un giro de 360º. Por lo tanto los valores angulares están acotados entre 0º y 360º y lo más importante, se establece una continuidad en 360º=0º que tras la codificación de estas señales se puede ver como una discontinuidad:

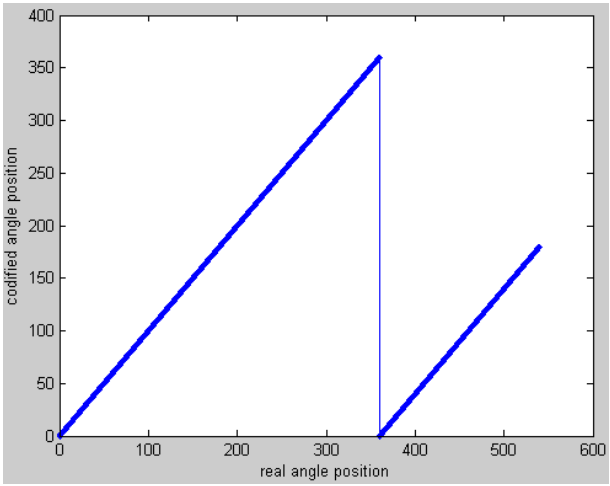


Figura 4.3-1.

Esta discontinuidad supone un grave problema en el procesado de la señal y, más concretamente, en el cálculo de la estima de la media muestral, necesario para aumentar la precisión. Imaginemos que se calcula la media de un grupo de muestras cercanas a la discontinuidad. Tan cercanas que unas pertenecen a ángulos anteriores a la discontinuidad (<360º) y las otras pertenecen a ángulos posteriores a dicha discontinuidad (≥0º). En este caso se obtendría un resultado para la media muy distinto a los valores extremos de la discontinuidad, lo que lo convierte en un valor erróneo. Por ejemplo, si tratásemos con el grupo de muestras *X={359.917º, 359.892º, 359.953º, 0.002º}*, la media obtenida sería:

$$\hat{s} = \frac{1}{4}\sum_{i=1}^{4}(359.917^\circ + 359.892^\circ + 359.953^\circ + 0.002^\circ) = \frac{1079.764^\circ}{4} = 269.941^\circ$$

Se puede ver que el resultado obtenido es erróneo porque difiere mucho de los límites de la discontinuidad (0º y 360º). Además este problema se agrava al repetir iterativamente el

proceso de calcular medias. De forma que cada nuevo nivel de precisión introduciría más muestras erróneas en la discontinuidad.

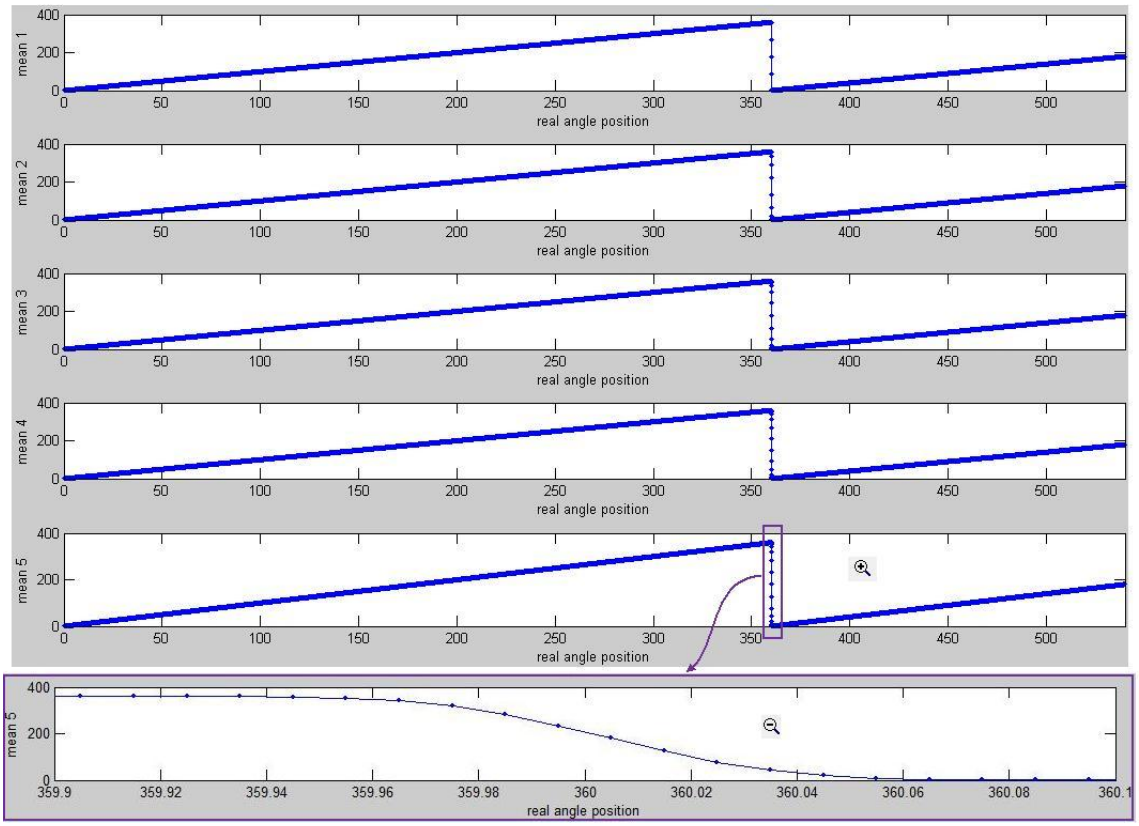


Figura 4.3-2. Error producido al filtrar la señal en torno a la discontinuidad.

En la Figura 4.3-2 se puede ver cómo afecta el problema de la discontinuidad a cada nivel de filtrado. Por otra parte, en la Figura 4.3-3 se muestra como el problema se incrementa al aumentar el nivel de precisión. El margen de ángulos en los que se produciría este problema viene determinado por el ruido asociado a la medida ofrecida por el sensor. Este error puede ser de hasta ±0.01º para una señal estacionaria (ver estudio teórico), por lo que el rango de ángulos en los que se produciría error sería 360±0.01º. Si la señal no es estacionaria este rango aumenta.

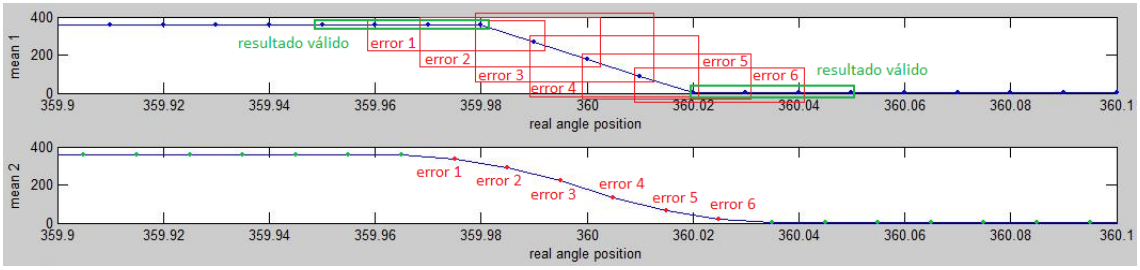


Figura 4.3-3. Generación de resultados erróneos.

Resolver este problema implica forzar una salida de 0º o 359.999º al realizar la acumulación. Si esto se hace en cada bloque acumulador, el problema queda resuelto y no se extiende de unos niveles a otros.

Con el fin de erradicar este problema se diseñará un bloque que implemente un algoritmo para forzar las salidas válidas. La idea consiste en detectar la incoherencia que se produce

cuando alguna de las muestras es muy cercana a 0º o a 360º, pero la media obtenida está muy alejada de cualquiera de estos valores. Es en esos casos en los que se debe actuar sobre la media para que deje de ser un resultado erróneo. Inicialmente se pensó en forzar directamente uno de los valores extremos, pero esto introduciría un nuevo error de precisión. Así que se ha desarrollado otra forma de corregir media sin necesidad de forzar directamente un valor de 0º o 359.999º.

La idea básica del método consiste en detectar cuando se produce la discontinuidad y, en tal caso, sumar 360º, es decir, un giro completo a las muestras que cercanas a 0º, dejando sin modificar las que poseen valores cercanos a 360º. Tras esta modificación se realiza la media con normalidad. De forma que ahora el resultado será muy cercano a 360º. En caso de que la media obtenida fuera mayor de 360º, esto generaría un problema, pero sólo en un sistema decimal, ya que al tratarse de señales binarias sin signo, se produce *overflow*[13] y el valor de la media pasa de ser ligeramente superior a 360º a ser ligeramente superior a 0º.

Por tanto para realizar este proceso con señales binarias se diseña un bloque independiente que se introducirá en cada bloque acumulador y que recibe el nombre de ***ZPcorrect4***. Las funciones de este bloque son:

- Detectar los grupos de muestras cercanas a la discontinuidad.
- Detectar cuántas de esas muestras lo hacen desde la derecha, es decir, desde la proximidad a 0º y sumar un giro de 360º por cada una de ellas.

El primer punto se puede realizar mediante una división del espacio muestral en las 3 zonas que se muestran en la Figura 4.3-4.

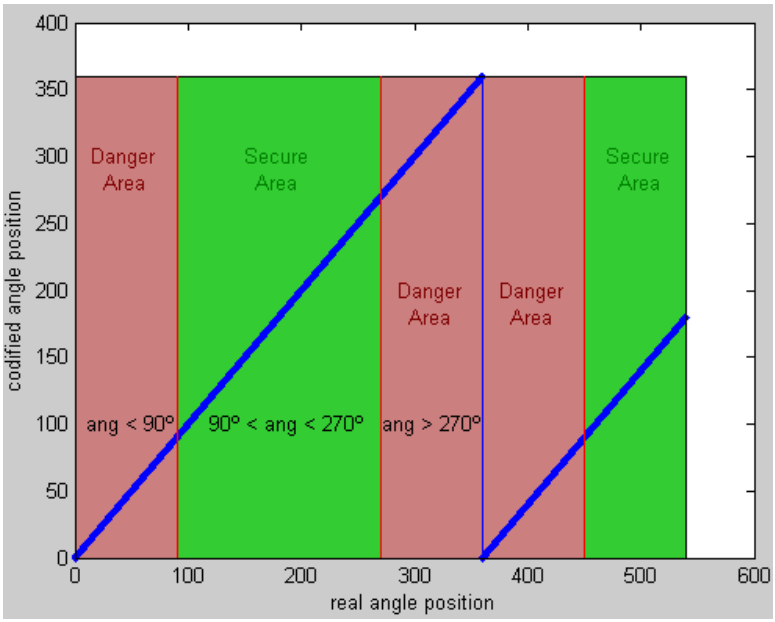


Figura 4.3-4: Representación de las zonas de susceptibles de sufrir el error *zero-pass*.

Estas zonas pueden parecer muy amplias, pero facilitan enormemente la detección de muestras cercanas a la discontinuidad. Las dos fronteras definidas coinciden con 90º y 270º.

[13] Ejemplo de *Overflow*: “0000.0000.0001.0100” + “1111.1111.1111.1111” = “0000.0000.0001.0011”.

Ángulos que equivalen a ¼ y ¾ de vuelta completa, respectivamente. Al transformar estos ángulos a señales binarias

$$360° = 2^{16} = 65536_{10} = 1.0000.0000.0000.0000_2$$

$$359.9945068° = 65535_{10} = 1111.1111.1111.1111_2$$

$$270° = 49152_{10} = 1100.0000.0000.0000_2$$

$$90° = 16384_{10} = 0100.0000.0000.0000_2$$

se aprecia cómo se simplifica la definición de estas zonas, ya que solo será necesario observar los dos bits más significativos (MSB_2) de la señal, para saber en qué cuadrante se encuentra el ángulo medido. Así

$$Zona\ 1 \equiv zona\ sensible - \ 1^{er}\ cuadrante\ \rightarrow 0° < \alpha < 90° \rightarrow MSB_2 = "00"$$

$$Zona\ 2\ \equiv zona\ segura \begin{cases} 2º\ cuadrante\ \rightarrow 90° < \alpha < 180° \rightarrow MSB_2 = "01" \\ 3^{er}\ cuadrante\ \rightarrow 180° < \alpha < 270° \rightarrow MSB_2 = "10" \end{cases}$$

$$Zona\ 3 \equiv zona\ sensible - 4º\ cuadrante \rightarrow 180° < \alpha < 360° \rightarrow MSB_2 = "11"$$

De esta forma el "detector de *zona sensible*" solo necesita comprobar si los dos bits más significativos son iguales a "00" o a "11" (también podría comprobar simplemente si son iguales entre sí). Pero además, el detector debe asegurarse de que esto se cumpla en las 4 muestras sobre las que se va a realizar la media muestral. Lo contrario indicaría que unas muestras proceden de la *zona sensible* y otras de la *zona segura*. Esto solo puede significar que el grupo de muestras pertenece a un ángulo muy próximo a los límites entre zonas (90º o 270º). Y como estos límites están suficientemente alejados[14] de la discontinuidad, no es necesario corregir la media muestral.

Para realizar el diseño del bloque que realiza la corrección *zero-pass* es necesario especificar antes el diseño del acumulador en el que se va a incluir. Por tanto, será el siguiente apartado sobre la discusión de los métodos de acumulación el que defina la arquitectura del bloque *ZPcorrect4* para cada caso.

[14] Por "suficientemente alejados", se entiende que el ruido de las muestras es mucho más pequeño que la distancia a la discontinuidad.

4.3.2. Métodos de acumulación de muestras

Acumulación simple 4in-1out

El primer acumulador diseñado es el más simple de los tres métodos que se analizan en este apartado. Consiste en una acumulación de 4 muestras sucesivas, hasta conseguir una salida como resultado de la suma de las muestras almacenadas.

Con este método la tasa de salida de datos de cada bloque acumulador es 4 veces menor que la que tiene a su entrada. Es decir, este bloque produce una salida de datos válida por cada 4 muestras recibidas, por lo que el tiempo entre salidas es igual al tiempo necesario para la acumulación de 4 entradas sucesivas (4 ciclos de reloj). Esta característica viola radicalmente el objetivo de trabajo en tiempo real[15], por lo que este diseño no será el definitivo, pero si servirá como acercamiento al problema.

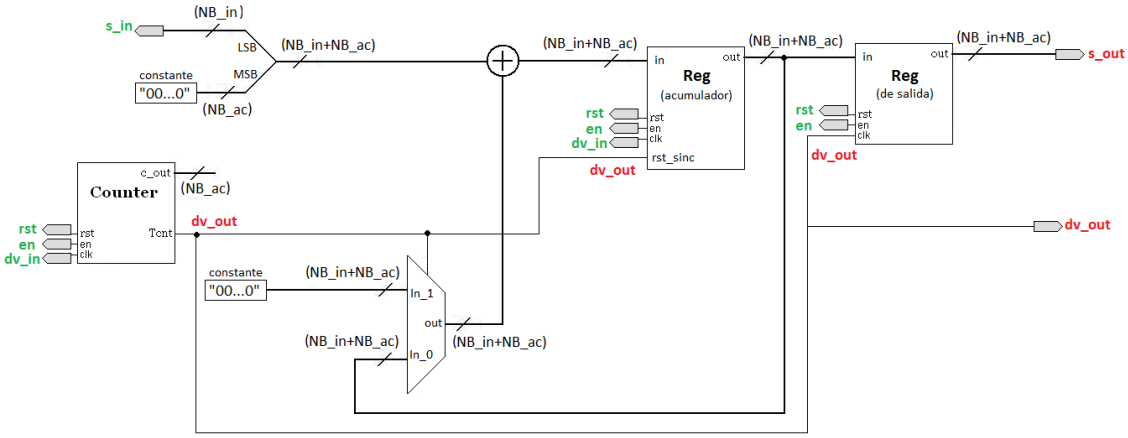


Figura 4.3-5: Diagrama de bloques del acumulador más simple. Espera la llegada de cuatro muestras, las suma y actualiza con la salida en el mismo instante en que llega la primera de las siguientes 4 muestras.

En la Figura 4.3-5 se muestra el diagrama de bloques. En él se puede ver que se utiliza un registro (*acumulador*) de $NB_{in}+NB_{ac}$ bits para acumular las 4 muestras, donde NB_{in} es el número de bits de la señal antes del bloque acumulador y $NB_{ac}=2bits$ es el número de bits que se acumulan, por lo que la salida tiene $NB_{in}+NB_{ac}=NB_{in}+2$ bits.

Además la salida del registro acumulador pasa por un ciclo *feedback* en el que se añade la última muestra que llega a la suma acumulada de las muestras anteriores. Esto se repite hasta obtener la suma de 4 muestras. Entonces, cada 4 flancos de subida del reloj[16] se detiene la acumulación (se deja de sumar) y se da por válida la salida.

Para contar los cuatro flancos de subida de la señal reloj se utiliza un bloque *Counter*. De este bloque contador se utilizará la salida de un solo bit llamada *terminal_counter (Tcnt)*. Esta señal

[15] En el capítulo 3, apartado 3.1, bajo el subtitulo *Objetivos finales – Últimos cambios* se enumeran los requisitos del sistema y, entre ellos, el punto a) habla de la tasa de salida de datos.
[16] El reloj del sistema es la señal *DV* (*data valid*) cuyo flanco de subida indica que la señal de entrada de datos es válida.

tiene un valor lógico '0' siempre, excepto cuando se alcanza el último estado[17], instante en que cambia a un valor lógico '1' para, después, volver a '0' con el siguiente flanco de subida del reloj. Esta señal (*Tcnt*) sirve para reiniciar la acumulación y también permite la salida de la última suma realizada, que se almacena en otro registro.

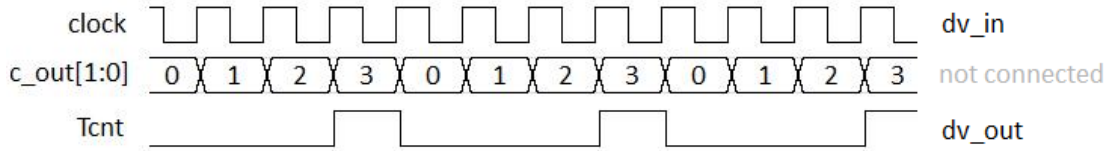


Figura 4.3-6. Principales señales del bloque *Counter.*

Hasta aquí se ha descrito la arquitectura y el funcionamiento del bloque ***Ac4***. Pero aún no se ha tenido en cuenta el problema *zero-pass*. En este diseño del acumulador no se almacenan las muestras sino la suma acumulada de las mismas según van llegando. Esto complica el diseño del bloque corrector, porque no se puede esperar a disponer de todas las muestras para realizar o no la corrección. Sino que se debe comprobar el rango angular de cada muestra según llega y de, alguna forma, almacenar esa información hasta acumular las cuatro muestras.

Una solución válida para el bloque corrector consiste en ir incrementando un contador al recibir muestras con el MSB='0' (muestras que podrían necesitar corrección) y a la vez comprobar si cada muestra pertenece o no a la zona sensible. Así, al terminar de acumular un grupo de 4 muestras, el contador indicaría cuantas veces debe sumarse un giro completo de 360º, pero la corrección solo se realizaría si se ha comprobado que las cuatro muestras pertenecen a la zona sensible.

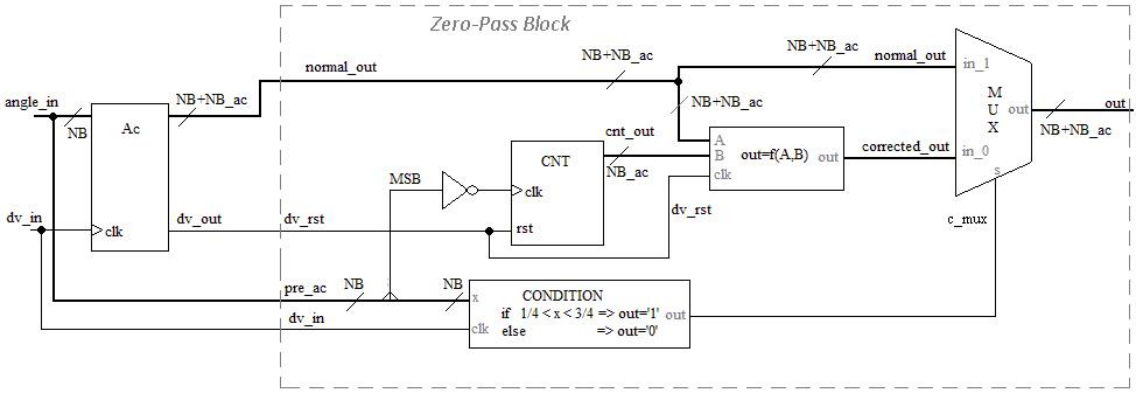


Figura 4.3-7. Diagrama de bloques del corrector *ZPcorrect4* para el primer tipo de acumulador.

La Figura 4.3-7 muestra el diagrama de bloques del circuito que corrige la acumulación. La nomenclatura que aparece en dicha Figura es muy generalista, por lo que concretaremos para *NB=2bits* y *NB_ac={16,18,20,22,24}* dependiendo del nivel de filtrado en el que se sitúe el acumulador. En esta configuración se sitúa el corrector a la salida del bloque acumulador, aunque también se podría introducir dentro del mismo, con lo que se obtendría directamente la salida corregida. En el diagrama se pueden observar dos bloques complejos: *CONDITION* y

[17] El último estado del contador corresponde al final de la cuenta de 0 a 3, es decir, el cuarto flanco de subida de la señal de reloj que indica que se ha recibido la cuarta muestra.

out=f(A,B). El primero de ellos realizaría la función de identificar la zona (sensible o segura) de cada medida, a través del análisis de sus dos bits más significativos, tal y como se explicó en el apartado anterior (4.3.1 Problema del Zero-Pass). La salida de CONDITION será '1' si la muestra está en zona segura y '0' si está en zona sensible. Esta salida se utilizará como señal de control del multiplexor, para seleccionar entre la acumulación normal o la corregida. El bloque *out=f(A,B)* es el que debe realizar la corrección propiamente dicha. Tendría como entradas la acumulación sin corregir y la salida del contador que indica el número de muestras a corregir. Su salida es la acumulación corregida, que será seleccionada por el multiplexor cuando la señal *c_mux* procedente del bloque *CONDITION* tenga un valor lógico '0'.

Si se reflexiona sobre el funcionamiento temporal del diagrama de bloques expuesto, es fácil deducir que, en realidad, la última de las cuatro muestras es la que determina si se aplica la corrección. Aunque este no es el comportamiento ideal, es válido. Esto es, si la última muestra está en la zona segura, la salida será la acumulación normal, lo que significa que se cumple el comportamiento ideal. Pero si esta cuarta muestra pertenece a la zona sensible, la salida del bloque será la corregida aunque haya muestras anteriores que pertenezcan a la zona segura. Este caso concreto solo puede ocurrir en dos puntos del espacio muestral. Estos puntos son las fronteras de las zonas sensible y segura, 90^o y 270^o (ver Figura 4.3-4). En cualquiera de estos casos, las salidas normal y corregida son iguales. En el primer caso, porque las cuatro muestras tendrían el MSB='0' y, por tanto, la suma de cuatro giros de 360^o produciría *overflow*, obteniéndose un resultado igual al que se tiene antes de corregir (ver ejemplo en la Figura 4.3-8). En el segundo caso (270^o) todas tendrían el MSB='1', con lo que no se sumaría en la corrección.

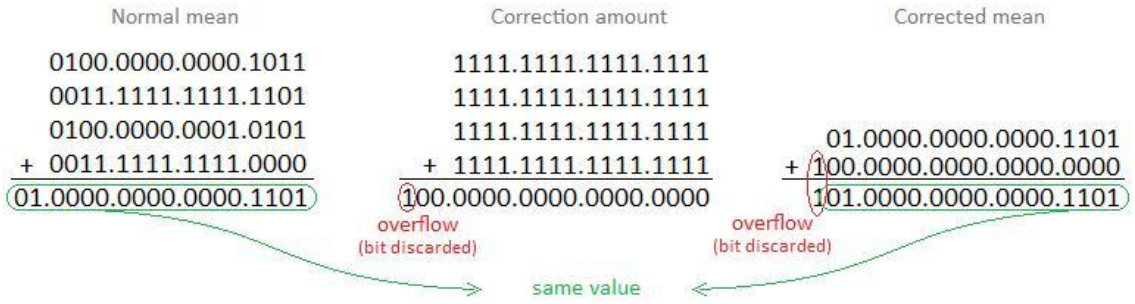


Figura 4.3-8. Ejemplo en el que la salida corregida y la acumulación normal son iguales. En torno a 90^o.

Acumulación por Sliding-Average

El segundo método de acumulación se basa en una conocida técnica llamada *Sliding-Average* [16]. Esta técnica define una ventana deslizante sobre la que calcula la media. El sistema resulta ideal para el objetivo que persigue el bloque acumulador. Por contra, veremos que obliga a incrementar considerablemente el número de registros, lo que puede llegar a comprometer la fabricación final del circuito por problemas de tamaño.

Gracias a la ventana deslizante se puede cumplir el objetivo del procesado en tiempo real enunciado en el capítulo 3. La longitud de la ventana es igual al número de muestras sobre las que se calcula la media, que en este caso son cuatro. Sobre un grupo de muestras ya establecidas, en cada ciclo de reloj se debe calcular un valor de media y después se desplazará la ventana una muestra.

En un bloque que realiza un procesado en tiempo real, los datos deben almacenarse el menor tiempo posible. Descartándose en cuanto hayan sido utilizados, para dejar paso a los siguientes. Por tanto, el número de datos a almacenar será igual al tamaño de la ventana menos uno. Esto significa que se deben almacenar al menos 3 muestras. La cuarta muestra será la recién llegada y junto a las otras tres completará la ventana y permitirá obtener un nuevo valor de la media. Todo esto se puede conseguir con un diagrama de bloques como el de la Figura 4.3-9.

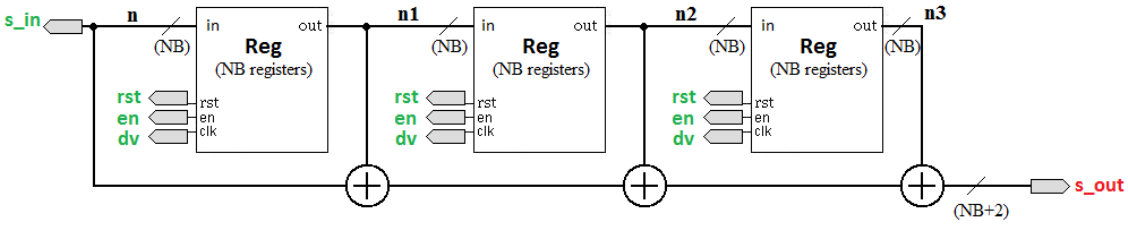


Figura 4.3-9. Diagrama de bloques del acumulador Ac4xSA.

Los tres registros almacenan las muestras anteriores a la actual, junto con la cual forman la ventana deslizante. Los sumadores son asíncronos y sus retardos son mucho menores que el periodo del reloj que hace desplazarse a la ventana. Por lo tanto, el resultado será válido a partir de un tiempo igual al retardo de los sumadores después del flanco de subida de la señal *dv*, que actúa de reloj. Pero a efectos prácticos, la media obtenida lleva un retraso de un ciclo de reloj (o de *dv*, es lo mismo) ya que el siguiente acumulador no dará por válida la señal hasta detectar el siguiente flanco de subida del reloj.

Todavía queda resolver el problema del *zero-pass*. Este diseño permite simplificar mucho la parte correctora, gracias a que dispone de las cuatro muestras para analizarlas en el mismo momento, evitando así algoritmos complicados.

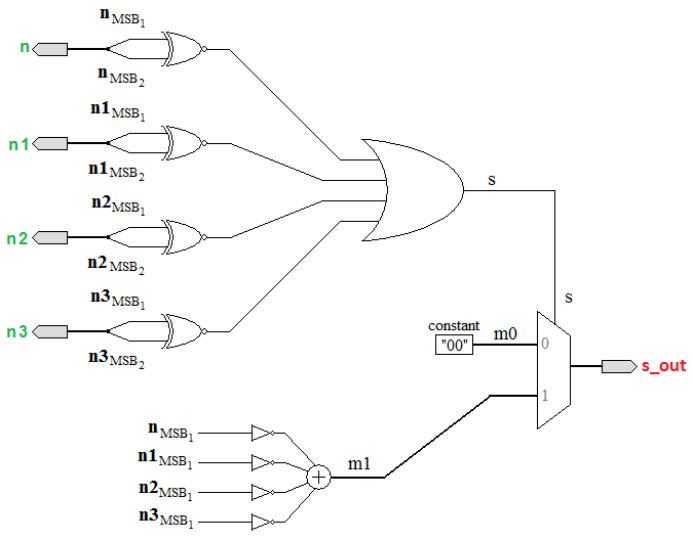


Figura 4.3-10. Representación del bloque ZPcorrectSA.

En la Figura 4.3-10 se muestra la arquitectura propuesta para el bloque corrector. La parte superior estría compuesta por una fase de cuatro puertas XNOR y otra fase OR. Sirve para analizar los dos bits más significativos de cada muestra y determinar si todas ellas pertenecen a la *zona sensible* del rango de valores angulares válidos. En tal caso la señal interna *S='1'* y el multiplexor seleccionará la señal *m1* para conectarla a la salida *s_out*. *m1* es una señal de dos bits que indica cuántas de las cuatro muestras representan un ángulo más cercano a 0º que a 360º (es decir <180º - ver apartado *4.3.1 Problema del Zero-Pass*). *m0* es una señal de dos bits que siempre tiene valor "00" y sirve para cuando no hay que corregir la media. La salida *s_out* es también una señal de dos bits, mientras que las señales de salida de cada acumulador tendrán 18, 20, 22, 24 y 26 bits respectivamente. Esto quiere decir que hay que extender la señal de corrección (la salida del bloque corrector) para que tenga la misma longitud que la señal de salida de cada acumulador. Esto se hará creando una señal de los bits correspondientes en cada acumulador, igualando sus dos bits más significativos a la salida del bloque corrector y dejando el resto con valor lógico '0'. Por tanto, lo que se ha hecho es un simple desplazamiento digital a izquierda, lo que implica multiplicar la salida del corrector por un giro completo (360º). Así la señal obtenida será el resultado de sumar tantos giros completos como muestras cercanas a 0º se detectaron.

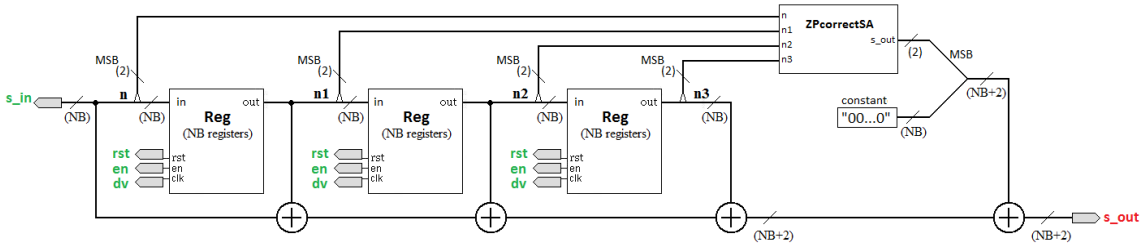


Figura 4.3-11. Diagrama de bloques del acumulador Ac4xSA con corrección *zero-pass*.

Se puede ver en la Figura 4.3-11, como quedaría el bloque corrector dentro del diseño del acumulador basado en ventana deslizante. Al igual que se incluye el bloque corrector dentro del bloque acumulador, se podría haber dejado la parte de extensión de la señal correctora en el interior del bloque corrector. Pero se entiende que dejando esa parte fuera se gana en generalización y **reutilización** de sistemas. Así el bloque corrector puede ser idéntico para cada acumulador sin importar el tamaño final de la salida del mismo. Será en cada acumulador en el

que se establezca un valor concreto para la variable generalista *NB={16,18,20,22,24}* que determina la longitud (en bits) de la señal de entrada.

Si bien los resultados de este método son ideales en cuanto a los objetivos de procesado de la señal, la necesidad de almacenar tres muestras en cada bloque acumulador, implica la necesidad de utilizar muchos registros. Ya que cada bloque ***Reg*** de los que aparecen en la Figura 4.3-11, así como en la Figura 4.3-9 o incluso en la Figura 4.3-5, no simboliza un solo registro, sino una batería de entre 16 y 26 registros (uno para cada bit) en paralelo, según el nivel de acumulación y el método utilizado. Por ejemplo, en el primer acumulador del método anterior (ver Figura 4.3-5) son necesarios *18bits·2Reg=36registros* mientras que el método de ventana deslizante utilizaría *16bits·3Reg=48registros*. Es decir un aumento de 12 registros en el primer acumulador, 14 en el segundo, 16 en el tercero, 18 en el cuarto y 20 en el quinto. En total este método necesita unos 80 registros más que el anterior (más adelante se valorará esta deficiencia teniendo en cuenta que en el método anterior se utilizaban contadores y comparadores que no son necesarios ahora).

Acumulación por semi-Sliding-Average

El método de *semi-Sliding-Average* es un refinamiento del método *Sliding-Average* para utilizar menos registros, manteniendo el objetivo de procesado en tiempo real (tasa de salida igual a la tasa de llegada).

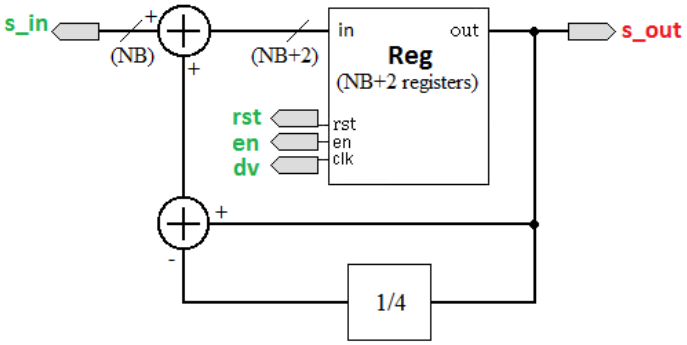


Figura 4.3-12. Diagrama de bloques del método de acumulación *semiSliding-Average*.

Como se ve en la Figura 4.3-12, este método solo necesita un bloque ***Reg*** por acumulador. Su funcionamiento se basa en una ventana de longitud creciente, puesto que en la suma intervienen todas las muestras anteriores a la actual. Pero la participación de cada muestra en dicha suma, es inversamente proporcional a la distancia temporal a la actual. Es decir, las muestras más nuevas tienen una mayor participación, pero al ir llegando más muestras su participación se hace menor, hasta resultar prácticamente insignificante.

$$\hat{s}_{semiSA}(n) = \frac{1}{4}\left(s(n-1) + \frac{3}{4}\hat{s}_{semiSA}(n-2)\right)$$

A partir de la ecuación anterior, se deduce que este método no realiza una buena estimación de la media sino que filtra la señal en función de sus valores anteriores. Aún así puede ser un método válido para incrementar la precisión de una señal con ruido.

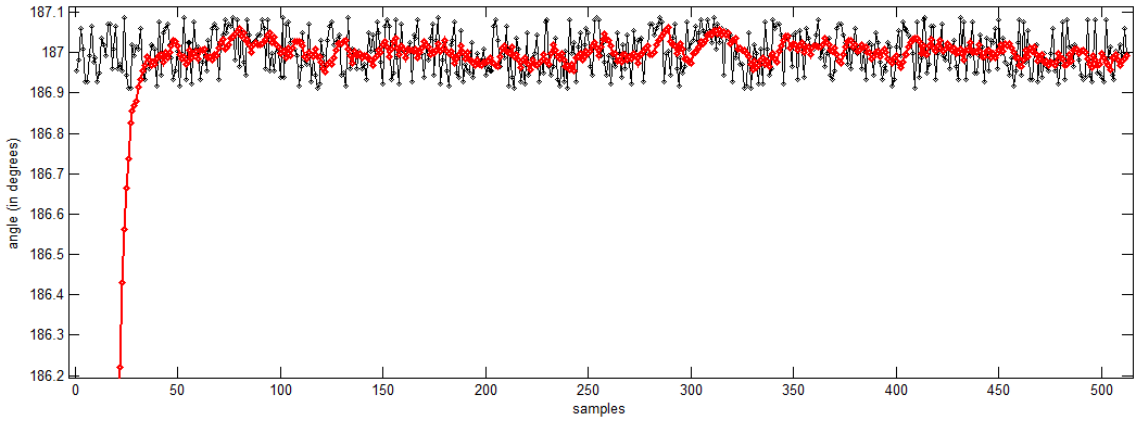


Figura 4.3-13. En negro muestras ruidosas en torno al ángulo 187º. En rojo la salida del primer acumulador con el método *semiSliding-Average*.

Pero reducir el número de registros tiene un precio. En la Figura 4.3-13 se puede ver como la salida del acumulador (en rojo) tiene un periodo transitorio bastante grande (aproximadamente 25-30 muestras en el ejemplo) hasta llegar al valor deseado. Esto supone una gran limitación en cuanto a la capacidad que el sistema tiene para adaptarse a cambios bruscos del ángulo medido.

Por otra parte, este sistema no va a ser válido en el contexto de las medidas angulares puesto que no se puede corregir el problema de *zero-pass* de forma sencilla. Si antes era necesario conocer la zona angular de la que procedían cada una de las cuatro muestras que se utilizaban para la media, ahora sería necesario hacer lo mismo para todas las muestras anteriores a la actual (y la actual también). Lo que supone un problema que se va complicando muestra a muestra.

Decisión del acumulador a implementar

Tras las explicaciones anteriores la decisión parece evidente. El acumulador por *semi-Sliding-Average* se puede descartar, tanto por el periodo transitorio que resulta demasiado extenso, como por la aparente imposibilidad de obtener un bloque corrector. Por otra parte, ya se vio que el primer método no cumplía con uno de los objetivos del proyecto, el de procesado en tiempo real. Sin embargo utilizaba menos registros que el método de *Sliding-Average*. Aunque si se analiza en detalle esta última diferencia se puede demostrar que no es tan grande como se mostraba anteriormente:

- El acumulador *4in-1out* necesita dos bloques *Reg* (con NB+2 registros y NB=16 el primer acumulador), un contador de 4 estados (necesita 2 registros), un multiplexor y 2·NB+2=(2+1)NB conexiones a la tensión de lógica ‘0’ para señales constantes (con). Es decir,

$$5acumuladores = 2 \cdot (18 + 20 + 22 + 24 + 26) + 4 \cdot 5 = 240\, registros$$

- El acumulador por *Sliding-Average* utiliza tres bloques *Reg* (con NB registros), un multiplexor (en el bloque corrector) y NB+2 conexiones a la tensión de lógica ‘0’ para señales constantes. Es decir,

$$5acumuladores = 3 \cdot (16 + 18 + 20 + 22 + 24) = 300\, registros$$

Por tanto la diferencia, en cuanto a registros, es favorable al acumulador más simple (el *4in-1out*) ya que necesita unos 60 registros menos. Pero finalmente se decide utilizar el acumulador por *Sliking-Average*, porque a pesar de implicar un aumento del 25% en el uso de registros, simplifica el diseño, cumple con todos los objetivos y utiliza menos conexiones de alimentación.

4.3.3. Técnica Adaptive-Averaging

La técnica de "*adaptive averaging*" proporciona un incremento de la relación señal-a-ruido para frecuencias bajas y estacionarias, a la vez que no limita la capacidad del canal para las altas frecuencias del APS basado en el sensor de efecto Hall. Un bloque de "*adaptive averaging*" consiste en un filtro digital (DF) de múltiples niveles y una salida selectora (OS). El DF y la OS pueden representarse por una serie de acumuladores y un multiplexor, respectivamente. El objetivo de la OS es modificar dinámicamente el ancho de banda del filtro digital mediante la conmutación de las salidas a un determinado nivel de filtrado. Esta conmutación entre niveles de filtrado la realiza una unidad de control (CU) de acuerdo con las propiedades dinámicas de la señal medida.

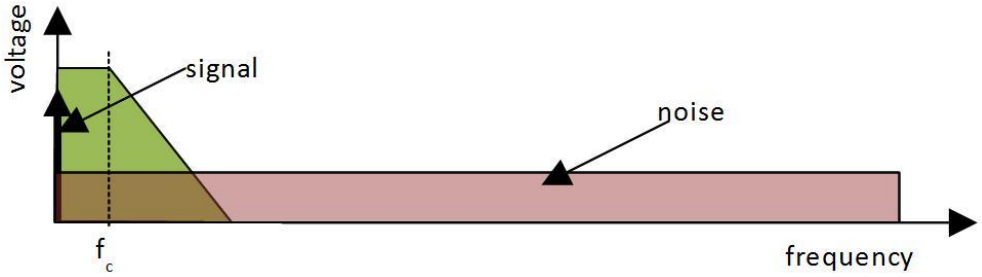


Figura 4.3-14. Señal estacionaria.

Si la señal es estacionaria, es decir, si la posición angular del imán no cambia durante un periodo de tiempo largo, la constante de tiempo del DF será la mayor fijada por la CU. De esta forma, el ancho de banda del filtro desciende hasta el mínimo y el ruido de alta frecuencia (ruido por encima de la frecuencia de corte f_c del filtro) queda eliminado por el filtro, por lo que la relación señal-a-ruido se ve incrementada.

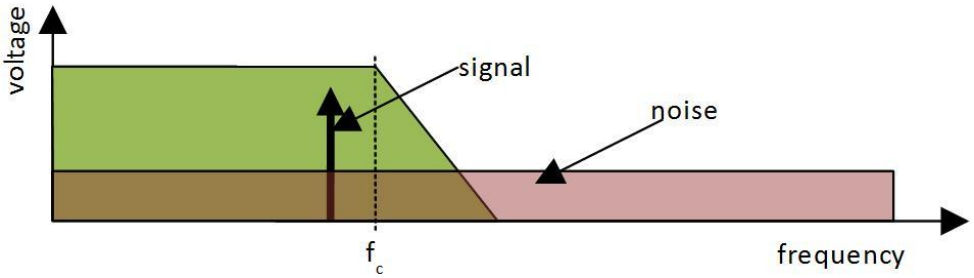


Figura 4.3-15. Señal dinámica (no estacionaria).

Al contrario que con la señal estacionaria, la señal medida se considerará dinámica, por ejemplo cuando el imán esté en movimiento. Este movimiento se detecta y la constante de tiempo del DF disminuye. El ancho de banda se aumenta y la señal medida no resulta alterada por el filtro.

Beneficios de la técnica "adaptive averaging"

Entre los beneficios de esta técnica tenemos:

- Un incremento de la relación señal-a-ruido para señales estacionarias y de baja frecuencia gracias al filtrado digital.
- No distorsión de la señal filtrada, es decir, no hay límites en frecuencia. Esto es así porque el ancho de banda del filtro queda adaptado a los comportamientos dinámicos de la señal.
- Adaptación automática del ancho de banda (no es necesaria la intervención del usuario).

4.3.4. Unidad de control – Máquina de estados

La máquina de estados tiene que implementar la parte principal del algoritmo de "*adaptive averaging*". Esta parte evalúa los resultados de todas las comparaciones que se realizan a la salida de los acumuladores. Y decide si es posible pasar a un nivel de filtrado superior para incrementar la precisión de la medida angular, si es necesario saltar a niveles inferiores o si el sistema debe mantener el nivel de filtrado actual.

La filosofía de la máquina de estados es muy simple: si la señal es suficientemente estable, avanza hasta el último estado (el que permite la máxima acumulación, es decir, la máxima precisión). Pero esto debe hacerse poco a poco, es decir, cada vez que se permita una nueva acumulación es necesario observar la señal resultante para saber si es estable. En caso positivo se continua avanzando hasta el máximo nivel de filtrado.

La parte más importante de este algoritmo es la que permite volver rápidamente a niveles de menor precisión cuando la señal deja de ser estable. La máquina de estados conoce cuándo se pierde la estabilidad gracias a la señales *S*. Estas señales provienen de comparadores situados a la salida de los acumuladores. Comparadores que analizan la estabilidad de la señal en cada uno de los niveles de precisión y comunican el resultado a la UC a través de una señal binaria (una de cada comparador: S1, S2…). La codificación de estas señales es simple: un valor lógico '1' indica inestabilidad en el correspondiente nivel de filtrado, mientras que un valor lógico '0' indica lo contrario.

La máquina de estados necesitará, al menos, un estado para cada nivel de filtrado. Si tenemos cinco acumuladores, hay 6 niveles de filtrado diferentes, por tanto, 6 estados. Pero para conseguir el funcionamiento deseado hay que utilizar más estados. Cuando la UC esté en funcionamiento, avanzar un estado implicará una nueva acumulación. El problema se produce cuando un cambio en el ángulo medido provoca una inestabilidad en la señal. Cuando se recupera la estabilidad de la señal, se debe esperar un tiempo prudencial antes de intentar de intentar recuperar el nivel de filtrado anterior a la inestabilidad. Tiempo necesario para que la inestabilidad se propague a los niveles superiores. Este tiempo de propagación sirve para asegurar que la señal de nivel superior, aunque pueda ser inestable, será una salida correcta[18].

[18] Se considera señal incorrecta aquella que es resultado de calcular una media con muestras de distintos ángulos. Esto ocurre cuando se produce un cambio de ángulo.

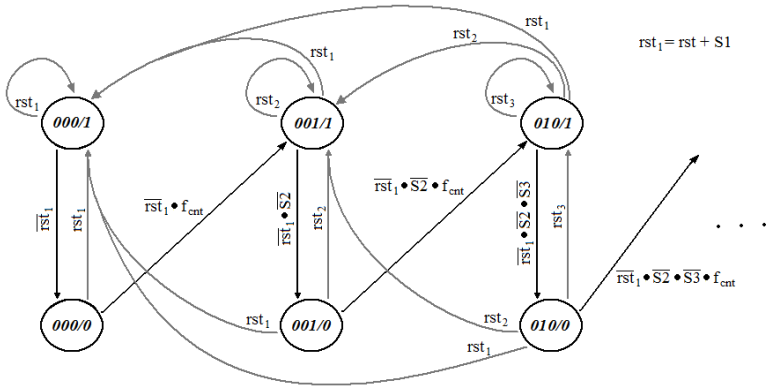


Figura 4.3-16. Diagrama de estados simplificado de la unidad de control.

En la Figura 4.3-16 se muestra una parte del diagrama de estados. Cada estado se representa con un óvalo que contiene cuatro cifras. Estas cifras representan las salidas del bloque UC. Concretamente las tres primeras cifras forman una señal digital de 3bits que se utilizará más adelante como señal selectora del multiplexor que conecta la salida del circuito a cada nivel de filtrado. De forma que si esas tres primeras cifras son *"000"* debe seleccionarse el nivel 0 (nivel sin incremento de precisión), si las cifras son *"001"* se seleccionará la salida del primer acumulador, si son *"010"* se seleccionará la salida del segundo acumulador, etc.

En el diagrama de la Figura 4.3-16 se puede ver el primer estado en la esquina superior izquierda. A este estado se llega si se produce un cambio brusco del ángulo medido o si hace un *reset* general del circuito. Se toma este estado como inicial para explicar el funcionamiento de la máquina de estados. Como se ha dicho anteriormente en este primer estado la salida del circuito es igual a la entrada, es decir, se conecta la salida a la entrada sin pasar por ningún acumulador. Las señales *S={S1,S2,S3,S4,S5}* son las que (junto con el *reset*) determinan los cambios entre estados. Por tanto, en el nivel 0, la señal *S1* determinará si se avanza al nivel 1. Esto es, si *S1='0'*, quiere decir que la señal a nivel 0 es estable y se puede pasar al nivel 1, pero si *S1='1'* se debe mantener el nivel 0. Con en el sistema en el nivel 1, será necesario que *S1* y *S2* tengan valor lógico *'0'*. En caso contrario si *S1='1'* se volverá al nivel 0, pero si *S1='0'* y *S2='1'* se mantiene el nivel 1. Así, conforme se avanza en el diagrama de estados se deben analizar más señales *S*. Se avanza a un nuevo estado si todas las analizadas tienen valor lógico '0', y si no se vuelve al nivel más bajo, cuya señal S tenga valor lógico '1'.

Los saltos a estados de nivel inferior se hacen tan rápido como se detecta una señal S de nivel inferior con valor '1', es decir, son inmediatos. Pero los saltos a estados de nivel superior deben esperar un tiempo prudencial igual a cuatro ciclos de reloj[19] y cuyo origen ya se ha explicado en este apartado. Por este motivo se establecen otros 5 estados, que son los que aparecen en la parte inferior del diagrama de la Figura 4.3-17. Se puede decir que estos estados sólo sirven para "esperar" y por eso mantienen iguales las tres primeras cifras, que son las que determinan la salida del circuito. La cuarta cifra (separada en el diagrama por una barra "/") se utiliza para permitir un inminente cambio a un estado de nivel superior. Si se trata de un '0' quiere decir que aún se está esperando el tiempo prudencial, pero si es un '1' indica que

[19] Cuatro ciclos de reloj es el tiempo durante el cual una misma muestra influye en la señal de salida de un acumulador. Puede decirse que es el tiempo que una muestra pasa dentro de un acumulador. Por tanto, si es la muestra que ha provocado la inestabilidad, cuatro ciclos de reloj es el tiempo durante el cual la salida del acumulador será errónea. Esta es la definición del "tiempo prudencial".

es seguro pasar al siguiente nivel. Evidentemente, durante el tiempo que el sistema se encuentra en uno de estos estados intermedios, la UC debe continuar analizando las señales S de la misma forma que lo hace cuando se encuentra en los otros estados. Porque, como ya hemos dicho, los cambios a niveles inferiores deben ser inmediatos independientemente del estado actual.

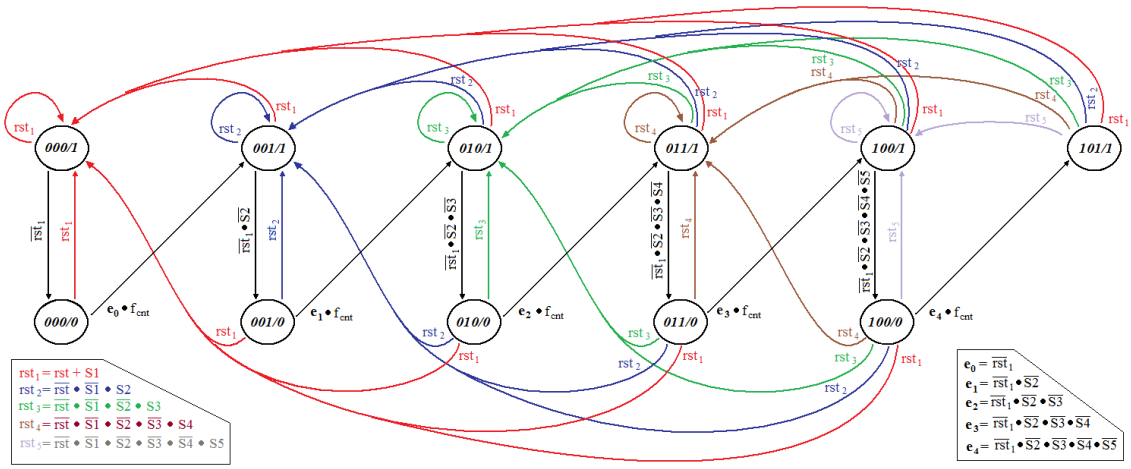


Figura 4.3-17. Diagrama de estados de la unidad de control. Las flechas de colores indican cambios a estados de nivel inferior (inmediatos). Las flechas negras indican saltos a los estados de nivel superior. La señal f_{cnt} procede de un contador de cuatro estados e indica que ha transcurrido el "tiempo prudencial".

En la Figura 4.3-17 se muestral un diagrama de estados completo. En cada flecha se indica la señal que produce el cambio. Se añade una leyenda para explicar el significado de algunas de esas señales. Como forma alternativa para explicar el funcionamiento de la UC se presenta el diagrama de flujo de la Figura 4.3-18.

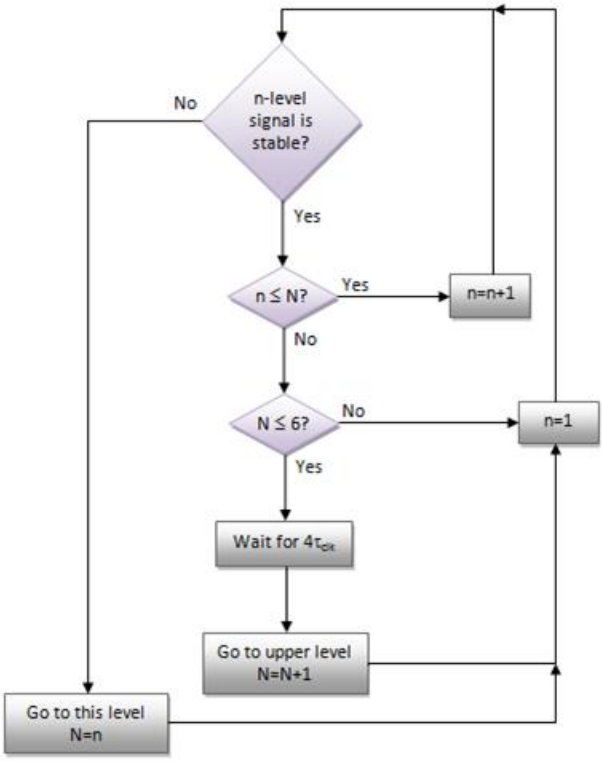


Figura 4.3-18. Diagrama de flujo de la unidad de control (UC).

4.3.5. Comparadores

Estos bloques deberán analizar las salidas de cada acumulador para comprobar su estabilidad y, así, facilitar a la unidad de control (UC) la información necesaria para decidir la salida del circuito (las señales *S*).

Lo primero que hay que plantearse es el **concepto de señal estable**. Este es un punto muy importante, porque es el único que va a influir en la decisión de aumentar o no la precisión de la señal. Ya se comentó este concepto en el apartado 4.3.3 de este mismo capítulo, cuando se hablo de señal estacionaria. Según el mismo criterio, se tomará como señal estable aquella cuyo valor tenga una variación máxima igual al ruido que se le supone.

En el apartado 4.2, se decía que si la señal es estacionaria, la entrada del sistema tendrá 11bits estables (los más significativos – MSB) y 5bits inestables (los menos significativos – LSB. Ver Figura 4.2.1). Por tanto, comprobar la estabilidad de la señal consiste en comprobar si los 11bits estables son realmente estables.

Para poder realizar dicha comprobación es necesario disponer de dos muestras que poder comparar. Por este motivo, cada comparador dispondrá de un bloque *Reg* para almacenar una muestra. La comparación, por tanto, debe ser síncrona. De forma que los comparadores producirán salidas (actualizaciones de las señales "*S*") a una tasa igual a la de entrada de muestras (se sigue cumpliendo el objetivo de procesado en tiempo real).

La operación puede parecer muy sencilla. En principio, debería ser suficiente con comparar los bits de estabilidad de la muestra actual y de la almacenada del ciclo de reloj anterior. Determinando como señal estable aquella en la que se compruebe que los bits estables son iguales a los de la muestra anterior, mientras que se entenderá por inestable el caso contrario. Pero si atendemos al origen de la inestabilidad surge un problema. Hay que tener en cuenta que la inestabilidad se debe a un ruido aditivo que afecta a la medida del ángulo realizada por el sensor. Esto significa que en una señal estacionaria o estable, la inestabilidad de los LSB puede afectar a los bits considerados estables. En la Figura 4.3-19 se muestran tres ejemplos de cómo afecta al ruido a la medida. En todos ellos se ha supuesto el peor caso posible en cuanto al ruido, que es que cambien todos los bits considerados inestables. En los casos (b) y (c) puede verse como el ruido aditivo produce un acarreo que termina alterando uno o más de los bits considerados estables.

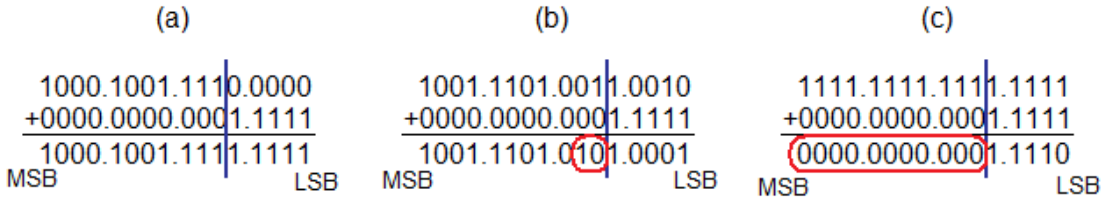


Figura 4.3-19. Efecto del ruido en los bits estables. (a) máximo ruido, solo cambian los bits inestables; (b) máximo ruido, dos bits teóricamente estables cambian; (c) máximo ruido, todos los bits estables cambian.

Se entiende, entonces, que lo pertinente es comprobar cuán diferente es la muestra actual de la almacenada. Si esa diferencia fuera menor al ruido máximo[20] la señal se consideraría

[20] Se entiende por ruido máximo el cambio de los 5 bits menos significativos, o lo que es lo mismo una diferencia de $\pm 11111_2$.

estable. En caso contrario se entendería que la señal no es suficientemente estable como para incrementar su precisión (*S='0'*). Para realizar la comprobación habría que realizar una resta entre las dos muestras y comprobar si el resultado es mayor que "11111".

Hasta aquí queda explicado el principio de operación básico de un bloque comparador. Pero se desea dar al bloque cierta flexibilidad en cuanto al ruido. Aunque el sensor actual ofrece 11bits estables, se pretende que el circuito sea compatible con otras versiones que pudieran ofrecer una señal con menor estabilidad. Por este motivo se incluye una entrada al circuito de 2bits, llamada *noise*. Esta entrada se utilizará para definir el rango de ruido con el que se debe trabajar. Es decir, esta señal se utilizaría con otras versiones que tuvieran un ruido mayor (hasta 8bits de inestabilidad), indicando cuantos bits más de inestabilidad tiene la salida del sensor. Esta señal es exterior al circuito incrementador de la precisión, por tanto, dependerá de la implementación final del chip sensor. Podría fijarse a un valor, previo estudio del ruido la versión del sensor que se introduzca en el chip, o podría dejarse como señal externa del chip para ser configurada en función de los resultados del chip. La codificación de la señal se representa en la tabla 4.3.1.

Noise[1:0]	Noise Range	Extra-bits of instability	*Input signal*		*Signal after 5 accumulators*	
			Stable bits	**Instable bits**	**Stable bits**	**Instable bits**
00	No extra noise	0	11	5	16	0
01	Low noise	1	10	6	15	1
10	Medium noise	2	9	7	14	2
11	High noise	3	8	8	13	3

Tabla 4.3-1. Rangos de ruido.

La señal *noise* se convierte, así, en una entrada más a cada bloque comparador. Obligando al mismo a implementar 4 comparaciones en paralelo, una para cada rango de ruido. En el apartado de edición de modelos VHDL se explicará como optimizar estas operaciones.

4.3.6. Descripción final de la operación realizada por el circuito

El principio de trabajo de la técnica "*Adaptive Averaging*" (AA) a nivel de bloque, quedará caracterizado con una señal de entrada de 16-bits con 11-bits de estabilidad (por ejemplo, para una señal estacionaria los primeros 11-MSBs no cambiaran, pero los últimos 5-LSBs pueden fluctuar).

El bloque AA consiste en cinco niveles de filtrado, una unidad de control (UC) y un selector de salidas (MUX). Cada nivel de filtrado está compuesto por un acumulador de 4-bits (Ac4), un registro (Reg) y una unidad de comparación (Compt):

- Los datos digitales quedan validados en la entrada mediante la señal "*data valid*" (dv).
- Los acumuladores realizan una operación conocida como "*sliding sum*", es decir, una suma con desplazamiento de cuatro datos sucesivos que va recibiendo a la entrada. La longitud de los datos de salida es N+2 para una entrada de longitud N (una entrada de 16-bits genera una salida de 18-bits, etc.). Por tanto, el incremento de estabilidad a la salida es de un bit (una entrada con 11-bits estables genera una salida con 12 bits estables, etc.)

- El comparador detecta las diferencias entre los bits estables de dos valores sucesivos. Y para permitirlo es necesario almacenar el valor previo en un registro.
- Si no se detectan diferencias durante cuatro valores sucesivos, la señal de entrada se considera como estacionaria para la resolución correspondiente al nivel actual y la unidad de control conmuta la salida al siguiente nivel.
- El mismo procedimiento se realiza en los sucesivos niveles de filtrado hasta llegar al quinto y último nivel.
- Si se detecta un movimiento, la salida se conmuta al nivel de filtrado inferior. La prioridad siempre la tiene el nivel de filtrado más bajo donde se haya detectado la diferencia. La selección de la salida se evalúa en la UC.
- Los bits de estabilidad, pueden ser modificados por una entrada de 2-bits, llamada "*noise*" y que llega hasta el comparador.

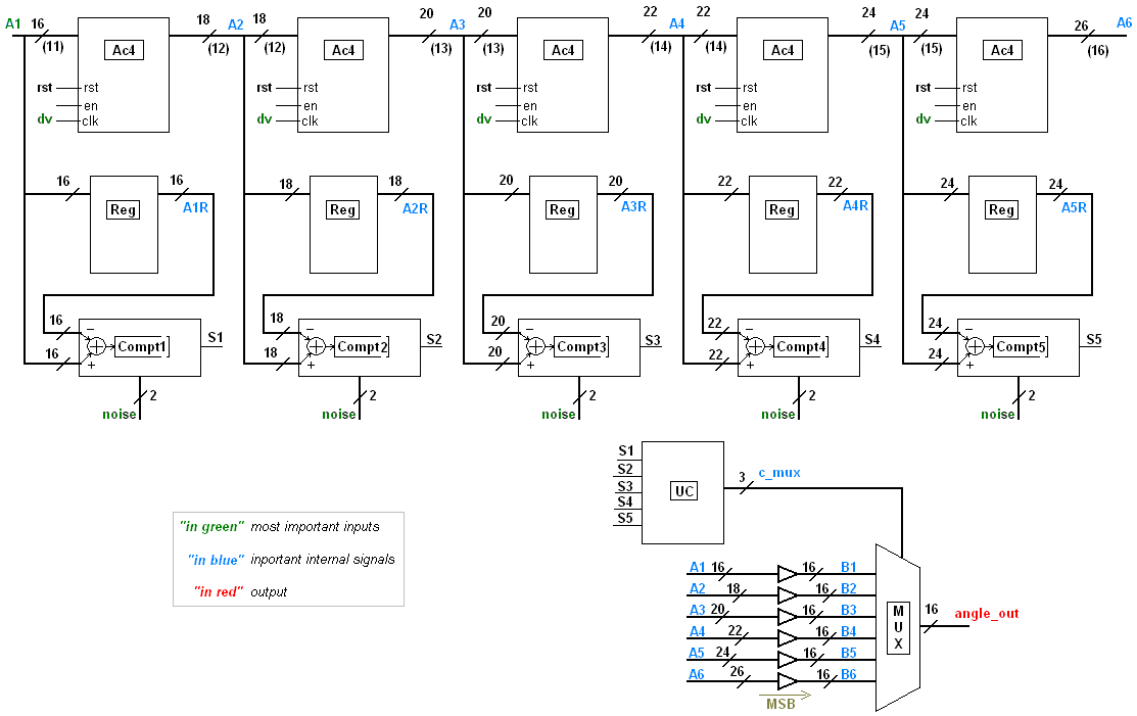


Figura 4.3-20. Diagrama de bloques del circuito, previo a la implementación.

4.3.7. Señales exteriores del diseño

Una vez decidida la funcionalidad y la arquitectura interna del circuito se pueden definir las señales externas del mismo. Es decir, las entradas y salidas que tendrá el circuito cuando se integré como una parte más de la circuitería de procesado del nuevo chip sensor. A continuación se muestra una descripción detallada de cómo deben ser esas entradas y salidas.

Entradas

- ***Clk***: señal de reloj del sistema (un sólo bit). Esta señal indica producida por el sensor (ver estudio teórico, capítulo 2), en su flanco de subida, cuando la entrada de datos es válida, por eso, fuera de este bloque recibe el nombre de *data valid signal* y tiene se etiqueta con ***DV***.

- ***Rst***: *reset* general del sistema (un sólo bit). Cuando esta señal binaria este a '1', el circuito deberá reiniciarse, dejando todos los registros a cero. Este *reset* debe ser asíncrono, por lo que los distintos subsistemas deben ser sensibles a esta señal sin necesidad de esperar a la llegada de un nuevo flanco de reloj.

- ***En***: señal de habilitación del bloque (un sólo bit). Debe estar siempre en alta ('1') para permitir el funcionamiento del circuito. Si no fuera así las señales del circuito no se actualizarán aunque llegue un nuevo flanco de reloj, a no ser que la señal de *reset* ***rst='1'*** ya que la prioridad del *reset* es mayor a la de cualquier otra señal del sistema.

- ***S_in[15:0]***: señal de entrada de datos (16 bits en paralelo). Representa el ángulo medido por el sensor.

- ***Noise[1:0]***: señal de ajuste de ruido (2 bits en paralelo). Sirve para indicar si la entrada de datos viene con más o menos ruido, es decir, con más o menos bits de inestabilidad. Esta señal se añade como una opción, para implementaciones del sensor con distintas precisiones, dando lugar a 4 rangos de ruido.

- ***Mode_test***: señal de test (un sólo bit). Sólo se utiliza para testear el circuito, de forma que en una supuesta versión comercial podría eliminarse. Se utiliza para habilitar la entrada ***out_select[2:0]*** cuando ***mode_test='1'***.

- ***Out_select[2:0]***: señal de selección de salida (3 bits en paralelo). Servirá para seleccionar el nivel de precisión que se desea observar a la salida del sistema, pero sólo en la fase de testeo del circuito, como se ha comentado en el apartado anterior. Estas dos últimas señales se incluyen también como opciones para hacer más robusto nuestro diseño.

Salidas

- ***S_out[15:0]***: señal de salida de datos (16 bits en paralelo). Representa el ángulo medido con precisión aumentada.

4.4. Etapas de diseño. Top-Down digital design flow

Este apartado detalla los pasos típicos del flujo de diseño *top-down* en VHDL / Verilog y que han sido los seguidos para obtener nuestro circuito digital de filtrado adaptativo.

A lo largo del apartado se van a considerar las siguientes herramientas:

- Modelsim v6.1b o superior, de Mentor Graphics.
- Design Compile rand Design Vision 2005.09 o superior, de Synopsys.
- Encounter 4.1 e IC 5.0.33 o superior, de Cadence Design Systems.

El kit de diseño utilizado es el Hit-Kit 3.70 AMS (Señales Analógicas y Mixtas) proporcionado por el fabricante Austriamicrosystems. El proceso utilizado es el C35B4 *(0.35 micron 4-metal CMOS)*, para el circuito digital que procesa la señal del sensor. Para el propio sensor se utiliza un proceso similar pero con la opción de alto voltaje (*high voltage*) y recibe el nombre de H35B4.

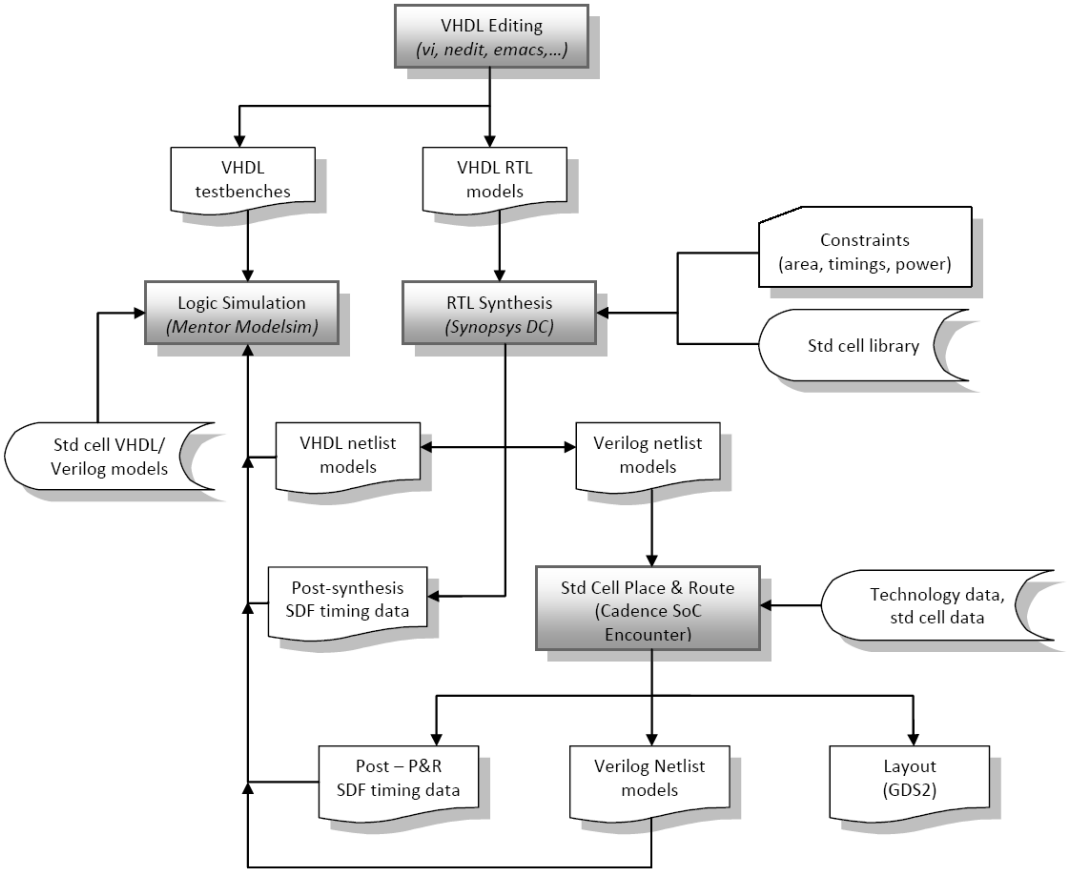


Figura 4.4-1. Flujo de diseño Top-Down.

La Figura 4.4-1 es una representación del flujo *top-down* que incluye los siguientes pasos:

- **Creación de modelos VHDL RTL**
  El objetivo de esta parte el desarrollar modelos VHDL sintetizables a nivel RTL[21]. Los modelos se definen como una separación clara entre las partes de control (por ejemplo, maquinas de estado) de las partes operativas (unidades aritméticas y lógicas).

[21] RTL – Register Transfer Level, significa nivel de transferencia de registros.

Los registros se utilizan para retener datos de pequeño tamaño entre ciclos de reloj. Las memorias RAM/ROM se usan para almacenar grandes cantidades de datos o código de programas. Bloques como los FSMs, ALUs o registros son habitualmente descritos como modelos de comportamiento que no implican una implementación particular. Las herramientas utilizadas en este paso van desde simples editores de texto a dedicados entornos gráficos que generan código VHDL automáticamente.

- **Simulación RTL**
  Los modelos VHDL RTL se validan mediante simulaciones generadas por archivos *testbench* que también se escriben en VHDL.

- **Síntesis RTL**
  El proceso de síntesis realiza una inferencia a una posible realización a nivel de puerta de una descripción RTL para que cumpla las restricciones de área, retardos y consumo de potencia establecidas por el usuario. Las restricciones de diseño se definen fuera de los modelos VHDL mediante comandos específicos del *soft-ware* utilizado. Las puertas lógicas especificas de cada tecnología pertenecen a bibliotecas proporcionadas por el fabricante o por compañías de propiedad intelectual (*IP companies*) como parte del llamado kit de diseño. Las librerías de puertas comunes incluyen unos pocos cientos de puertas lógicas y combinacionales. Cada función lógica se implementa con diferentes puertas para cumplir las distintas necesidades de *fanout* o longitud de las señales. La librería de puertas está descrita en un formato específico para cada herramienta de diseño y define, para cada puerta, su área, sus características de potencia y retardos y sus limitaciones ambientales.

  El paso de síntesis genera diversas salidas: un netlist VHDL a nivel de puerta, un netlist Verilog a nivel de puerta y una descripción SDF. El primer netlist se utiliza habitualmente para las simulaciones post-synthesis, mientras que el segundo es más adecuado como entrada para el paso *place&route*. La descripción SDF incluye información de los retardos para simulación. Nótese que los retrasos considerados en este paso son correctos para las puertas, pero sólo son estimaciones para las interconexiones.

- **Simulaciones post-síntesis a nivel de puerta**
  Los *testbenches* (archivos de simulación *testbench*) utilizados para la validación de modelos RTL pueden ser reutilizados (posiblemente con algunas modificaciones para utilizar los netlist VHL a nivel de puerta). La simulación a nivel de puerta se hace uso de los modelos VHDL para las puertas lógicas que proporciona el kit de diseño.

- **Place&Route - Emplazamiento y conexión de las celdas estándar**
  El paso *place&route* (P&R) realiza otra inferencia, ahora, a una realización geométrica del netlist a nivel de puerta, llamada *layout*. El estilo de diseño de celdas estándar coloca las celdas lógicas en filas de altura similar. Como consecuencia, todas las puertas lógicas de la librería tienen la misma altura, pero pueden tener distintas anchuras. Cada celda tiene una salida de alimentación en su parte superior y otra para conexión a masa en la parte inferior.

Actualmente, las conexiones entre puertas se realizan a través de las celdas, desde que el procesado de corrientes permite ocupar varias capas de metal (en nuestro caso 4 capas de metal para el proceso AMS C35). Como consecuencia, las filas pueden verse limitadas o modificadas para permitir que las conexiones de alimentación y masa sean compartidas por las sucesivas filas.

El paso R&P genera diversas salidas: una descripción geométrica (layout) en formato GDS2, una descripción SDF y un netlist Verilog a nivel de puerta. La descripción SDF incluye ahora retrasos de interconexión. El netlist puede ser diferente del leído como entrada, ya que el paso P&R puede hacer mejores optimizaciones de tiempos durante el emplazamiento, la generación del *clock tree* (árbol de reloj) y el enrutamiento (por ejemplo, mediante la inserción de un *buffer*).

- **Simulaciones a nivel de puerta post-layout**
  El netlist Verilog puede ser simulado mediante el uso de los *testbenches* VHDL y los datos SDF más precisos extraídos del layout.

- **Integración a nivel de sistema**
  La descripción del layout se integra como bloque en un sistema superior previamente diseñado.

4.4.1. Organización del diseño

Dado el número de herramientas EDA y archivos utilizados, es muy recomendable (y así se hizo) organizar el trabajo de la forma apropiada. Con este fin, se puede utilizar el comando *create_eda_project*[22] para crear una estructura de directorios para guardar los archivos del diseño. El comando se utiliza como se muestra a continuación:

```
create_eda_project <project_name>
```

donde <*project_name*> es el nombre del directorio principal que almacenará todos los archivos del proyecto.

En el desarrollo de este diseño se realizaron multitud de "proyectos" en los que se iba avanzando hacia el funcionamiento final del mismo. En la creación del proyecto definitivo se introdujo el comando:

```
create_eda_project Proyecto_v15/Precition_Increaser
```

De esta forma, el directorio */Proyecto_v15/Precition_Increaser* contiene los archivos de configuración para la simulación lógica (Modelsim), síntesis lógica (Synopsys DC) y el emplazamiento y enrutamiento de celdas estándar (Cadence SoC Encounter). Como consecuencia, es necesario que todas estas aplicaciones se lancen desde este punto. Con la excepción de las herramientas *full-custom layout* (Cadence IC) que deben ser ejecutadas desde el subdirectorio LAY, que contiene diferentes archivos de configuración.

[22] Se trabajaba en entorno Linux.

Estructura de los directorios del proyecto

```
Precition_Increaser/      # directorio principal del proyecto
  .synopsys_dc.setup      # archivo de instalación de Synopsys
  Modelsim.ini            # archivo de instalación de Modelsim
  DOC/                    # documentación (pdf, text, etc.)
  HDL/                    # archivos VHDL / Verilog
    GATE/                 # netlists a nivel de puerta
    RTL/                  # descripciones RTL
    TBENCH/               # archivos testbench
  IP/                     # bloques externos (memorias, etc)
  LAY/                    # archivos de layout full-custom
  LIB/                    # librerías de diseño
    MSIM/                 # librerías Modelsim (VHDL, Verilog)
    SNPS/                 # librerías Synopsys (VHDL, Verilog)
  PAR/                    # archivos "place & route"
    BIN/                  # comandos, scripts
    CONF/                 # archivos de configuración
    CTS/                  # archivos de síntesis del "clock tree"
    DB/                   # bases de datos
    DEX/                  # archivos de intercambio de diseños
    LOG/                  # archivos "log"
    RPT/                  # archivos "report"
    SDC/                  # archivos de restricciones
    TEC/                  # archivos de la tecnología
    TIM/                  # archivos de sincronización
  SIM/                    # archivos de simulación
    BIN/                  # comandos, scripts
    OUT/                  # archivos de salidas (formas de onda, etc)
  SYN/                    # archivos de síntesis
    BIN/                  # comandos, scripts
    DB/                   # bases de datos
    RPT/                  # archivos "report"
    SDC/                  # archivos de restricciones
    TIM/                  # archivos de sincronización
  TST/                    # archivos de test
    BIN/                  # comandos, scripts
    RPT/                  # archivos "report"
    TV/                   # vectores de test
```

4.4.2. Configuración del kit de diseño y aplicaciones EDA

Para utilizar las herramientas EDA y el kit de diseño, se añadir un archivo *edadk.conf* en el directorio principal del proyecto. Sin él no se pueden ejecutar las aplicaciones. Por eso deben lanzarse desde el directorio principal. El contenido del archivo *edadk.conf* debe ser el siguiente (el orden no es relevante):

```
mgc msim 6.2c
snps syn 2005.09
cds soce 4.1
cds ic 5.1.41
dk ams hk371
```

Este archivo sólo se reconoce en las máquinas Linux *immsunsrv1* y *immsunsrv2*[23].

[23] *immsunsrv1* y *immsunsrv2* son los dos servidores con los que trabaja del grupo de investigación del LMSI3 de la EPFL (École Polytechnique Fédérale de Lausanne).

4.4.3. Instalación del kit AMS de Austriamicrosystems

Para instalar los archivos necesarios para utilizar el kit de diseño hay que ejecutar el comando *ams_setup* en el directorio principal, como se muestra a continuación:

- Para instalar las herramientas de simulación lógica y síntesis, con las propiedades de la tecnología CMOS de 0.35µm y 4 capas de metal, llamada H35B4:

```
ams_setup -p h35b4 -t synopsys_dc
```

  Este comando crea los archivos de configuración de Modelsim y Synopsys, a la vez que establece las librerías necesarias para compilar y simular de los modelos VHDL/Verilog.

- Para instalar el mismo kit de diseño para el emplazamiento y enrutamiento de las celdas estándar:

```
ams_setup -p h35b4 -t cadence_soce
```

  Este comando crea diversos archivos en el directorio PAR (place&route).

Entorno gráfico de Modelsim

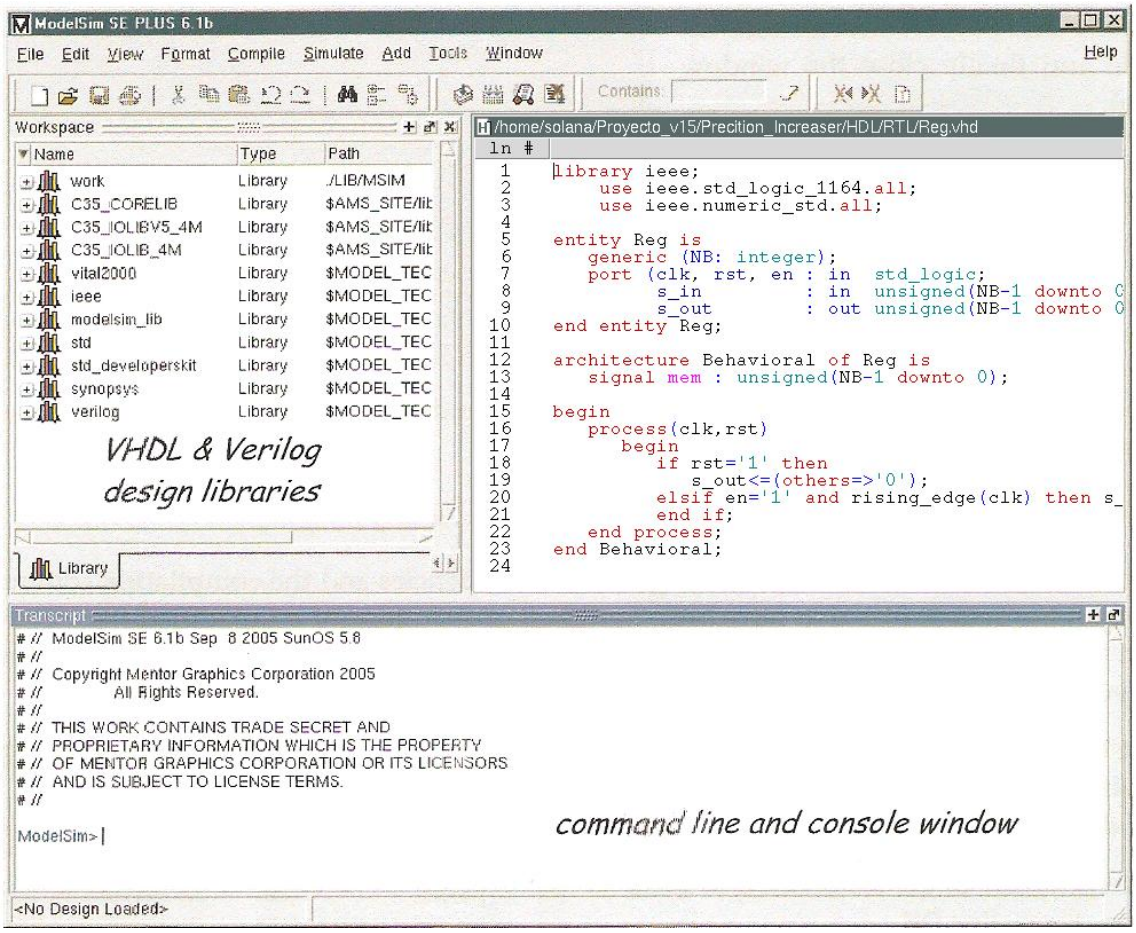


Figura 4.4-2. Vista de la aplicación Modelsim.

4.5. Edición de módulos VHDL y síntesis lógica

En este apartado se pretende comentar la implementación final de cada bloque. Se seguirá un orden de menor a mayor complejidad, es decir, primero se explicarán los bloques simples y después los que incluyen a otros bloques en su implementación.

4.5.1. Registros

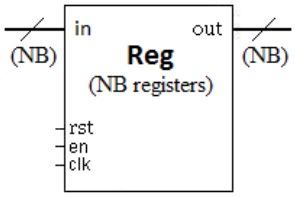


Figura 4.5-1. Vista exterior de un boque *Reg* de NB bits.

Se pretende que el bloque *Reg* sea reutilizable independientemente del tamaño de los datos que se conecten a su entrada. Por eso lo primero que se debe hacer es definir una variable genérica NB, cuyo valor se establecerá al simular (en el archivo *testbench*) y más tarde, en la síntesis lógica del circuito completo. Definida la variable genérica NB, esta se utiliza para establecer el tamaño (en número de líneas – una línea por bit) de las señales del bloque *Reg*.

La función del bloque *Reg*, como la de cualquier registro, es la de almacenar un dato hasta la llegada de un flanco de reloj. En este bloque, como en todos los demás se establece el flanco de subida, como el elegido para realizar el cambio de estado. De esta forma el registro cambia el dato que almacena por el que se encuentra en entrada cuando el detecta un flanco de subida en el reloj. La salida del bloque muestra siempre la señal almacenada, tan pronto como esta cambia, cambia también la salida.

Para almacenar el valor de la señal de entrada se utiliza una señal interna definida como "*mem*" (del mismo tipo y tamaño que la entrada y la salida, *s_in* y *s_out* respectivamente). Al realizar la síntesis lógica, esta señal junto con la instrucción *if then* utilizada se traducen en un *flip-flop* o biestable tipo-D para cada bit, como el que se muestra en la Figura 4.5-2.

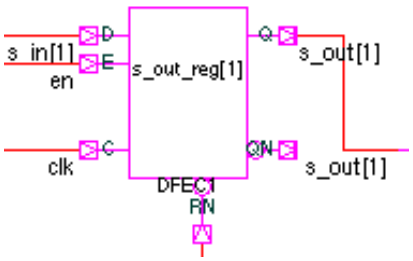


Figura 4.5-2. *Flip-flop* o biestable tipo-D.

Por tanto, el bloque consiste en una batería de NB biestables tipo-D conectados en paralelo, como se puede ver en el ANEXO III de diagramas de bloques sintetizados.

El bloque *Reg* también incluye la posibilidad de ser reiniciado (valor guardado = "00…0") o de ser habilitado/deshabilitado. Por ese motivo se incluyen las entradas *rst* y *en*. Estas señales

deberán encontrarse junto con el reloj *clk* en la lista sensible[24] del proceso que implemente el funcionamiento del bloque.

4.5.2. Multiplexor de 2 entradas

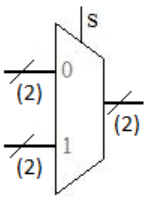


Figura 4.5-3. Vista exterior de un multiplexor de 2 entradas.

Este multiplexor se utilizará en el bloque corrector de la media. Aunque las entradas y salidas de este multiplexor serán siempre buses de 2 bits, en la implementación del bloque se vuelve a definir una variable genérica para definir el tamaño de las señales de entrada, así como de la salida. Esto se hace con el fin de continuar con la filosofía generalista y de reutilización de los modelos VHDL. Así si se quisiera utilizar un multiplexor de dos entradas en otra parte del diseño o incluso en otro circuito diferente, se podría utilizar este bloque independientemente del tamaño de los datos. (Finalmente se utiliza también para seleccionar la salida del circuito entre el *modo test* o el *modo normal* – ver apartado 4.3.7).

En este caso el bloque es asíncrono, motivo por el cual no se utiliza señal de reloj. Además se pretende que la salida esté directamente conectada a una de las dos entradas, en función de la señal de selección *S*. Esto se puede hacer en VHDL con un proceso en el que se incluyan todas las señales de entrada (*a, b* y *S*) en la lista sensible, o mediante una instrucción *when else* en una sola línea. Se eligió la segunda opción por ser más simple.

La salida se conecta a la primera entrada si *S='0'*, en caso contrario se conectaría a la segunda entrada.

4.5.3. Multiplexor de salida

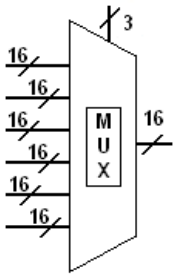


Figura 4.5-4. Vista exterior del multiplexor de salida.

El multiplexor de salida se utiliza para seleccionar la salida del circuito entre las salidas de cada acumulador y la entrada al primer acumulador. Como hay 5 acumuladores, con la entrada del primer acumulador hay 6 posibles salidas. Por tanto se necesita una señal de selección de, al menos, 3bits. Siguiendo con la idea de hacer modelos reutilizables, se vuelve a definir la señal NB de la misma forma en que se hacía en módulos anteriores.

[24] En la lista sensible de un proceso en VHDL se enumeran las señales, cuyos eventos determinan la ejecución del proceso. Definen por tanto la forma de concurrencia del proceso y, por tanto, debe prestarse especial atención a su definición.

La única diferencia real a la hora de programar este multiplexor, respecto al anterior, es el número de entradas. Este parámetro no es fácilmente escalable. Si en el caso anterior se resolvía con una simple instrucción *when else*, en este caso habrá que crear un proceso que permita introducir una estructura *case*. En la estructura *case* se especificará la señal de entrada que se debe conectar a la salida para cada combinación de la señal de selección *S*. Las señales de la lista sensible de este proceso son todas las entradas, incluida la señal *S*. Las combinaciones establecidas se especifican en la tabla 4.5-1.

Señal de selección '*S*'	Señal conectada a la salida
000	Entrada al 1er Ac
001	Salida del 1er Ac
010	Salida del 2º Ac
011	Salida del 3er Ac
100	Salida del 4º Ac
101	Salida del 5º Ac
resto	Entrada al 1er Ac

Tabla 4.5-1. Combinaciones de selección de entradas del *MuxOut*.

4.5.4. Comparadores

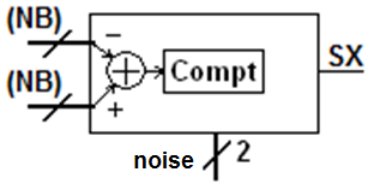


Figura 4.5-5. Vista exterior de un bloque comparador.

Según lo discutido en el apartado 4.5-5, este bloque debe, primero realizar una resta entre las dos señales de entrada para después comprobar si la diferencia obtenida está dentro del rango de ruido establecido por la señal *noise*. La resta se introduce en el código VHDL mediante una simple operación aritmética que se traducirá en un sumador de NB estados y una conversión en Ca2 (complemento a 2) de una de las señales. Pero una comparación > o <, que en código VHDL puede parecer simple, se convertiría en un bloque de compleja implementación. Para evitarlo se ha diseñado una forma alternativa de realizar la comparación, utilizando solamente comparaciones de igualdad, que son más fáciles de implementar.

Comparación alternativa:

Se trata de comparar la diferencia entre las señales de entrada y el ruido máximo permitido. Al realizar una resta entre dos muestras sucesivas puede ocurrir que el resultado sea positivo, o que sea negativo, en función del sentido de giro del sensor, del paso por el origen angular (0°=360°) o por efecto del ruido aditivo que tiene un valor máximo de $\pm 11111_2$. Si el resultado es positivo, se representará con una señal binaria en la que los bits más significativos serán ceros, pero si la diferencia fuera negativa los MSB serán unos. En cualquier caso, si la diferencia es menor que el máximo ruido permitido, todos los "bits de estabilidad" deben ser iguales entre sí. Todos unos o todos ceros, pero iguales entre sí. Por tanto, se pueden separar los "bits de estabilidad" de la señal diferencia y reducir el problema de la comparación a comprobar si los bits son estables entre sí. Si lo son se considerará que la señal es estable.

Por notación y para simplificar la explicación llamaremos *señal Y* a la señal diferencia, y *señal Z* a los "bits de estabilidad" de la señal *Y*.

Para realizar esta comprobación, lo más evidente sería decir que la señal es estable si

Z=="00…0" or Z=="11...1".

Esto supone dos comparaciones para cada bit de la señal *Z*, que se pueden reducir a una sola si definimos otra señal *Zp* resultado de realizar un desplazamiento circular a izquierdas (o a derechas, no es relevante). Así, comparando solamente estas señales entre sí, se puede determinar unívocamente si todos los bits de *Z* son iguales entre sí. Es decir, la señal sería estable, si y solo si

Z==Zp.

Por otra parte, queda por resolver el asunto de los rangos de ruido que establece la señal de entrada *noise*. La diferencia entre los distintos rangos de ruido está en el número de "bits de estabilidad", o visto de otra forma, el tamaño que tendrá la señal *Z*. Así, con el rango de ruido menor *Z* tendría una longitud de 11bits, mientras que con el rango mayor su longitud sería de sólo 8btis. Pero estas comparaciones se pueden hacer de forma concurrente sin necesidad de duplicar recursos. Basta con definir la longitud de *Z* y *Zp* como la correspondiente al peor caso (*High Noise* → *L=8bits*), compararlas como se explicaba anteriormente y para cada uno de los otros rangos de ruido, se irá comparando un bit cualquiera de la señal *Z*, con el más significativo de los que quedan entre los descartados de *Y*.

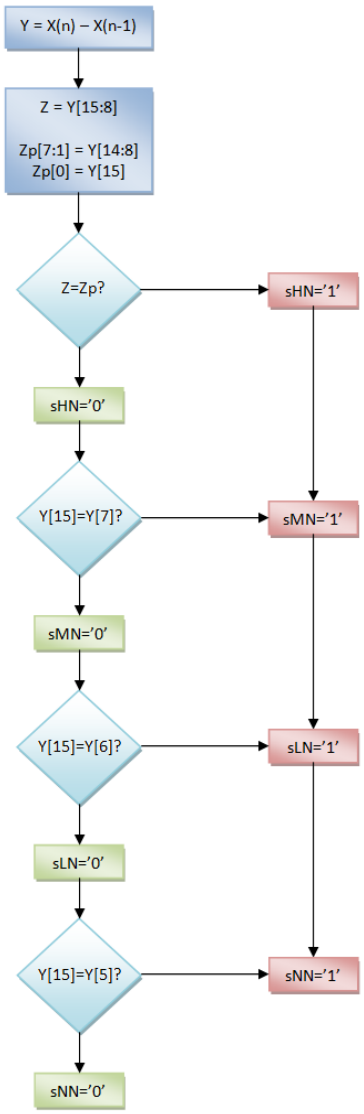


Figura 4.5-6. Diagrama de flujo de ul bloque comparador con NB=16bits.

La Figura 4.5-6 muestra, de forma clara, el funcionamiento de este sistema de comparación. Al final se obtienen 4 señales de un solo bit (sHN, sMN, sLN y sNN) que determinan para cada rango de ruido si la señal es estable. La señal *noise* actuará como señal selectora en un multiplexor que tomará como entradas a estas señales y cuya salida será la salida del bloque comparador (externamente llamada señal *S*).

noise[1:0]	Noise Range	Signal	Output	
00	No Noise	*sNN*	0	Stable
			1	Instable
01	Low Noise	*sLN*	0	Stable
			1	Instable
10	Medium Noise	*sMN*	0	Stable
			1	Instable
11	High Noise	*sHN*	0	Stable
			1	Instable

Tabla 4.5-2. Señal de estabilidad para cada rango de ruido.

Por último, decir que en la definición VHDL de este modelo sigue los principios de generalidad y reutilización, como los anteriores. Por eso se definen las variables genéricas *NB* y *NBnoise* para determinar el tamaño de las entradas de datos y el número de bits del menor rango de ruido, respectivamente. Será al sintetizar cuando se les de valores a estas variables. *NB* tendrá un valor diferente según el acumulador con el que se asocie, mientras que *NBnoise=5*bits para todos los comparadores.

4.5.5. Bloque ZPcorrectSA

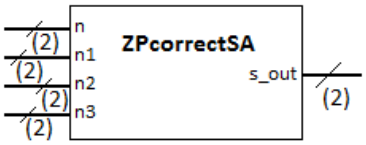


Figura 4.5-7. Vista exterior de un bloque *ZPcorrect*.

La programación del bloque de corrección rompe con la tendencia procedimental que se utiliza para definir la operación de los otros modelos. En la discusión sobre las posibilidades de implementación se llegó a un diseño muy concreto para este bloque. Por tanto, la programación VHDL de este bloque se limita a implementar el esquema de puertas lógicas representado en la Figura 4.3-10.

4.5.6. Acumuladores

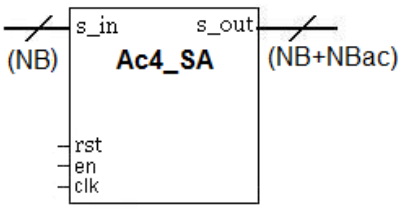


Figura 4.5-8. Vista exterior de un bloque acumulador.

Gracias a la estructuración por modelos que se está realizando, el bloque acumulador queda muy simplificado. Su programación VHDL consiste en la simple conexión de los 3 bloques *Reg* y el bloque de corrección *ZPcorrectSA* (SA de *Sliding-Average*). Además se continúa con la intención de diseñar los modelos VHDL para que sean lo más generales y reutilizables posible, mediante la definición de la variable genérica *NB*, que se utiliza para definir el tamaño de las señales.

4.5.7. Contador

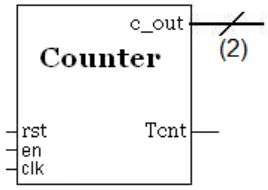


Figura 4.5-9. Vista exterior de un bloque *Counter* de NE estados.

La programación utilizada para definir el modelo contador es muy sencilla. Se trata de un proceso estándar en el que se incrementa una señal interna (*count*) cada vez que llega un flanco de subida de reloj. Como es costumbre se define la variable genérica *NB* para la definición del tamaño de las señales. El valor de la señal interna *count* se almacena en un bloque *Reg* de NB=$\log_2$NE bits, siendo NE el número de estados del contador.

También se incluyen una señal de reset (*rst*) y otra de habilitación (*en*). Además, y aunque no se represente en la Figura 4.5-9, el bloque dispone de una entrada de *NB* bits, llamada *finalCount,* que se utiliza para establecer el valor final de la cuenta.

Dispone de dos salidas: una de *NB* bits (típicamente 2) que ofrece el resultado de la cuenta y otra de un solo bit que sólo tiene valor lógico '1' cuando el contador está en el último estado de la cuenta. Esta última señal, llamada *Tcnt (terminal count)* es la más utilizada, ya que informa del final de la cuenta.

4.5.8. Unidad de control

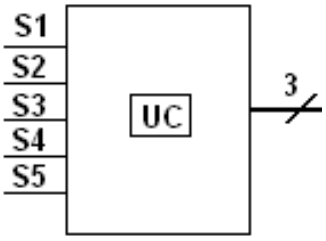


Figura 4.5-10. Vista exterior de la unidad de control. No se representan las entradas de reloj ni reset aunque en el modelo programado si aparecen.

Como en el caso de los acumuladores y del bloque corrector, el principio de operación de la unidad de control se explicó en detalle anteriormente (en el apartado 4.3–4). Por tanto, de la programación VHDL de este modelo, sólo queda decir que se utiliza una estructura *case when* para definir el comportamiento de cada estado. El estado actual se almacena en un bloque *Reg* de 4 registros. Y para definir el "tiempo de precaución" se utiliza la señal *Tcnt* del modelo *Counter*, con cuatro estados.

Este modelo no es fácilmente escalable y por tanto poco reutilizable. Esto se debe a la utilización de la estructura case, que debería reorganizarse si se ampliara el número de estados. Además habría que tener en cuenta más señales.

Limitaciones del código VHDL

Es en este bloque en el único en el que se han detectado ciertas limitaciones del entorno VHDL o tal vez del compilador de Modelsim. El comportamiento ideal de la unidad de control implicaría la necesidad de analizar los eventos producidos en cualquiera de las señales *S* procedentes de los comparadores de forma asíncrona. Esto no se ha conseguido implementar. Aunque se pueden incluir multitud de señales en la lista sensible de un proceso, el compilador sólo permite que se analicen los eventos de una de ellas. Es decir, el proceso se inicia ante evento de cualquier señal que se encuentre en su lista sensible, pero sólo tomar una de ellas como señal de reloj. Al intentar identificar un evento de una señal concreta, como por ejemplo un flanco de subida, el compilador entiende que esa señal es el reloj del proceso y no permite que un mismo proceso tenga dos señales de reloj.

Ante esta limitación se optó por no incluir a las señales *S* en la lista sensible del proceso principal de la unidad de control, limitando dicha lista a la señal de reloj y al reset. De esta forma los cambios entre estados ocurren ante cambios de la señal de reloj, convirtiendo la unidad de control en un bloque síncrono. Pero para minimizar el daño causado por los retrasos, la actualización de los estados se sincroniza con cualquier evento de la señal de reloj, ya sea flanco de subida o de bajada. De esta forma, el retardo producido por este bloque no será relevante para el comportamiento del circuito incrementador de la precisión.

4.5.9. Incrementador de la Precisión – Top-Level

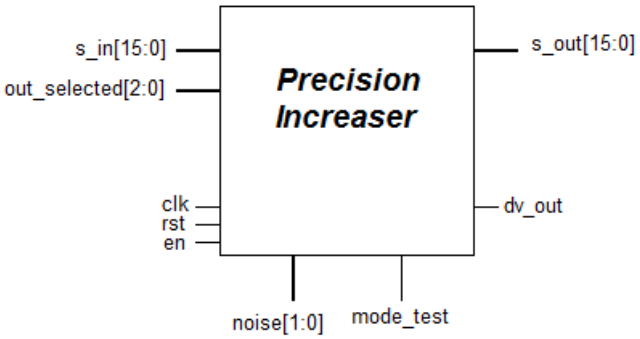


Figura 4.5-11. Vista exterior del modelo *Top-Level*. El incrementador de la precisión.

Este es el bloque de mayor jerarquía. En él se conectan todos los anteriores para formar el diseño final del circuito incrementador de la precisión, tal y como se indicaba en esquema de la Figura 4.3-20.

Faltaría añadir las señales *mode_test* y *out_selected*. La primera de ellas sirve para elegir la forma de seleccionar la salida del circuito. Si se conecta a masa (valor lógico ‘0’) el circuito funcionará de forma autónoma, siendo la unidad de control la responsable de seleccionar la salida señal que se conecta a la salida y con ello el nivel de filtrado de la señal de salida. Si la entrada *mode_test* se conecta a una tensión de 3.3V (valor lógico ‘1’) se deshabilita la unidad de control, pasando a ser la señal externa *out_selected* la responsable de seleccionar el nivel

de filtrado que se quiere ver en la salida del circuito. Como ya se explicó esto se hace para permitir la observabilidad de la medida en todos los niveles de filtrado independientemente de su estabilidad. Propiedad que resultará muy útil en la fase de test. Estas señales se conectan mediante un multiplexor de dos entradas (modelo definido anteriormente). Las entradas del multiplexor serán la salida de la unidad de control (*c_muxUC*) y la señal externa *out_selected*. La señal de control será *mode_test* y la salida (*c_mux*) se conectará como señal selectora al multiplexor final, el que selecciona la salida final del circuito entre sus seis entradas (los seis niveles de filtrado).

Por último se añade un bloque *Reg* de 16 registros antes del primer acumulador para asegurar la sincronía de la señal de entrada de datos con la señal de reloj. Esto hace que el circuito sea muy robusto ante posibles fallos de sincronismo externos.

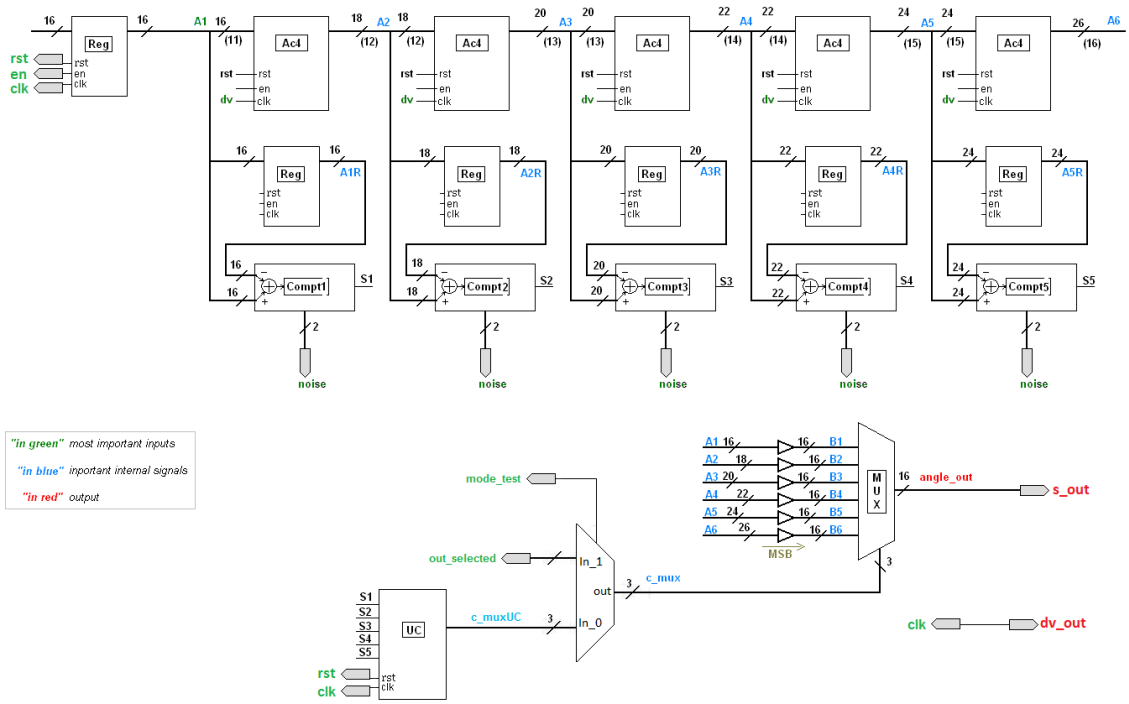


Figura 4.5-12. Representación del esquema definitivo del circuito diseñado.

4.6. Resultados - Simulación de archivos VHDL

Para comprobar el correcto funcionamiento del circuito diseñado es muy útil realizar simulaciones lógicas como las siguientes. Las simulaciones de archivos VHDL se ponen en marcha ejecutando archivos *testbench* programados también en VHDL (ver ANEXO IV). En estos archivos se incluye el módulo general del circuito de forma similar a como un módulo VHDL añade en su código a otros módulos ya definidos. Por tanto, en el archivo *testbench* se debe definir una señal para cada entrada y salida del módulo a simular. En los archivos de simulación, además de especificar los valores de las variables genéricas del módulo a simular, se deben establecer valores para las entradas del módulo. Estos valores pueden dejarse fijos de principio a fin de la simulación o se puede cambiar. Es necesario definir una variable temporal. Esta variable servirá para generar la señal de reloj que se conecta a la entrada correspondiente del bloque *Precision_Increaser_SA*. El resto de señales se deben sincronizar con la señal de reloj.

Comienza, ahora, la fase de generación de vectores de entrada. Es decir, hay que decidir que entradas se van a introducir en función de la respuesta que se espere del circuito.

Antes de llegar a la simulación del módulo de mayor jerarquía, que incluye a todos los otros, incluso antes de llegar implementarlo, es aconsejable ir realizando diversas simulaciones para ir comprobando el funcionamiento de cada submódulo. De no hacerlo, se corre el riesgo de encontrar errores al simular el circuito completo y no saber identificarlos y/o localizarlos. Siguiendo este consejo, durante el desarrollo del diseño y su implementación se fueron realizando multitud de simulaciones para cada uno de los distintos submódulos. Se pudo comprobar por tanto el correcto funcionamiento de ellos. Estas simulaciones no son relevantes en cuanto a los resultados finales del circuito, por lo que se omiten. Bastará con saber que los módulos responden al comportamiento que se espera de ellos.

Por tanto, queda por comprobar el comportamiento del circuito cuando la señal es estacionaria y cuando deja de serlo.

Las simulaciones que se muestran a continuación se han realizado con la señal *mode_test* desactivada, para permitir el funcionamiento de la unidad de control, y con el ruido mínimo. En la figura 4.6-1 se puede ver el resultado de la primera simulación. La primera señal que aparece (desde arriba) es la entrada de datos. Se puede ver que es una señal estacionaria durante un tiempo suficientemente largo, hasta que se produce un cambio que hace reaccionar al circuito. La señal de salida es *s_out* (la octava desde arriba). Se puede ver como produce una salida errónea justo en el momento del cambio del ángulo de entrada, pero sólo durante un ciclo de reloj. Si se observa la señal de salida de la unidad de control (la decima desde arriba), se puede ver que no se actualiza hasta un ciclo después de producirse el cambio de la entrada. Este retardo es el que provoca la salida errónea, ya que durante ese ciclo de reloj la salida del circuito continúa conectada a la salida del último acumulador. Es decir, durante un ciclo de reloj se mantiene el máximo nivel de filtrado aunque la señal no es estable. El retardo de la UC se debe a que se ha implementado de forma síncrona (ya se razonó este problema en el apartado 4.5 de edición de módulos VHDL). Por tanto este error no es corregible.

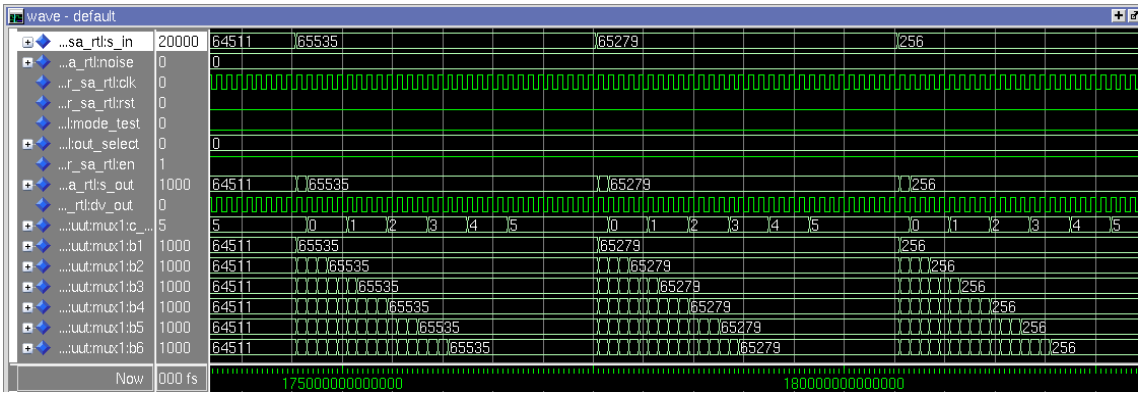


Figura 4.6-1. Respuesta a los cambios angulares.

Si se observan las señales *b1, b2, b3, b4, b5* y *b6* (que son las salidas de cada acumulador) se puede ver cómo la nueva estabilidad de la señal se va extendiendo por los acumuladores. Y cómo la señal de salida de la UC (*mux1:c_...*) va seleccionando salidas de nivel superior según va comprobando la estabilidad de la señal.

En las ilustraciones 4.6-2, 4.6-3 y 4.6-4 se muestran los resultados de otra simulación con parámetros similares, en la que se produce un cambio de la entrada de datos antes de que se produzca el restablecimiento del máximo nivel de filtrado. Esto lleva a la máquina de estados a mantener una salida de nivel inferior durante más tiempo, hasta que comprueba que la señal vuelve a ser estable. Por lo demás los resultados son similares. La diferencia entre las ilustraciones es el *zoom* aplicado. Se ha intentado hacer más observable el progreso de la señal en las salidas de los acumuladores, pero como puede comprobarse resulta bastante difícil.

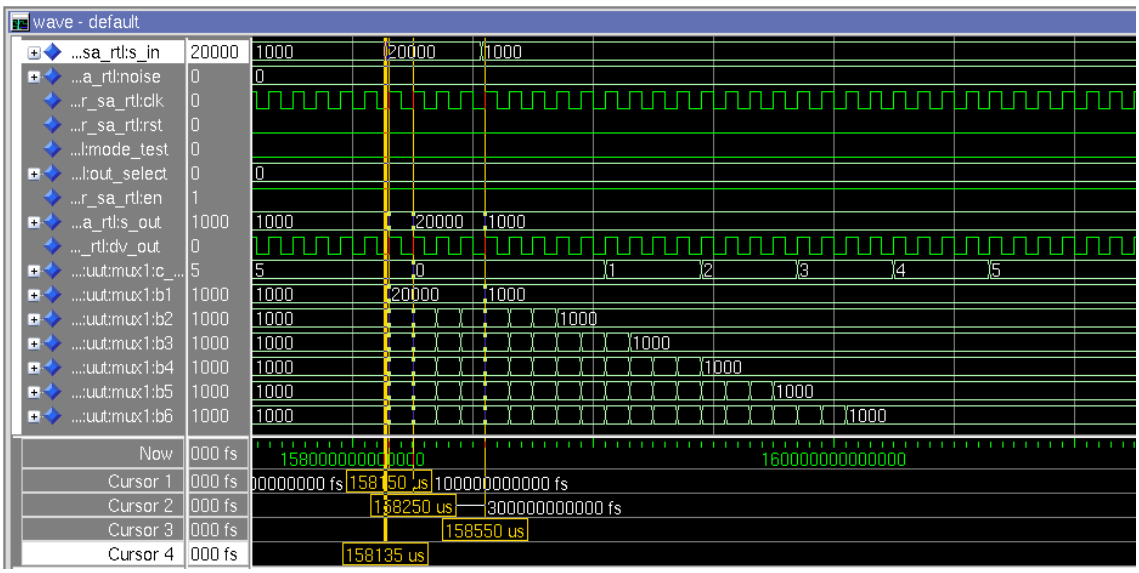


Figura 4.6-2. Respuesta a un gran cambio angular con medidas temporales.

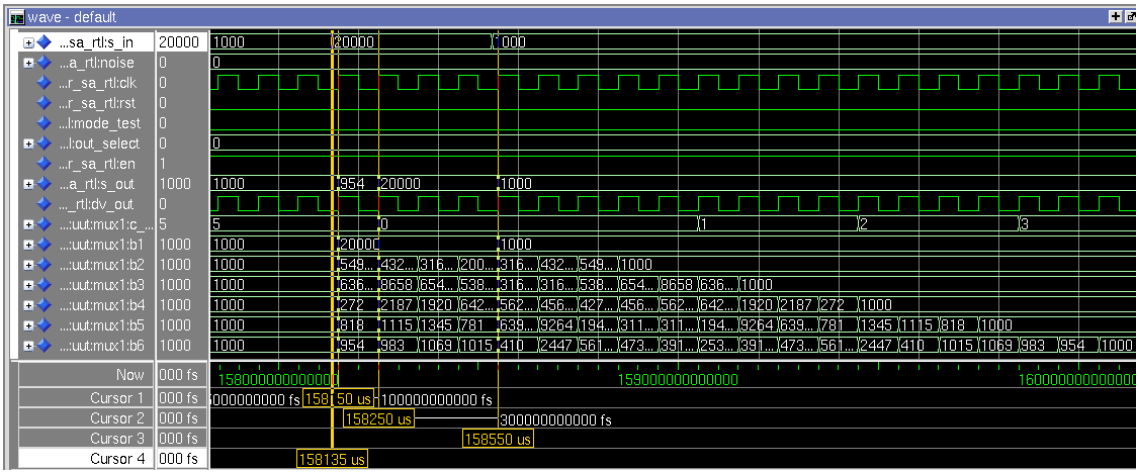


Figura 4.6-3. Respuesta a un gran cambio angular con medidas temporales (zoom 1).

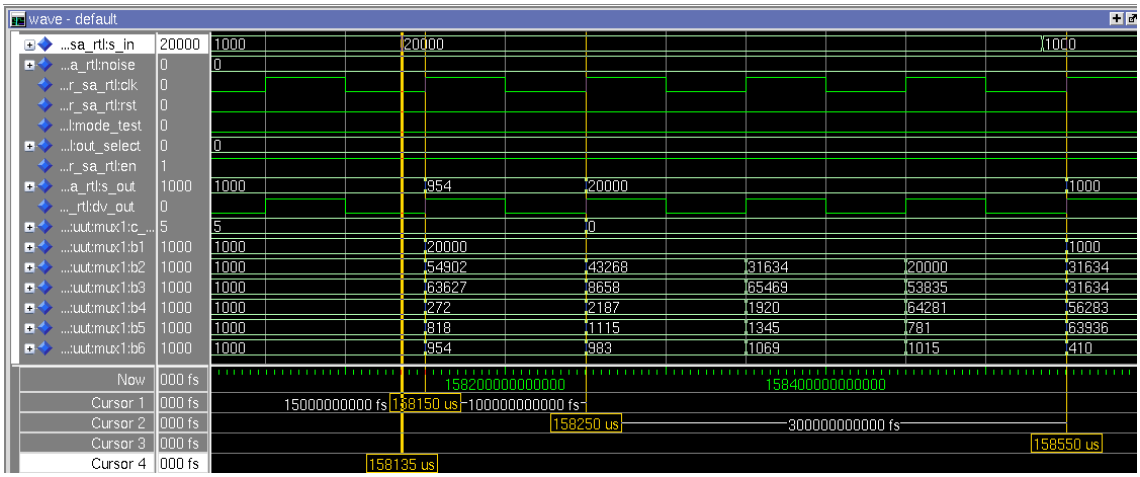


Figura 4.6-4. Respuesta a un gran cambio angular con medidas temporales (zoom 2).

Otras simulaciones realizadas más interesantes incluyen una variación constante de la entrada. No de forma brusca, sino limitándose al margen de ruido establecido. Por desgracia estas simulaciones necesitan ser observadas dinámicamente. Es decir, hay que observar las señales en un periodo de tiempo demasiado grande como para ser reflejado gráficamente en esta memoria. Pero se pueden comentar sus resultados.

Las simulaciones “con ruido” implican escribir un archivo *testbench* con muchísimos cambios de entradas, uno por cada ciclo de reloj. Además hay que establecer una concordancia entre los cambios angulares que se introducen y el ruido que se quiere simular. Calcular y escribir a mano los valores de entrada puede convertirse en una tarea especialmente tediosa. Por ese motivo se definieron una serie de archivos Matlab para la generación de archivos *testbench* con ruido (ver ANEXO IV).

De esta forma se realizaron simulaciones con distintos rangos de ruido. Se pudo observar que cuando el ruido introducido en la señal de entrada era menor que el indicado con la entrada *noise*, la unidad de control mantenía el máximo nivel de filtrado, lo que resulta lógico ya que los comparadores estarían esperando un ruido mayor.

Sin embargo cuando el ruido es considerablemente mayor al indicado con la señal *noise*, la unidad de control mantiene la salida conectada directamente a la entrada. Es decir, no realiza ningún incremento de la precisión.

Un punto intermedio se encuentra cuando el ruido introducido en la medida angular es ligeramente superior al límite establecido por la señal *noise*. En este caso la UC alterna entre niveles de filtrado intermedios, pero mayoritariamente en los más bajos. En un comportamiento ideal (con la UC asíncrona) este caso se resolvería manteniendo un nivel de filtrado intermedio de forma casi constante. Pero al ser la UC asíncrona, se produce una salida errónea cada vez que hay que pasar a un nivel de filtrado inferior. Y ante una señal que puede superar los límites de ruido con cierta frecuencia, este problema obliga a mantener niveles de filtrado muy bajos.

Con estas simulaciones se consideró que el circuito mostraba un comportamiento suficientemente bueno como para seguir con el proceso de diseño. Por ese motivo, los próximos apartados explican los pasos hasta llegar al netlist post-síntesis, que será importable a dos plataformas de *Cadence: Virtuoso* y *Enconter*.

4.6.1. Síntesis lógica

El paso de síntesis lógica produce el esquemático del circuito a partir del código VHDL implementado y el netlist que se importará a otras plataformas para llegar hasta el layout, si procede. La aplicación utilizada para este paso es ***Synopsys Design Vision***.

Lo primero que se debe hacer es analizar los archivos VHDL RTL. Esto se hace mediante **File→Analyze…** en el menú principal de *Design Vision*. Hay que tener en cuenta que se deben seleccionar todos los archivos para el análisis y se debe hacer en orden de jerarquía de los módulos. Aunque también se puede utilizar la opción **Automatic Ordering**.

La siguiente fase es la elaboración del diseño. Mediante **File→Elaborate…** se abre una ventana en la que se debe seleccionar el módulo de mayor jerarquía. Y en la misma ventana se le dan valores a todas las variables genéricas que se definen en el circuito. Las fases siguientes son:

- Establecimiento de las condiciones de operación: **Attributes→Operating Environment →Operating Conditions…** Seleccionando WORST-IND (85°C, 3V –el ideal es 3.3V).
- Definición del reloj: **Attributes→Specify Clock…** Este paso es necesario para realizar los análisis temporales. Se definió un reloj de 10µs, ya que la tasa de refresco ofrecida por el sensor con el que este circuito va a trabajar es de 100KHz.
- Definición de las restricciones de tamaño: **Attributes→Optimization Constraints→Design Constraints…** Mediante *set_max_area 0* se fuerza al sintetizador a optimizar el diseño para ocupar el área mínima posible.
- Mapeo y optimización: **Design→Compile Design… OK**. Con este último paso se obtiene el netlist final del circuito. Este es el momento de guardar el diseño como

SYN/DB/precision_increaser_sa_mapped.dcc

También se generan los esquemáticos, ya compilados, que se muestran en el ANEXO III.

4.6.2. Emplazamiento en celdas estándar y enrutamiento

En este apartado se presentan los pasos principales para configurar el emplazamiento y enrutamiento del Netlist sintetizado a nivel de puerta, utilizando las celdas estándar del kit de diseño AMS. La aplicación utilizada es *Cadence Encounter*. Se sigue trabajando en entorno Unix. El comando *cdsdoc* da acceso a la documentación Cadence, aunque la aplicación también tiene un menú de ayuda.

Para iniciar la aplicación Encounter basta con introducir el comando *encounter* en una nueva ventana Unix. A partir de ahora llamaremos a esta ventana "*consola Encounter*". Aquí es donde se pueden introducir todos los comandos de Encounter y donde la aplicación mostrará todos sus mensajes.

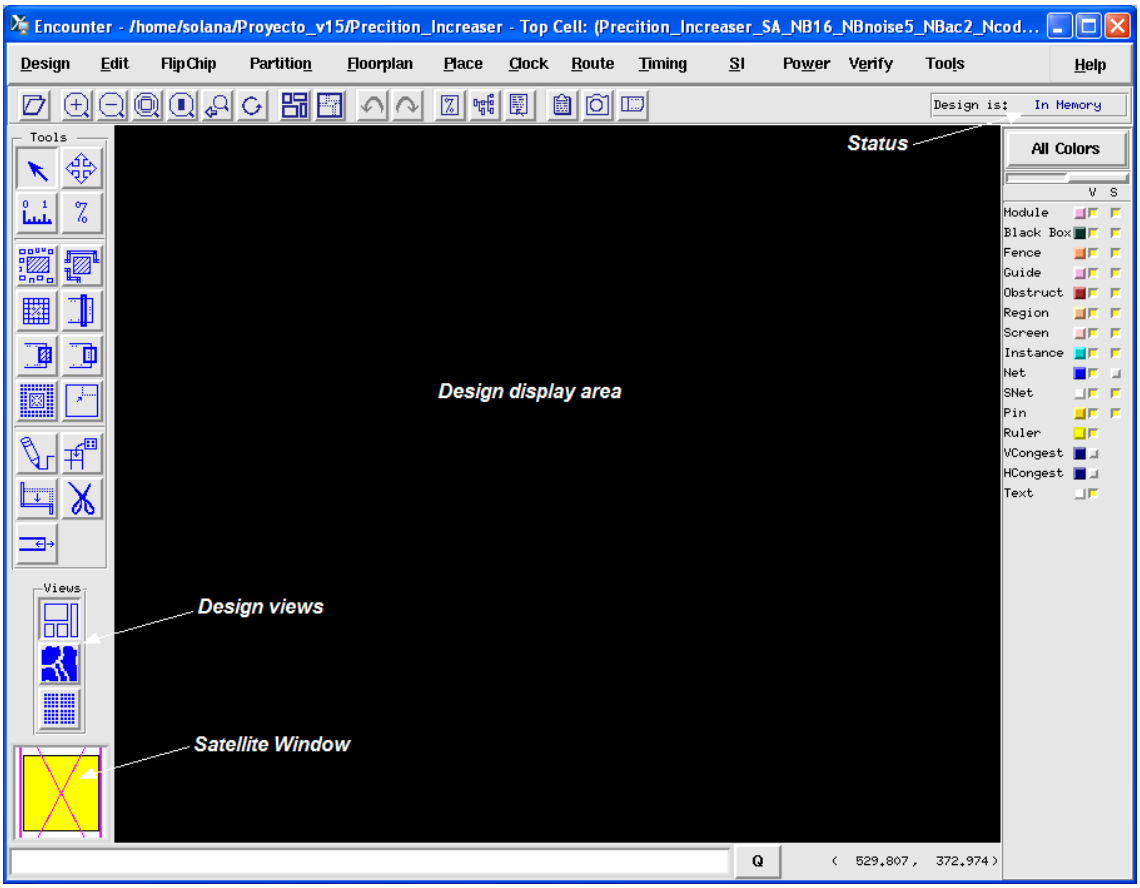


Figura 4.6-5. Vista de la aplicación Encounter de Cadence.

Al iniciar la aplicación aparece otra ventana que llamaremos la "ventana principal". Esta incluye tres vistas de diseño diferentes entre las que se puede cambiar durante la sesión: la vista *Floorplan,* la vista de Ameba y la vista Física.

La primera de ellas, la vista *Floorplan* representa las líneas de conexión, el emplazamiento de los bloques y las líneas de alimentación y tierra. La vista de Ameba representa los contornos de los módulos y submódulos después del emplazamiento, mostrando la localización física de cada módulo. La vista Física representa el emplazamiento detallado de los bloques de módulos, celdas estándar, conexiones, etc.

Importación del diseño

La importación del diseño a la aplicación *Encounter* implica especificar la siguiente información de configuración:

- **Librerías y archives del diseño.** Esto incluye la información relativa a la tecnología utilizada y las librerías de celdas en formato LEF (*Layout Exchange Format*). Los archivos LEF proporcionan información de las capas metálicas y de conexión, así como las reglas de diseño que se utilizarán para las tareas de enrutamiento.

- **Netlist a nivel de puerta.** Esto es el netlist (Verilog) para ser situado y enrutado.

- **Librerías temporales.** Esto incluye información temporal sobre las celdas (retrasos, tiempos *setup/hold,* etc.).

- **Información de alimentación.** Esto incumbe a las conexiones de alimentación que se utilizarán en el layout.

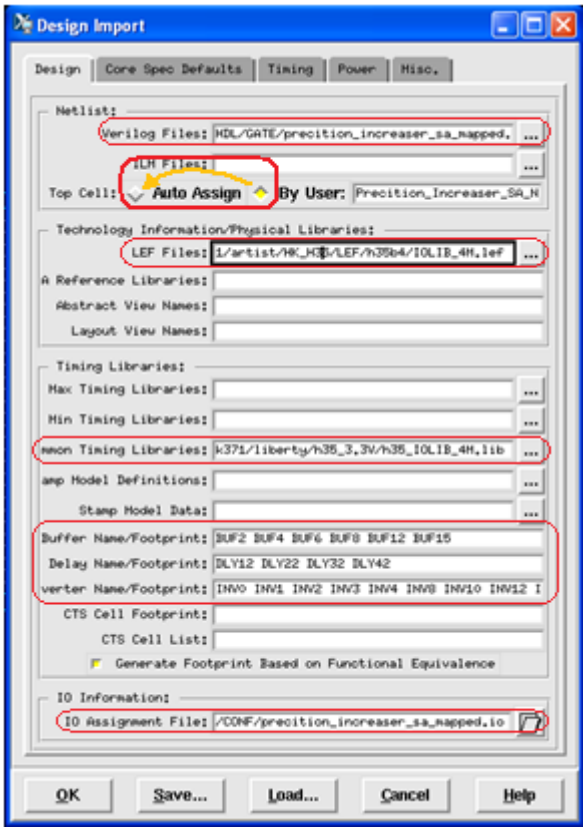


Figura 4.6-6. Ventana de importación de diseños.

La ventana de importación de diseño, que se muestra en la Figura 4.6-6, se abre desde el menú principal seleccionando **Design→Design Import…**. Después, pulsando el botón **Load…** se debe cargar el archivo

PAR/CONF/h35b4_std.conf

Este archivo define la configuración básica de la importación. Al importarlo se rellenan automáticamente algunos de los campos señalados en la Figura 4.6-6. Pero hay dos campos que hay que cargar por separado. El primero de ellos es *Verilog Files*, campo en el que hay que cargar el netlist que se encuentra en

HDL/GATE/precision_increaser_sa_mapped.v

En el campo *IO Assignment File* debe incluirse el archivo de conexiones exteriores que se debe escribir previamente. Este archivo debe encontrarse en

PAR/CONF/precision_increaser_mapped.io

y tendrá el contenido que se muestra en la Figura 4.6-7.

precition_increaser_sa_mapped.io - WordPad

Archivo Edición Ver Insertar Formato Ayuda

```
# angle_decoder.io
# Syntax:
#       Pin: <pin-name> <orientation>
# where <orientation> may be either one of:
#       n       north   (top)
#       e       east    (right)
#       s       south   (bottom)
#       w       west    (left)

Pin: clk e
Pin: rst e
Pin: en e
Pin: noise[0] e
Pin: noise[1] e
Pin: mode_test e
Pin: out_select[0] e
Pin: out_select[1] e
Pin: out_select[2] e
```

```
Pin: s_out[0] n
Pin: s_out[1] n
Pin: s_out[2] n
Pin: s_out[3] n
Pin: s_out[4] n
Pin: s_out[5] n
Pin: s_out[6] n
Pin: s_out[7] n
Pin: s_out[8] n
Pin: s_out[9] n
Pin: s_out[10] n
Pin: s_out[11] n
Pin: s_out[12] n
Pin: s_out[13] n
Pin: s_out[14] n
Pin: s_out[15] n
```

```
Pin: s_in[0] s
Pin: s_in[1] s
Pin: s_in[2] s
Pin: s_in[3] s
Pin: s_in[4] s
Pin: s_in[5] s
Pin: s_in[6] s
Pin: s_in[7] s
Pin: s_in[8] s
Pin: s_in[9] s
Pin: s_in[10] s
Pin: s_in[11] s
Pin: s_in[12] s
Pin: s_in[13] s
Pin: s_in[14] s
Pin: s_in[15] s
```

Figura 4.6-7. Archivo de entradas y salidas del circuito *Precision Increaser.*

En la pestaña *Timing* (ver Figura 4.6-8) se pueden especificar las restricciones temporales que se utilizaron para la síntesis. Para ello sólo hay que seleccionar el archivo generado durante la síntesis lógica

SYN/SDC/precision_increaser_sa_mapped.sdc

De hecho, *Encouter* sólo utilizará la información de restricciones de tiempo de este archivo.

En la pestaña *Power* se pueden añadir las conexiones de alimentación y tierra. Los nombres de estas conexiones deben ser los mismos que se utilizaban en el archivo LEF que describe las celdas estándar.

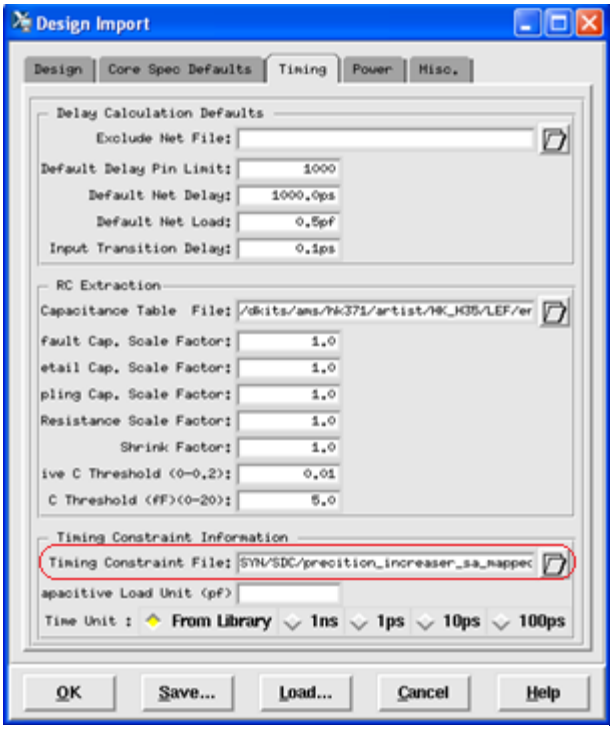


Figura 4.6-8. Restricciones temporales.

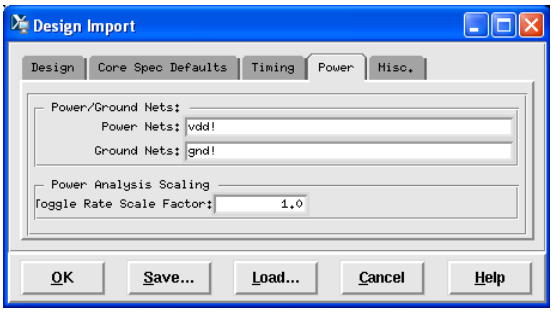


Figura 4.6-9. Conexión de alimentación y masa.

El resto de propiedades pueden dejarse como están. Establecida la configuración básica, es un buen momento para guardarla. Para ello se pulsa el botón **Save…** y se guarda la configuración en un nuevo archivo con el nombre

PAR/CONF/precision_increaser_sa_mapped.conf

Especificación Floorplan

Esta parte define la forma del layout, la rejilla de posibles rutas, las filas que acogerán el núcleo de las celdas y sus conexiones de entrada y salida (si fuera necesario) y la localización de las esquinas de las celdas (si fuera necesario).

La ventana que se muestra en la Figura 4.4-24 se abre seleccionando **Floorplan→Specify Floorplan…** en el menú.

En el campo *Ratio (H/W)* se debe especificar la relación altura/anchura que tendrá el layout. Debido a la configuración del circuito en el que se pretende insertar este diseño, este debe tener una forma rectangular, tal que el ancho sea el doble que el alto. Por tanto se debe introducir un valor de 0.5 en este campo.

La utilización del layout es, por defecto, del 85%. Esto significa que el 15% del espacio se dejará libre para una posible futura inserción de *buffers* o para la recolocación de las celdas y sus conexiones. Tras realizar este proceso algunas veces, con valores distintos para este campo (*Core Utilization*) se decide establecer una utilización del 80% del espacio, porque es el valor máximo con el que se ha conseguido que no se produjera ninguna violación de geometría, conexiones o retardos.

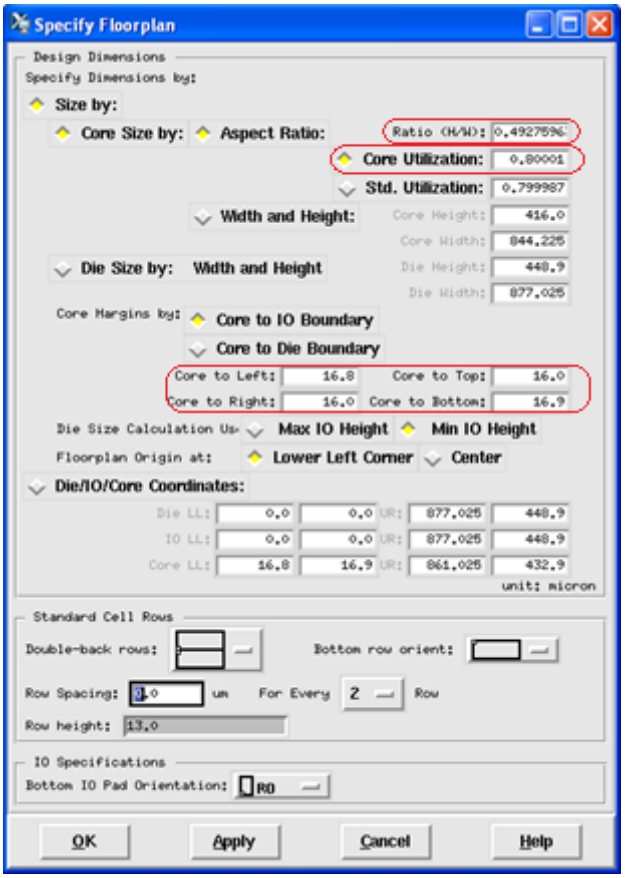


Figura 4.6-10. Especificación de los parámetros del plano.

El espaciado entre celdas y líneas de transmisión debe ser suficiente como para permitir la inserción de líneas de alimentación y de otras conexiones exteriores. Según los diseños anteriores realizados en el laboratorio se decide establecer una distancia entre núcleos de 16µm, que permite introducir una línea de alimentación y otra de masa de 4µm de anchura cada una.

Pulsando **Apply** se actualizarán los parámetros a los valores actuales y pulsando **OK** podremos visualizar el resultado sobre el panel de diseño (ver Figura 4.6-10).

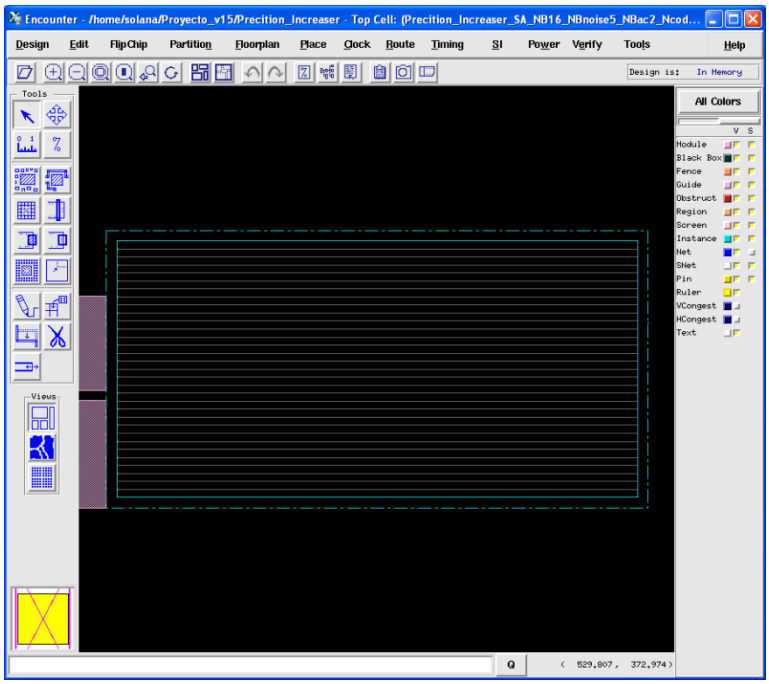


Figura 4.6-11. Plano de pre-layout.

Se decide guardar este paso en *PAR/DB/precision_increaser_sa-fplan.enc*.

Creación de anillos de alimentación

Este paso genera los anillos de tensión V_{DD} y tierra alrededor del núcleo y, opcionalmente añade un número de líneas horizontales y/o verticales a través del núcleo para asegurar una apropiada distribución de la alimentación en núcleos grandes.

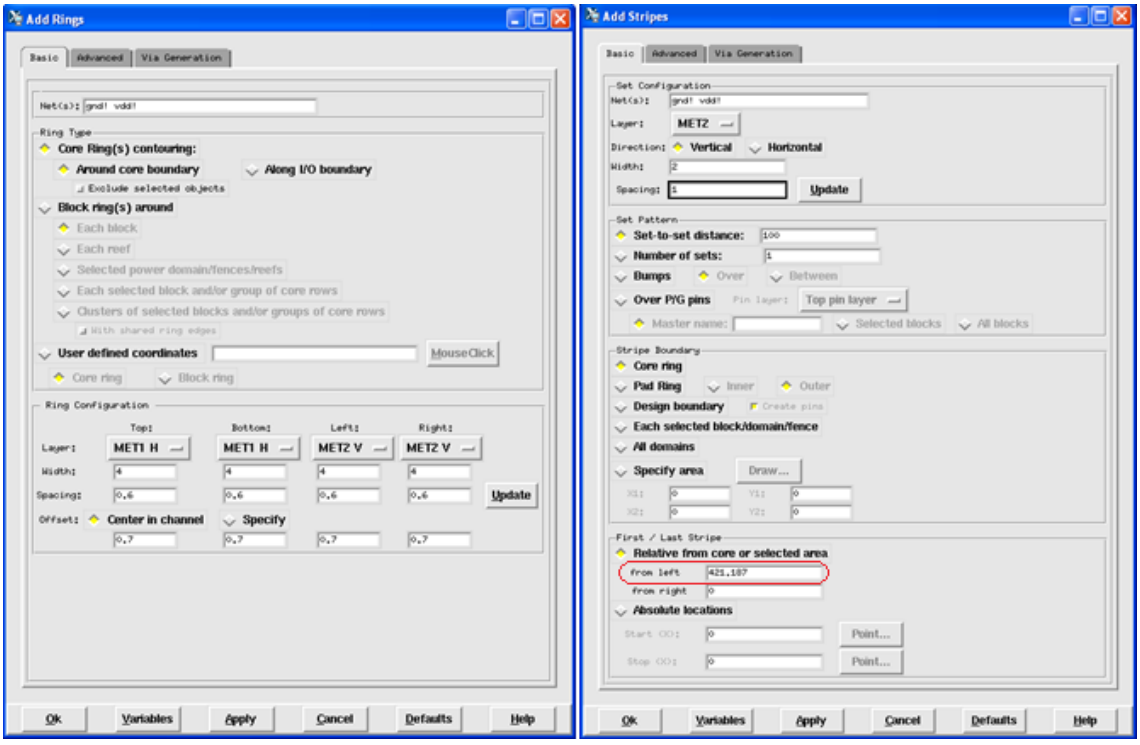


Figura 4.6-12. Establecimiento de líneas de transmisión.

Este paso se realiza sobre las ventanas que se muestran en la Figura 4.6-12, donde el punto más importante se señala en rojo. Este campo sirve para introducir una línea de alimentación y otra de tierra en el interior del núcleo. Exactamente se ha establecido un valor de 425.187µm para situar las líneas en la parte central del núcleo, tal y como se observa en la Figura 4.6-13 que muestra el resultado de este paso.

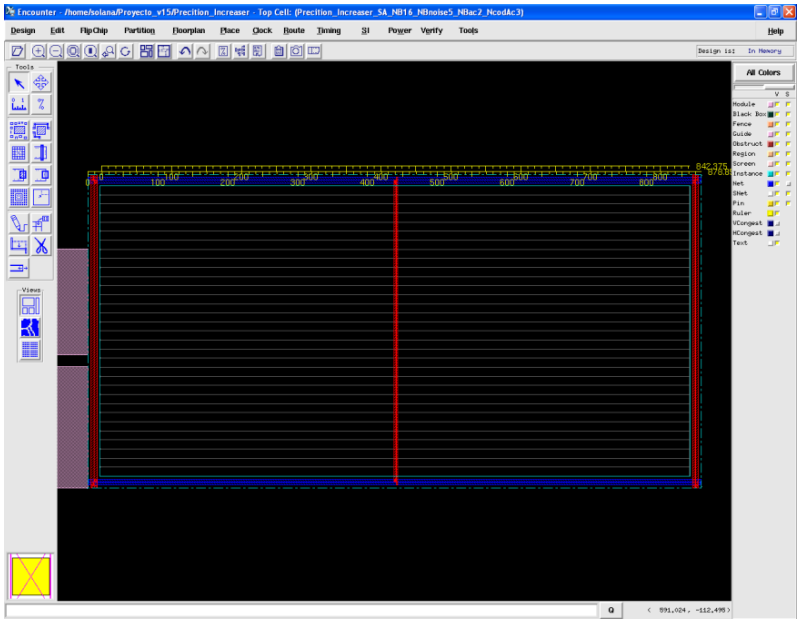


Figura 4.6-13. Visualización de los anillos de alimentación (1).

Si se hace lo mismo, pero introduciendo líneas horizontales se obtiene el resultado que se muestra en la Figura 4.5-14.

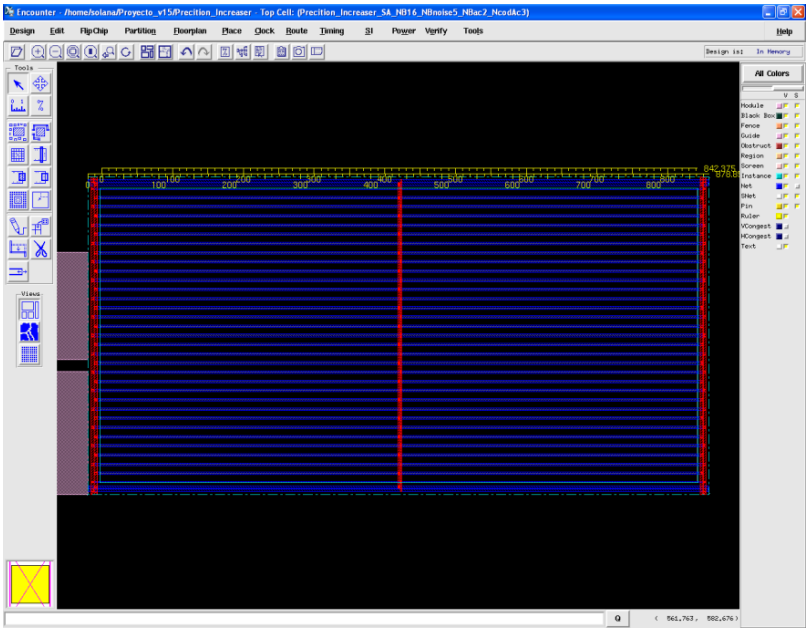


Figura 4.6-14. Visualización de los anillos de alimentación (2).

Conexiones globales

En este paso se asignan los *pins* de las líneas de alimentación y tierra. El netlist Verilog que se importó en el primer paso no hace mención alguna en relación a estas conexiones. Sin embargo, las celdas que se van a utilizar si tienen *pins* para estas conexiones que necesitan ser enrutados a las líneas globales de alimentación y tierra definidas anteriormente. **Floorplan→Global Net Connections…** abre la ventana de la Figura 4.6-15 (izquierda).

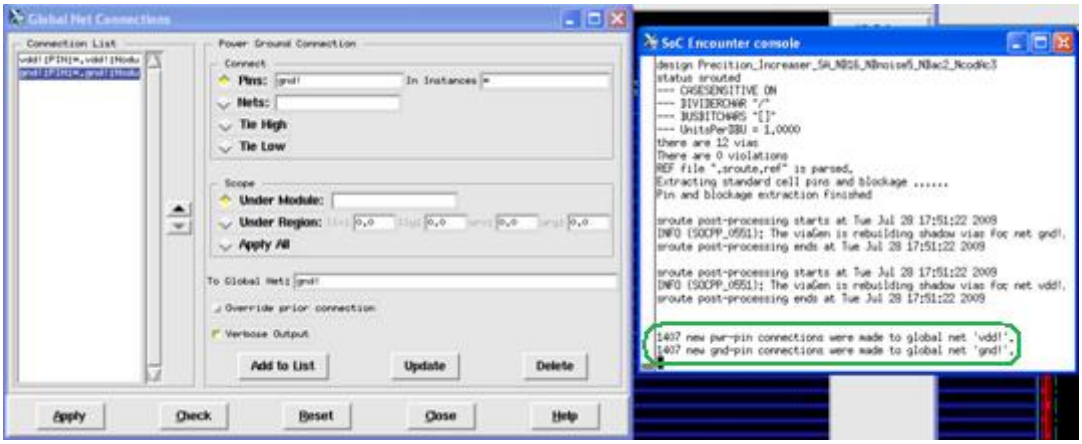


Figura 4.6-15. ***Global Net Connections.***

El panel de la izquierda aparece inicialmente vacio. Por tanto, para cada conexión V_{DD} o de tierra hay que añadir el nombre del *pin*, que debe coincidir con los utilizados anteriormente.

Tras pulsar **Apply** y, después **Close**, la *consola Encounter* (a la derecha de la Figura 4.6-15) muestra el número de conexiones realizadas (1407 conexiones para V_{DD} y 1407 para tierra).

Emplazamiento de celdas CAP

El kit de diseño AMS requiere que las celdas de terminación física se sitúen al final de cada fila. Las llamadas celdas CAP se utilizan para proporcionar la polarización de los sustratos P + y N+.

Para hacer esto, se debe seleccionar **Place→Filler→Add End Cap…**. Después se deben rellenar los campos como se muestra en la Figura 4.4-30 y al pulsar **OK**, se añadirán las celdas CAP al diseño.

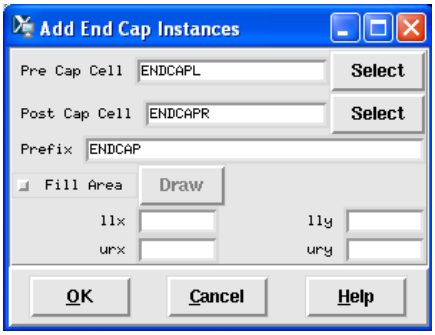


Figura 4.6-16. Emplazamiento de celdas CAP.

Definición de las condiciones de operación

Las condiciones de operación definen la temperatura, el proceso y las condiciones de voltaje del diseño. Estas propiedades afectan directamente a la optimización de los retardos.

Seleccionando **Timming→Specify Analysis Conditions→Specify Operating Condition/PVT…** se abre la ventana de la Figura 4.6-17.

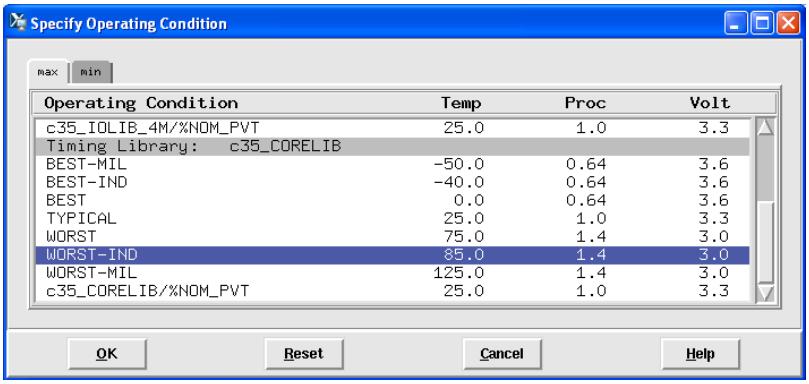


Figura 4.6-17. Condiciones de funcionamiento.

Se debe seleccionar *WORST-IND* en la pestaña *max* y *BEST-IND* en la pestaña *min*. Las condiciones de operación seleccionadas en *max* se utilizarán para establecer las restricciones de retardos, mientras que las seleccionadas en *min* servirán para mantenerlas.

Las condiciones seleccionadas se pueden visualizar en la *consola Encounter* con el comando

```
Encounter 10> getOpCond -v
min: BEST-IND  proc: 0.6400 volt: 3.6000 temp: -40.000
max: WORST-IND proc: 1.4000 volt: 3.0000 temp:  85.000
```

Emplazamiento de las celdas

Ahora se van a situar las celdas importadas en el netlist Verilog. Para ello se abre la ventana de la Figura 4.6-18 en **Place→Place…**. Se podría seleccionar la opción *Timing Driven* para que se optimice la colocación de las celdas que contienen el camino crítico[25], pero no se ha hecho porque no se obtuvo en la fase de síntesis.

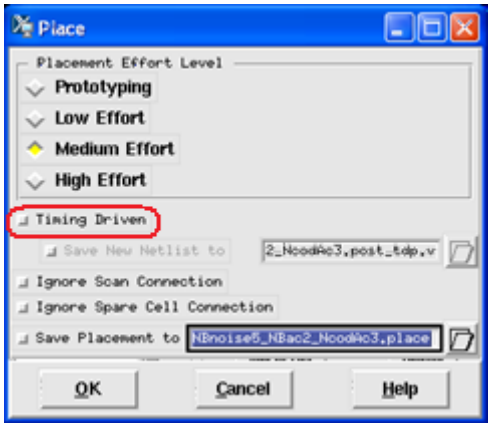


Figura 4.6-18. Emplazamiento de las celdas.

[25] El camino crítico es el que produce un mayor retardo. Se debe localizar en el paso de síntesis.

Tras pulsar **OK** y esperar un tiempo que puede ser relativamente largo si se selecciona *Timmin Driven y Medium Effort* se puede ver el resultado que se muestra en la Figura 4.6-19. Después conviene guardar el diseño con el nombre *PAR/DB/precision_increaser_sa-placed.enc*.

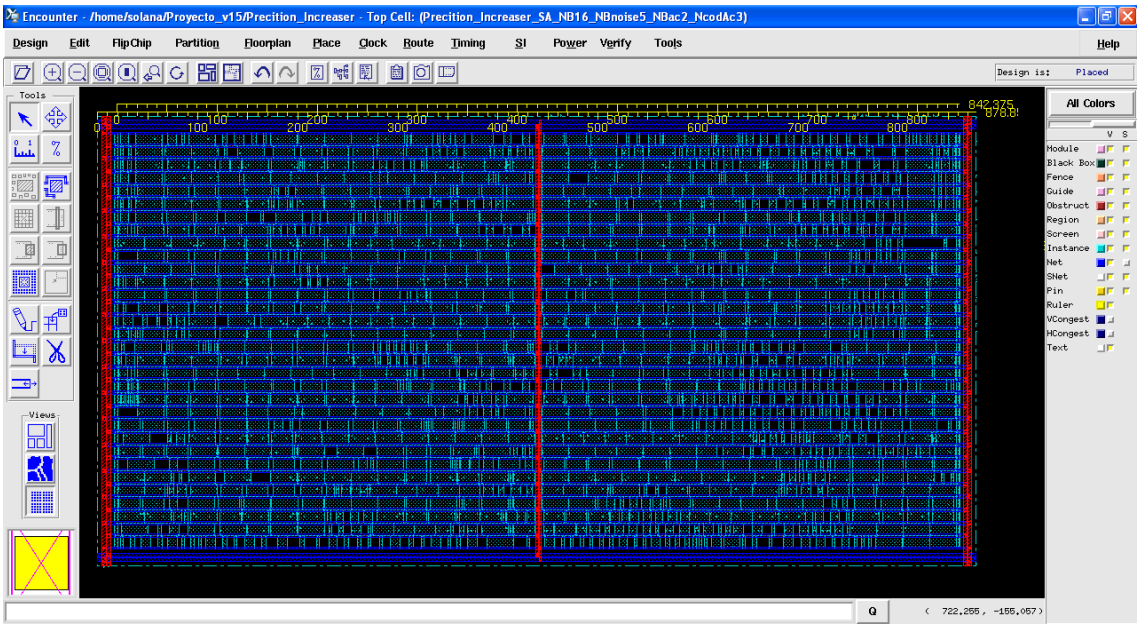


Figura 4.6-19. Visualización del emplazamiento de las celdas estándar del diseño.

Análisis temporal post-emplazamiento

El motor de análisis temporal de *Encounter* puede será utilizado ahora para obtener una idea de las propiedades temporales del diseño. Seleccionando **Timming→Timing Analysis…**y pulsando **OK** se obtiene en la *consola Encounter* el resultado del análisis, que es el que se muestra en la Figura 4.6-20. Préstese especial atención a la parte final del informe generado, donde se expone que no se han encontrado violaciones temporales.

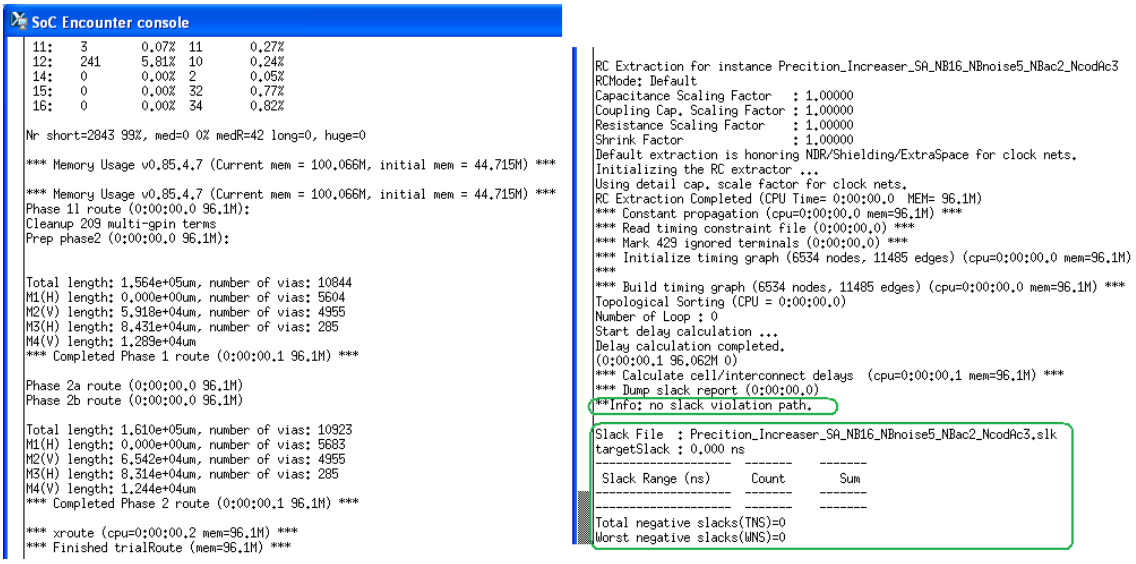


Figura 4.6-20. Análisis temporal posterior al emplazamiento de las celdas importadas.

Enrutamiento del diseño

En este apartado se generarán todos los hilos necesarios para conectar las celdas de acuerdo al netlist importado.

Simplemente se debe abrir **Route→Nanoroute…**, seleccionar la opción *Timing Driven*, mover la barra de esfuerzo a su punto máximo y pulsar **OK**.

Así se podrá ver la disposición final del layout de este bloque (ver Figura 4.6-22).

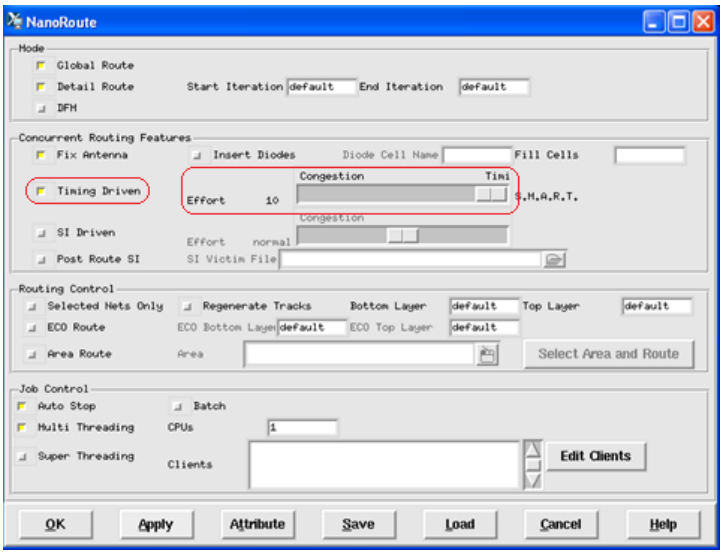


Figura 4.6-21. Enrutamiento del diseño.

El diseño quedará guardado en el archivo *PAR/DB/precision_increaser_sa-routed.enc*.

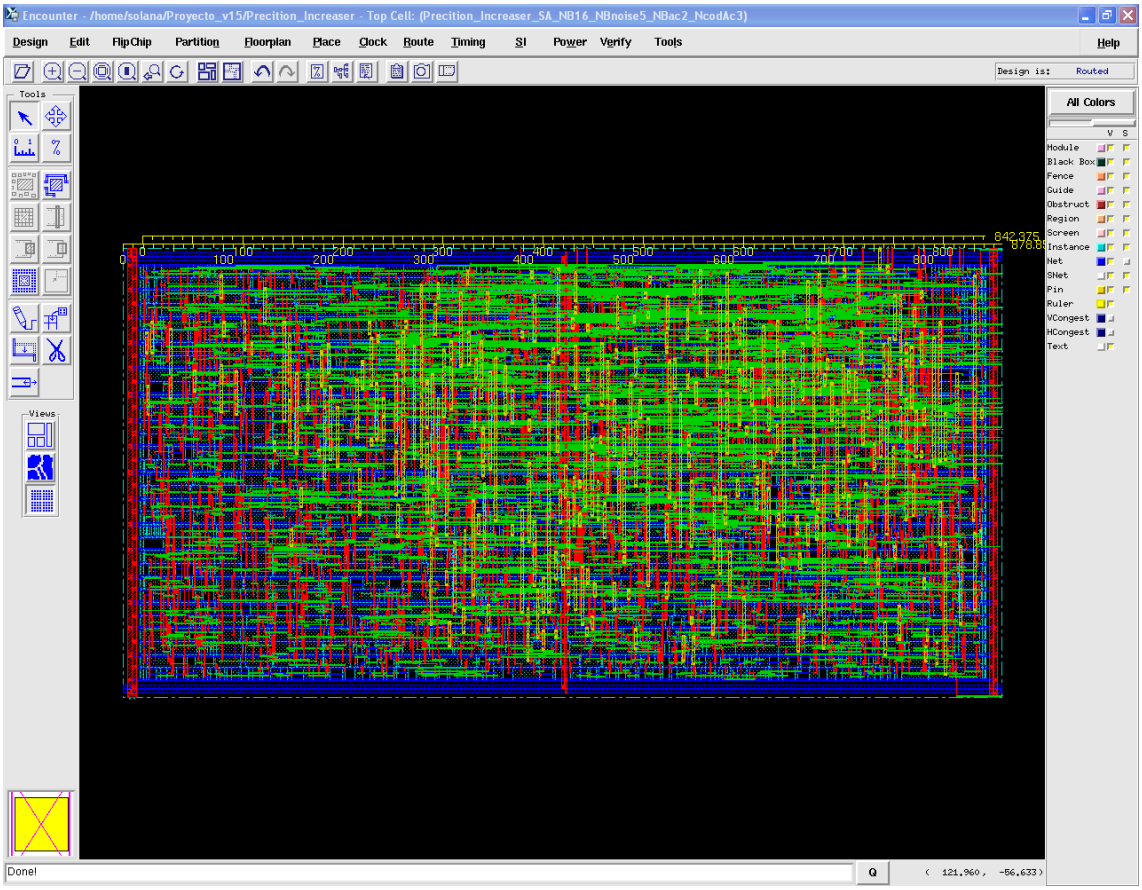


Figura 4.6-22. Vista del layout posterior al nano-enrutamiento.

En el ANEXO I se pueden ver una serie de informes generados después de la fase de enrutamiento. El primero de ellos es el más interesante ya que incluye datos como el número de celdas o líneas utilizadas.

4.6.3. Exportación del netlist a Virtuoso

El netlist que se importó para la fase de emplazamiento y enrutamiento se puede importar también desde la plataforma de Cadence, Virtuoso. Con esta aplicación se incluyó el diseño del circuito incrementador de la precisión (bloque IP) en otro diseño mayor, el del nuevo chip que incluye una nueva versión del sensor.

En la conexión al resto del chip se incluyó un multiplexor a la salida del bloque IP. Con este multiplexor se puede elegir entre la señal de salida de datos del bloque (es decir, la de precisión aumentada) o la señal de entrada de datos del mismo (es decir, sin incremento de precisión). Este multiplexor se utilizará en la fase de test del circuito para observar la el efecto de real del circuito diseñado en este PFC.

En la plataforma Virtuoso se realizaron algunas simulaciones para comprobar el correcto funcionamiento del bloque incrementador de la precisión. Se prestó especial atención a la capacidad de adaptación a los cambios de la señal de entrada (la medida angular), ya que la operación principal de este bloque debe ser esta, la de *adaptive-averaging*.

Las simulaciones fueron positivas y el diseño quedó conectado con el resto del chip y pendiente de su preparación para enviarlo a fábrica.

4.6.4. Test (pendiente)

Aunque en las expectativas iniciales del proyecto incluían la fase de test físico del circuito tras su fabricación, problemas presupuestarios en el laboratorio llevaron a un retraso obligado en las fechas de fabricación y entrega del chip. El retraso superó la duración de realización de este proyecto y, por tanto, no se han podido incluir los esperados test físicos sobre el chip.

5. Conclusiones

En la primera parte de esta memoria se ha presentado el sensor de posición angular basado efecto Hall. Un circuito integrado completo capaz de extraer el ángulo de un anillo Hall con 64 conexiones. En el que el campo magnético es medido mediante la detección del desfase de una señal analógica y un contador convierte directamente la fase en una salida digital. Hay que destacar la importancia de la técnica de *subspinning* que cancela el *offset* residual.

A continuación se realizó un estudio práctico de una versión de entrenamiento del sensor. Se realizaron diversas medidas, con las que se completo el estudio del sensor con el que se iba a trabajar.

Tras el estudio del sensor, se planteó la posibilidad de tratar digitalmente la señal para intentar incrementar su relación señal-a-ruido.

Finalmente se desarrolló un circuito capaz de estabilizar completamente una señal que llega con sus cinco bits menos significativos aplicados por un ruido que no se puede eliminar del sensor. Hay que recordar que este ruido está producido por el efecto de una anisotropía de 90° en la sensibilidad del anillo Hall. Y que esta anisotropía se crea en el proceso de fabricación. Esta anisotropía es prácticamente independiente de las variaciones en temperatura, por lo que podría aumentarse la precisión de la señal mediante la utilización de una simple curva de calibración tomada a temperatura ambiente. Pero un elemento que necesita calibración, necesita un mantenimiento y esto no siempre es posible. Las aplicaciones de un sensor como el aquí estudiado son muchísimas. Por ejemplo, medir la posición de apertura de un panel solar en un satélite que orbite a La Tierra, o gracias a su reducido tamaño puede utilizarse en la ruedecilla de un *mouse* de ordenador, etc. Y en muchas de las posibles aplicaciones, algunas de ellas aún por descubrir, sería muy útil un sensor que no necesite calibración.

Durante el desarrollo del proyecto se ha podido comprobar que la técnica de filtrado adaptativo resulta especialmente útil ante una señal de ancho de banda variable. Gracias al algoritmo implementado por la Unidad de Control del circuito, se consigue adaptar el ancho de banda del filtro al de la señal procedente del sensor (ver 4.3.3 Técnica Adaptive-Averaging).

Respecto a los resultados reales del circuito, hay que tener en cuenta que este proyecto se ha desarrollado en la Escuela Politécnica Federal de Lausana (EPFL) en Suiza, bajo un programa de intercambio Erasmus. Esto significa que el desarrollo del proyecto ha tenido una duración limitada. Desafortunadamente, el grupo de investigación LMSI3 de la Escuela de Microsistemas y Microelectronica de la EPFL tiene previsto disolverse en el año 2010, por lo que el presupuesto dedicado a nuevos proyectos ha sufrido un importante recorte. Como consecuencia, el chip que incluye el circuito diseñado en este PFC no ha sido fabricado hasta 2 meses después de terminarse la estancia. Y por tanto no pudo ser testado. Por lo tanto, las únicas conclusiones que se pueden sacar por el momento se limitan a los buenos resultados que se obtuvieron tanto con las simulaciones lógicas en Modelsim, como con los análisis post-síntesis efectuados con Encounter y, por supuesto, con las simulaciones realizadas con la

plataforma de Cadence, Virtuoso. En esta última plataforma se comprobó el comportamiento del circuito tras su inclusión en el netlist del chip que incluye a la nueva versión del sensor.

Una vez comprobada su utilidad en la mejora de la precisión del sensor fabricado, las ***futuras líneas de trabajo*** deberían ir por mejorar este circuito. Sería interesante estudiar la posibilidad de establecer más niveles intermedios de filtrado, para incrementar la sensibilidad de la parte adaptativa del sistema.

En el apartado 4.5-8 se comenta la implementación de la Unidad de Control (UC) del diseño, tal vez el módulo más complejo. Se comentaba que esta unidad debe ser idealmente asíncrona, pero que la multitud de eventos que intervienen en ella ha obligado a una implementación síncrona. Si se continuara con el desarrollo de este diseño, sería muy interesante investigar una forma de redefinir este módulo para poder implementarlo de forma asíncrona. Aunque tal vez esto obligue a cambiar el compilador de VHDL o el mismo lenguaje de programación. El sincronismo de esta unidad con la señal de reloj del sistema genera una salida errónea (durante un ciclo de reloj) cada vez que se debe disminuir el nivel de filtrado. Para evitar desestabilizar la señal se decidió introducir un retardo prudencial para la recuperación de los niveles superiores de filtrado. Este retardo podría ser disminuido o, incluso, eliminado si se consiguiera una implementación asíncrona de la UC.

Para el problema del *Zero-pass* (ver 4.3.1Problema del Zero-Pass y 4.5.5Bloque ZPcorrectSA), que se produce al calcular la media de valores angulares en el límite 0°-360°, se obtuvo una solución bastante satisfactoria, pero nada es perfecto y seguramente se podría llegar a desarrollar un algoritmo que implique la utilización de menos puertas lógicas. Sobre todo sería interesante investigar una solución realizable para el caso de una acumulación de muestras del tipo *semi-Sliding-Average*. Este método se descartó por este problema, pero si se consiguiera solucionar se podría reducir el número de registros, aproximadamente a un tercio de los utilizados actualmente, lo que tendría una repercusión muy positiva sobre el tamaño final del layout del circuito.

Por otra parte, el futuro cercano de un sensor como este pasa por ofrecer una medida de la velocidad de giro, además de la de posición. Si bien este cálculo puede realizarse de forma simple mediante un software externo, la utilidad potencial del sensor se vería enormemente incrementada integrando esta funcionalidad en el chip.

6. Referencias

[1] Vert-X Technology, Novotechnik U.S., Inc., http://www.novotechnik.com/technology/technology.html

[2] 1 C.Shott, R.Ratz, S.Huber, "*CMOS three axis Hall sensor and joystick application*", Procedimientos de Sensores IEEE, Viena, Austria, Oct.24-27,2004,vol.2,pp.9977-980

[3] "*Absolute Magnetic Rotary Encoder producers analog output*", Mayo, 2005, (model AS5043)

[4] http://www.allmagnetics.com/smco.htm

[5] Halbach K. Strong rare earth cobalt quadrupoles. IEEE Trans. Nucl. Sci 26, 3882 (1979).

[6] H. Raich and P. Bluemer *Design and construction of a dipolar Halbach array with a homogeneous field from identical bar magnets: NMR Mandhalas*, Concepts in Magnetic Resonance Part B **23B**, 16 (2004).

[7] P. Kejik, S. Reymond, R. S. Popovic, "Circular Hall Transducer for Angular Position Sensing", Transducer '07 & Eurosensors XXI conference, June 10-14, 2007, Lyon, France, Digest of technical papers vol.2, pp 2593-2596.

[8] Spinning current method of reducing the offset voltage of a hall device: http://www.freepatentsonline.com/6064202.html http://www.patentstorm.us/patents/6064202.html

[9] Spinning-current method for offset reduction in silicon Hall plates, by Munter, Peter Jan Adriaan, Ph.D. Thesis Technische Univ., Delft (Netherlands). Electronic Instrumentation Lab.: http://adsabs.harvard.edu/abs/1992PhDT.....26M

[10] Sitio web de AustriaMicrosystems: http://www.austriamicrosystems.com/

[11] "*Start circuit for a bandgap reference cell*" – http://www.freepatentsonline.com/5087830.html

[12] "*Band gap voltage reference circuit*" – http://www.patentstorm.us/patents/4524318/claims.html

[13] http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=ICDSE7000141000006000510000001&idtype=cvips&gifs=yes

[14] http://www.freepatentsonline.com/EP0221290.html

[15] http://www.faqs.org/patents/app/20080252358

[16] http://blogspinner.countwordula.com/2006/02/sliding-average-explanation.html

[17] Pavel Kejik, Serge Reymond, Radivoje S. Popovic, "*Purely CMOS Angular Position Sensor Based on a New Hall Microchip*", École Polytechnique Fédérale de Lausanne.

[18] S. Reymond, P. Kejik, R. S. Popovic, "*True 2D CMOS integrated Hall sensor*", Swiss Federal Institute of Technology - Lausanne, Switzerland.

[19] P. Kejik, S. Reymond, R. S. Popovic, "*CIRCULAR HALL TRANSDUCER ANGULAR POSITION SENSING*", École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.

[20] Alain Vachoux, "*Top-down digital design flow*", Version 3.1 (November 2006), Microelectronic Systems Lab STI-IMM-LSM, EPFL.

[21] SYNOPSYS, "*FPGA COMPILER II / FPGA EXPRESS, VHDL Reference Manual*", Version 1999.05, May 1999.

7. ANEXOS

ANEXO I. Informes *post-routing*

Precition_increaser_mapped_area.rtp

```
****************************************
Report : area
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Mon Jul 20 21:02:58 2009
****************************************

Library(s) Used:

    c35_CORELIB (File: /dkits/ams/hk371/synopsys/h35_3.3V/h35_CORELIB.db)

Number of ports:               41
Number of nets:               261
Number of cells:               18
Number of references:          18

Combinational area:       159613.890625
Noncombinational area:    151077.531250
Net Interconnect area:    41022.000000

Total cell area:          310692.187500
Total area:               351714.187500
Information: Updating design information... (UID-85)

****************************************
Report : area
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Tue Jul 21 12:12:58 2009
****************************************

Library(s) Used:

    c35_CORELIB (File: /dkits/ams/hk371/synopsys/h35_3.3V/h35_CORELIB.db)

Number of ports:               41
Number of nets:               261
Number of cells:               18
Number of references:          18

Combinational area:       159613.890625
Noncombinational area:    151077.531250
Net Interconnect area:    41022.000000

Total cell area:          310692.187500
Total area:               351714.187500
```

Precition_increaser_mapped_resources.rtp

```
****************************************
Report : resources
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Mon Jul 20 21:08:13 2009
****************************************


No resource sharing information to report.

No implementations to report

No multiplexors to report

****************************************
Report : resources
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Tue Jul 21 12:16:30 2009
****************************************


No resource sharing information to report.

No implementations to report

No multiplexors to report
```

Precition_increaser_mapped_timming.rtp

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
        -sort_by group
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Mon Jul 20 21:06:11 2009
****************************************

Operating Conditions: WORST-IND   Library: c35_CORELIB
Wire Load Model Mode: enclosed

  Startpoint: reg0/s_out_reg[14]
              (rising edge-triggered flip-flop clocked by DV)
  Endpoint: ac5/reg1/s_out_reg[23]
            (rising edge-triggered flip-flop clocked by DV)
  Path Group: DV
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3 10k c35_CORELIB
  ZPcorrect4SA_0     10k                   c35_CORELIB
  Ac4xv3_SA_NB16_NBac2 10k                 c35_CORELIB
  Ac4xv3_SA_NB16_NBac2_DW01_add_0 10k      c35_CORELIB
  ZPcorrect4SA_4     10k                   c35_CORELIB
  Ac4xv3_SA_NB18_NBac2 10k                 c35_CORELIB
  Ac4xv3_SA_NB18_NBac2_DW01_add_0 10k      c35_CORELIB
  ZPcorrect4SA_3     10k                   c35_CORELIB
  Ac4xv3_SA_NB20_NBac2 10k                 c35_CORELIB
  Ac4xv3_SA_NB20_NBac2_DW01_add_1 10k      c35_CORELIB
  Ac4xv3_SA_NB20_NBac2_DW01_add_0 10k      c35_CORELIB
```

```
  ZPcorrect4SA_2     10k                    c35_CORELIB
  Ac4xv3_SA_NB22_NBac2 10k                  c35_CORELIB
  Ac4xv3_SA_NB22_NBac2_DW01_add_1 10k       c35_CORELIB
  Ac4xv3_SA_NB22_NBac2_DW01_add_0 10k       c35_CORELIB

  Point                                                    Incr       Path
  ------------------------------------------------------------------------
  clock DV (rise edge)                                   500.00     500.00
  clock network delay (ideal)                              0.00     500.00
  reg0/s_out_reg[14]/C (DFEC1)                             0.00     500.00 r
  reg0/s_out_reg[14]/Q (DFEC1)                             2.30     502.30 r
  reg0/s_out[14] (Reg_NB16_0)                              0.00     502.30 r
  ac1/s_in[14] (Ac4xv3_SA_NB16_NBac2)                      0.00     502.30 r
  ac1/correct/n1[0] (ZPcorrect4SA_0)                       0.00     502.30 r
  ac1/correct/U2/Q (INV1)                                  1.00     503.30 f
  ac1/correct/U5/Q (XNR21)                                 1.32     504.62 f
  ac1/correct/U8/Q (XOR21)                                 0.69     505.32 f
  ac1/correct/U7/Q (NAND22)                                0.37     505.68 r
  ac1/correct/U6/Q (INV3)                                  0.16     505.84 f
  ac1/correct/correc[0] (ZPcorrect4SA_0)                   0.00     505.84 f
  ac1/U44/Q (NAND20)                                       0.50     506.34 r
  ac1/U46/Q (CLKIN0)                                       0.73     507.08 f
  ac1/add_3_root_add_0_root_add_39/U1_1/CO (ADD32)         0.89     507.97 f
  ac1/U41/Q (NAND20)                                       0.53     508.50 r
  ac1/U43/Q (CLKIN0)                                       0.68     509.18 f
  ac1/U38/Q (NAND20)                                       0.82     509.99 r
  ac1/U40/Q (CLKIN0)                                       0.71     510.70 f
  ac1/U35/Q (NAND20)                                       0.83     511.54 r
  ac1/U37/Q (CLKIN0)                                       0.71     512.24 f
  ac1/U32/Q (NAND20)                                       0.84     513.08 r
  ac1/U34/Q (CLKIN0)                                       0.71     513.79 f
  ac1/U29/Q (NAND20)                                       0.84     514.63 r
  ac1/U31/Q (CLKIN0)                                       0.71     515.33 f
  ac1/U26/Q (NAND20)                                       0.84     516.17 r
  ac1/U28/Q (CLKIN0)                                       0.71     516.88 f
  ac1/U23/Q (NAND20)                                       0.84     517.72 r
  ac1/U25/Q (CLKIN0)                                       0.71     518.42 f
  ac1/U20/Q (NAND20)                                       0.84     519.26 r
  ac1/U22/Q (CLKIN0)                                       0.71     519.97 f
  ac1/U17/Q (NAND20)                                       0.84     520.80 r
  ac1/U19/Q (CLKIN0)                                       0.71     521.51 f
  ac1/U14/Q (NAND20)                                       0.84     522.35 r
  ac1/U16/Q (CLKIN0)                                       0.71     523.06 f
  ac1/U11/Q (NAND20)                                       0.84     523.89 r
  ac1/U13/Q (CLKIN0)                                       0.71     524.60 f
  ac1/U8/Q (NAND20)                                        0.84     525.44 r
  ac1/U10/Q (CLKIN0)                                       0.71     526.15 f
  ac1/U5/Q (NAND20)                                        0.84     526.98 r
  ac1/U7/Q (CLKIN0)                                        0.71     527.69 f
  ac1/U3/Q (XOR20)                                         1.28     528.97 f
  ac1/add_0_root_add_0_root_add_39/B[15] (Ac4xv3_SA_NB16_NBac2_DW01_add_0)
0.00   528.97 f
  ac1/add_0_root_add_0_root_add_39/U1_15/CO (ADD32)        0.99     529.96 f
  ac1/add_0_root_add_0_root_add_39/U1_16/S (ADD32)         1.52     531.49 r
  ac1/add_0_root_add_0_root_add_39/SUM[16] (Ac4xv3_SA_NB16_NBac2_DW01_add_0)
0.00   531.49 r
  ac1/s_out[16] (Ac4xv3_SA_NB16_NBac2)                     0.00     531.49 r
  ac2/s_in[16] (Ac4xv3_SA_NB18_NBac2)                      0.00     531.49 r
  ac2/correct/n1[0] (ZPcorrect4SA_4)                       0.00     531.49 r
  ac2/correct/U7/Q (INV3)                                  0.28     531.77 f
  ac2/correct/U2/Q (XNR20)                                 0.84     532.61 r
  ac2/correct/U14/Q (NAND41)                               0.15     532.75 f
  ac2/correct/U5/Q (NAND22)                                0.40     533.15 r
  ac2/correct/U4/Q (INV3)                                  0.16     533.31 f
  ac2/correct/correc[0] (ZPcorrect4SA_4)                   0.00     533.31 f
  ac2/U50/Q (NAND20)                                       0.50     533.81 r
  ac2/U52/Q (CLKIN0)                                       0.73     534.55 f
  ac2/add_3_root_add_0_root_add_39/U1_1/CO (ADD32)         0.89     535.44 f
```

```
  ac2/U47/Q (NAND20)                                      0.53     535.97 r
  ac2/U49/Q (CLKIN0)                                      0.68     536.65 f
  ac2/U44/Q (NAND20)                                      0.82     537.46 r
  ac2/U46/Q (CLKIN0)                                      0.71     538.17 f
  ac2/U41/Q (NAND20)                                      0.83     539.00 r
  ac2/U43/Q (CLKIN0)                                      0.71     539.71 f
  ac2/U38/Q (NAND20)                                      0.84     540.55 r
  ac2/U40/Q (CLKIN0)                                      0.71     541.26 f
  ac2/U35/Q (NAND20)                                      0.84     542.09 r
  ac2/U37/Q (CLKIN0)                                      0.71     542.80 f
  ac2/U32/Q (NAND20)                                      0.84     543.64 r
  ac2/U34/Q (CLKIN0)                                      0.71     544.35 f
  ac2/U29/Q (NAND20)                                      0.84     545.18 r
  ac2/U31/Q (CLKIN0)                                      0.71     545.89 f
  ac2/U26/Q (NAND20)                                      0.84     546.73 r
  ac2/U28/Q (CLKIN0)                                      0.71     547.44 f
  ac2/U23/Q (NAND20)                                      0.84     548.27 r
  ac2/U25/Q (CLKIN0)                                      0.71     548.98 f
  ac2/U20/Q (NAND20)                                      0.84     549.82 r
  ac2/U22/Q (CLKIN0)                                      0.71     550.53 f
  ac2/U17/Q (NAND20)                                      0.84     551.36 r
  ac2/U19/Q (CLKIN0)                                      0.71     552.07 f
  ac2/U14/Q (NAND20)                                      0.84     552.91 r
  ac2/U16/Q (CLKIN0)                                      0.71     553.62 f
  ac2/U11/Q (NAND20)                                      0.84     554.45 r
  ac2/U13/Q (CLKIN0)                                      0.71     555.16 f
  ac2/U8/Q (NAND20)                                       0.84     556.00 r
  ac2/U10/Q (CLKIN0)                                      0.71     556.71 f
  ac2/U5/Q (NAND20)                                       0.84     557.54 r
  ac2/U7/Q (CLKIN0)                                       0.71     558.25 f
  ac2/U3/Q (XOR20)                                        1.28     559.53 f
  ac2/add_0_root_add_0_root_add_39/B[17] (Ac4xv3_SA_NB18_NBac2_DW01_add_0)
0.00   559.53 f
  ac2/add_0_root_add_0_root_add_39/U1_17/CO (ADD32)       0.99     560.52 f
  ac2/add_0_root_add_0_root_add_39/U1_18/S (ADD32)        1.52     562.04 r
  ac2/add_0_root_add_0_root_add_39/SUM[18] (Ac4xv3_SA_NB18_NBac2_DW01_add_0)
0.00   562.04 r
  ac2/s_out[18] (Ac4xv3_SA_NB18_NBac2)                    0.00     562.04 r
  ac3/s_in[18] (Ac4xv3_SA_NB20_NBac2)                     0.00     562.04 r
  ac3/correct/n1[0] (ZPcorrect4SA_3)                      0.00     562.04 r
  ac3/correct/U7/Q (INV3)                                 0.28     562.33 f
  ac3/correct/U2/Q (XNR20)                                0.84     563.16 r
  ac3/correct/U14/Q (NAND41)                              0.15     563.31 f
  ac3/correct/U5/Q (NAND22)                               0.40     563.71 r
  ac3/correct/U4/Q (INV3)                                 0.15     563.86 f
  ac3/correct/correc[0] (ZPcorrect4SA_3)                  0.00     563.86 f
  ac3/add_2_root_add_0_root_add_39/B[0] (Ac4xv3_SA_NB20_NBac2_DW01_add_1)
0.00   563.86 f
  ac3/add_2_root_add_0_root_add_39/U55/Q (NAND20)         0.44     564.30 r
  ac3/add_2_root_add_0_root_add_39/U57/Q (CLKIN0)         0.75     565.05 f
  ac3/add_2_root_add_0_root_add_39/U1_1/CO (ADD32)        0.88     565.93 f
  ac3/add_2_root_add_0_root_add_39/U52/Q (NAND20)         0.47     566.40 r
  ac3/add_2_root_add_0_root_add_39/U54/Q (CLKIN0)         0.64     567.04 f
  ac3/add_2_root_add_0_root_add_39/U49/Q (NAND20)         0.73     567.76 r
  ac3/add_2_root_add_0_root_add_39/U51/Q (CLKIN0)         0.64     568.40 f
  ac3/add_2_root_add_0_root_add_39/U46/Q (NAND20)         0.73     569.13 r
  ac3/add_2_root_add_0_root_add_39/U48/Q (CLKIN0)         0.64     569.76 f
  ac3/add_2_root_add_0_root_add_39/U43/Q (NAND20)         0.73     570.49 r
  ac3/add_2_root_add_0_root_add_39/U45/Q (CLKIN0)         0.64     571.12 f
  ac3/add_2_root_add_0_root_add_39/U40/Q (NAND20)         0.73     571.85 r
  ac3/add_2_root_add_0_root_add_39/U42/Q (CLKIN0)         0.64     572.49 f
  ac3/add_2_root_add_0_root_add_39/U37/Q (NAND20)         0.73     573.21 r
  ac3/add_2_root_add_0_root_add_39/U39/Q (CLKIN0)         0.64     573.85 f
  ac3/add_2_root_add_0_root_add_39/U34/Q (NAND20)         0.73     574.57 r
  ac3/add_2_root_add_0_root_add_39/U36/Q (CLKIN0)         0.64     575.21 f
  ac3/add_2_root_add_0_root_add_39/U31/Q (NAND20)         0.73     575.94 r
  ac3/add_2_root_add_0_root_add_39/U33/Q (CLKIN0)         0.64     576.57 f
  ac3/add_2_root_add_0_root_add_39/U28/Q (NAND20)         0.73     577.30 r
```

```
  ac3/add_2_root_add_0_root_add_39/U30/Q (CLKIN0)          0.64     577.93 f
  ac3/add_2_root_add_0_root_add_39/U25/Q (NAND20)          0.73     578.66 r
  ac3/add_2_root_add_0_root_add_39/U27/Q (CLKIN0)          0.64     579.30 f
  ac3/add_2_root_add_0_root_add_39/U22/Q (NAND20)          0.73     580.02 r
  ac3/add_2_root_add_0_root_add_39/U24/Q (CLKIN0)          0.64     580.66 f
  ac3/add_2_root_add_0_root_add_39/U19/Q (NAND20)          0.73     581.39 r
  ac3/add_2_root_add_0_root_add_39/U21/Q (CLKIN0)          0.64     582.02 f
  ac3/add_2_root_add_0_root_add_39/U16/Q (NAND20)          0.73     582.75 r
  ac3/add_2_root_add_0_root_add_39/U18/Q (CLKIN0)          0.64     583.38 f
  ac3/add_2_root_add_0_root_add_39/U13/Q (NAND20)          0.73     584.11 r
  ac3/add_2_root_add_0_root_add_39/U15/Q (CLKIN0)          0.64     584.75 f
  ac3/add_2_root_add_0_root_add_39/U10/Q (NAND20)          0.73     585.47 r
  ac3/add_2_root_add_0_root_add_39/U12/Q (CLKIN0)          0.64     586.11 f
  ac3/add_2_root_add_0_root_add_39/U7/Q (NAND20)           0.73     586.83 r
  ac3/add_2_root_add_0_root_add_39/U9/Q (CLKIN0)           0.64     587.47 f
  ac3/add_2_root_add_0_root_add_39/U4/Q (NAND20)           0.73     588.20 r
  ac3/add_2_root_add_0_root_add_39/U6/Q (CLKIN0)           0.64     588.84 f
  ac3/add_2_root_add_0_root_add_39/U2/Q (XOR20)            1.17     590.01 f
  ac3/add_2_root_add_0_root_add_39/SUM[19] (Ac4xv3_SA_NB20_NBac2_DW01_add_1)
0.00   590.01 f
  ac3/add_0_root_add_0_root_add_39/B[19] (Ac4xv3_SA_NB20_NBac2_DW01_add_0)
0.00   590.01 f
  ac3/add_0_root_add_0_root_add_39/U1_19/CO (ADD32)        0.99     591.01 f
  ac3/add_0_root_add_0_root_add_39/U1_20/S (ADD32)         1.55     592.55 r
  ac3/add_0_root_add_0_root_add_39/SUM[20] (Ac4xv3_SA_NB20_NBac2_DW01_add_0)
0.00   592.55 r
  ac3/s_out[20] (Ac4xv3_SA_NB20_NBac2)                     0.00     592.55 r
  ac4/s_in[20] (Ac4xv3_SA_NB22_NBac2)                      0.00     592.55 r
  ac4/correct/n1[0] (ZPcorrect4SA_2)                       0.00     592.55 r
  ac4/correct/U9/Q (INV3)                                  0.28     592.83 f
  ac4/correct/U2/Q (XNR20)                                 0.84     593.67 r
  ac4/correct/U16/Q (NAND41)                               0.15     593.82 f
  ac4/correct/U7/Q (NAND22)                                0.40     594.22 r
  ac4/correct/U6/Q (INV3)                                  0.15     594.36 f
  ac4/correct/correc[0] (ZPcorrect4SA_2)                   0.00     594.36 f
  ac4/add_1_root_add_0_root_add_39/B[0] (Ac4xv3_SA_NB22_NBac2_DW01_add_1)
0.00   594.36 f
  ac4/add_1_root_add_0_root_add_39/U65/Q (NAND20)          0.44     594.81 r
  ac4/add_1_root_add_0_root_add_39/U67/Q (CLKIN0)          0.76     595.57 f
  ac4/add_1_root_add_0_root_add_39/U1_1/CO (ADD32)         0.88     596.45 f
  ac4/add_1_root_add_0_root_add_39/U62/Q (NAND20)          0.47     596.92 r
  ac4/add_1_root_add_0_root_add_39/U64/Q (CLKIN0)          0.62     597.54 f
  ac4/add_1_root_add_0_root_add_39/U59/Q (NAND20)          0.72     598.26 r
  ac4/add_1_root_add_0_root_add_39/U61/Q (CLKIN0)          0.63     598.90 f
  ac4/add_1_root_add_0_root_add_39/U56/Q (NAND20)          0.72     599.62 r
  ac4/add_1_root_add_0_root_add_39/U58/Q (CLKIN0)          0.63     600.25 f
  ac4/add_1_root_add_0_root_add_39/U53/Q (NAND20)          0.72     600.98 r
  ac4/add_1_root_add_0_root_add_39/U55/Q (CLKIN0)          0.63     601.61 f
  ac4/add_1_root_add_0_root_add_39/U50/Q (NAND20)          0.72     602.33 r
  ac4/add_1_root_add_0_root_add_39/U52/Q (CLKIN0)          0.63     602.97 f
  ac4/add_1_root_add_0_root_add_39/U47/Q (NAND20)          0.72     603.69 r
  ac4/add_1_root_add_0_root_add_39/U49/Q (CLKIN0)          0.63     604.32 f
  ac4/add_1_root_add_0_root_add_39/U44/Q (NAND20)          0.72     605.05 r
  ac4/add_1_root_add_0_root_add_39/U46/Q (CLKIN0)          0.63     605.68 f
  ac4/add_1_root_add_0_root_add_39/U41/Q (NAND20)          0.72     606.41 r
  ac4/add_1_root_add_0_root_add_39/U43/Q (CLKIN0)          0.63     607.04 f
  ac4/add_1_root_add_0_root_add_39/U38/Q (NAND20)          0.72     607.76 r
  ac4/add_1_root_add_0_root_add_39/U40/Q (CLKIN0)          0.63     608.40 f
  ac4/add_1_root_add_0_root_add_39/U35/Q (NAND20)          0.72     609.12 r
  ac4/add_1_root_add_0_root_add_39/U37/Q (CLKIN0)          0.63     609.75 f
  ac4/add_1_root_add_0_root_add_39/U32/Q (NAND20)          0.72     610.48 r
  ac4/add_1_root_add_0_root_add_39/U34/Q (CLKIN0)          0.63     611.11 f
  ac4/add_1_root_add_0_root_add_39/U29/Q (NAND20)          0.72     611.83 r
  ac4/add_1_root_add_0_root_add_39/U31/Q (CLKIN0)          0.63     612.47 f
  ac4/add_1_root_add_0_root_add_39/U26/Q (NAND20)          0.72     613.19 r
  ac4/add_1_root_add_0_root_add_39/U28/Q (CLKIN0)          0.63     613.82 f
  ac4/add_1_root_add_0_root_add_39/U23/Q (NAND20)          0.72     614.55 r
  ac4/add_1_root_add_0_root_add_39/U25/Q (CLKIN0)          0.63     615.18 f
```

```
  ac4/add_1_root_add_0_root_add_39/U20/Q (NAND20)          0.72     615.91 r
  ac4/add_1_root_add_0_root_add_39/U22/Q (CLKIN0)          0.63     616.54 f
  ac4/add_1_root_add_0_root_add_39/U17/Q (NAND20)          0.72     617.26 r
  ac4/add_1_root_add_0_root_add_39/U19/Q (CLKIN0)          0.63     617.90 f
  ac4/add_1_root_add_0_root_add_39/U14/Q (NAND20)          0.72     618.62 r
  ac4/add_1_root_add_0_root_add_39/U16/Q (CLKIN0)          0.63     619.25 f
  ac4/add_1_root_add_0_root_add_39/U11/Q (NAND20)          0.72     619.98 r
  ac4/add_1_root_add_0_root_add_39/U13/Q (CLKIN0)          0.63     620.61 f
  ac4/add_1_root_add_0_root_add_39/U8/Q (NAND20)           0.72     621.34 r
  ac4/add_1_root_add_0_root_add_39/U10/Q (CLKIN0)          0.63     621.97 f
  ac4/add_1_root_add_0_root_add_39/U5/Q (NAND20)           0.72     622.69 r
  ac4/add_1_root_add_0_root_add_39/U7/Q (CLKIN0)           0.63     623.32 f
  ac4/add_1_root_add_0_root_add_39/U3/Q (XOR20)            1.04     624.36 f
  ac4/add_1_root_add_0_root_add_39/SUM[22] (Ac4xv3_SA_NB22_NBac2_DW01_add_1)
0.00   624.36 f
  ac4/add_0_root_add_0_root_add_39/B[22] (Ac4xv3_SA_NB22_NBac2_DW01_add_0)
0.00   624.36 f
  ac4/add_0_root_add_0_root_add_39/U4/Q (NAND20)           0.70     625.06 r
  ac4/add_0_root_add_0_root_add_39/U5/Q (CLKIN0)           0.57     625.64 f
  ac4/add_0_root_add_0_root_add_39/U2/Q (XOR21)            1.42     627.05 r
  ac4/add_0_root_add_0_root_add_39/SUM[23] (Ac4xv3_SA_NB22_NBac2_DW01_add_0)
0.00   627.05 r
  ac4/s_out[23] (Ac4xv3_SA_NB22_NBac2)                     0.00     627.05 r
  ac5/s_in[23] (Ac4xv3_SA_NB24_NBac2)                      0.00     627.05 r
  ac5/reg1/s_in[23] (Reg_NB24_3)                           0.00     627.05 r
  ac5/reg1/s_out_reg[23]/D (DFEC1)                         0.00     627.05 r
  data arrival time                                                 627.05

  clock DV (rise edge)                                  1500.00    1500.00
  clock network delay (ideal)                              0.00    1500.00
  ac5/reg1/s_out_reg[23]/C (DFEC1)                         0.00    1500.00 r
  library setup time                                      -0.14    1499.86
  data required time                                               1499.86
  ---------------------------------------------------------------------------
  data required time                                               1499.86
  data arrival time                                                -627.05
  ---------------------------------------------------------------------------
  slack (MET)                                                       872.81



****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
        -sort_by group
Design : Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3
Version: X-2005.09-SP4
Date   : Tue Jul 21 12:15:01 2009
****************************************

Operating Conditions: WORST-IND   Library: c35_CORELIB
Wire Load Model Mode: enclosed

  Startpoint: reg0/s_out_reg[14]
              (rising edge-triggered flip-flop clocked by DV)
  Endpoint: ac5/reg1/s_out_reg[23]
            (rising edge-triggered flip-flop clocked by DV)
  Path Group: DV
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  Precition_Increaser_SA_NB16_NBnoise5_NBac2_NcodAc3 10k c35_CORELIB
  ZPcorrect4SA_0     10k                   c35_CORELIB
  Ac4xv3_SA_NB16_NBac2 10k                 c35_CORELIB
  Ac4xv3_SA_NB16_NBac2_DW01_add_0 10k      c35_CORELIB
```

```
  ZPcorrect4SA_4     10k                    c35_CORELIB
  Ac4xv3_SA_NB18_NBac2 10k                  c35_CORELIB
  Ac4xv3_SA_NB18_NBac2_DW01_add_0 10k       c35_CORELIB
  ZPcorrect4SA_3     10k                    c35_CORELIB
  Ac4xv3_SA_NB20_NBac2 10k                  c35_CORELIB
  Ac4xv3_SA_NB20_NBac2_DW01_add_1 10k       c35_CORELIB
  Ac4xv3_SA_NB20_NBac2_DW01_add_0 10k       c35_CORELIB
  ZPcorrect4SA_2     10k                    c35_CORELIB
  Ac4xv3_SA_NB22_NBac2 10k                  c35_CORELIB
  Ac4xv3_SA_NB22_NBac2_DW01_add_1 10k       c35_CORELIB
  Ac4xv3_SA_NB22_NBac2_DW01_add_0 10k       c35_CORELIB

  Point                                                   Incr       Path
  ------------------------------------------------------------------------
  clock DV (rise edge)                                  500.00     500.00
  clock network delay (ideal)                             0.00     500.00
  reg0/s_out_reg[14]/C (DFEC1)                            0.00     500.00 r
  reg0/s_out_reg[14]/Q (DFEC1)                            2.30     502.30 r
  reg0/s_out[14] (Reg_NB16_0)                             0.00     502.30 r
  ac1/s_in[14] (Ac4xv3_SA_NB16_NBac2)                     0.00     502.30 r
  ac1/correct/n1[0] (ZPcorrect4SA_0)                      0.00     502.30 r
  ac1/correct/U2/Q (INV1)                                 1.00     503.30 f
  ac1/correct/U5/Q (XNR21)                                1.32     504.62 f
  ac1/correct/U8/Q (XOR21)                                0.69     505.32 f
  ac1/correct/U7/Q (NAND22)                               0.37     505.68 r
  ac1/correct/U6/Q (INV3)                                 0.16     505.84 f
  ac1/correct/correc[0] (ZPcorrect4SA_0)                  0.00     505.84 f
  ac1/U44/Q (NAND20)                                      0.50     506.34 r
  ac1/U46/Q (CLKIN0)                                      0.73     507.08 f
  ac1/add_3_root_add_0_root_add_39/U1_1/CO (ADD32)        0.89     507.97 f
  ac1/U41/Q (NAND20)                                      0.53     508.50 r
  ac1/U43/Q (CLKIN0)                                      0.68     509.18 f
  ac1/U38/Q (NAND20)                                      0.82     509.99 r
  ac1/U40/Q (CLKIN0)                                      0.71     510.70 f
  ac1/U35/Q (NAND20)                                      0.83     511.54 r
  ac1/U37/Q (CLKIN0)                                      0.71     512.24 f
  ac1/U32/Q (NAND20)                                      0.84     513.08 r
  ac1/U34/Q (CLKIN0)                                      0.71     513.79 f
  ac1/U29/Q (NAND20)                                      0.84     514.63 r
  ac1/U31/Q (CLKIN0)                                      0.71     515.33 f
  ac1/U26/Q (NAND20)                                      0.84     516.17 r
  ac1/U28/Q (CLKIN0)                                      0.71     516.88 f
  ac1/U23/Q (NAND20)                                      0.84     517.72 r
  ac1/U25/Q (CLKIN0)                                      0.71     518.42 f
  ac1/U20/Q (NAND20)                                      0.84     519.26 r
  ac1/U22/Q (CLKIN0)                                      0.71     519.97 f
  ac1/U17/Q (NAND20)                                      0.84     520.80 r
  ac1/U19/Q (CLKIN0)                                      0.71     521.51 f
  ac1/U14/Q (NAND20)                                      0.84     522.35 r
  ac1/U16/Q (CLKIN0)                                      0.71     523.06 f
  ac1/U11/Q (NAND20)                                      0.84     523.89 r
  ac1/U13/Q (CLKIN0)                                      0.71     524.60 f
  ac1/U8/Q (NAND20)                                       0.84     525.44 r
  ac1/U10/Q (CLKIN0)                                      0.71     526.15 f
  ac1/U5/Q (NAND20)                                       0.84     526.98 r
  ac1/U7/Q (CLKIN0)                                       0.71     527.69 f
  ac1/U3/Q (XOR20)                                        1.28     528.97 f
  ac1/add_0_root_add_0_root_add_39/B[15] (Ac4xv3_SA_NB16_NBac2_DW01_add_0)
0.00   528.97 f
  ac1/add_0_root_add_0_root_add_39/U1_15/CO (ADD32)       0.99     529.96 f
  ac1/add_0_root_add_0_root_add_39/U1_16/S (ADD32)        1.52     531.49 r
  ac1/add_0_root_add_0_root_add_39/SUM[16] (Ac4xv3_SA_NB16_NBac2_DW01_add_0)
0.00   531.49 r
  ac1/s_out[16] (Ac4xv3_SA_NB16_NBac2)                    0.00     531.49 r
  ac2/s_in[16] (Ac4xv3_SA_NB18_NBac2)                     0.00     531.49 r
  ac2/correct/n1[0] (ZPcorrect4SA_4)                      0.00     531.49 r
  ac2/correct/U7/Q (INV3)                                 0.28     531.77 f
  ac2/correct/U2/Q (XNR20)                                0.84     532.61 r
```

```
  ac2/correct/U14/Q (NAND41)                                0.15     532.75 f
  ac2/correct/U5/Q (NAND22)                                 0.40     533.15 r
  ac2/correct/U4/Q (INV3)                                   0.16     533.31 f
  ac2/correct/correc[0] (ZPcorrect4SA_4)                    0.00     533.31 f
  ac2/U50/Q (NAND20)                                        0.50     533.81 r
  ac2/U52/Q (CLKIN0)                                        0.73     534.55 f
  ac2/add_3_root_add_0_root_add_39/U1_1/CO (ADD32)          0.89     535.44 f
  ac2/U47/Q (NAND20)                                        0.53     535.97 r
  ac2/U49/Q (CLKIN0)                                        0.68     536.65 f
  ac2/U44/Q (NAND20)                                        0.82     537.46 r
  ac2/U46/Q (CLKIN0)                                        0.71     538.17 f
  ac2/U41/Q (NAND20)                                        0.83     539.00 r
  ac2/U43/Q (CLKIN0)                                        0.71     539.71 f
  ac2/U38/Q (NAND20)                                        0.84     540.55 r
  ac2/U40/Q (CLKIN0)                                        0.71     541.26 f
  ac2/U35/Q (NAND20)                                        0.84     542.09 r
  ac2/U37/Q (CLKIN0)                                        0.71     542.80 f
  ac2/U32/Q (NAND20)                                        0.84     543.64 r
  ac2/U34/Q (CLKIN0)                                        0.71     544.35 f
  ac2/U29/Q (NAND20)                                        0.84     545.18 r
  ac2/U31/Q (CLKIN0)                                        0.71     545.89 f
  ac2/U26/Q (NAND20)                                        0.84     546.73 r
  ac2/U28/Q (CLKIN0)                                        0.71     547.44 f
  ac2/U23/Q (NAND20)                                        0.84     548.27 r
  ac2/U25/Q (CLKIN0)                                        0.71     548.98 f
  ac2/U20/Q (NAND20)                                        0.84     549.82 r
  ac2/U22/Q (CLKIN0)                                        0.71     550.53 f
  ac2/U17/Q (NAND20)                                        0.84     551.36 r
  ac2/U19/Q (CLKIN0)                                        0.71     552.07 f
  ac2/U14/Q (NAND20)                                        0.84     552.91 r
  ac2/U16/Q (CLKIN0)                                        0.71     553.62 f
  ac2/U11/Q (NAND20)                                        0.84     554.45 r
  ac2/U13/Q (CLKIN0)                                        0.71     555.16 f
  ac2/U8/Q (NAND20)                                         0.84     556.00 r
  ac2/U10/Q (CLKIN0)                                        0.71     556.71 f
  ac2/U5/Q (NAND20)                                         0.84     557.54 r
  ac2/U7/Q (CLKIN0)                                         0.71     558.25 f
  ac2/U3/Q (XOR20)                                          1.28     559.53 f
  ac2/add_0_root_add_0_root_add_39/B[17] (Ac4xv3_SA_NB18_NBac2_DW01_add_0)
0.00   559.53 f
  ac2/add_0_root_add_0_root_add_39/U1_17/CO (ADD32)         0.99     560.52 f
  ac2/add_0_root_add_0_root_add_39/U1_18/S (ADD32)          1.52     562.04 r
  ac2/add_0_root_add_0_root_add_39/SUM[18] (Ac4xv3_SA_NB18_NBac2_DW01_add_0)
0.00   562.04 r
  ac2/s_out[18] (Ac4xv3_SA_NB18_NBac2)                      0.00     562.04 r
  ac3/s_in[18] (Ac4xv3_SA_NB20_NBac2)                       0.00     562.04 r
  ac3/correct/n1[0] (ZPcorrect4SA_3)                        0.00     562.04 r
  ac3/correct/U7/Q (INV3)                                   0.28     562.33 f
  ac3/correct/U2/Q (XNR20)                                  0.84     563.16 r
  ac3/correct/U14/Q (NAND41)                                0.15     563.31 f
  ac3/correct/U5/Q (NAND22)                                 0.40     563.71 r
  ac3/correct/U4/Q (INV3)                                   0.15     563.86 f
  ac3/correct/correc[0] (ZPcorrect4SA_3)                    0.00     563.86 f
  ac3/add_2_root_add_0_root_add_39/B[0] (Ac4xv3_SA_NB20_NBac2_DW01_add_1)
0.00   563.86 f
  ac3/add_2_root_add_0_root_add_39/U55/Q (NAND20)           0.44     564.30 r
  ac3/add_2_root_add_0_root_add_39/U57/Q (CLKIN0)           0.75     565.05 f
  ac3/add_2_root_add_0_root_add_39/U1_1/CO (ADD32)          0.88     565.93 f
  ac3/add_2_root_add_0_root_add_39/U52/Q (NAND20)           0.47     566.40 r
  ac3/add_2_root_add_0_root_add_39/U54/Q (CLKIN0)           0.64     567.04 f
  ac3/add_2_root_add_0_root_add_39/U49/Q (NAND20)           0.73     567.76 r
  ac3/add_2_root_add_0_root_add_39/U51/Q (CLKIN0)           0.64     568.40 f
  ac3/add_2_root_add_0_root_add_39/U46/Q (NAND20)           0.73     569.13 r
  ac3/add_2_root_add_0_root_add_39/U48/Q (CLKIN0)           0.64     569.76 f
  ac3/add_2_root_add_0_root_add_39/U43/Q (NAND20)           0.73     570.49 r
  ac3/add_2_root_add_0_root_add_39/U45/Q (CLKIN0)           0.64     571.12 f
  ac3/add_2_root_add_0_root_add_39/U40/Q (NAND20)           0.73     571.85 r
  ac3/add_2_root_add_0_root_add_39/U42/Q (CLKIN0)           0.64     572.49 f
```

```
  ac3/add_2_root_add_0_root_add_39/U37/Q (NAND20)          0.73     573.21 r
  ac3/add_2_root_add_0_root_add_39/U39/Q (CLKIN0)          0.64     573.85 f
  ac3/add_2_root_add_0_root_add_39/U34/Q (NAND20)          0.73     574.57 r
  ac3/add_2_root_add_0_root_add_39/U36/Q (CLKIN0)          0.64     575.21 f
  ac3/add_2_root_add_0_root_add_39/U31/Q (NAND20)          0.73     575.94 r
  ac3/add_2_root_add_0_root_add_39/U33/Q (CLKIN0)          0.64     576.57 f
  ac3/add_2_root_add_0_root_add_39/U28/Q (NAND20)          0.73     577.30 r
  ac3/add_2_root_add_0_root_add_39/U30/Q (CLKIN0)          0.64     577.93 f
  ac3/add_2_root_add_0_root_add_39/U25/Q (NAND20)          0.73     578.66 r
  ac3/add_2_root_add_0_root_add_39/U27/Q (CLKIN0)          0.64     579.30 f
  ac3/add_2_root_add_0_root_add_39/U22/Q (NAND20)          0.73     580.02 r
  ac3/add_2_root_add_0_root_add_39/U24/Q (CLKIN0)          0.64     580.66 f
  ac3/add_2_root_add_0_root_add_39/U19/Q (NAND20)          0.73     581.39 r
  ac3/add_2_root_add_0_root_add_39/U21/Q (CLKIN0)          0.64     582.02 f
  ac3/add_2_root_add_0_root_add_39/U16/Q (NAND20)          0.73     582.75 r
  ac3/add_2_root_add_0_root_add_39/U18/Q (CLKIN0)          0.64     583.38 f
  ac3/add_2_root_add_0_root_add_39/U13/Q (NAND20)          0.73     584.11 r
  ac3/add_2_root_add_0_root_add_39/U15/Q (CLKIN0)          0.64     584.75 f
  ac3/add_2_root_add_0_root_add_39/U10/Q (NAND20)          0.73     585.47 r
  ac3/add_2_root_add_0_root_add_39/U12/Q (CLKIN0)          0.64     586.11 f
  ac3/add_2_root_add_0_root_add_39/U7/Q (NAND20)           0.73     586.83 r
  ac3/add_2_root_add_0_root_add_39/U9/Q (CLKIN0)           0.64     587.47 f
  ac3/add_2_root_add_0_root_add_39/U4/Q (NAND20)           0.73     588.20 r
  ac3/add_2_root_add_0_root_add_39/U6/Q (CLKIN0)           0.64     588.84 f
  ac3/add_2_root_add_0_root_add_39/U2/Q (XOR20)            1.17     590.01 f
  ac3/add_2_root_add_0_root_add_39/SUM[19] (Ac4xv3_SA_NB20_NBac2_DW01_add_1)
0.00   590.01 f
  ac3/add_0_root_add_0_root_add_39/B[19] (Ac4xv3_SA_NB20_NBac2_DW01_add_0)
0.00   590.01 f
  ac3/add_0_root_add_0_root_add_39/U1_19/CO (ADD32)        0.99     591.01 f
  ac3/add_0_root_add_0_root_add_39/U1_20/S (ADD32)         1.55     592.55 r
  ac3/add_0_root_add_0_root_add_39/SUM[20] (Ac4xv3_SA_NB20_NBac2_DW01_add_0)
0.00   592.55 r
  ac3/s_out[20] (Ac4xv3_SA_NB20_NBac2)                     0.00     592.55 r
  ac4/s_in[20] (Ac4xv3_SA_NB22_NBac2)                      0.00     592.55 r
  ac4/correct/n1[0] (ZPcorrect4SA_2)                       0.00     592.55 r
  ac4/correct/U9/Q (INV3)                                  0.28     592.83 f
  ac4/correct/U2/Q (XNR20)                                 0.84     593.67 r
  ac4/correct/U16/Q (NAND41)                               0.15     593.82 f
  ac4/correct/U7/Q (NAND22)                                0.40     594.22 r
  ac4/correct/U6/Q (INV3)                                  0.15     594.36 f
  ac4/correct/correc[0] (ZPcorrect4SA_2)                   0.00     594.36 f
  ac4/add_1_root_add_0_root_add_39/B[0] (Ac4xv3_SA_NB22_NBac2_DW01_add_1)
0.00   594.36 f
  ac4/add_1_root_add_0_root_add_39/U65/Q (NAND20)          0.44     594.81 r
  ac4/add_1_root_add_0_root_add_39/U67/Q (CLKIN0)          0.76     595.57 f
  ac4/add_1_root_add_0_root_add_39/U1_1/CO (ADD32)         0.88     596.45 f
  ac4/add_1_root_add_0_root_add_39/U62/Q (NAND20)          0.47     596.92 r
  ac4/add_1_root_add_0_root_add_39/U64/Q (CLKIN0)          0.62     597.54 f
  ac4/add_1_root_add_0_root_add_39/U59/Q (NAND20)          0.72     598.26 r
  ac4/add_1_root_add_0_root_add_39/U61/Q (CLKIN0)          0.63     598.90 f
  ac4/add_1_root_add_0_root_add_39/U56/Q (NAND20)          0.72     599.62 r
  ac4/add_1_root_add_0_root_add_39/U58/Q (CLKIN0)          0.63     600.25 f
  ac4/add_1_root_add_0_root_add_39/U53/Q (NAND20)          0.72     600.98 r
  ac4/add_1_root_add_0_root_add_39/U55/Q (CLKIN0)          0.63     601.61 f
  ac4/add_1_root_add_0_root_add_39/U50/Q (NAND20)          0.72     602.33 r
  ac4/add_1_root_add_0_root_add_39/U52/Q (CLKIN0)          0.63     602.97 f
  ac4/add_1_root_add_0_root_add_39/U47/Q (NAND20)          0.72     603.69 r
  ac4/add_1_root_add_0_root_add_39/U49/Q (CLKIN0)          0.63     604.32 f
  ac4/add_1_root_add_0_root_add_39/U44/Q (NAND20)          0.72     605.05 r
  ac4/add_1_root_add_0_root_add_39/U46/Q (CLKIN0)          0.63     605.68 f
  ac4/add_1_root_add_0_root_add_39/U41/Q (NAND20)          0.72     606.41 r
  ac4/add_1_root_add_0_root_add_39/U43/Q (CLKIN0)          0.63     607.04 f
  ac4/add_1_root_add_0_root_add_39/U38/Q (NAND20)          0.72     607.76 r
  ac4/add_1_root_add_0_root_add_39/U40/Q (CLKIN0)          0.63     608.40 f
  ac4/add_1_root_add_0_root_add_39/U35/Q (NAND20)          0.72     609.12 r
  ac4/add_1_root_add_0_root_add_39/U37/Q (CLKIN0)          0.63     609.75 f
  ac4/add_1_root_add_0_root_add_39/U32/Q (NAND20)          0.72     610.48 r
```

```
  ac4/add_1_root_add_0_root_add_39/U34/Q (CLKIN0)          0.63     611.11 f
  ac4/add_1_root_add_0_root_add_39/U29/Q (NAND20)          0.72     611.83 r
  ac4/add_1_root_add_0_root_add_39/U31/Q (CLKIN0)          0.63     612.47 f
  ac4/add_1_root_add_0_root_add_39/U26/Q (NAND20)          0.72     613.19 r
  ac4/add_1_root_add_0_root_add_39/U28/Q (CLKIN0)          0.63     613.82 f
  ac4/add_1_root_add_0_root_add_39/U23/Q (NAND20)          0.72     614.55 r
  ac4/add_1_root_add_0_root_add_39/U25/Q (CLKIN0)          0.63     615.18 f
  ac4/add_1_root_add_0_root_add_39/U20/Q (NAND20)          0.72     615.91 r
  ac4/add_1_root_add_0_root_add_39/U22/Q (CLKIN0)          0.63     616.54 f
  ac4/add_1_root_add_0_root_add_39/U17/Q (NAND20)          0.72     617.26 r
  ac4/add_1_root_add_0_root_add_39/U19/Q (CLKIN0)          0.63     617.90 f
  ac4/add_1_root_add_0_root_add_39/U14/Q (NAND20)          0.72     618.62 r
  ac4/add_1_root_add_0_root_add_39/U16/Q (CLKIN0)          0.63     619.25 f
  ac4/add_1_root_add_0_root_add_39/U11/Q (NAND20)          0.72     619.98 r
  ac4/add_1_root_add_0_root_add_39/U13/Q (CLKIN0)          0.63     620.61 f
  ac4/add_1_root_add_0_root_add_39/U8/Q (NAND20)           0.72     621.34 r
  ac4/add_1_root_add_0_root_add_39/U10/Q (CLKIN0)          0.63     621.97 f
  ac4/add_1_root_add_0_root_add_39/U5/Q (NAND20)           0.72     622.69 r
  ac4/add_1_root_add_0_root_add_39/U7/Q (CLKIN0)           0.63     623.32 f
  ac4/add_1_root_add_0_root_add_39/U3/Q (XOR20)            1.04     624.36 f
  ac4/add_1_root_add_0_root_add_39/SUM[22] (Ac4xv3_SA_NB22_NBac2_DW01_add_1)
0.00   624.36 f
  ac4/add_0_root_add_0_root_add_39/B[22] (Ac4xv3_SA_NB22_NBac2_DW01_add_0)
0.00   624.36 f
  ac4/add_0_root_add_0_root_add_39/U4/Q (NAND20)           0.70     625.06 r
  ac4/add_0_root_add_0_root_add_39/U5/Q (CLKIN0)           0.57     625.64 f
  ac4/add_0_root_add_0_root_add_39/U2/Q (XOR21)            1.42     627.05 r
  ac4/add_0_root_add_0_root_add_39/SUM[23] (Ac4xv3_SA_NB22_NBac2_DW01_add_0)
0.00   627.05 r
  ac4/s_out[23] (Ac4xv3_SA_NB22_NBac2)                     0.00     627.05 r
  ac5/s_in[23] (Ac4xv3_SA_NB24_NBac2)                      0.00     627.05 r
  ac5/reg1/s_in[23] (Reg_NB24_3)                           0.00     627.05 r
  ac5/reg1/s_out_reg[23]/D (DFEC1)                         0.00     627.05 r
  data arrival time                                                 627.05

  clock DV (rise edge)                                  1500.00    1500.00
  clock network delay (ideal)                              0.00    1500.00
  ac5/reg1/s_out_reg[23]/C (DFEC1)                         0.00    1500.00 r
  library setup time                                      -0.14    1499.86
  data required time                                               1499.86
  ---------------------------------------------------------------------------
  data required time                                               1499.86
  data arrival time                                                -627.05
  ---------------------------------------------------------------------------
  slack (MET)                                                       872.81
```

ANEXO II. Código VHDL

Incrementador de la precisión – Top-Level

```
-- by Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Precition_Increaser_SA is
   generic (NB     : natural := 16;
            NBnoise: natural := 5;--Hay que dejar de forzar a 16b
            NBac   : natural := 2;
            NcodAc : natural := 3);--(NB_out-NB_ini)/4;
   port (clk, rst, en : in  std_logic;
         noise        : in  unsigned(1 downto 0);
         s_in         : in  unsigned(NB-1 downto 0);
         mode_test    : in  std_logic;
         out_select   : in  unsigned(NcodAc-1 downto 0);
         s_out        : out unsigned(NB-1 downto 0);
         dv_out       : out std_logic);
         --c_mux_out  : out unsigned(NcodAc-1 downto 0));
end entity Precition_Increaser_SA;

architecture Behavioral of Precition_Increaser_SA is
   signal pre_out: unsigned(NB-1 downto 0);
   signal A1,A1R : unsigned(NB-1 downto 0);
   signal A2,A2R : unsigned(NB+NBac-1 downto 0);
   signal A3,A3R : unsigned(NB+NBac*2-1 downto 0);
   signal A4,A4R : unsigned(NB+NBac*3-1 downto 0);
   signal A5,A5R : unsigned(NB+NBac*4-1 downto 0);
   signal A6     : unsigned(NB+NBac*5-1 downto 0);
   signal c_muxIN: unsigned(NcodAc-1 downto 0);
   signal c_muxUC: unsigned(NcodAc-1 downto 0);
   signal c_mux  : unsigned(NcodAc-1 downto 0);
   signal dv     : std_logic;
   signal S1,S2,S3,S4,S5    : std_logic;
   signal B1,B2,B3,B4,B5,B6 : unsigned(NB-1 downto 0);

   --We'll need some accumulators. They implement solution to ZeroCrossing problem.
   component Ac4xv3_SA is
       generic(NB : natural;
               NBac : natural);
       port (clk,rst,en   : in  std_logic;
             s_in         : in  unsigned(NB-1 downto 0);
             s_out        : out unsigned(NB+NBac-1 downto 0));
   end component;

   --We'll need too one register after each accumulator to compare
   --the proximity between their successive outputs
   component Reg is
       generic(NB: natural);
       port(clk,rst,en  : in  std_logic;
            s_in        : in  unsigned(NB-1 downto 0);
            s_out       : out unsigned(NB-1 downto 0));
   end component;

   --The comparison mentioned just before will be done by the next blocks.
```

```
    component Comparator is
        generic(NB     : natural:=16;
                NBnoise: natural:=5);
        port   (clk    : in std_logic;
                x,xr   : in unsigned(NB-1 downto 0);
                noise  : in unsigned(1 downto 0);
                s      : out std_logic);
    end component;

    --After every comparisons the next block is the responsible of to choice the
    --best output between all the accumulators
    component UC_SA_v3 is
        generic(NBcnt       : natural :=2);
        port(rst,en         : in  std_logic;
             S1,S2,S3,S4,S5 : in  std_logic;
             dv             : in  std_logic;
             c_mux          : out unsigned(NcodAc-1 downto 0));
    end component;

    --Finally a Multiplexor gives us the output selected by last block.
    component MUXoutSA is
        generic(NB: natural);
        port(clk                    : in  std_logic;
             c_mux                  : in  unsigned(2 downto 0);
             B1,B2,B3,B4,B5,B6      : in  unsigned(NB-1 downto 0);
             pre_out                : out unsigned(NB-1 downto 0));
    end component;

    --General multiplexor used to select between mode test and normal mode
    component Mux2 is
    generic(NB : natural :=16);
    port (a,b  : in  unsigned(NB-1 downto 0);
          S    : in  std_logic;
          y    : out unsigned(NB-1 downto 0));
    end component;


begin

B1<=A1;
B2<=A2(NB+NBac-1   downto NBac);
B3<=A3(NB+NBac*2-1 downto NBac*2);
B4<=A4(NB+NBac*3-1 downto NBac*3);
B5<=A5(NB+NBac*4-1 downto NBac*4);
B6<=A6(NB+NBac*5-1 downto NBac*5);

--Outputs connections
s_out <= pre_out;
dv<=clk;
dv_out<=dv;

c_muxIN<=out_select;

--Accumulators connection
ac1: Ac4xv3_SA generic map(NB,NBac)        port map(dv,rst,en,A1,A2);
ac2: Ac4xv3_SA generic map(NB+NBac,NBac)   port map(dv,rst,en,A2,A3);
ac3: Ac4xv3_SA generic map(NB+NBac*2,NBac) port map(dv,rst,en,A3,A4);
ac4: Ac4xv3_SA generic map(NB+NBac*3,NBac) port map(dv,rst,en,A4,A5);
ac5: Ac4xv3_SA generic map(NB+NBac*4,NBac) port map(dv,rst,en,A5,A6);

--Registers connection
```

```
   --IMPORTANT (it may be changed by hand)
   --De weight of the registers depends on the number of
   --precision bits at the input.

--we'll use this register also for the first comparison
reg0: Reg generic map(NB)        port map(dv,rst,en,s_in,A1);

--reg1: Reg generic map(NB)        port map(dv,rst,en,A1,A1R);
A1R<=A1;
reg2: Reg generic map(NB+NBac)   port map(dv,rst,en,A2,A2R);
reg3: Reg generic map(NB+NBac*2) port map(dv,rst,en,A3,A3R);
reg4: Reg generic map(NB+NBac*3) port map(dv,rst,en,A4,A4R);
reg5: Reg generic map(NB+NBac*4) port map(dv,rst,en,A5,A5R);

--Comparators connection
ch1: Comparator generic map(NB,NBnoise)          port map(dv,s_in,A1R,noise,S1);
ch2: Comparator generic map(NB+NBac,NBnoise+1)   port map(dv,A2,A2R,noise,S2);
ch3: Comparator generic map(NB+NBac*2,NBnoise+2) port map(dv,A3,A3R,noise,S3);
ch4: Comparator generic map(NB+NBac*3,NBnoise+3) port map(dv,A4,A4R,noise,S4);
ch5: Comparator generic map(NB+NBac*4,NBnoise+4) port map(dv,A5,A5R,noise,S5);

--UC connection
uc2 : UC_SA_v3 generic map(2) port map(rst,en,S1,S2,S3,S4,S5,dv,c_muxUC);
mux1: MUXoutSA generic map(NB) port map(clk,c_mux,B1,B2,B3,B4,B5,B6,pre_out);
muxout: Mux2   generic map(NcodAc) port map(c_muxUC,c_muxIN,mode_test,c_mux);
end Behavioral;
```

Máquina de estados

```
-- by Jorge Solana Munoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity UC_SA_v3 is
   generic(NBcnt        : natural :=2); --Max number of samples accumulated
   port (rst,en         : in  std_logic;
         S1,S2,S3,S4,S5 : in  std_logic;
         dv             : in  std_logic;
         c_mux          : out unsigned(2 downto 0));

end entity UC_SA_v3;

architecture Behavioral of UC_SA_v3 is
   signal cnt     : unsigned(NBcnt-1 downto 0);
   signal fcnt    : std_logic; --Accumulation Finnished
   signal topCnt  : unsigned(NBcnt-1 downto 0);-- := to_unsigned(3,NBcnt);
   signal rstCnt  : std_logic;
   signal resetCnt: std_logic;
   signal reset   : std_logic;
   signal state,nextstate : unsigned(3 downto 0);
   signal priority: std_logic;

   component Counter is
       generic(NB: natural:=2);
       port(clk,rst,en: in  std_logic;
            finalCount: in  unsigned(NB-1 downto 0);
            c_out     : out unsigned(NB-1 downto 0);
            Tcnt      : out std_logic);
   end component;
   component Reg is
       generic(NB: natural);
       port(clk,rst,en  : in  std_logic;
            s_in        : in  unsigned(NB-1 downto 0);
            s_out      : out unsigned(NB-1 downto 0));
   end component;

begin
    --First 3 bits of 'state' tell us in which output we are;
    --the last one is to allow or not to continue with the accumulations.
    rstCnt<=state(0);
    c_mux<=state(3 downto 1);
    --In all the cases if we have a reset or a '1' from the first comparison
    --we'll reset all the system to the first case.
    reset<=rst or S1;
    topCnt<="10";
process(rst,dv)
    begin
        case state is
            --First case is output0 without permission to accumulate.
            ---If there isn't reset and the first comparison is good
            ---we'll allow the next accumulation
            ---by putting the reset of the counter to '0'.
            when "0001" =>
                if reset='1' then nextstate<="0001";
                else nextstate<="0000"; end if;
            --Second case is output1 with permission to accumulate.
            ---When we continue receiving good news from the first comparison
```

```
            ---during the count (3clocks) we'll be able to change the output to the
            ---next accumulator. That will happen on the next clock's rising edge.
            ---(because we put  register)
            when "0000" =>
                if reset='1' then nextstate<="0001";
                elsif fcnt='1' then nextstate<="0011";
                else nextstate<="0000"; end if;
            --Third case is output1 without permission to accumulate.
            when "0011" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                else nextstate <="0010"; end if;
            --Third case is output1 with permission to accumulate.
            when "0010" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif fcnt='1' then nextstate<="0101";
                else nextstate<="0010"; end if;
            --Third case is output2 without permission to accumulate.
            when "0101" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                else nextstate<="0100"; end if;
            when "0100" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                elsif fcnt='1' then nextstate<="0111";
                else nextstate<="0100"; end if;
            when "0111" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                elsif S4='1' then nextstate<="0111";
                else nextstate<="0110"; end if;
            when "0110" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                elsif S4='1' then nextstate<="0111";
                elsif fcnt='1' then nextstate<="1001";
                else nextstate<="0110"; end if;
            when "1001" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                elsif S4='1' then nextstate<="0111";
                elsif S5='1' then nextstate<="1001";
                else nextstate<="1000"; end if;
            when "1000" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
                elsif S3='1' then nextstate<="0101";
                elsif S4='1' then nextstate<="0111";
                elsif S5='1' then nextstate<="1001";
                elsif fcnt='1' then nextstate<="1011";
                else nextstate<="1000";end if;
            when "1011" =>
                if reset='1' then nextstate<="0001";
                elsif S2='1' then nextstate<="0011";
```

```
                elsif S3='1' then nextstate<="0101";
                elsif S4='1' then nextstate<="0111";
                elsif S5='1' then nextstate<="1001";
                else nextstate<="1011"; end if;
            when others => nextstate<="0000";
        end case;
    end process;

resetCnt<=rstCnt or rst;
--Counter connection
count: Counter generic map(NBcnt) port map(dv,resetCnt,en,topCnt,cnt,fcnt);
reg0 : Reg     generic map(4)     port map(dv,rst,en,nextstate,state);
end Behavioral;
```

Acumuladores

```
-- By Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Ac4xv3_SA is

   generic (NB : integer :=16;
            NBac : integer := 2);

   port (clk, rst, en : in  std_logic;
         s_in         : in  unsigned(NB-1 downto 0);
         s_out        : out unsigned(NB+NBac-1 downto 0));

end entity Ac4xv3_SA;

architecture Behavioral of Ac4xv3_SA is
   signal n,n1,n2,n3,sum : unsigned(NB+NBac-1 downto 0);
   signal correc   : unsigned(NB+NBac-1 downto 0);

   component ZPcorrect4SA is
       port (n1,n2,n3,n4 : in  unsigned(1 downto 0);
             correc      : out unsigned(1 downto 0));
   end component;

   component Reg is
       generic(NB       : integer);
       port(clk,rst,en  : in  std_logic;
            s_in        : in  unsigned(NB-1 downto 0);
            s_out       : out unsigned(NB-1 downto 0));
   end component;

begin
   n (NB-1      downto 0) <=s_in;
   n (NB+NBac-1 downto NB)<=(others=>'0');
   n1(NB+NBac-1 downto NB)<=(others=>'0');
   n2(NB+NBac-1 downto NB)<=(others=>'0');
   n3(NB+NBac-1 downto NB)<=(others=>'0');
   sum<=n+n1+n2+n3;
   correc(NB+NBac-3 downto 0)<=(others=>'0');
   s_out<=sum+correc;

--register connection
reg1: Reg generic map(NB) port map(clk,rst,en,n(NB-1 downto 0),n1(NB-1 downto 0));
reg2: Reg generic map(NB) port map(clk,rst,en,n1(NB-1 downto 0),n2(NB-1 downto 0));
reg3: Reg generic map(NB) port map(clk,rst,en,n2(NB-1 downto 0),n3(NB-1 downto 0));

--ZPcorrect4As connection
correct: ZPcorrect4SA port map(n(NB-1 downto NB-2),n1(NB-1 downto NB-2),n2(NB-1
downto NB-2),n3(NB-1 downto NB-2),correc(NB+NBac-1 downto NB+NBac-2));
end Behavioral;
```

Bloque corrector

```
-- By Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity ZPcorrect4SA is
   generic(NB_ac     : integer:=2);
   port (n1,n2,n3,n4 : in  unsigned(NB_ac-1 downto 0);
         correc      : out unsigned(NB_ac-1 downto 0));

end entity ZPcorrect4SA;

architecture Behavioral of ZPcorrect4SA is
   signal s    : std_logic;
   signal m0,m1: unsigned(NB_ac-1 downto 0);
   signal k1,k2,k3,k4 : unsigned(NB_ac-1 downto 0);

begin
    --if NB_ac!=2 you must change next part
   s<=(n1(1)xnor n1(0)) or (n2(1)xnor n2(0)) or    (n3(1)xnor n3(0)) or (n4(1)xnor
n4(0));
   m0<=(others=>'0');
   k1(1)<='0';
   k2(1)<='0';
   k3(1)<='0';
   k4(1)<='0';
   k1(0)<=not n1(0);
   k2(0)<=not n2(0);
   k3(0)<=not n3(0);
   k4(0)<=not n4(0);
   m1<=k1+k2+k3+k4;
   --miniMux
   correc<=m0 when s='0' else m1;

end Behavioral;
```

Comparadores

```
-- By Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Comparator is
   generic (NB     : natural :=16;
            NBnoise: natural :=5);
   port (clk    : in std_logic;
         x,xr   : in unsigned(NB-1 downto 0);
         noise  : in unsigned(1 downto 0);
         s      : out std_logic);
end entity Comparator;

architecture Behavioral of Comparator is
   signal y  : unsigned(NB-1 downto 0);
   signal z  : unsigned(NB-NBnoise-3-1 downto 0);
   signal zp : unsigned(NB-NBnoise-3-1 downto 0);
   signal sNN,sLN,sMN,sBN: std_logic;

begin
    y<=x-xr;

    z<=y(NB-1 downto NBnoise+3);
    zp(NB-NBnoise-3-1 downto 1)<=z(NB-NBnoise-3-2 downto 0);
    zp(0)<=z(NB-NBnoise-3-1);

    sBN<='0' when z=zp else '1'; --we see if all de bits of z are the similars
    sMN<='0' when sBN='0' and y(NB-1)=y(NBnoise+2) else '1';
    sLN<='0' when sMN='0' and y(NB-1)=y(NBnoise+1) else '1';
    sNN<='0' when sLN='0' and y(NB-1)=y(NBnoise) else '1';

 process(clk,noise)--,sNN,sLN,sMN,sBN)
     begin
           case noise is
                 when "00" => s<=sNN;
                 when "01" => s<=sLN;
                 when "10" => s<=sMN;
                 when "11" => s<=sBN;
                 when others => s<=sNN;
           end case;
 end process;
end Behavioral;
```

Multiplexor de salida

```
-- by Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity MUXoutSA is
   generic (NB : natural :=16);
   port (clk                  : in  std_logic;
         c_mux                : in  unsigned(2 downto 0);
         B1,B2,B3,B4,B5,B6    : in  unsigned(NB-1 downto 0);
         pre_out              : out unsigned(NB-1 downto 0));

end entity MUXoutSA;

architecture Behavioral of MUXoutSA is

begin
   -- MUX (asincrono)
   process(c_mux,B1,B2,B3,B4,B5,B6)
    begin
        case c_mux is
            when "000" => pre_out<=B1;
            when "001" => pre_out<=B2;
            when "010" => pre_out<=B3;
            when "011" => pre_out<=B4;
            when "100" => pre_out<=B5;
            when "101" => pre_out<=B6;
            when others=> pre_out<=B1;
        end case;
   end process;
end Behavioral;
```

Contador de 4 estados

```
-- by Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Counter is
   generic(NB: natural:=2);

   port(clk,rst,en: in  std_logic;
        finalCount: in  unsigned(NB-1 downto 0);
        c_out     : out unsigned(NB-1 downto 0);
        Tcnt      : out std_logic);
end entity Counter;

architecture Behavioral of Counter is
   signal count : unsigned(NB-1 downto 0);
   signal fin   : std_logic;

begin
   process (rst,clk)
   begin
      if rst='1' or (count=finalCount and rising_edge(clk)) then
         count <=(others=>'0');
      else if en='1' then
            if clk'event and clk = '1' then
               count <= count + 1;
            end if;
          end if;
      end if;
   end process;
   fin<='1' when count=finalCount and rst='0' else '0';

   c_out <= count;
   Tcnt <= fin;

end Behavioral;
```

Registros

```
-- by Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Reg is
   generic (NB: integer);
   port (clk, rst, en   : in  std_logic;
         s_in           : in  unsigned(NB-1 downto 0);
         s_out          : out unsigned(NB-1 downto 0));
end entity Reg;

architecture Behavioral of Reg is
   signal mem : unsigned(NB-1 downto 0);

begin
   process(clk,rst)
      begin
         if rst='1' then
            s_out<=(others=>'0');
         elsif en='1' and rising_edge(clk) then s_out<=s_in;
         end if;
   end process;
end Behavioral;
```

Multiplexor de dos entradas

```
-- by Jorge Solana Muñoz
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity Mux2 is
   generic(NB : natural :=16);
   port (a,b  : in  unsigned(NB-1 downto 0);
         S    : in  std_logic;
         y    : out unsigned(NB-1 downto 0));
end entity Mux2;

architecture Behavioral of Mux2 is
   begin
       y<=a when S='0' else b;
end Behavioral;
```

ANEXO III. Esquemáticos post-síntesis lógica

Figura ANEXO III - 1. Comparadores.

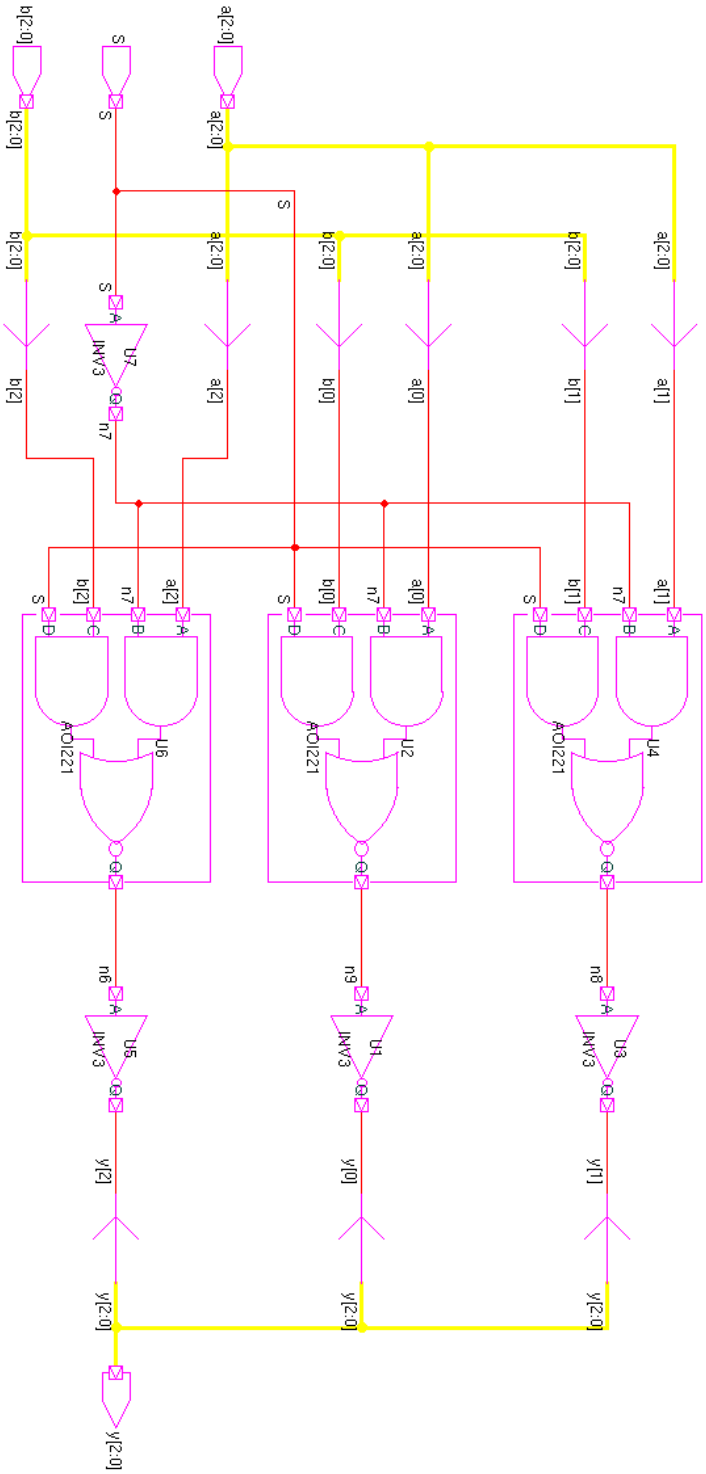


Figura ANEXO III - 2. Circuito incrementador de la precisión (parte 1).

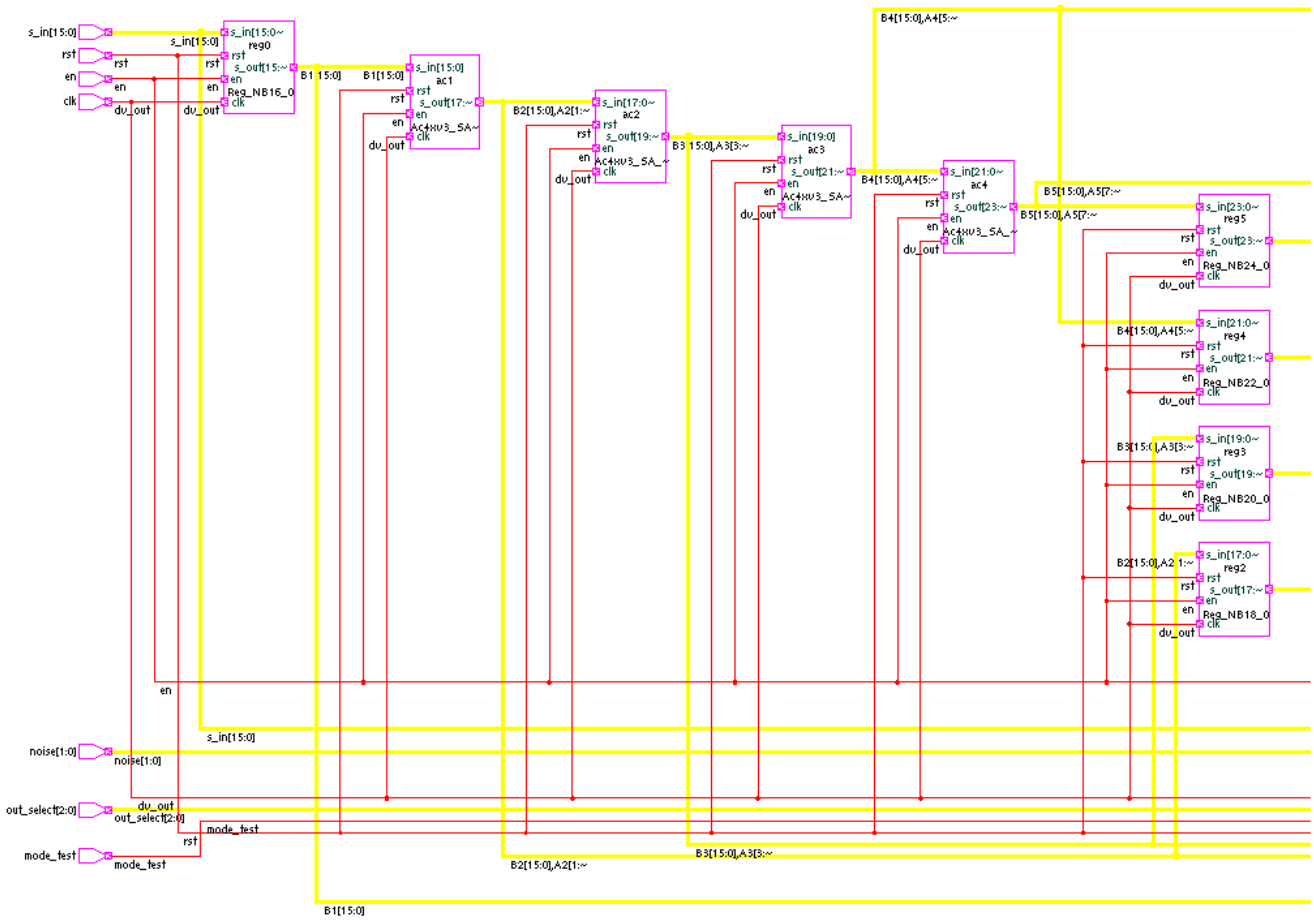


Figura ANEXO III - 3. Acumulador con NB=16 (parte 1).

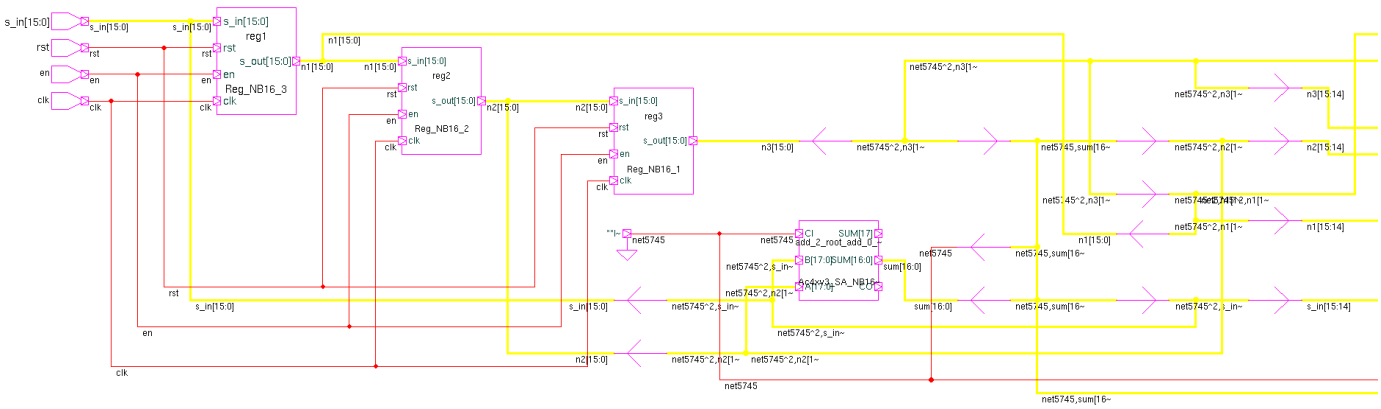


Figura ANEXO III - 2. Circuito Incrementador de la precisión (parte 2).

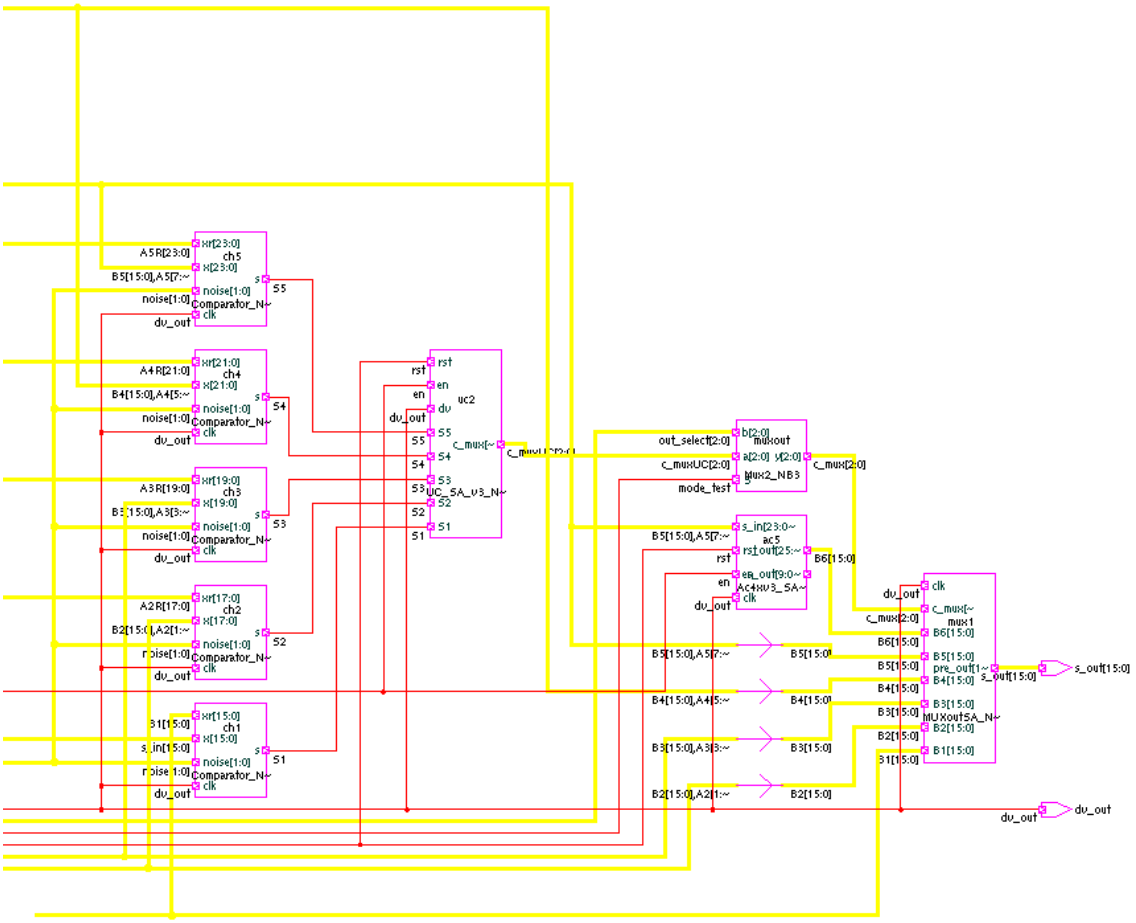


Figura ANEXO III - 3. Acumulador con NB=16 (parte 2).

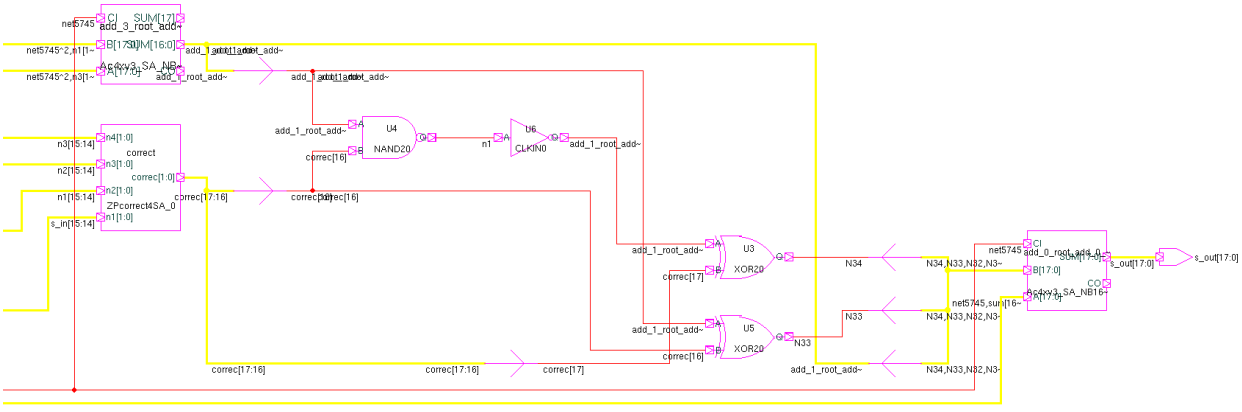


Figura ANEXO III - 4. Multiplexor de 2 entradas.

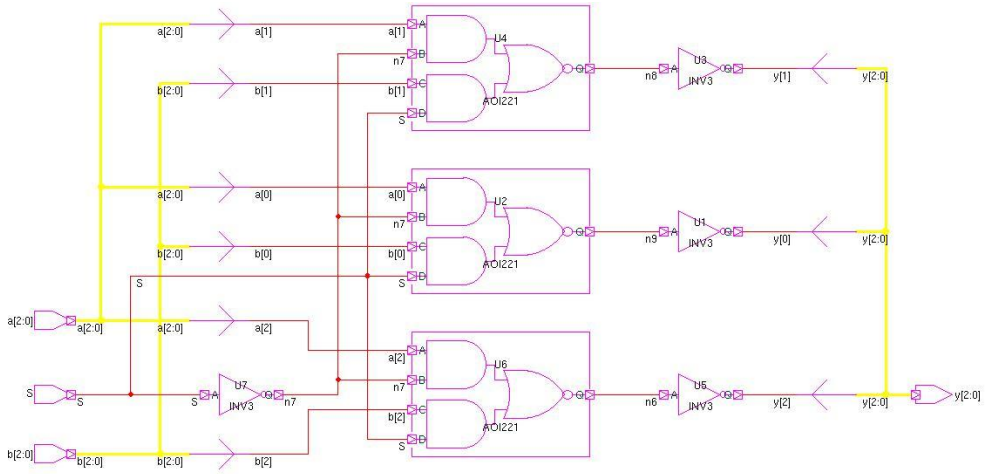


Figura ANEXO III - 5. Bloque ZPcorrectSA.

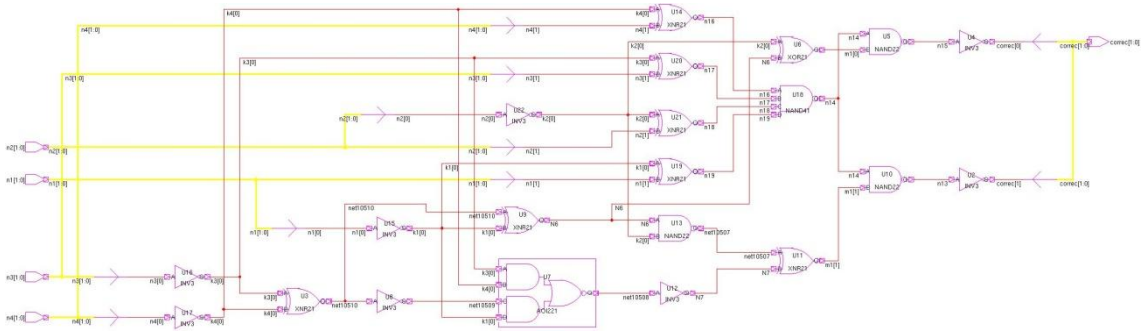
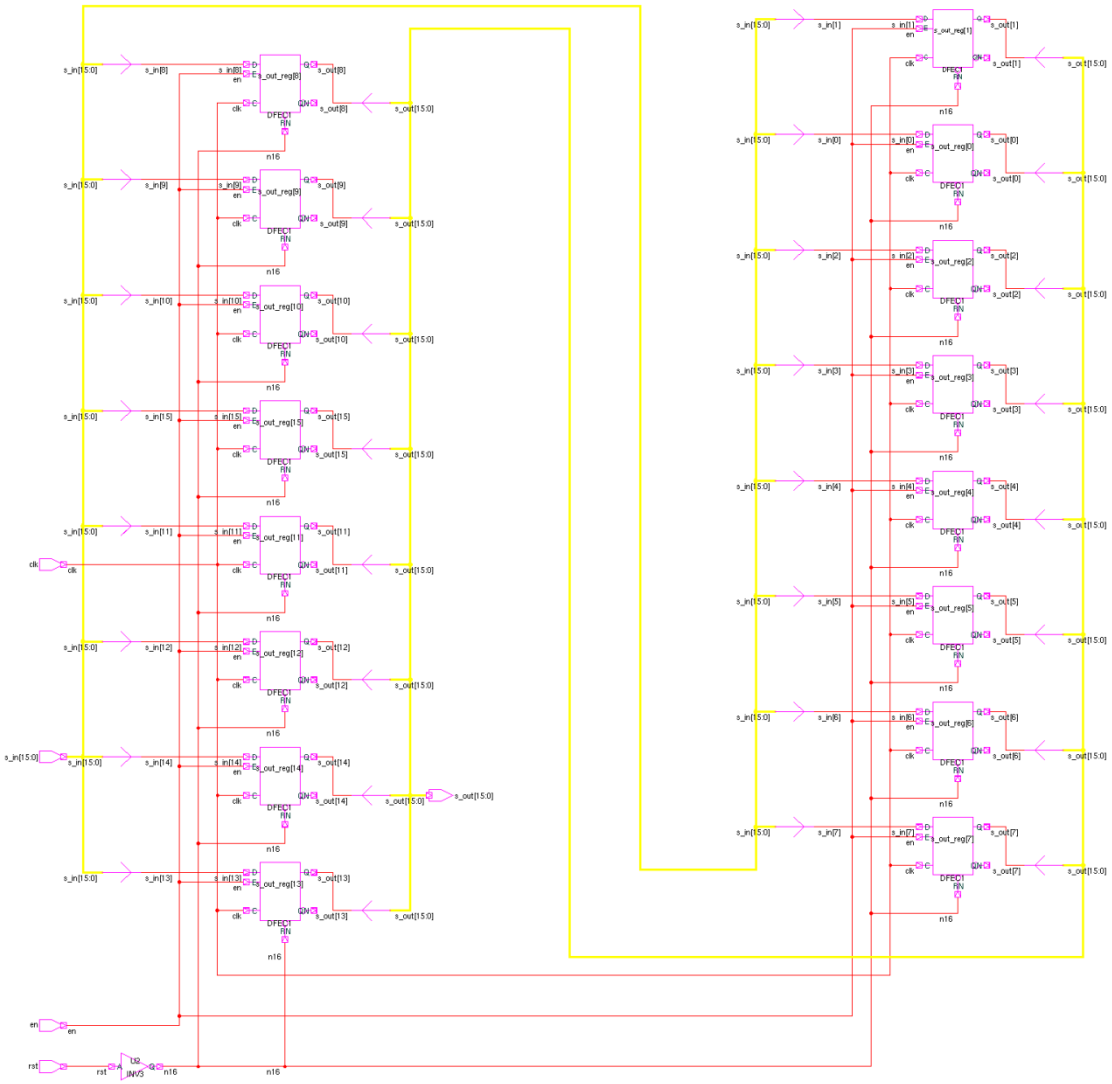


Figura ANEXO III - 6. Bloque *Reg* con NB=16.



ANEXO IV. Código Matlab para la generación de archivos testbench

ejecutable.m

```
% Aplication
clear all, close all, clc

% Example:
speed=0;            %angular speed
RfRt=10e3;          %refresh rate of samples
clk_per=1/RfRt*1e6; %clock period in miliseconds
ninputs=512;        %number of samples
inputsNB=16;        %inputs length
noiseNB=5;          %noise length - noise ratio
noiseR=0;           %external output to fit the noise(0,1,2 o 3)

ini=35;             %initial angle
[inputs,r,inSN]=inputsGeneration(speed,RfRt,ninputs,inputsNB,noiseNB,ini);
ini=187;            %new initial angle
[inputs,r,inSN]=inputsGeneration(speed,RfRt,ninputs,inputsNB,noiseNB,ini);
ini=125;            %new initial angle
[inputs3,r,inSN3]=inputsGeneration(speed,RfRt,ninputs,inputsNB,noiseNB,ini);

inputs=[inputs inputs2 inputs3];
inSN=[inSN inSN2 inSN3];
ninputs=ninputs*3;

mode=['SlideA';'semiSA';'Av_by4'];
[A1,A2,A3,A4,A5,A6]=soloAcum(inputs,inputsNB,r,mode(2,:));

InFile_v13=sprintf('INPUTS');
InFile_v13=sprintf('%s\n ------------------------ \n',InFile_v13);

for u=1:length(inputs)    %para caso de acumular 4 muestras
    InFile_v13=sprintf('%s %d-> %d \t',InFile_v13,u,inputs(u));
    if mod(u,6)==0
        InFile_v13=sprintf('%s\n ',InFile_v13);
    end;
end

%%%%Creamos el archivo%%%%
InputsFile_v13=fopen('InputsFile_v13(PrecitionIncreaser_v13).doc','w');
fwrite(InputsFile_v13,InFile_v13);
fclose(InputsFile_v13);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
InOutFile_v13=sprintf('INOUT');
InOutFile_v13=sprintf('%s\n ------------------------ \n',InOutFile_v13);
InOutFile_v13=sprintf('%s INPUTS \t __A1__ \t __A2__ \t __A3__ \t __A4__ \t __A5__ \t __A6__
\n',InOutFile_v13);
for u=1:length(inputs)    %para caso de acumular 4 muestras
    InOutFile_v13=sprintf('%s%d-> %d \t %d \t %d \t %d \t %d \t %d \t
\n',InOutFile_v13,u,inputs(u),A1(u),A2(u),A3(u),A4(u),A5(u),A6(u));
end
%%%%Creamos el archivo%%%%
InOutputsFile_v13=fopen('InOutputsFile_v13(PrecitionIncreaser_v13).doc','w');
fwrite(InOutputsFile_v13,InOutFile_v13);
fclose(InOutputsFile_v13);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TestBenchFile_v13=sprintf('--TestBench');
TestBenchFile_v13=sprintf('%s-- by Jorge Solana Munoz\n library ieee;\n  use
ieee.std_logic_1164.all;\n use ieee.numeric_std.all;\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s entity tb_Precition_Increaser_SA_rtl is end;\n \n architecture
bench of tb_Precition_Increaser_SA_rtl is\n',TestBenchFile_v13);
```

```
TestBenchFile_v13=sprintf('%s    constant CLK_PER : time := %d us;\n    constant NB   :
natural:=%d;\n      constant NBnoise  :
natural:=%d;\n',TestBenchFile_v13,clk_per,inputsNB,noiseNB);
TestBenchFile_v13=sprintf('%s    constant NBac : natural:=%d;\n constant NBcodac :
natural:=%d;\n \n',TestBenchFile_v13,2,3);
TestBenchFile_v13=sprintf('%s   signal s_in   : unsigned(NB-1 downto 0);\n     signal noise
: unsigned(1 downto 0);\n    signal clk:      std_logic:=''0'';\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s   signal rst    : std_logic;\n      signal en     :
std_logic;\n signal s_out  : unsigned(NB-1 downto 0);\n   signal dv_out :
std_logic;\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s   signal mode_test    : std_logic;\n      signal out_select
: unsigned(NBcodac-1 downto 0);\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s \n-- speed=%d; RfRt=%d; ninputs=%d; inputsNB=%d; noiseNB=%d;
ini=%d; noiseR=%d;%(0,1,2 o 3)\n
\n',TestBenchFile_v13,speed,RfRt,ninputs,inputsNB,noiseNB,ini,noiseR);
TestBenchFile_v13=sprintf('%s \n begin\n    UUT: entity
work.Precition_Increaser_SA(Behavioral)\n       generic map(NB,NBnoise,NBac,NBcodac)\n
port map (clk,rst,en,noise,s_in,mode_test,out_select,s_out,dv_out);\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s    clk <= not clk after CLK_PER/2;\n \n    stimulus: process\n
begin\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n s_in<=(others=>''0'');\n rst<=''0'';\n
en<=''0'';\n noise<="%s";\n',TestBenchFile_v13,dec2bin(noiseR,2));
TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n s_in<=to_unsigned(3453,NB);\n
mode_test<=''0'';\n out_select<="000";\n',TestBenchFile_v13,dec2bin(noiseR,2));
TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n rst<=''1'';\n en<=''1'';\n wait for
CLK_PER;\n rst<=''0'';\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n
noise<="%s";\n',TestBenchFile_v13,dec2bin(noiseR,2));
for k=1:ninputs
    TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n
s_in<=to_unsigned(%d,%d);\n',TestBenchFile_v13,inputs(k),inputsNB);
end

TestBenchFile_v13=sprintf('%s wait for CLK_PER*1024;\n rst<=''1'';\n
en<=''0'';\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s wait for CLK_PER;\n rst<=''0'';\n
en<=''1'';\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s wait for CLK_PER*3000;\n',TestBenchFile_v13);
TestBenchFile_v13=sprintf('%s end process stimulus;\n end architecture
bench;\n',TestBenchFile_v13);

%%%%Creamos el archivo%%%%
TBFile_v13=fopen('TestBenchFile_v13(PrecitionIncreaser_v13).doc','w');
fwrite(TBFile_v13,TestBenchFile_v13);
fclose(TBFile_v13);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

soloAcum.m

```
function [A1,A2,A3,A4,A5,A6]=soloAcum(inputs,inputsNB,r,mode)

% inputs --> is the vector of inputs used in the simulation. It's the
%            result of add the teorical inputs for an especificated speed,
%            with a known noise.
% option --> char vector to choice between the kind of average to do it.
%
% See help accumulator.m

% Before any accumulation
A1=inputs;                                  %original inputs
% A1bin=dec2bin(floor(A1/r),inputsNB);      %inputs in binarial codification
% A1bin=A1bin(:,end-inputsNB+1:end);        %to be sure we have 16bits
% out1=bin2dec(A1bin(:,1:inputsNB))';       %outputs in decimal
% A1=mod(A1,360);

%After first accumulation
A2=accumulator(A1,inputsNB,mode);           %original inputs
% A2bin=dec2bin(floor(A2/r),inputsNB+2);    %inputs in binarial codification
% A2bin=A2bin(:,end-inputsNB-1:end);        %to be sure we have 16bits
% out2=bin2dec(A2bin(:,1:inputsNB))';       %outputs in decimal
% A2=mod(A2,360*4);

%After second accumulation
A3=accumulator(A2,inputsNB+2,mode);         %original inputs
% A3bin=dec2bin(floor(A3/r),inputsNB+4);    %inputs in binarial codification
% A3bin=A3bin(:,end-inputsNB-3:end);        %to be sure we have 16bits
% out3=bin2dec(A3bin(:,1:inputsNB))';       %outputs in decimal
% A2=mod(A2,360*4^2);

%After thirst accumulation
A4=accumulator(A3,inputsNB+4,mode);         %original inputs
% A4bin=dec2bin(floor(A4/r),inputsNB+6);    %inputs in binarial codification
% A4bin=A4bin(:,end-inputsNB-5:end);        %to be sure we have 16bits
% out4=bin2dec(A4bin(:,1:inputsNB))';       %outputs in decimal
% A2=mod(A2,360*4^3);

%After fourth accumulation
A5=accumulator(A4,inputsNB+6,mode);         %original inputs
% A5bin=dec2bin(floor(A5/r),inputsNB+8);    %inputs in binarial codification
% A5bin=A5bin(:,end-inputsNB-7:end);        %to be sure we have 16bits
% out5=bin2dec(A5bin(:,1:inputsNB))';       %outputs in decimal
% A2=mod(A2,360*4^4);

%After fifth and last accumulation
A6=accumulator(A5,inputsNB+8,mode);         %original inputs
% A6bin=dec2bin(floor(A6/r),inputsNB+10);   %inputs in binarial codification
% A6bin=A6bin(:,end-inputsNB-9:end);        %to be sure we have 16bits
% out6=bin2dec(A6bin(:,1:inputsNB))';       %outputs in decimal
% A2=mod(A2,360*4^5);
```

accumulator.m

```
function out=accumulator(in,NB,mode)
% This fuction make the addition of each 4 sucesive values of the input
limit90=2^NB/4;
limit270=limit90*3;
oneRound=2^NB;

switch mode
    % First we'll see how to do it in case of Sliding Average
    case 'SlideA'    %Best results but it's the biggest option (in reg)
        a=[0 0 0 in];
        out(1:3)=[0 0 0];
        for k=1:length(in)
            out(k+3)=a(k)+a(k+1)+a(k+2)+a(k+3);
            aux=[a(k) a(k+1) a(k+2) a(k+3)];
            % Solution to "zero crossing" problem
            if (sum(aux>=limit90 & aux<=limit270)==0 & sum(aux<limit90)~=4)
                out(k+3)=out(k+3)+ sum(aux<limit90)*oneRound;
            end
        end
        out=out(4:end);
        out=mod(out,oneRound*4);
    % Semi Sliding Average
    case 'semiSA'   %Smallest option but results less sactifactory
        a=[0 0 0 in];
        %Solution to "zero crossing" problem
        preout(1)=0;
        for k=1:length(in)%+1
            preout(k+1)=preout(k)*3/4+a(k);
            aux=[a(k) a(k+1) a(k+2) a(k+3)];
            %out(k)=preout(k+1);
            if (sum(aux>=limit90 & aux<=limit270)==0 & sum(aux<limit90)~=4 & a(k)<limit90)
                preout(k+1)=preout(k+1)+oneRound;
            end
        end
        out=preout(2:end);
        out=mod(out,oneRound*4);
    %Basic Average, one output for each four inputs
    case 'Av_by4'   %First slution discarded by his limited results
        a=in;
        %Solution to "zero crossing" problem
        out=[];
        for k=0:length(in)/4-1
            out(k+1)=a(k*4+1)+a(k*4+2)+a(k*4+3)+a(k*4+4);%out(k)
            aux=[a(4*k+1) a(4*k+2) a(4*k+3) a(4*k+4)];
            if (sum(aux>=limit90 & aux<=limit270)==0 & sum(aux<limit90)~=4)
                out(k+1)=out(k+1)+sum(aux<limit90)*oneRound;
            end
        end
        out=mod(out,oneRound*4);
    otherwise
        out=-1;
        disp('Error writing one of the options: SlideA - semiSA - Av_by4');
end
```

ANEXO V. Código Matlab para la simulación de los acumuladores.

Añadiendo este código al archivo *ejecutable.m* del ANEXO IV, se pueden visualizar diversas representaciones gráficas que ayudan a comprender las diferencias entre los tres modos de acumulación de muestras estudiados.

```
figure(1)
subplot(6,1,1),plot(1:ninputs,A1,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,2),plot(1:ninputs,A2,'.-'),hold on,plot(1:ninputs,2^(inputsNB+2),'r-'),hold off
subplot(6,1,3),plot(1:ninputs,A3,'.-'),hold on,plot(1:ninputs,2^(inputsNB+4),'r-'),hold off
subplot(6,1,4),plot(1:ninputs,A4,'.-'),hold on,plot(1:ninputs,2^(inputsNB+6),'r-'),hold off
subplot(6,1,5),plot(1:ninputs,A5,'.-'),hold on,plot(1:ninputs,2^(inputsNB+8),'r-'),hold off
subplot(6,1,6),plot(1:ninputs,A6,'.-'),hold on,plot(1:ninputs,2^(inputsNB+10),'r-'),hold off

angleA1=A1*360/2^inputsNB;
angleA2=A2*360/2^(inputsNB+2);
angleA3=A3*360/2^(inputsNB+4);
angleA4=A4*360/2^(inputsNB+6);
angleA5=A5*360/2^(inputsNB+8);
angleA6=A6*360/2^(inputsNB+10);

figure(2)
subplot(6,1,1),plot(1:ninputs,angleA1,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,2),plot(1:ninputs,angleA2,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,3),plot(1:ninputs,angleA3,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,4),plot(1:ninputs,angleA4,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,5),plot(1:ninputs,angleA5,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,6),plot(1:ninputs,angleA6,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
figure,plot(1:ninputs,angleA1,'k.-'),hold on,plot(1:ninputs,angleA2,'r.-'),hold off


A1bin=dec2bin(A1,inputsNB); A1o=bin2dec(A1bin(:,1:inputsNB));
A2bin=dec2bin(A2,inputsNB+2); A2o=bin2dec(A2bin(:,1:inputsNB));
A3bin=dec2bin(A3,inputsNB+4); A3o=bin2dec(A3bin(:,1:inputsNB));
A4bin=dec2bin(A4,inputsNB+6); A4o=bin2dec(A4bin(:,1:inputsNB));
A5bin=dec2bin(A5,inputsNB+8); A5o=bin2dec(A5bin(:,1:inputsNB));
A6bin=dec2bin(A6,inputsNB+10); A6o=bin2dec(A6bin(:,1:inputsNB));

figure(3)
subplot(6,1,1),plot(1:ninputs,A1o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,2),plot(1:ninputs,A2o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,3),plot(1:ninputs,A3o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,4),plot(1:ninputs,A4o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,5),plot(1:ninputs,A5o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off
subplot(6,1,6),plot(1:ninputs,A6o,'.-'),hold on,plot(1:ninputs,2^inputsNB,'r-'),hold off

figure(4)
subplot(6,1,1),plot(1:ninputs,A1o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,2),plot(1:ninputs,A2o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,3),plot(1:ninputs,A3o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,4),plot(1:ninputs,A4o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,5),plot(1:ninputs,A5o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,6),plot(1:ninputs,A6o*r/360,'.-'),hold on,plot(1:ninputs,360,'r-'),hold off

figure(5)
plot(1:ninputs,A1o*r,'.-k'),hold on,plot(1:ninputs,360,'r-')
plot(1:ninputs,A2o*r,'.-b'),%plot(1:ninputs,2^(inputsNB+2),'r-')
plot(1:ninputs,A3o*r,'.-c'),%plot(1:ninputs,2^(inputsNB+4),'r-')
plot(1:ninputs,A4o*r,'.-g'),%plot(1:ninputs,2^(inputsNB+6),'r-')
plot(1:ninputs,A5o*r,'.-y'),%plot(1:ninputs,2^(inputsNB+8),'r-')
plot(1:ninputs,A6o*r,'.-m'),%plot(1:ninputs,2^(inputsNB+10),'r-')
```

```
hold off
legend('A1','A2','A3','A4','A5','A6');

figure(6)
x=ini+(0:ninputs-1)*speed/RfRt*360;
plot(x,A1o*r,x,A2o*r,x,A3o*r,x,A4o*r,x,A5o*r,x,A6o*r),legend('A1','A2','A3','A4','A5','A6');

%Noise
eA1=A1o-inSN';
eA2=A2o-inSN';
eA3=A3o-inSN';
eA4=A4o-inSN';
eA5=A5o-inSN';
eA6=A6o-inSN';
figure(7)
plot(1:ninputs,eA1*r/360,'.-k'),hold on%,plot(1:ninputs,360,'r-')
plot(1:ninputs,eA2*r/360,'.-b'),%plot(1:ninputs,2^(inputsNB+2),'r-')
plot(1:ninputs,eA3*r/360,'.-c'),%plot(1:ninputs,2^(inputsNB+4),'r-')
plot(1:ninputs,eA4*r/360,'.-g'),%plot(1:ninputs,2^(inputsNB+6),'r-')
plot(1:ninputs,eA5*r/360,'.-y'),%plot(1:ninputs,2^(inputsNB+8),'r-')
plot(1:ninputs,eA6*r/360,'.-m'),%plot(1:ninputs,2^(inputsNB+10),'r-')
hold off

figure(8)
subplot(6,1,1),plot(1:ninputs,eA1*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,2),plot(1:ninputs,eA2*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,3),plot(1:ninputs,eA3*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,4),plot(1:ninputs,eA4*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,5),plot(1:ninputs,eA5*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
subplot(6,1,6),plot(1:ninputs,eA6*r/360,'.-')%,hold on,plot(1:ninputs,360,'r-'),hold off
```