5 CONSIDERACIONES DE DISEÑO ADC BASADO EN TIEMPO

En este capítulo se describirá el diseño de los modelos llevados a cabo para hacer distintas consideraciones para implementar el ADC basado en tiempo en el que se basa en la arquitectura descrita en el capítulo anterior. Para abordar el diseño, en primer lugar se ha implementado la máquina de estados, ya que se trata del elemento más importante del sistema porque controla todos los demás bloques de la arquitectura. En función de cómo se diseñe la máquina de estados se diseñarán los demás bloques del sistema.

Para implementar la máquina de estados se ha realizado un diseño en código VHDL, a partir del cual se ha trasladado tanto a un modelo en Matlab/Simulink en el que se ha incrustado toda la arquitectura para analizar la viabilidad de esta, como a un modelo diseñado en verilog implementado en Cadence para añadir elementos reales, tales como transistores CMOS para realizar un análisis más completo de la arquitectura.

Una vez trasladada la máquina de estados tanto a Matlab/Simulink como a Cadence se han implementado los demás bloques del sistema para estudiar la arquitectura haciendo distintos análisis paramétricos en los que se estudiarán distintas especificaciones del ADC con la arquitectura que se describe en el capítulo anterior.

5.1 Máquina de estados

Para describir la máquina de estados en primer lugar se realizará una descripción funcional de esta en la que se explicará el funcionamiento a alto nivel. Una vez explicada la funcionalidad se analizará el código VHDL para entender el funcionamiento interno. Finalmente se realizarán varias simulaciones de este diseño VHDL para ver el funcionamiento en los distintos casos que se pueden dar.

5.1.1 Funcionalidad

En primer lugar se describirán las distintas entradas y salidas de las que consta la máquina de estados. Posteriormente se explicará cada estado por separado, describiendo las acciones que se realizan en cada uno de los estos y las condiciones que se deben dar para cambiar de estado.

En la *Figura 42* se muestra un esquema de la máquina de estados en el que se pueden ver las distintas entradas y salidas de la máquina de estados.

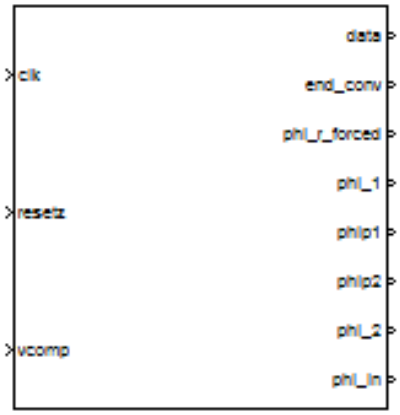

Figura 42. Entradas-Salidas de la Máquina de Estados

Entradas:

- clk: reloj del sistema. Entra en juego en el proceso de sincronismo en el sistema, produciéndose las transiciones de los distintos estados durante el flanco de subida.
- resetz: reset del sistema. Se trata de un reset activo a nivel bajo.

- vcomp: salida del comparador de la arquitectura. Se trata de una señal que controla una serie de flip-flops que controlaran a su vez el cambio de algunas señales de salida sin la necesidad de que se produzca un cambio de estado.

Salidas:

- data: Se trata de un dato digital de diez bits que se corresponde con la salida de datos digital. Se produce una conversión cada veintidós ciclos de reloj.
- end_conv: señal que cuando está a '1' indica que ha finalizado la conversión de un bit y que sirve para saber cuándo hay que tomar el dato, es decir, cuándo hay que muestrear.
- phi_r_forced: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para resetar los dos condesadores de los que consta la arquitectura.
- phi_1: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para que la corriente de referencia I_{ref} cargue el condensador C_1.
- phip1: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para que la salida del condensador C_1 se conecte al terminal positivo del comparador.
- phi_2: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para que la corriente de referencia I_{ref} cargue el condensador C_2.
- phip2: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para que la salida del condensador C_2 se conecte al terminal positivo del comparador.
- phi_in: señal que cuando está a '1' propicia que la circuitería analógica se reorganice para que la corriente de entrada I_{in} cargue el condensador C_1, ya que se trata de la carga inicial en la conversión.

En cuando al diseño, se ha sido capaz de implementar una máquina de estados con ocho estados: *Reposo*, *Initial_charge*, *C2_charge*, *Neg_res*, *Pos_res*, *Assign_1*, *Assign_zero* y *Wait_end*.

- Estado 0: *Reposo*. En este estado se resetean varios de los flip-flops controlados por la salida del comparador antes mencionados en la explicación, se indica que el estado siguiente será *Initial_Charge* y se modifica la salida phi_r_forced asignando un '1' de manera que se indica el reseteo de los condensadores. Por otro lado se modifican una serie de señales internas de la máquina de estados de manera que se indica que para el siguiente estado data será igual a "0", end_conv será '0' y phi_in será 1'. Finalmente, se inicializa un contador que controla que la duración de la conversión sea como máximo veintidós ciclos de reloj.

- Estado 1: *Initial_charge.* En este estado se resetean varios de los flip-flops controlados por la salida del comparador antes mencionados en la explicación y se indica que el estado siguiente será *C2_charge*. Por otro lado durante el tiempo que la máquina se encuentra en este estado, que será un ciclo de reloj, la salida phi_in está a '1', de manera que se cargará C_1 con I_{in}. Por último se inicializa un con contador de dos bits, el cual será utilizado para contar el número de ciclos hasta que se igualan las tensiones en los condensadores para calcular los dos bits más significativos en la fase MSB de la conversión.

- Estado 2: *C2_charge*. En este estado se ponen tanto phi_2 como phip2 a '1' mientras no se igualan las tensiones en los condensadores, es decir, se carga el condensador C_2 con I_{ref} hasta que se llega al mismo nivel de tensión, lo cual se detecta por la salida del comparador; y por otra parte se va incrementando tanto el contador para obtener los dos bist MSBs como el contador general que controla el número de ciclos que dura la conversión. Una vez se produce lo antes comentado, se indica que debe haber una transición al estado *neg_res*, se pone phi_1 a '1' para iniciar la carga de C_1 con I_{ref} y finalmente se asignan los dos bits más significativos al dato de salida.

- Estado 3: *neg_res*. Este estado se corresponde con la conversión de un bit en una etapa de resolución negativa, es decir, se calcula el negado el bit correspondiente. Por tanto, se controla la carga de los condensadores C_1 y C_2 con I_{ref} y la conexión a los terminales positivo y negativo del comparador. Por otro lado, en primer lugar se comprueba si el residuo cae tan cerca del flanco de reloj del paso previo que no se podrá detectar, en cuyo caso, significa que se corresponde con la asignación de un '1',

por lo que se establece que el siguiente estado será *assign_1*; y en segundo lugar se comprueba si el residuo cae tan cerca del flanco de reloj del paso posterior que no se podrá detectar, en cuyo caso, significa que se corresponde con la asignación de un '0', por lo que se establece que el siguiente estado será *assign_0*. En el caso de que no se de ninguna de estas dos circunstancias, y ya se hayan producido las dos comparaciones necesarias para la conversión de un bit (esta circunstancia se indica mediante banderas), se asignara un '0' o un '1', según corresponda, y se establecerá que el estado siguiente será *Pos_res.*

- Estado 4: *Pos_res*. Este estado se corresponde con la conversión de un bit en una etapa de resolución positiva, es decir, se calcula el bit correspondiente propiamente dicho. Por tanto, se controla la carga de los condensadores C_1 y C_2 con I_{ref} y la conexión a los terminales positivo y negativo del comparador. Por otro lado, en primer lugar se comprueba si el residuo cae tan cerca del flanco de reloj del paso previo que no se podrá detectar, en cuyo caso, significa que se corresponde con la asignación de un '0', por lo que se establece que el siguiente estado será *assign_0*; y en segundo lugar se comprueba si el residuo cae tan cerca del flanco de reloj del paso posterior que no se podrá detectar, en cuyo caso, significa que se corresponde con la asignación de un '1', por lo que se establece que el siguiente estado será *assign_1*. En el caso de que no se de ninguna de estas dos circunstancias, y ya se hayan producido las dos comparaciones necesarias para la conversión de un bit (esta circunstancia se indica mediante banderas), se asignara un '0' o un '1', según corresponda, y se establecerá que el estado siguiente será *Neg_res.* Por otro lado en este estado también se comprueba si se ha llegado al final de la conversión (veintidós ciclos de reloj), de tal manera que se establezca que el siguiente estado sea *Reposo* (la salida end_conv se pondrá a '1') o por el contrario aún haya que esperar a que termine la conversión, cuando se establecerá que el siguiente estado sea *Wait_end.*
- Estado 5: *Assign_1.* Si durante la conversión se pasa por este estado significa que el residuo ha caído tan cerca tanto del flanco de reloj del paso previo o tan cerca del flanco de reloj del paso posterior que no es capaz de detectarse. En este estado se asignará un '1' a todos los bits que queden por convertir, ya que así lo establecen las condiciones del residuo. Por otro lado en este estado también se comprueba si se ha llegado al final de la conversión (veintidós ciclos de reloj), de tal manera que se establezca que el siguiente estado sea *Reposo* (la salida end_conv se pondrá a '1') o por el contrario aún haya que esperar a que termine la conversión, cuando se establecerá que el siguiente estado sea *Wait_end*
- Estado 6: *Assign_zero.* Si durante la conversión se pasa por este estado significa que el residuo ha caído tan cerca tanto del flanco de reloj del paso previo o tan cerca del flanco de reloj del paso posterior que no es capaz de detectarse. En este estado se asignará un '0' a todos los bits que queden por convertir, ya que así lo establecen las condiciones del residuo. Por otro lado en este estado también se comprueba si se ha llegado al final de la conversión (veintidós ciclos de reloj), de tal manera que se establezca que el siguiente estado sea *Reposo* (la salida end_conv se pondrá a '1') o por el contrario aún haya que esperar a que termine la conversión, cuando se establecerá que el siguiente estado sea *Wait_end.*
- Estado 7: *Wait_end*. Si se pasa por este estado es porque la conversión ha finalizado, pero aun no han transcurrido los veintidós ciclos de reloj necesariós para obtener una conversión, de manera que en este estado lo único que se hace es esperar a que pasen estos veintidós ciclos, y cuando se de este hecho se establecerá que el estado siguiente será *Reposo*, y se cambiará la señal de salida end_conv poniéndola a '1'.

5.1.2 Implementación en código VHDL

El diseño de la máquina de estados en primer lugar se ha implementado en lenguaje VHDL, para posteriormente en base a este diseño pasar tanto implementar en Matlab/Simulink mediante stateflow como en Cadence mediante síntesis con PKS.

5.1.2.1 Descripción del código VHDL

A la hora del diseño VHDL, se ha tenido presente que se está ante un sistema síncrono, pues se basa en la utilización de una señal de reloj que coordina los cambios en el circuito, sucediéndose en instantes muy precisos, concretamente en los flancos activos (de subida en este caso).

Todo sistema síncrono se puede descomponer en dos procesos concurrentes, uno que recoge la funcionalidad característica del sistema y un proceso de sincronismo. Para el diseño VHDL se ha seguido este esquema de codificación en dos procesos, ya que si bien requiere la introducción de un número mayor de líneas de código la descripción es más clara, más legible y más fácil de depurar.

Como se ha mencionado anteriormente se utilizan utilizarán flip-flops activos por flanco. Estos están controlados por la salida del comparador, los cuales nos servirán para controlar la carga y descarga de los condensadores dentro de los propios estados, así como para controlar las condiciones que se deben dar en las distintas transiciones de los estados. Por lo tanto no debe extrañar que se produzcan cambios en la salida de la máquina de estados sin que se produzca un cambio de estado.

A continuación se describirá el código VHDL de forma detallada, de manera que junto con la explicación anterior de la funcionalidad de los distintos estados, quede totalmente descrito el funcionamiento de la máquina de estados.

En primer lugar se mostrará el diseño que ha seguido para implementar los flip-flops D utilizados a lo largo del diseño.

<table>
<tr><th colspan="3">Flip-Flop qb</th></tr>
<tr><td>Descripción</td><td colspan="2">Un biestable o flip-flop D es un dispositivo de almacenamiento temporal, cuya salida adquiere el valor de la entrada D cuando se activa la entrada de sincronismo, el reloj clk.</td></tr>
<tr><td>Entidad</td><td colspan="2"><pre>
entity flipflop_qb is
    Port (d : in STD_LOGIC;
           clk : in STD_LOGIC;
           resetz : in STD_LOGIC;
           q : out STD_LOGIC;
           qb : out STD_LOGIC);
end flipflop_qb;
</pre></td></tr>
<tr><td colspan="3">Descripción de los puertos</td></tr>
<tr><td>d</td><td colspan="2">Entrada del biestable que se verá reflejada a la salida</td></tr>
<tr><td>clk</td><td colspan="2">Reloj, señal de sincronismo</td></tr>
<tr><td>resetz</td><td colspan="2">Reset asíncrono activo a nivel bajo</td></tr>
<tr><td>q</td><td colspan="2">Salida del biestable que refleja el mismo valor de la entrada</td></tr>
<tr><td>qb</td><td colspan="2">Salida negada del biestable que refleja el valor de la entrada negado</td></tr>
<tr><td colspan="3">Diseño VHDL</td></tr>
<tr><td colspan="2">Código VHDL</td><td>Descripción</td></tr>
</table>

`` architecture Behavioral of flipflop_qb is --Señales signal p_q,qr: std_logic; begin comb: process(d,qr) begin p_q<=d; end process; q<=qr; qb<=not(qr); sync: process(clk,resetz) begin if(resetz='0') then qr<='0'; elsif(clk='1' and clk'event) then qr<=p_q; end if; end process; end Behavioral; ```	El diseño del biestable se descompone en dos procesos concurrentes, uno que recoge la funcionalidad característica del propio flip-flop denominado "comb" y otro, denominado proceso de sincronismo "sync", en el cual se detecta cuando se producen los cambiós a través de la detección del flanco de subida

A continuación se mostrará una figura en la que se observa el funcionamiento del biestable tras realizar una simulación utilizando ModelSim.

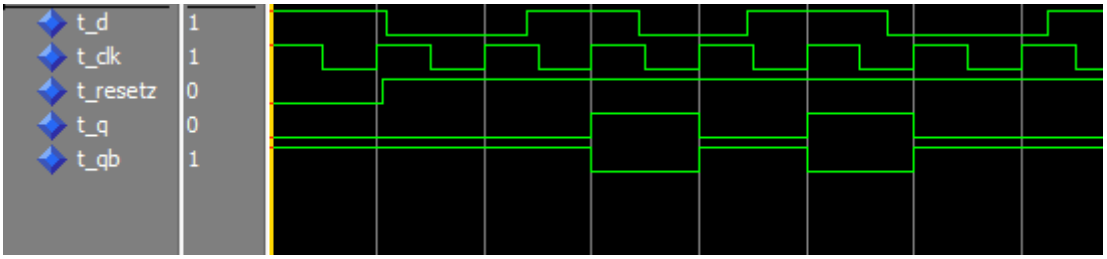

Figura 43. Simulación flip-flip D en Modelsim

Cur_based_adc_sm	
Descripción	Se trata de una máquina de estados que controla la lógica del convertidor analógico digital basado en tiempo. Como se ha explicado en la descripción del convertidor se usará el tiempo como variable. El diseño del convertidor está basado en la carga y descarga de dos condensadores. Esta máquina controlará por un lado la carga y descarga de los condensadores, y por otro lado la conexión de la salida de los condensadores a los terminales del comparador del convertidor.
Entidad	``` entity cur_based_adc_sm is Port (resetz : in STD_LOGIC; clk : in STD_LOGIC; vcomp: in STD_LOGIC; data: out STD_LOGIC_VECTOR (9 downto 0); end_conv: out STD_LOGIC; phi_in: out STD_LOGIC; phi_1: out STD_LOGIC; phi_2: out STD_LOGIC; phi_r_forced: out STD_LOGIC; phip1: out STD_LOGIC; phip2: out STD_LOGIC); end cur_based_adc_sm; ```
Descripción de los puertos	
resetz	Reset asíncrono activo a nivel bajo
clk	Reloj, señal de sincronismo
vcomp	Salida del comparador que actúa como entrada de la máquina de estados para controlar el cambio que se produce en los biestables dedicados a este efecto.

data	Salida de datos digital.
end_conv	Salida de indicación de fin de la conversión.
phi_in	Salida que indica fase de carga inicial. Carga del condensador C_1 mediante $I_{in.}$
phi_1	Salida que indica fase de carga del condensador C_1. Carga con I_{ref}.
phi_2	Salida que indica fase de carga del condensador C_2. Carga con I_{ref}.
phi_r_forced	Salida que indica fase de reseteo de ambos condensadores.
phip1	Salida que indica fase de conexión de la salida de C_1 con el terminal positivo del comparador.
phip2	Salida que indica fase de conexión de la salida de C_2 con el terminal positivo del comparador.

Diseño VHDL

Código VHDL		**Descripción**
Inicio del Diseño: Parte Funcional	`architecture Behavioral of cur based adc sm is`	Comienzo del nivel de abstracción más elevado de la descripción en la que se define el comportamiento del circuito.
Definición de Component	`component flipflop_qb` `  Port ( d : in  STD_LOGIC;` `  clk : in  STD_LOGIC;` `  resetz : in  STD_LOGIC;` `  q : out  STD_LOGIC;` `  qb : out STD_LOGIC);` `end component;`	Al inicio del diseño se incluyen elementos diseñados con anterioridad en el diseño del circuito. En este caso se incluye el flip-flop D descrito anteriormente como componente del diseño.
Estados	`type states is (Reposo, Initial_charge, C2_charge,` `  neg_res, pos_res,wait_end, assign_1,` `  assign_zero);`	Se define un nuevo tipo de datos estado (states) para los distintos estados de la máquina.
Señales	`signal state,p_state: states;` `signal p_data, data_s: std_logic_vector (9 downto 0);` `signal count_conv, p_count_conv: std_logic_vector (4 downto 0);` `signal counter, p_counter: std_logic_vector (1 downto 0);` `signal bit, p_bit: std_logic_vector (2 downto 0);` `signal p_phi_in, phi_in_s: std_logic;` `signal rst_cal, rst_ini, rst_comp_ff: std_logic;` `signal flag1, flag2, flag1_n, flag2_n: std_logic;` `signal comp_cal, comp_ini, comp_cal_n, comp_ini_n: std_logic;` `signal p_end_conv, end_conv_s: std_logic;`	En esta parte del código se definen una serie de señales internas. *state, p state*: señales utilizadas para identitificar los estados. La señal p_state se utiliza para actualizar el estado en el proceso de sincronismo. *p data, data s*: son señales asociadas a la salida de datos. Se trata de vectores de 10 bits. La señal p_data se utiliza para actualizar el dato en el proceso de sincronismo, y data_s se conecta directamente con la señal de salida. *count conv, p cont conv:* son señales asociadas al contador general para comprobar cuando termina la conversión. La señal p_count_conv se utiliza para actualizar en el proceso de sincronismo. Se trata de un vector de 4 bits, pues como mucho una conversión dura 22 ciclos de reloj.

		counter, p counter: se trata de señales que actúan como contador en la conversión de cada uno de los bits. Se trata de un vector de dos bits. La señal p_counter se utiliza para actualizar en el proceso de sincronismo. *bit, p bit*: son señales que indican cual de los 10 bits se está convirtiendo en cada momento. La señal p_bit se utiliza para actualizar en el proceso de sincronismo. Se trata de un vector de 3 bits. *p phi in, p phi in s*: señales asociadas a la señal phi_in. *rst cal, rst ini, rst cmp ff*: señales que actúan como reset de los distintos flip-flops utilizados en el diseño. *flag1, flag2, flag1 n, flag2 n*: banderas utilizadas como salidas de dos de los flip-flops que nos indican las distintas comparaciones que se producen en cada conversión. *comp cal, comp ini, comp cal n, comp ini n*: utilizadas como salidas de dos de los flip-flops para contralar las distintas conversiones de cada bit. *p end conv, end conv s*: señales asociadas a la indicación de conversión. p_end_conv se actualiza en el proceso de sincronismo, y end_conv_s se conecta directamente a la salida.
Conexión a señales de salida	`begin` `data<=data_s;` `phi_in<=phi_in_s;` `end_conv<=end_conv_s;`	En primer lugar se conectan las salidas de la máquina con las señales definidas dedicadas a este efecto.

Definición de flip-flops D	`ff_ini: flipflop_qb port map(` `d=>comp_ini_n,` `clk=>vcomp,` `resetz=>rst_ini,` `q=>comp_ini,` `qb=>comp_ini_n);` `ff_cal: flipflop_qb port map(` `d=>comp_cal_n,` `clk=>vcomp,` `resetz=>rst_cal,` `q=>comp_cal,` `qb=>comp_cal_n);` `ff_flag1: flipflop_qb port map(` `d=>flag1_n,` `clk=>comp_cal,` `resetz=>rst_comp_ff,` `q=>flag1,` `qb=>flag1_n);` `ff_flag2: flipflop_qb port map(` `d=>flag2_n,` `clk=>comp_cal_n,` `resetz=>rst_comp_ff,` `q=>flag2,` `qb=>flag2_n);`	En esta parte del código se definen los flip-flops D utilizados en el diseño, los cuales se han conectado a las señales incluidas para este efecto. A continuación se describirá la función de cada uno de estos flip-flops. *ff_ini*: este flip-flop se activa cuando se pone la salida del comparador a '1' por primera vez, de manera que se sabe cuando se está haciendo la primera carga de los condensadores. *ff_cal*: este flip-flop está controlado por vcomp, y su salida se utilizará como entrada a otro flip-flop para activar o desctivar banderas que indican en qué momento de la conversión de cada bit se encuentra el proceso para cambiar de estado según proceda. *ff_flag1*: este flip-flop está controlado por la salida de ff_cal. La salida es una bandera que indica en qué momento de la conversión de cada bit se encuentra, la cual condiciona la transición de estados. *ff_flag2*: este flip-flop está controlado por la salida de ff_cal. La salida es una bandera que indica en qué momento de la conversión de cada bit, la cual condiciona la transición de estados.
Inicio del proceso concurrente: Parte funcional	`comb: process(state, bit, comp_ini,` `comp_cal, comp_cal_n,` `flag1, flag2, count_conv)` `begin` `p_count_conv<=count_conv+1;` `p_state<=state;` `p_data<=data_s;` `p_counter<=counter;` `p_bit<=bit;` `p_end_conv<=end_conv_s;` `p_phi_in<='0';` `phi_1<='0';` `phi_2<='0';` `phi_r_forced<='0';` `phip1<='0';` `phip2<='0';` `rst_comp_ff<='1';` `rst_cal<='1';` `rst_ini<='1';`	En esta parte del código se inicia el proceso concurrente con la funcionalidad característica del diseño. En primer lugar se incluye una lista de sensibilidad con las señales que pueden producir que el proceso se evalúe completamente. Posteriormente se realizan una serie de asignaciones que sirven para conectar distintas señales internas, actualizar contadores, inicializar salidas directas del sistema y finalmente para activar todos los flip-flops del sistema mediente un reset. Esta última acción podría parecer discutible porque se podrían activar en el momento que no se requiera, pero es dentro los propios estados donde se desactivan si es necesario.

Estado 0: Reposo	`case state is` `  when Reposo =>` `  rst_comp_ff<='0';` `  rst_cal<='0';` `  rst_ini<='0';` `  p_phi_in<='1';` `  phi_r_forced<='1';` `  p_state<=Initial_charge;` `  p_data<="0000000000";` `  p_end_conv<='0';` `  p_count_conv<="00000";`	En este estado en las primeras tres líneas de código se desactivan los flip-flops ya que no deben actuar en este estado. Se establece que el próximo estado será *Initial_charge*, se marca el reseteo de los condensadores, se pone la salida digital a "0", se inicia el contador de la conversión total, y se pone a '0' la señal que indica el final de la conversión.
Estado 1: Initail_charge	`when Initial_charge =>` `  p_phi_in<='0';` `  rst_cal<='0';` `  rst_ini<='0';` `  p_state<=C2_charge;` `  p_counter<="00";`	En este estado se produce la carga de C_1 mediante I_{ref}, ya que phi_in será '1', actualizada antes de entrar en el estado. Por otra parte se resetean tanto ff_cal, como ff_ini, ya que no actúan en este estado. Por otra parte se establece que el siguiente estado será *C2_charge*. Por último se inicializa el contador para la conversión de los dos primeros bits en la fase MSB de la conversión.
Estado 2: C2_charge	`when C2_charge =>` `  rst_cal<='0';` `  if (comp_ini='0') then` `  phi_2<='1';` `  phip2<='1';` `  p_counter<=counter+1;` `  else` `  p_counter<="00";` `  p_state<=neg_res;` `  p_data(9 downto 8)<=counter;` `  phi_1<='1';` `  p_bit<="111";` `  end if;`	En este estado se carga el condensador C_2 con I_{ref} hasta que se igualan las tensiones en ambos condensadores. Si se observa el código en primer lugar se resetea ff_cal ya que no actúa en este estado. La primera condición establece que el comparador aún no se ha puesto a '1', ya que comp_ini cambia cuando se produce por primera vez este evento. Por lo tanto mientras se da esta condición el condensador C_2 se está cargando y la salida de este condensador está conectada al terminal positivo del comparador. (phi_2<='1'; phip2<='1';). Por último se actualiza el contador de la conversión de los bits MSBs Cuando se da la segunda condición significa que se ha producido la primera comparación, y por este motivo se establece que el siguiente estado es *neg_res*, se asignan los dos bits MSBs y se inicia la carga de C_1 modificando la salida phi_1<='1'.

Estado 3: Neg_res	`when neg_res =>` `  rst_ini<='0';` `  p_counter<=counter+1;` `  if (counter="11") then` `  p_data(conv_integer(bit))<='1';` `  p_state<=assign_1;` `  p_bit<=bit-1;` `  elsif (counter="10" and flag1='1' and flag2='1') then` `  p_data(conv_integer(bit))<='0';` `  p_state<=assign_zero;` `  p_bit<=bit-1;` `  elsif (flag1='1' and flag2='1') then` `  phi_2<='1';` `  p_counter<="00";` `  p_state<=pos_res;` `  p_data(conv_integer(bit))<=NOT(counter(0));` `  p_bit<=bit-1;` `  rst_cal<='0';` `  elsif (comp_cal='0') then` `  phi_2<='1';` `  phip2<='1';` `  elsif (comp_cal_n='0') then` `  phi_1<='1';` `  phip1<='1';` `  end if;`	En este estado en primer lugar se desactiva el ff_ini ya que éste solo entra en funcionamiento en la carga inicial y se actualiza el contador general de la conversión. En la primera condición se comprueba si el residuo cae tan cerca del flanco de reloj previo que no es capaz de detectarse, en cuyo caso se le asigna un '1' al bit correspondiente y se estable que el siguiente estado es *assign_1.* En la segunda condición se comprueba si el residuo cae tan cerca del flanco de reloj siguiente que no es capaz de detectarse, en cuyo caso se le asigna un '0' al bit correspondiente y se establece que el siguiente estado es *assign_zero.* En la tercera condición se comprueba si se han efectuado las comparaciones necesarias (tres) para la conversión del bit correspondiente, en cuyo caso se asigna el bit que se ha obtenido como resultado. Para esta comprobación se utilizan las banderas flag1 y flag2 que se obtienen de la salida de los biestables ff_flag1 y ff_flag2 respectivamente. Con las otras dos condiciones se controla la carga y la conexión al terminal positivo y negativo del comparador por parte de los condensadores. Para comprobar la condición se utiliza la salida de ff_cal, el cual está controlado por la salida del comparador vcomp.

Estado 4: Pos_res

```
when pos_res =>
   rst_ini<='0';
   p_counter<=counter+1;
   if (counter="11") then
      if (bit="000") then
           p_data(0)<='0';
           if (count_conv="10100") then
             p_end_conv<='1';
             p_state<=Reposo;
           else
             p_state<=wait_end;
           end if;
      else
           p_data(conv_integer(bit)) <='0';
           p_state<=assign_zero;
      end if;
   elsif (counter="10" and flag1='0' and flag2='0') then
      if (bit="000") then
           p_data(0)<='1';
           if (count_conv="10100") then
             p_end_conv<='1';
             p_state<=Reposo;
           else
             p_state<=wait_end;
           end if;
      else
           p_data(conv_integer(bit)) <='1';
           p_state<=assign_1;
      end if;

    elsif (flag1='0' and flag2='0') then
           rst_cal<='0';
           phi_1<='1';
           p_counter<="00";
           p_data(conv_integer(bit))<=counter(0);
           if (bit="000") then
               if (count_conv="10100") then
                 p_end_conv<='1';
                 p_state<=Reposo;
               else
                 p_state<=wait_end;
               end if;
           else
             p_state<=neg_res;
             p_bit<=bit-1;
           end if;
    elsif (comp_cal='0') then
      phi_1<='1';
      phip1<='1';
    elsif (comp_cal_n='0') then
      phi_2<='1';
      phip2<='1';
    end if;
```

En este estado en primer lugar se desactiva el ff_ini ya que éste solo entra en funcionamiento en la carga inicial y se actualiza el contador general de la conversión.

En la primera condición se comprueba si el residuo cae tan cerca del flanco de reloj previo que no es capaz de detectarse, en cuyo caso se le asigna un '0' al bit correspondiente y se estable que el siguiente estado es *assign_zero.* Dentro de esta condición a su vez se comprueba, si se da la primera, si se ha llegado al final de la conversión (veintidós ciclos de reloj) en cuyo caso se pasa al estado de *Reposo*, y si no se da se pasa al estado de *wait_end.*

En la segunda condición se comprueba si el residuo cae tan cerca del flanco de reloj previo que no es capaz de detectarse, en cuyo caso se le asigna un '1' al bit correspondiente y se estable que el siguiente estado es *assign_1.* Dentro de esta condición a su vez se comprueba, si se da la primera, si se ha llegado al final de la conversión (veintidós ciclos de reloj) en cuyo caso se pasa al estado de *Reposo*, y si no se da se pasa al estado de *wait_end.*

En la tercera condición se comprueba si se han efectuado las comparaciones necesarias (tres) para la conversión del bit correspondiente, en cuyo caso se asigna el bit que se ha obtenido como resultado. Para esta comprobación se utilizan las banderas flag1 y flag2 que se obtienen de la salida de los biestables ff_flag1 y ff_flag2 respectivamente. Dentro de esta condición a su vez se comprueba, si se da la primera, si se ha llegado al final de la conversión (veintidós ciclos de reloj) en cuyo caso se pasa al estado de *Reposo*, y si no se da se pasa al estado de *wait_end.* Con las otras dos condiciones se controla la carga y la conexión al terminal positivo y negativo del comparador por parte de los condensadores. Para comprobar la condición se utiliza la salida de ff_cal, el cual está controlado por la salida del comparador vcomp.

Estado 5: Assign_1	`when assign_1 =>` `  if (bit="000") then` `  p_data(0)<='1';` `  if(count_conv="10100") then` `  p_state<=Reposo;` `  p_end_conv<='1';` `  else` `  p_state<=wait_end;` `  end if;` `  else` `  p_data(conv_integer(bit))<='1';` `  p_bit<=bit-1;` `  end if;`	Cuando se pasa por este estado es porque se ha dado algunas de las condiciones que se describen en el estado anterior donde no se detecta el residuo. En primer lugar se comprueba si el bit que se está convirtiendo es el menos significativo, en cuyo caso si han transcurrido los veintidós ciclos de reloj necesarios para una conversión se pasa al estado de *Reposo,* y en otro caso se pasa al estado de *Wait_end* asignando primero un ´1´ al bit correspondiente. En caso de que no se esté en el último bit de la conversión se va asignando un '1' recursivamente a todos los bits restantes ya que según las condiciones del residuo se establece que todos los bits restantes de la conversión deben estar a ´1´.
Estado 6: Assign_zero	`when assign_zero =>` `  if (bit="000") then` `  p_data(0)<='0';` `  if(count_conv="10100") then` `  p_state<=Reposo;` `  p_end_conv<='1';` `  else` `  p_state<=wait_end;` `  end if;` `  else` `  p_data(conv_integer(bit))<='0';` `  p_bit<=bit-1;` `  end if;`	Cuando se pasa por este estado es porque se ha dado algunas de las condiciones que se describen en el estado anterior donde no se detecta el residuo. En primer lugar se comprueba si el bit que se está convirtiendo es el menos significativo, en cuyo caso si han transcurrido los veintidós ciclos de reloj necesarios para una conversión se pasa al estado de *Reposo,* y en otro caso se pasa al estado de *Wait_end* asignando primero un ´0´ al bit correspondiente. En caso de que no se esté en el último bit de la conversión se va asignando un '0' recursivamente a todos los bits restantes ya que según las condiciones del residuo se establece que todos los bits restantes de la conversión deben estar a ´0´.
Estado 7: Wait_end	`  when wait_end =>` `  if(count_conv="10100") then` `  p_state<=Reposo;` `  p_end_conv<='1';` `  end if;` `  end case;` `end process;`	Cuando se pasa por estado es porque ya se han convertido todos los bits, pero aun no se han transcurrido los veintidós ciclos de reloj necesarios para que se dé una conversión completa. Por tanto lo que se hace en este estado es esperar a que la conversión termine, en cuyo caso estable que el siguiente estado es el estado de *Reposo.* En esta parte del código también se finaliza el proceso concurrente con la funcionalidad característica del diseño

	sync: process(clk,resetz) begin if(resetz='0') then state<=Reposo; data_s<="0000000000"; counter<="00"; bit<="000"; end_conv_s<='0'; phi_in_s<='0'; count_conv<="00000"; elsif(clk='1' and clk'event) then state<=p_state; data_s<=p_data; counter<=p_counter; bit<=p_bit; end_conv_s<=p_end_conv; phi_in_s<=p_phi_in; count_conv<=p_count_conv; end if; end process;	En esta parte del código se describe el proceso de sincronismo del diseño, en el cual se puede observar que si se efectúa un reset a nivel bajo se inicializan las variables como contadores y el dato de salida, y se establece que el primer estado tras el reset será el estado de reposo. Por otro lado se si da un flanco de subida en el reloj, se actualizan todas las variables y se producen las transiciones a los distintos estados si es el caso.

Con esta queda descrito el diseño a nivel de código VHDL, no obstante en el anexo I se incluirá el código completo con los comentarios incluidos.

5.1.2.2 Simulación

A continuación, se muestran una serie de figuras en las que se pueden ver conversiones completas en las que se dan los tres casos posibles, es decir, que se detecten los residuos correctamente y no hay excepciones, que no se detecten los residuos por caer demasiado cerca del ciclo de reloj previo, y que no se detecte el residuo por caer demasiado cerca del ciclo de reloj siguiente. Para ver el correcto funcionamiento de la máquina de estados se han incluido las salidas y las entradas de la máquina de estados, junto con algunas de las señales internas más importantes como son las salidas de los flip-flops internos de la máquina o la señal que indica el estado en el que se encuentra la máquina en cada momento.

Para realizar estas simulaciones se ha implementado un test-bench en el cual se han incluido valores razonables para vcomp, ya que se trata de la salida del comparador, la cual es una señal que no se puede controlar a priori.

En la *Figura 44* se puede observar una conversión completa. Si se observa la señal que indica los estados, la secuencia de estados es Reposo, Initial_charge, C2_charge, Neg_res, Pos_res, estos dos últimos repetidamente hasta que se llega a Wait_end una vez que se han convertido todos los bits. Tras analizar esta cuestión se deduce que se trata de una conversión normal, ya que se detectan todos los residuos correctamente. Por otra parte también se pude observar que las salidas de los distintos flip-flops están controladas por la señal vcomp, que es la salida del comparador, de manera que la máquina es capaz de llevar la cuenta de las comparaciones mediante las banderas flag1 y flag2 propiciando las distintas transiciones de estado como se ha explicado anteriormente en la descripción del código. Si se observan tanto los contadores, como estas banderas se puede comprobar que las transiciones de estados se producen cuando se dan las condiciones que se han explicado en la descripción del código, por lo que se puede concluir que ante esta situación la máquina de estados funciona correctamente.

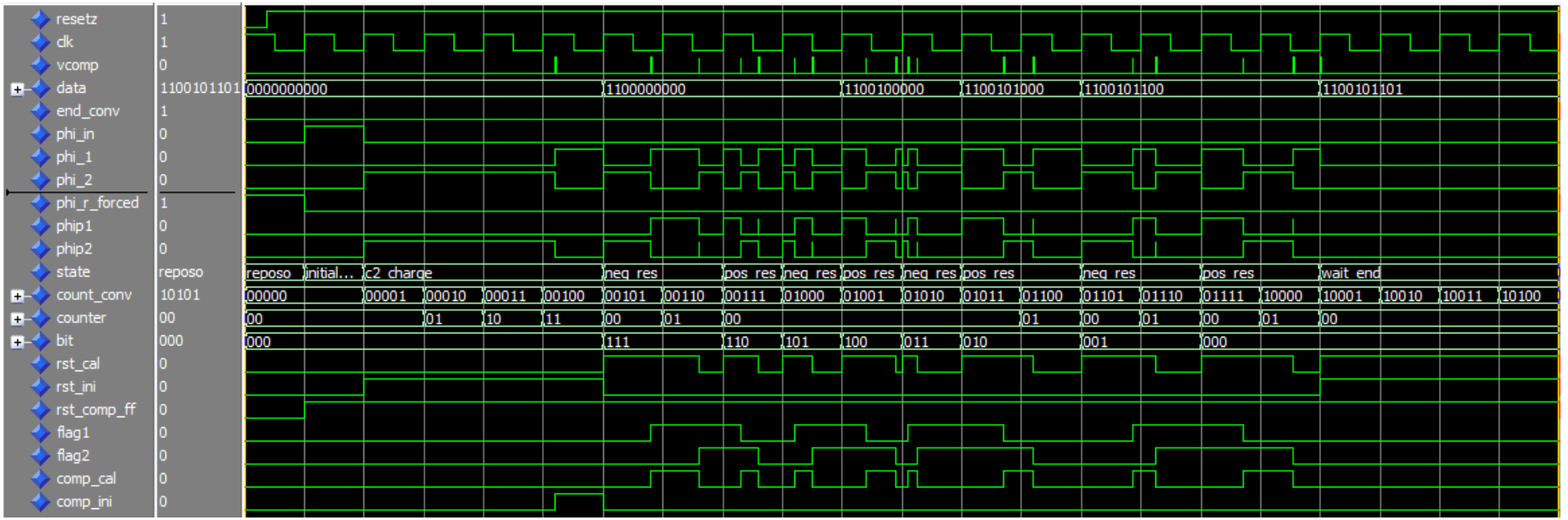

Figura 44. Conversión completa. Detección correcta de los residuos.

En la *Figura 45* se puede observar una conversión completa. Si se observa la señal que indica los estados, la secuencia de estados es Reposo, Initial_charge, C2_charge, Neg_res, Pos_res, estos dos últimos repetidamente hasta que se llega a Assign_1, que como se ha explicado anteriormente se debe a que el residuo no se detecta por caer demasiado cerca del ciclo de reloj previo. Finalmente se llega a Wait_end una vez que se han convertido todos los bits. Cabe destacar que una vez que se entra en Assign_1 los bits restantes que quedan por convertir se ponen todos a '1'. Por otra parte también se pude observar que las salidas de los distintos flip-flops están controladas por la señal vcomp, que es la salida del comparador, de manera que la máquina es capaz de llevar la cuenta de las comparaciones mediante las banderas flag1 y flag2 propiciando las distintas transiciones de estado como se ha explicado anteriormente en la descripción del código. Si se observan tanto los contadores, como estas banderas se puede comprobar que las transiciones de estados se producen cuando se dan las condiciones que se han explicado en la descripción del código, por lo que se puede concluir que ante esta situación la máquina de estados funciona correctamente.

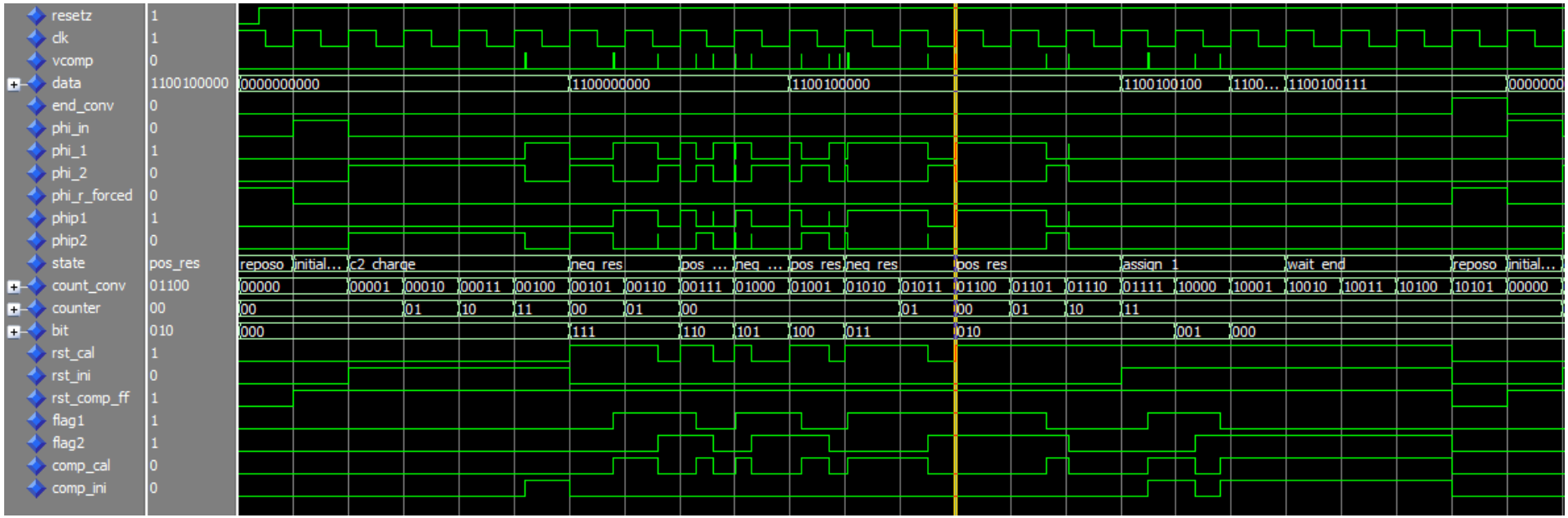

Figura 45. Conversión completa. Residuo demasiado cercano al ciclo de reloj previo

En la *Figura 46* se puede observar una conversión completa. Si se observa la señal que indica los estados, la secuencia de estados es Reposo, Initial_charge, C2_charge, Neg_res, Pos_res, estos dos últimos repetidamente hasta que se llega a Assign_zero, que como se ha explicado anteriormente se debe a que el residuo no se detecta por caer demasiado cerca del ciclo de reloj siguiente. Finalmente se llega a Wait_end una vez que se han convertido todos los bits. Cabe destacar que una vez que se entra en Assign_zero los bits restantes que quedan por convertir se ponen todos a '0'. Por otra parte también se pude observar que las salidas de los distintos flip-flops están controladas por la señal vcomp, que es la salida del comparador, de manera que la máquina es capaz de llevar la cuenta de las comparaciones mediante las banderas flag1 y flag2 propiciando las distintas transiciones de estado como se ha explicado anteriormente en la descripción del código. Si se observan tanto los contadores, como estas banderas se puede comprobar que las transiciones de estados se producen cuando se dan las condiciones que se han explicado en la descripción del código, por lo que se puede concluir que ante esta situación la máquina de estados funciona correctamente.

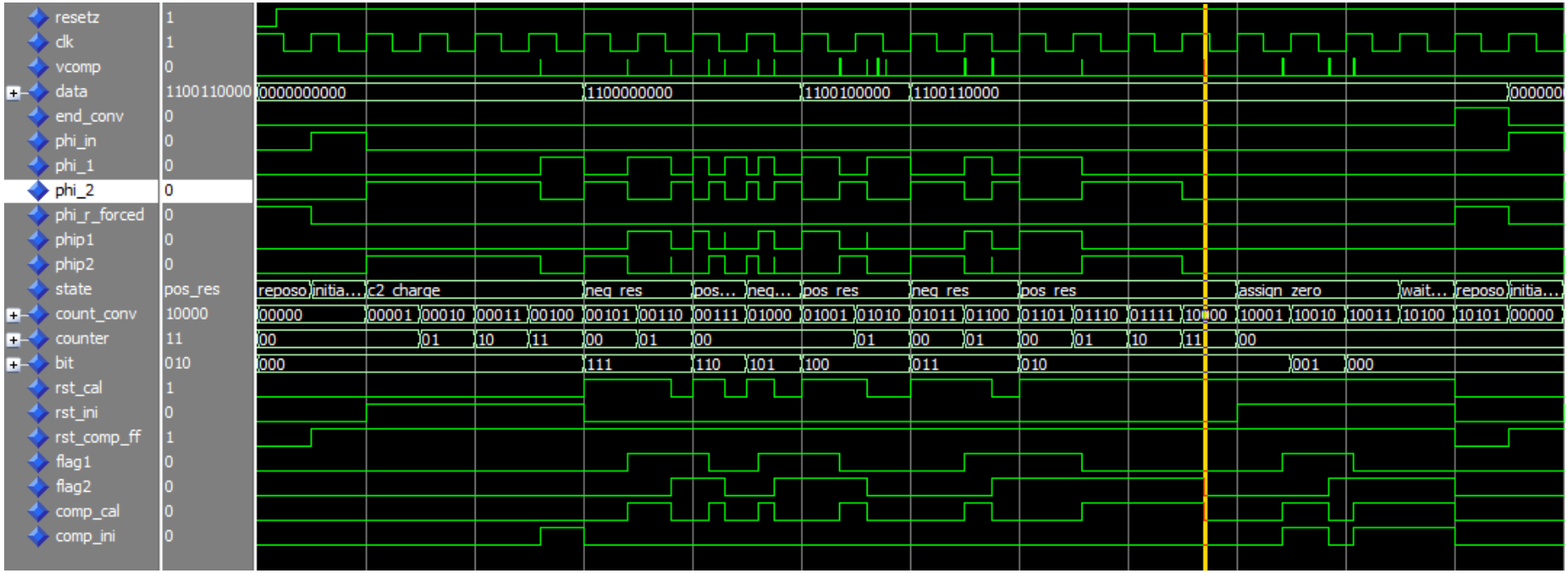

Figura 46. Conversión completa. Residuo demasiado cercano al ciclo de reloj siguiente

5.2 Implementación del Diseño en Matlab/Simulink

En este apartado se realizará una explicación de cómo se ha realizado la implementación del convertidor en Matlab/Simulink. Para implementar el convertidor se ha diseñado cada elemento por separado en Matlab/simulink, es decir, se han implementado por separado el comparador, la máquina de estados, switchs, capacidades y la lógica necesaria. Una vez realizado cada bloque, se han conectado para completar el diseño completo del convertidor.

Una vez realizado el diseño del comparador, y comprobado su correcto funcionamiento, mediante distintas simulaciones, se han realizado una serie de análisis paramétricos. En estos análisis paramétricos se ha intentado evaluar el funcionamiento real del convertidor, para lo cual se han estudiado algunas de las medidas explicadas en apartado 2.1.4. Concretamente se han estudiado tanto la INL y la ENOB para evaluar la variación de parámetros como el posible retraso del comparador, la tensión de offset dependiente e independiente de la señal del comparador y el desapareamiento de las capacidades. Se han tomado estos parámetros porque en un convertidor real es muy probable que se den todos estos efectos debido a causas como temperatura, topología del circuito, etc.

5.2.1 Diseño de los distintos bloques del sistema

5.2.1.1 Capacidad

Para el diseño de las capacidades en Matlab/Simulink se ha tenido en cuenta que en el modelo completo del convertidor cuenta con una fuente de corriente que carga un condensador (o no). Tanto la carga como la descarga del condensador estará controlada por unos switches, que a su vez estarán controlados por unas señales que generará la máquina de estados. En definitiva, en el convertidor, en lo que se refiere a los condensadores se puede encontrar lo que se ve en la *Figura 47.*

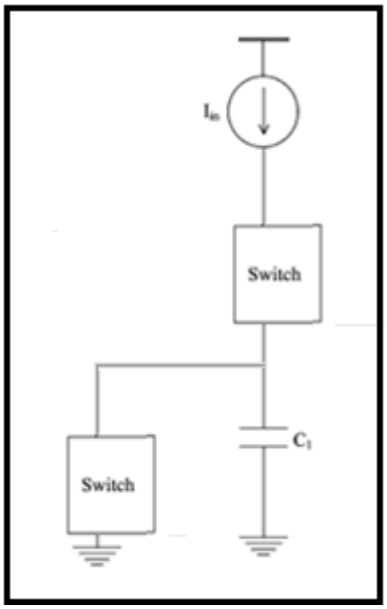

Figura 47. Bloque de Capacidad incluido en el ADC

Teniendo en cuenta lo comentado anteriormente, para diseñar el modelo en Matlab/Simulink se ha implementado un bloque que cuenta con dos puertos de entrada. Con el primer puerto se controla la fase de carga (1) y de descarga (0) del condensador, mientras que por el segundo se introduce la corriente de carga. En la *Figura 48* se puede observar el aspecto del bloque.

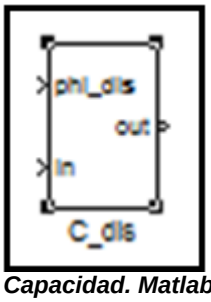

Figura 48. Capacidad. Matlab/Simulink

Para implementar este bloque se ha utilizado un switch proporcionado por Matlab/Simulink para alternar la carga y descarga del condensador, un integrador y dos ganancias que nos proporcionan la constantes de carga y de descarga del condensador. El valor de las contantes de se corresponde con el valor de las constantes de tiempo del condensador T=RC, donde R es la resistencia del switch, y C es la valor de la Capacidad, el cual será un parámetro sobre el cual se realizará un análisis para observar el efecto del mistmach de las capacidades. En la *Figura 49* se puede ver la implementación comentada anteriormente para realizar este bloque.

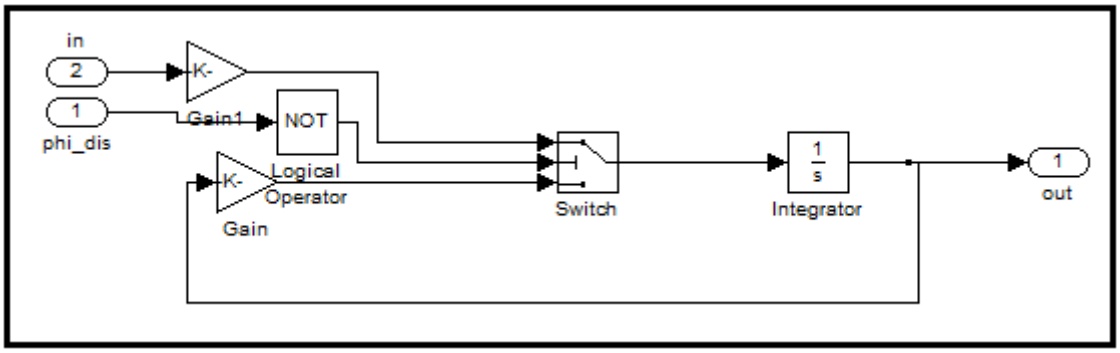

Figura 49. Implementación de capacidad. Matlab/Simulink

A continuación se puede observar en la *Figura 50* como se produce la carga y la descarga de la capacidad, donde se observa como cuando la fase de carga y descarga permanece a uno se produce la descarga del condensador, y cuando permanece a cero se produce la carga.

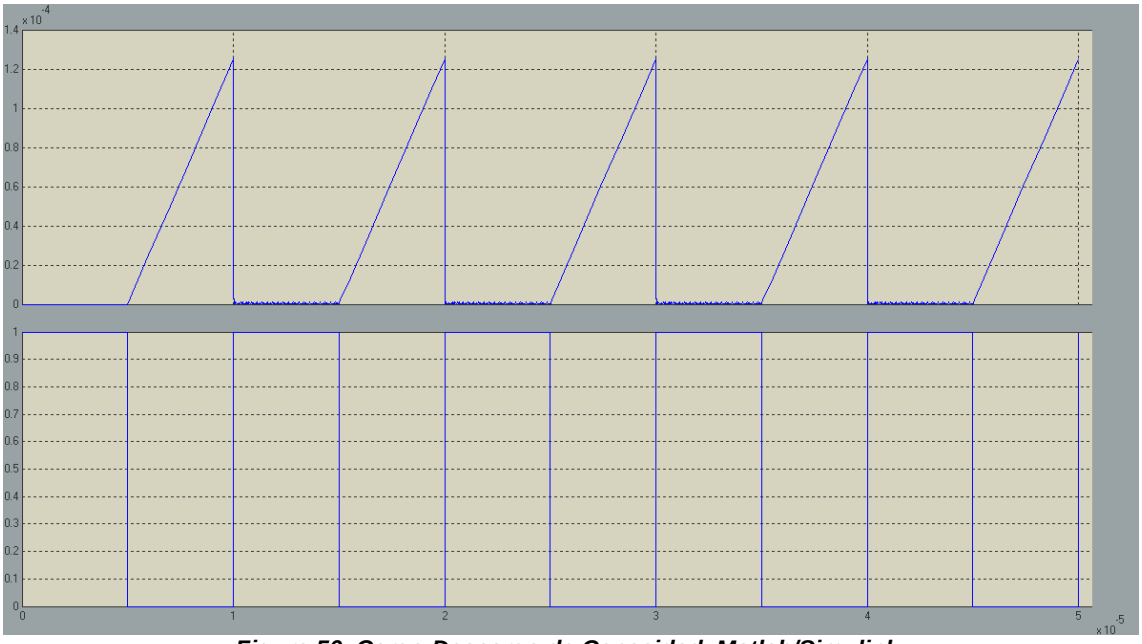

Figura 50. Carga-Descarga de Capacidad. Matlab/Simulink

5.2.1.2 Switches

Se ha implementado un switch que se conectará a los condensadores. Para esta función el switch propio de Matlab/Simulink no es válido ya que en el momento en el que la fase de carga está a cero, es decir, no se está cargando el condensador, es necesario que se mantenga el valor de tensión en este último para poder realizar la comparación. Este bloque consta de dos puertos de entrada, el primero para introducir la corriente de carga, y el segundo para introducir la fase de carga. A la salida tenemos la corriente de carga, ya sea el valor de corriente propiamente dicho en el caso de que se esté cargando el condensador, o cero en el caso de que se necesite mantener el valor de tensión en el comparador para que este pueda ser comparado. En la *Figura 51* se puede observar el aspecto del bloque.

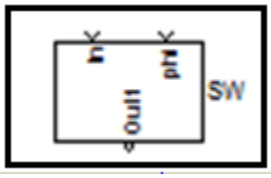

Figura 51. Switch. Matlab/Simulink

Para la implementación de este bloque simplemente se ha utilizado un switch que proporciona la propia herramienta, en el cual para la cuando la señal de carga se encuentra a uno la salida se corresponde con la señal de entrada del otro puerto, que se corresponde con la corriente de carga, mientras que cuando la señal de carga se encuentra a cero, en la salida se tiene un valor de corriente cero, la cual es necesaria para mantener el valor de tensión en el condensador. En la *Figura 52* se observa la implementación del switch comentada.

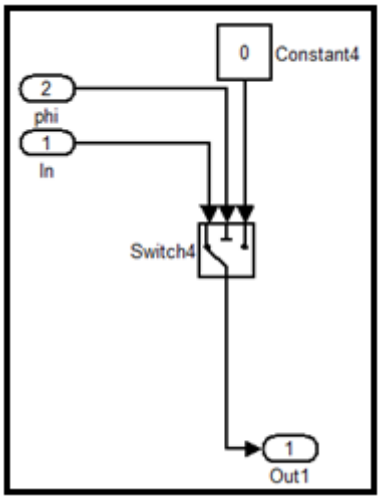

Figura 52. Implementación de switch. Matlab/Simulink

5.2.1.3 Lógica

Se han implementado dos bloques lógica, los cuales son necesarios para el correcto funcionamiento del convertidor.

El primer bloque de lógica, denominado *LogicPhiX*, consta de dos puertos de entrada y tres puertos de salida y será utilizado en el diseño como. A la entrada de los dos puertos llegan dos señales que se corresponden con dos salidas de la máquina de estados. Estas señales son *PhiX* (donde X es 1 o 2) y *Phi_r_f*, cuyas funciones se han explicado anteriormente en la descripción de la propia máquina de estados en el capítulo 5.1.1 En los puertos de salida de este bloque se generan tres señales con distintas funcionalidades. En primera señal es *Phi_r_IX* (donde X es 1 o 2), la cual tiene como destino otro bloque de lógica. La segunda señal es *Phi_r_cX* (donde X es 1 o 2), la cual tiene como función ordenar el reseteo de cada uno de los condensadores por separado, es decir, cuando X=1 se reseteará C_1 y cuando X=2 se reseteará C_2, en el caso de que esta señal tenga valor 1. Finalmente, la tercera señal es *PhiXs* (donde X es 1 o 2), la cual tiene como función ordenar la carga de los condensadores por

separado, es decir, cuando X=1 se cargará C_1 y cuando X=2 se cargará C_2, en el caso de que esta señal tenga valor 1. Puede parece que la función de esta última señal es la misma que tenían las salidas de la máquina de estados *Phi1* y *Phi2*, pero en realidad, estas dos últimas señales son entradas de este bloque de lógica y generan las señales *PhiXs* mediante la utilización de retrasos para generar fases no solapadas. En la *Figura 53* se puede observar el aspecto del bloque.

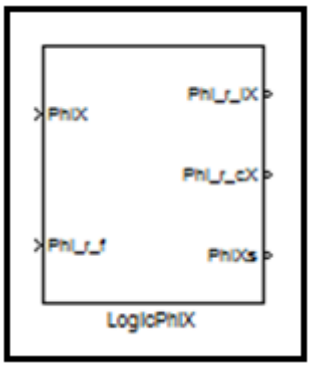

Figura 53. LogicPhiX. Matlab/Simulink

Para la implementación de *LogicPhiX* se han utilizado los bloques de lógica NOT, NOR y NAND proporcionados por Matlab/Simulink junto con retrasos en los que se puede asignar el valor que se estime oportuno proporcionados por la herramienta. La función de estos retrasos y estas puertas es la generación de las señales descritas anteriormente. En la *Figura 54* se observa la implementación del bloque comentada.

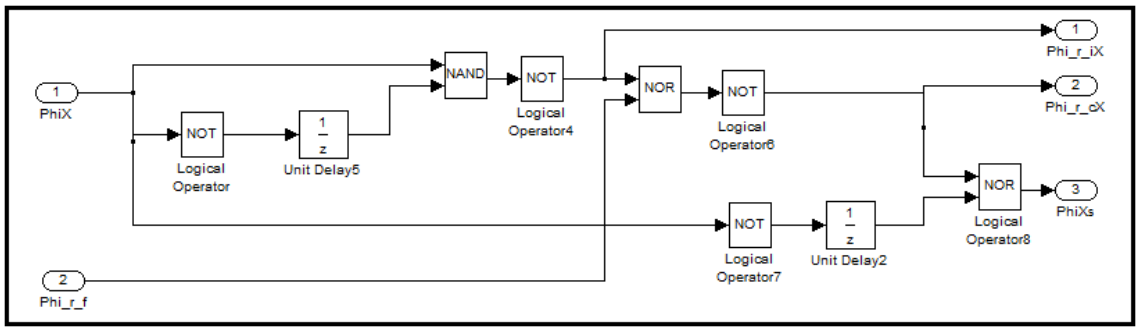

Figura 54.Implementación de LogicPhiX. Matlab/Simulink

El siguiente bloque, denominada *LogicComp_en_f*, tiene cuatro puertos de entrada en los cuales se reciben dos señales de la máquina de estados *Phip1* y *Phip2,* y dos señales generadas por el bloque de lógica anterior *Phi_r_l_1* y *Phi_r_l_2* . Por otra parte tiene un único puerto de salida en el cual se genera una salida *Comp_en_f,* la cual tiene como función habilitar la comparación, es decir, indica al comparador cuando debe comparar. En la *Figura 55* se puede observar el aspecto del bloque.

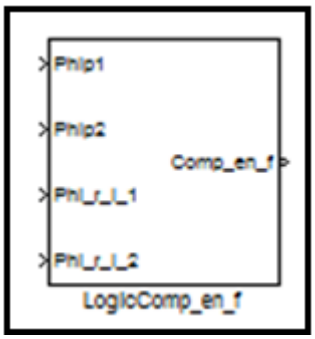

Figura 55. LogicComp_en_f. Matlab/Simulink

Para la implementación de *LogicPhiX* se han utilizado los bloques de lógica NOTy NOR proporcionados por Matlab/Simulink. En la *Figura 56*Figura 54 se observa la implementación del bloque comentada.

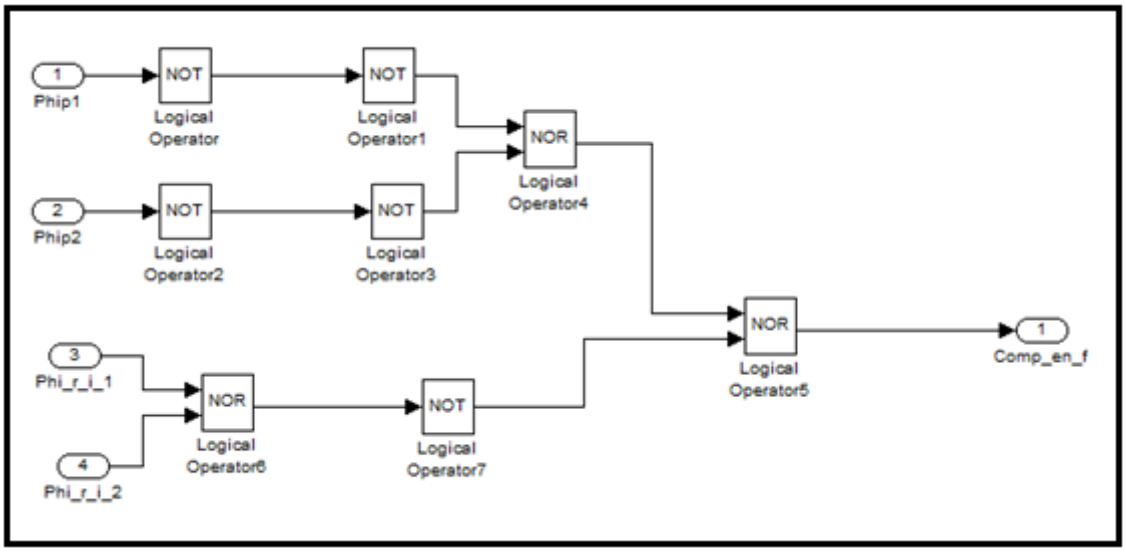

Figura 56. Implementación LogicComp_en_f. Matlab/Simulink

Una vez se han implementado los bloques descritos anteriormente, se ha creado un bloque denominado denominado *Phases,* en el cual se han introducido los bloques anteriores, conectados de manera adecuada, ya que todos los bloques anteriores están relacionados con la generación de fases para la carga y descarga de condensadores y la propia carga y descarga de estos últimos. De esta manera se tienen todos los bloques relacionados a su vez en un mismo bloque para así analizar los errores y realizar la depuración de una forma ordenada y eficiente. Este bloque consta de ocho puertos de entrada y tres puertos de salida. De los ocho puertos de entrada seis se corresponden con salidas de la máquina de estados cuya funcionalidad se ha explicado en el apartado 5.1.1, las cuales son *Phi_1, Phi_2, Phip1, Phip2, Phi_r_f y Phi_ini*. Los otros dos puertos de entrada se corresponden con las corrientes de carga de los condensadores, la corriente de referencia I_{ref} y la corriente de entrada I_{in}. Los tres puertos de salida se corresponden por un lado con las salidas de los dos bloques de capacidades, y por otro lado con la señal que habilita la comparación. En la *Figura 57* se puede observar el aspecto del bloque.

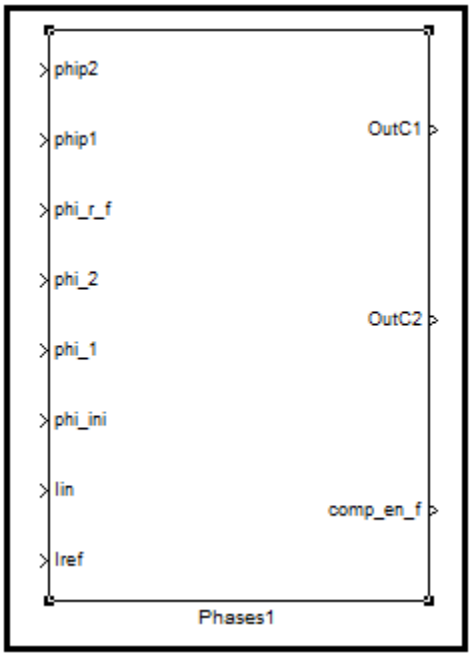

Figura 57.Phases. Matlab/Simulink

Para la implementación de este bloque, es decir, para conectar los bloques descritos anteriormente, se ha tenido en cuenta la funcionalidad de las distintas señales, las cuales se han explicado en capítulos anteriores. En la *Figura 58* se observa la implementación del bloque.

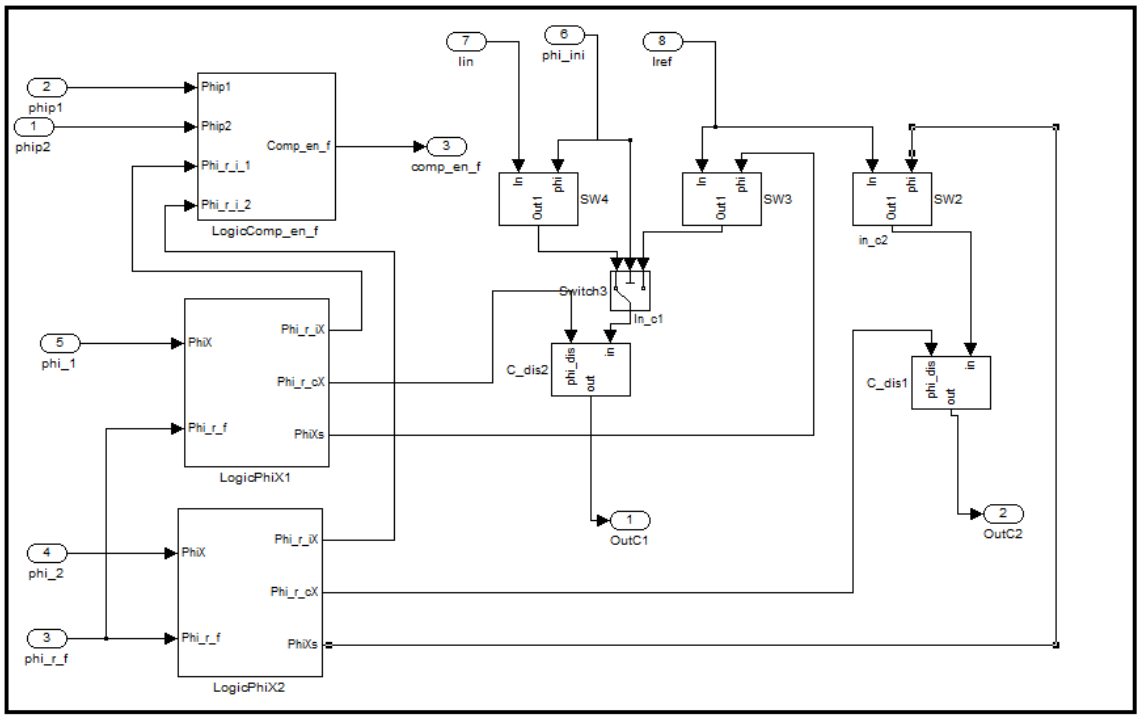

Figura 58. Implementación Phases. Matlab/Simulink

En las siguientes figuras se muestra el resultado de varias simulaciones donde se pueden observar distintas las señales más importantes del bloque anterior.

En la **¡Error! No se encuentra el origen de la referencia.** en la gráfica superior se observan tres señales. La señal de color azul se corresponde con la salida del condensador C_1. Se puede observar que la carga se produce cuando la señal de color rojo está activa (es decir, está a uno), la cual se corresponde con la señal *Phi1s*, que como se ha explicado

anteriormente habilita la carga de este condensador y se calcula en función de *Phi1*, siendo una versión retrasada de esta última, evitando que se solapen las fases. Otra señal que se puede observar en esta gráfica es la de color rosa, la cual se corresponde con *Phi_in*, a través de la cual se produce la carga del condensador C_1 en el estado de carga inicial.

En la **¡Error! No se encuentra el origen de la referencia.**, en la gráfica inferior se observan dos señales. La señal de color azul se corresponde con la salida del condensador C_2. Se puede observar que la carga se produce cuando la señal de color rojo está activa (es decir, está a uno), la cual se corresponde con la señal *Phi2s*, que como se ha explicado anteriormente habilita la carga de este condensador y se calcula en función de *Phi2*, siendo una versión retrasada de esta última, evitando que se solapen las fases.

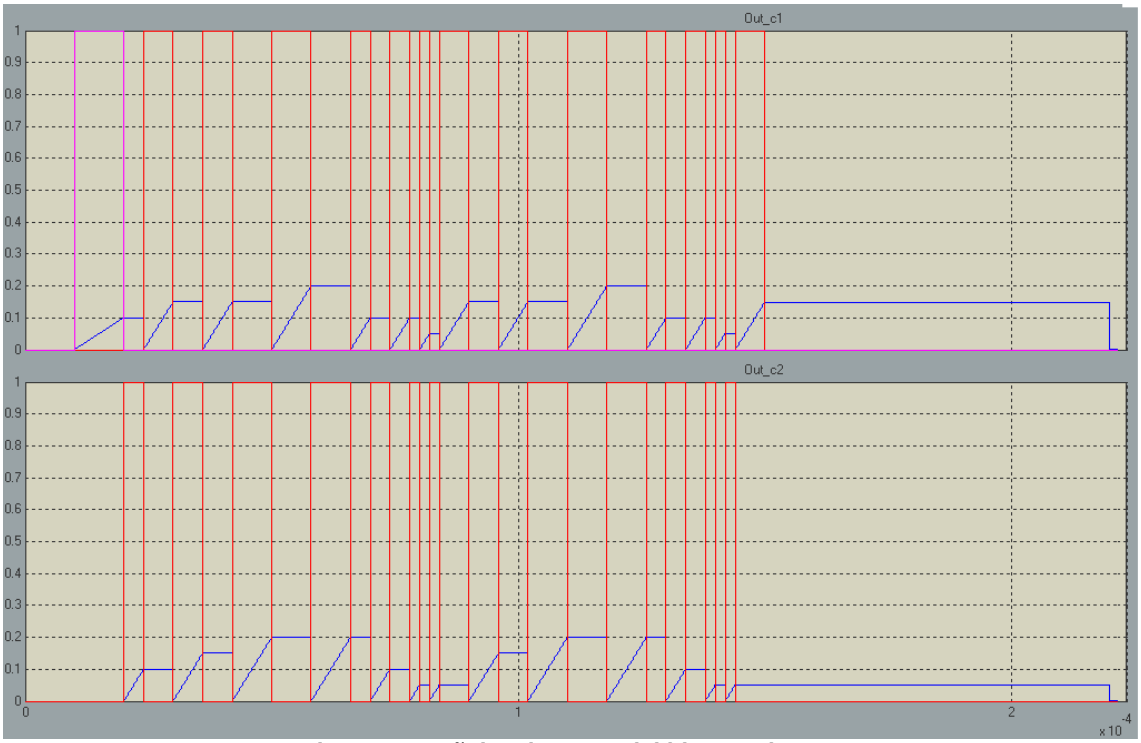

Figura 59. Señales de carga del bloque Phases

En la **¡Error! No se encuentra el origen de la referencia.** en la gráfica superior se observan dos señales. La señal de color azul se corresponde con la salida del condensador C_1, en la cual se puede observar cómo se produce la carga hasta que se alcanza un valor de señal determinado, el cual vendrá dado por el resultado de la comparación. Una vez que se alcanza éste valor determinado, la señal se mantiene hasta que la señal de color rosa se activa, en cuyo caso se reseteará el condensador y el valor de señal en el condensador pasará a cero. Esta señal de color rosa se corresponde con *Phi_r_c1*, la cual se calcula en función de dos de las salidas de la máquina de estados *Phi1* y *Phi_r_f.*

En la **¡Error! No se encuentra el origen de la referencia.** en la gráfica superior se observan dos señales. La señal de color azul se corresponde con la salida del condensador C_2, en la cual se puede observar cómo se produce la carga hasta que se alcanza un valor de señal determinado, el cual vendrá dado por el resultado de la comparación. Una vez que se alcanza éste valor determinado, la señal se mantiene hasta que la señal de color rojo se activa, en cuyo caso se reseteará el condensador y el valor de señal en el condensador pasará a cero. Esta señal de color rojo se corresponde con *Phi_r_c2*, la cual se calcula en función de dos de las salidas de la máquina de estados *Phi2* y *Phi_r_f.*

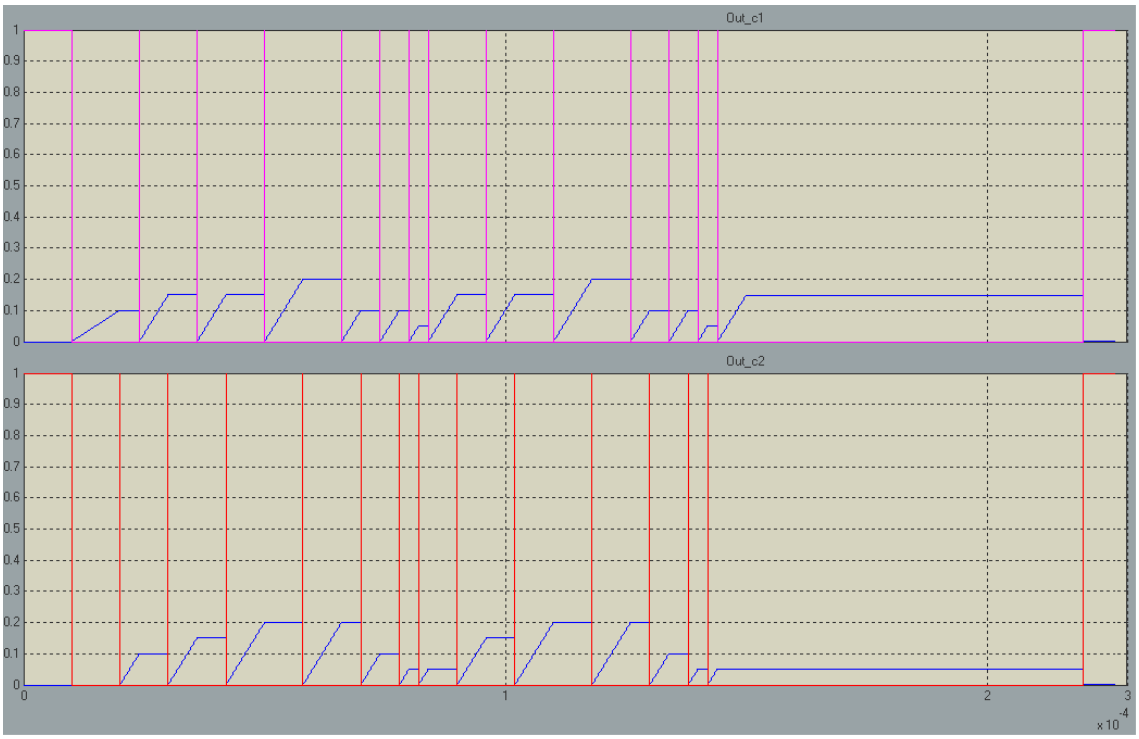

Figura 60. Señales de descarga del bloque Phases

5.2.1.4 Comparador

Un bloque de especial importancia en la arquitectura es el comparador, ya que el resultado de la conversión depende fuertemente del resultado de la comparación. Este bloque tiene dos puertos de entrada, y de un puerto de salida. Como en todo comparador, las entradas se corresponden con los terminales positivo y negativo, y el puerto de salida se corresponde con el resultado de la comparación, la cual ser activa cuando las dos señales conectadas a los terminales tengan el mismo valor y cero en otro caso. Se puede echar en falta en este bloque un puerto de entrada en la que se habilite (o no) la comparación, ya que hay una señal generada para este efecto por los bloques anteriormente descritos, pero esto no se da porque la habilitación se da mediante circuitería externa con una simple puerta AND conectada a la salida del comparador. En la *Figura 61* se puede observar el aspecto del bloque.

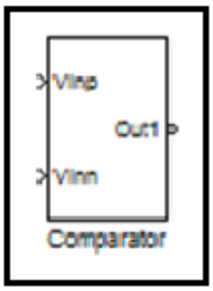

Figura 61. Comparador. Matlab/Simulink

Para implementar este bloque simplemente se ha utilizado un bloque *Add* que proporciona la herramienta para realizar restas (o sumas). Cuando al producirse la resta el resultado es cero entra en acción el elemento *Relay* de Matlab/Simulink, en el cual se pasa de cero a uno cuando se da este hecho. Por otro lado se ha incluido un bloque *Unit delay* para poder hacer un análisis paramétrico del retraso del comparador. Adicionalmente para añadir tanto un offset dependiente de la señal, como un offset independiente de la señal se ha incluido un bloque de ganancia (*gain*) para el primero, y un bloque constante (*constant*) para el segundo, de manera que se puedan analizar estos parámetros.

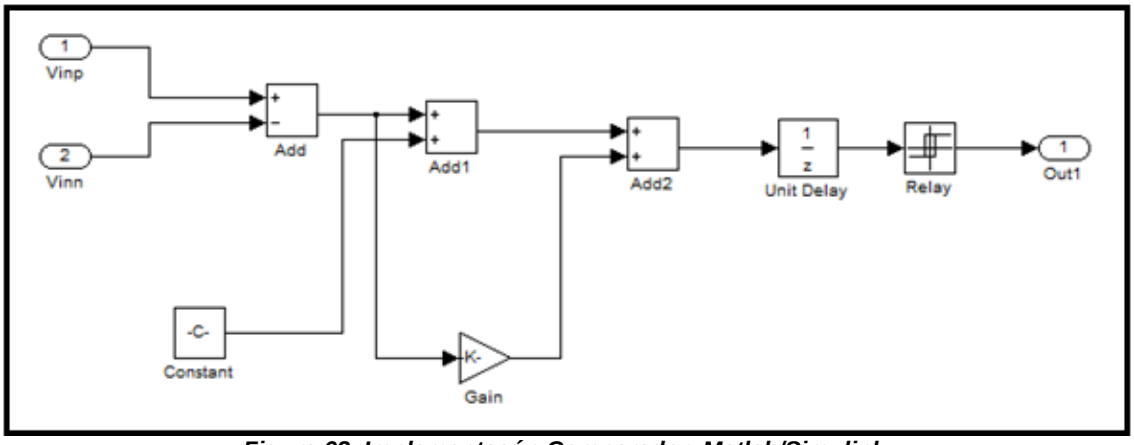

Figura 62. Implementacón Comparador. Matlab/Simulink

En las siguientes figuras se mostrará el resultado de la simulación del comparador variando los diferentes parámetros. Para mostrar un resultado más claro del funcionamiento se han utilizado valores mayores a los que se corresponden con el funcionamiento real del comparador. Estos valores serán escalados cuando se incluya este bloque en el modelo.

En primer lugar se ha conectado una rampa al terminal positivo, mientras que al terminal negativo se ha conectado una constante de 0.3, de manera que se puede observar que en el momento en el que la rampa alcanza este valor la salida del comparador cambia su valor de cero a uno. En la *Figura 63* se hace patente este efecto.

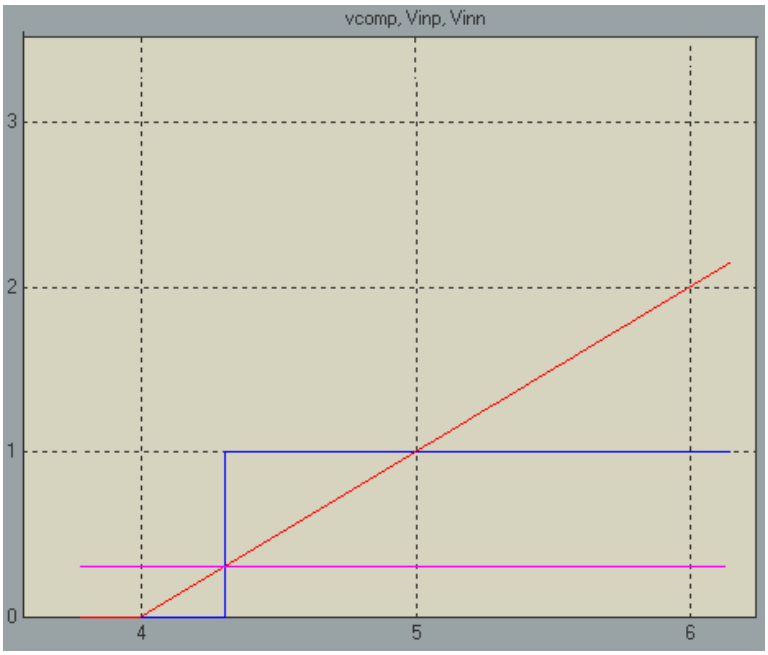

Figura 63. Funcionamiento del comparador. Matlab/Simulink

Para comprobar el correcto funcionamiento del retraso se le ha dado un valor de 0.2 a este mediante un parámetro denominado *Tdelay*. En este caso el cambio en la salida del comparador se producirá 0.2 unidades de tiempo posterior al instante en el que la rampa alcanza el valor de 0.3. Se observa en la *Figura 64* se puede observar como la rampa alcanza el valor de 0.3 en 4.3 unidades de tiempo, y efectivamente el cambio en la salida se produce en 4.5, 0.2 unidades de tiempo más tarde, de manera que el funcionamiento con retraso es correcto.

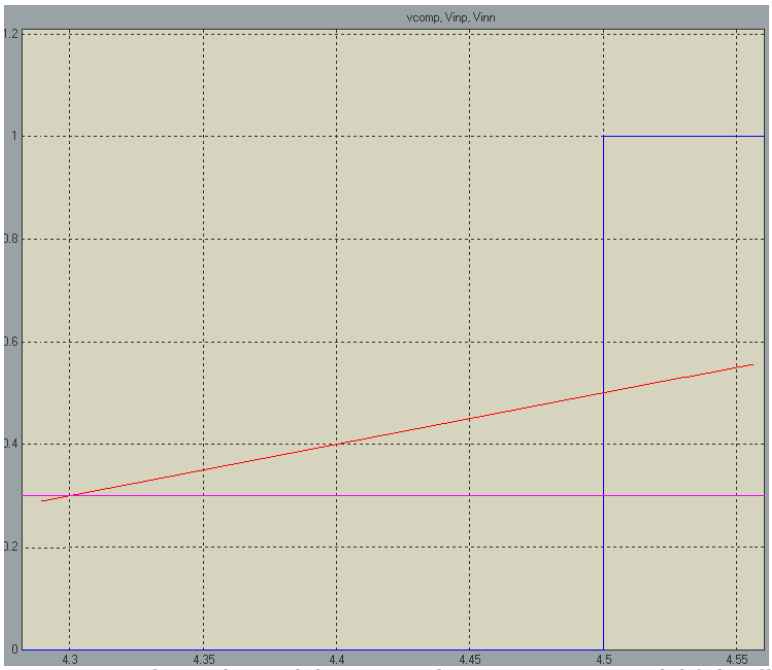

Figura 64. Funcionamiento del comparador con retraso. Matlab/Simulink

Para comprobar el correcto funcionamiento del offset independiente de la señal se le ha dado un valor de 0.1 a este mediante un parámetro denominado *V_offset*. En este caso el cambio en la salida del comparador se producirá cuando la rampa alcance el valor de la constante más 0.1. Se observa en la *Figura 65* que la rampa alcanza a la constante (0.3) y como el cambio se produce cuando llega a 0.4, es decir 0.1 por encima de su valor, de manera que el funcionamiento con offset es correcto.

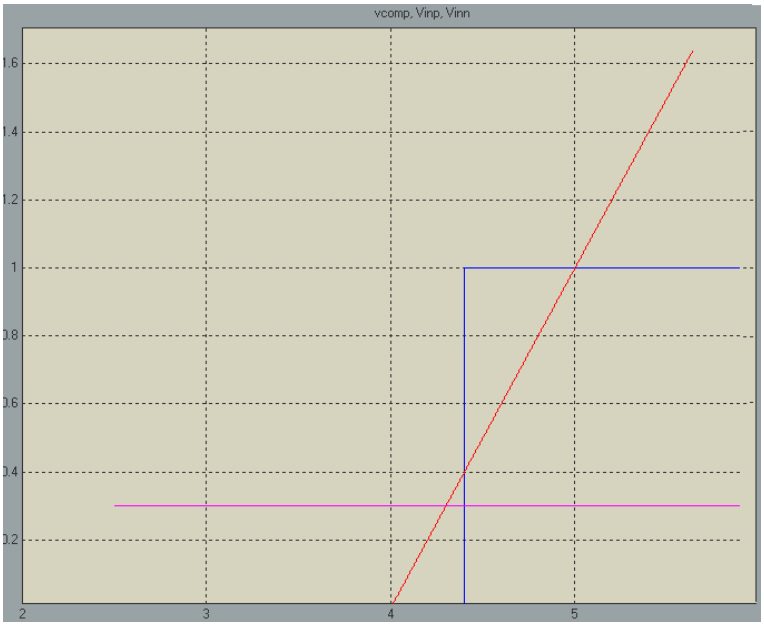

Figura 65. Funcionamiento del comparador con offset independiente. Matlab/Simulink

Para comprobar el correcto funcionamiento del offset dependiente de la señal se le ha dado un valor de 0.005 a este mediante un parámetro denominado *Depend_offset*. En este caso es más complicado saber cuándo se debe producir el cambio en el comparador a priori por tratarse de un offset dependiente de la señal. Lo que se observa en la *Figura 66* es que con este valor para el offset no se aprecia apenas diferencia con el funcionamiento normal, por lo que se puede concluir que no afecta demasiado.

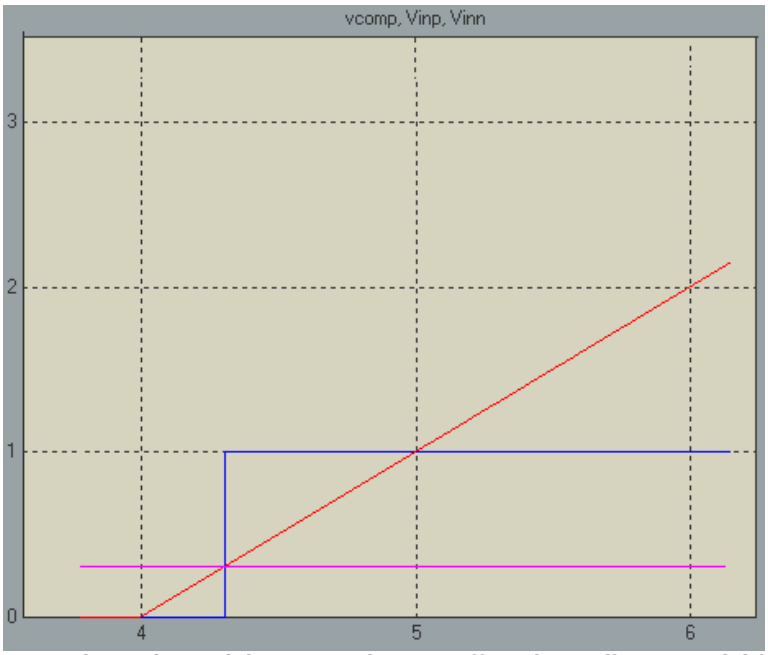

Figura 66. Funcionamiento del comparador con offset dependiente. Matlab/Simulink

5.2.1.5 Maquina de estados: Stateflow

Para implementar la máquina de estados en Matlab/Simulink se ha trasladado e diseñol realizado en VHDL mediante la un bloque *Stateflow* que proporciona la herramienta. *Stateflow* permite trabajar diseñando diagramas de flujo, de forma que proporciona una alternativa para diseñar una máquina de estados.

De la misma forma que se trabaja en VHDL, la implementación de la máquina de estados se realiza en dos partes bien diferenciadas. Por una parte se realizará el diseño funcional, y por otra se diseñará el diseño de la parte de sincronismo, ya que *Stateflow* ofrece la posibilidad de detectar eventos, de manera que se pueden detectar los flancos de reloj.

Cuando se explica la implementación de la máquina en VHDL se hace patente la necesidad de introducir distintos flip-flops D en la arquitectura. Por este motivo junto con el diseño funcional y de sincronismo se ha implementado un flip-flop D mediante *Stateflow*, en los cuales se ha vuelto a utilizar la detección de flancos de reloj.

En la *Figura 67* se muestra el bloque en el que se implementa la parte funcional de la máquina de estados. Se puede observar que tanto los puertos de entrada, como los puertos de salida se corresponden con señales internas o salidas propias de la máquina de estados según corresponda. Estas señales reciben el mismo nombre que en la implementación VHDL, de manera que se corresponde totalmente con el código VHDL. Por este motivo para el entendimiento tanto de este bloque, como de los posteriores se remite al apartado en el que se describe el código VHDL.

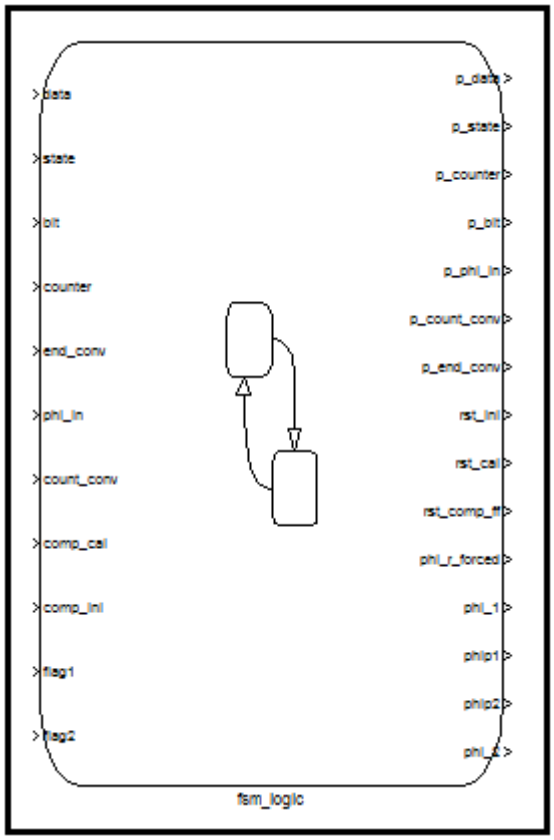

Figura 67. Máquina de estados. Funcional (fsm_logic). Matlab/Simulink

A continuación se mostrara el diagrama de flujo de cada estado, de tal forma que se podrá observar que la funcionalidad es la misma que se implementaba en VHDL.

```
/* fsm_logic */
{p_count_conv=count_conv+1;
p_state=state;
p_data=data;
p_counter=counter;
p_bit=bit;
p_end_conv=end_conv;
p_phi_in=0;
phi_1=0;
phi_2=0;
phi_r_forced=0;
phip1=0;
phip2=0;
rst_comp_ff=1;
rst_cal=1;
rst_ini=1;}
```

Figura 68. Inicio del proceso funcional. Stateflow

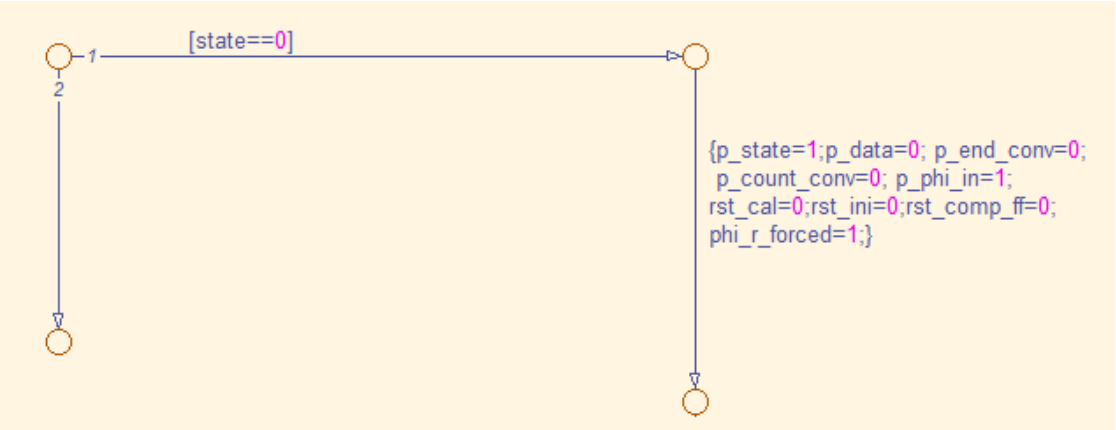

Figura 69. Estado 0: Reposo

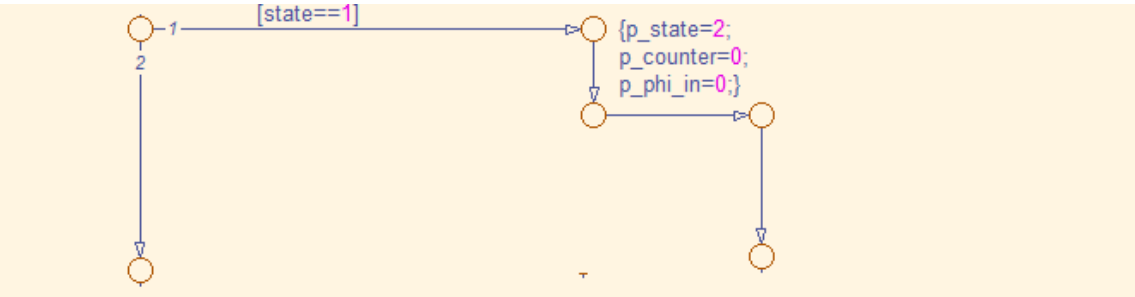

Figura 70. Estado 1: Initial_charge

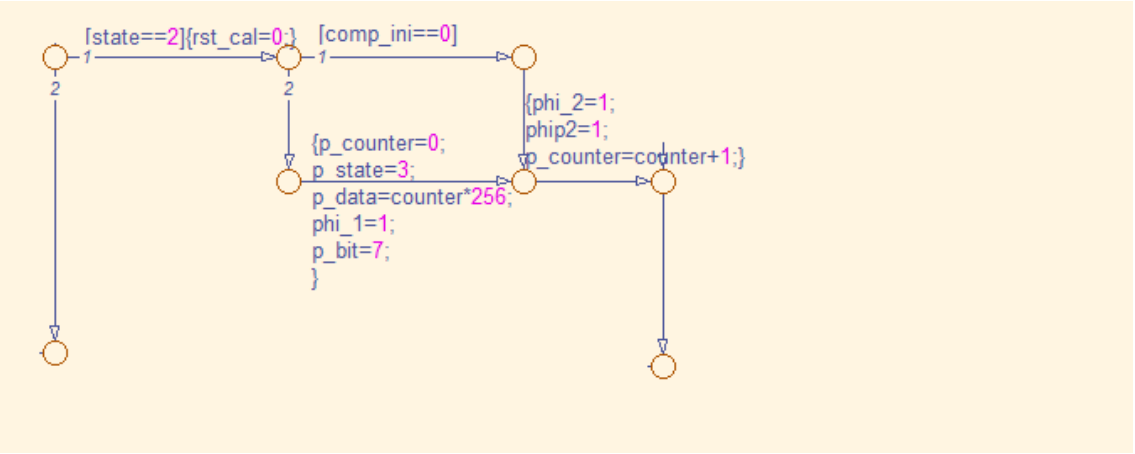

Figura 71. Estado 2: C2_charge

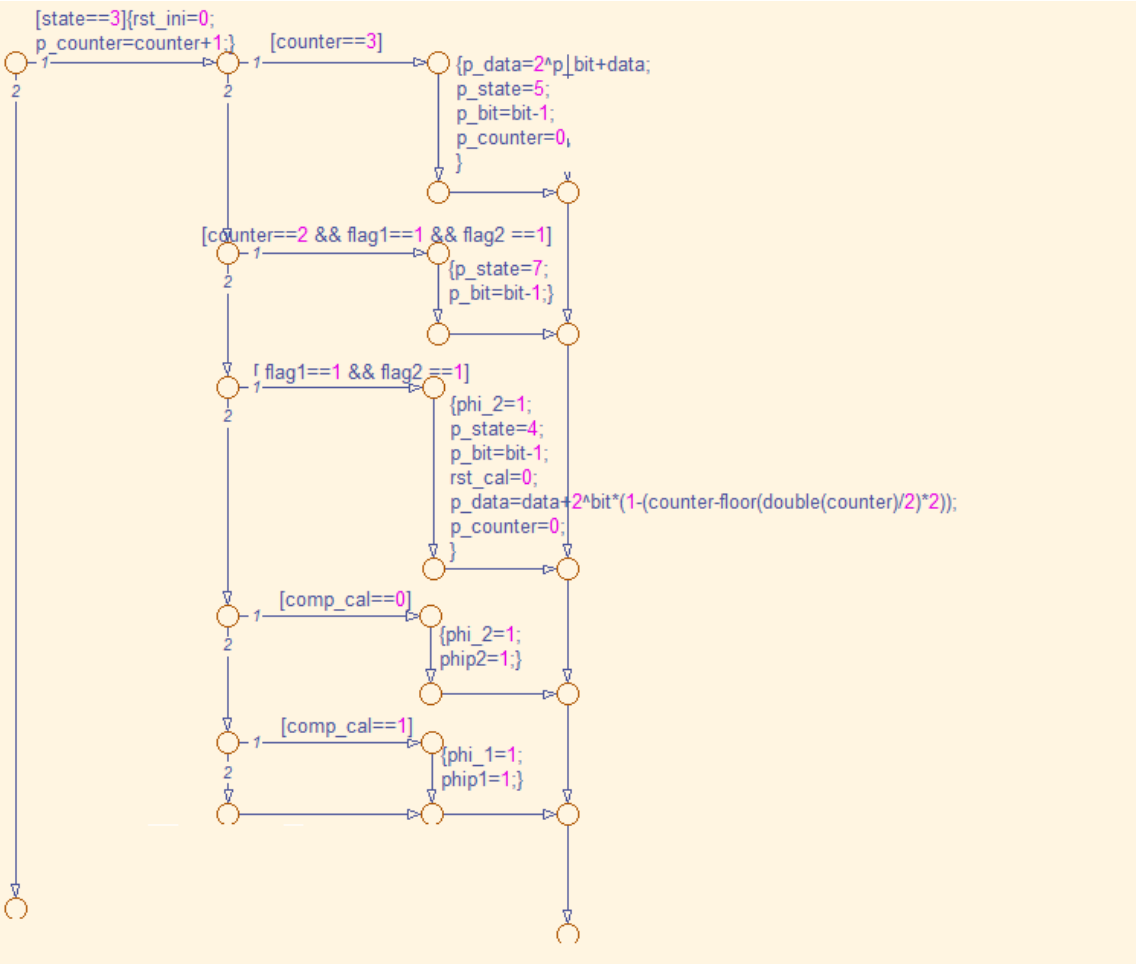

Figura 72. Estado 3: Neg_res

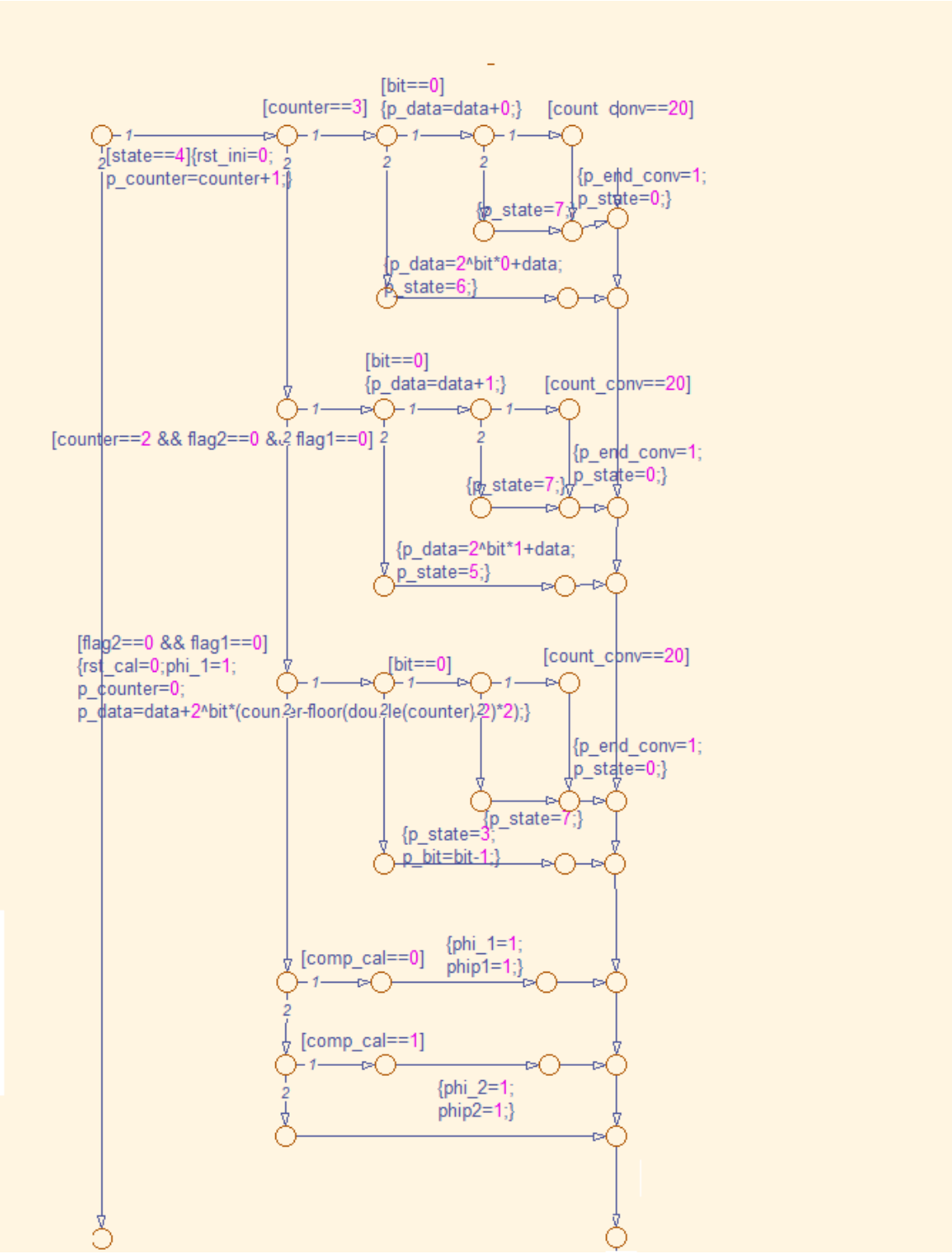

Figura 73. Estado 4: Pos_res

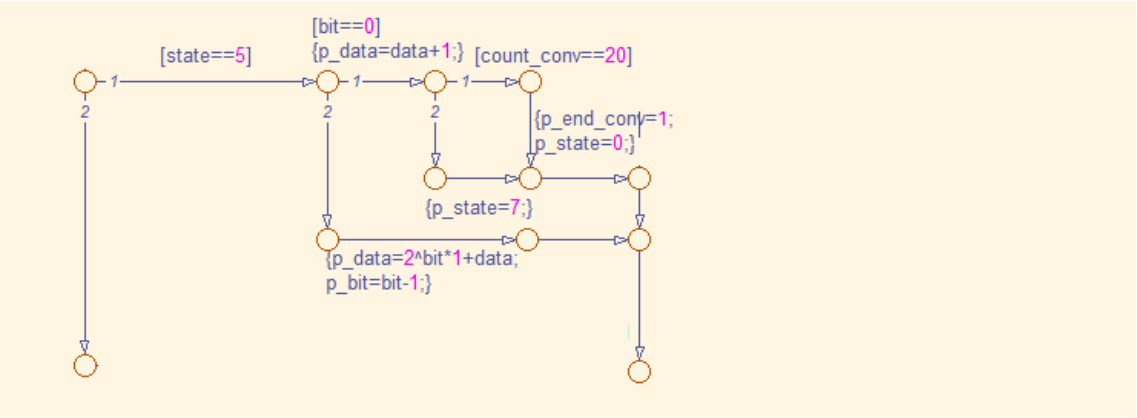

Figura 74. Estado 5: Assign_1

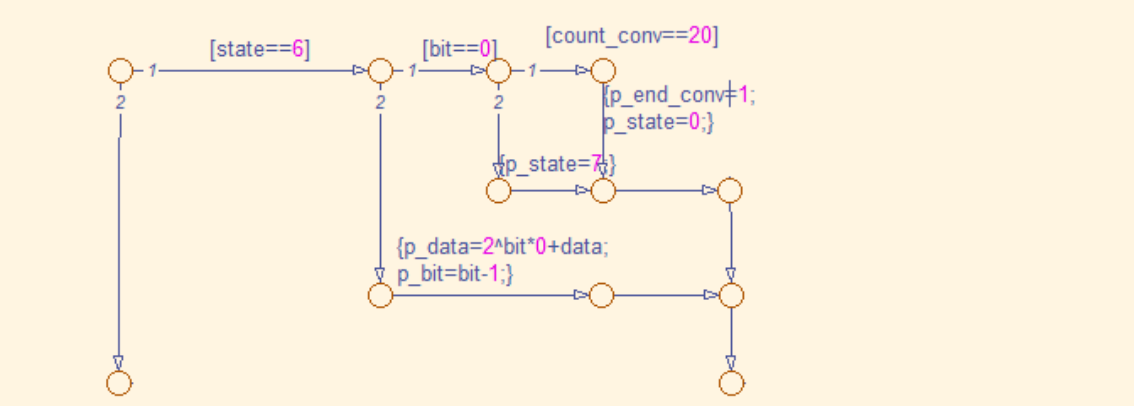

Figura 75. Estado 6: Assign_0

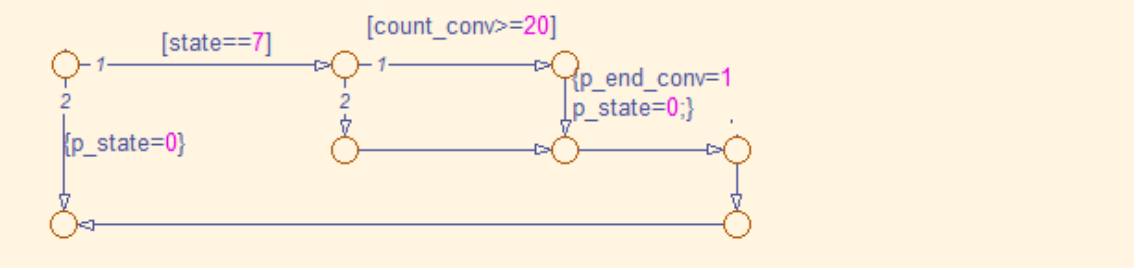

Figura 76. Estado 7: Wait_end

En la *Figura 77* se muestra la parte de la máquina de estados en la que se encuentra el proceso de sincronismo. En la figura se observa el bloque *fsm_state_top*, el cual se utiliza como parte de la máquina de estados por un lado, y por el otro lo que contiene este bloque en su interior, donde únicamente se han introducido unos bloques que proporciona Matlab/Simulink denominados *Mem* necesarios para retener el valor de la señal al bloque *fsm_state* implementado en *Sateflow*. Se puede observar que tanto los puertos de entrada, como los puertos de salida se corresponden con señales internas o salidas propias de la máquina de estados según corresponda. Estas señales reciben el mismo nombre que en la implementación VHDL, de manera que se corresponde totalmente con el código VHDL.

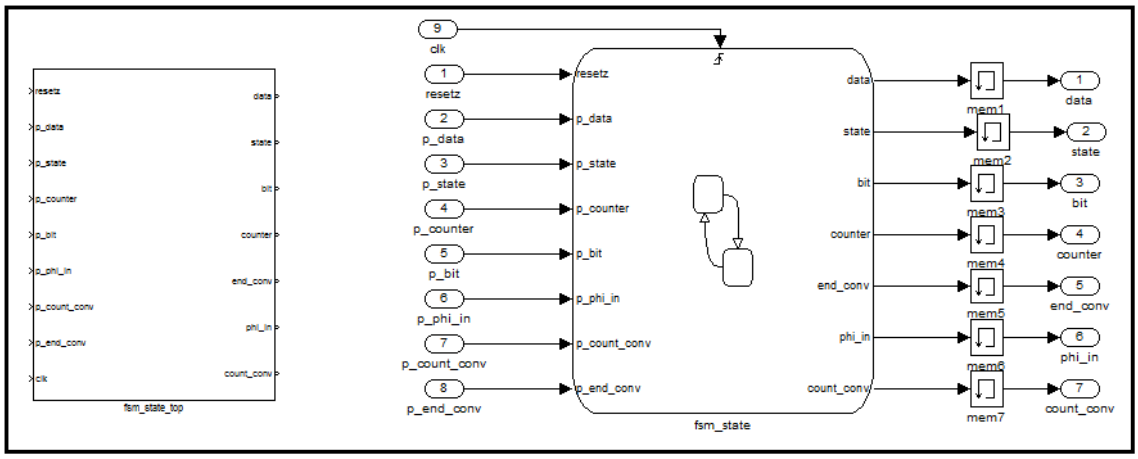

Figura 77. Máquina de estados. Función de sincronismo (fsm_state). Matlab/Simulink

A continuación se mostrara el diagrama de flujo del bloque *fsm_state* para detectar los flancos de reloj, de tal forma que se podrá observar que la funcionalidad es la misma que se implementaba en VHDL.

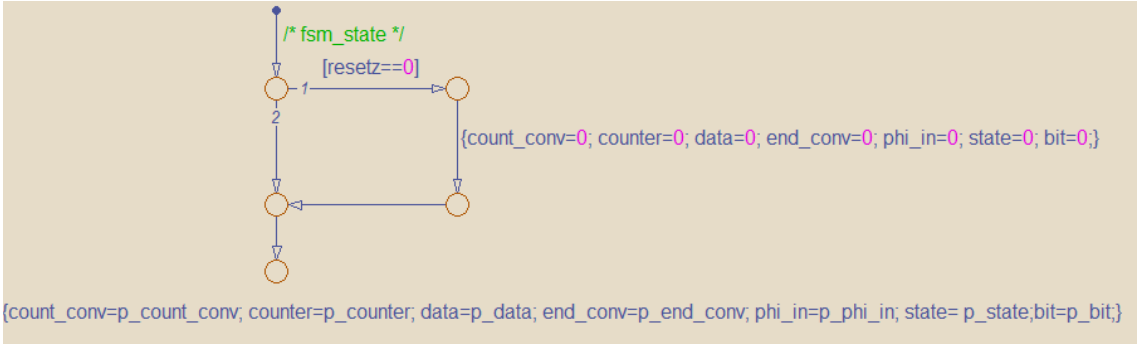

Figura 78. fsm_state. Stateflow

En la *Figura 79* se muestra la parte de la máquina de estados en la que diseña el flip-flop. En la figura se observa el bloque *ff_mem*, el cual se utiliza como parte de la máquina de estados por un lado, y por el otro lo que contiene este bloque en su interior, donde se han introducido unos bloques que proporciona Matlab/Simulink denominados *Mem* necesarios para retener el valor de la señal, junto con una circuitería para generar un reset asíncrono. Se puede observar que tanto los puertos de entrada, como los puertos de salida se corresponden con señales internas o salidas propias de la máquina de estados según corresponda. Estas señales reciben el mismo nombre que en la implementación VHDL, de manera que se corresponde totalmente con el código VHDL.

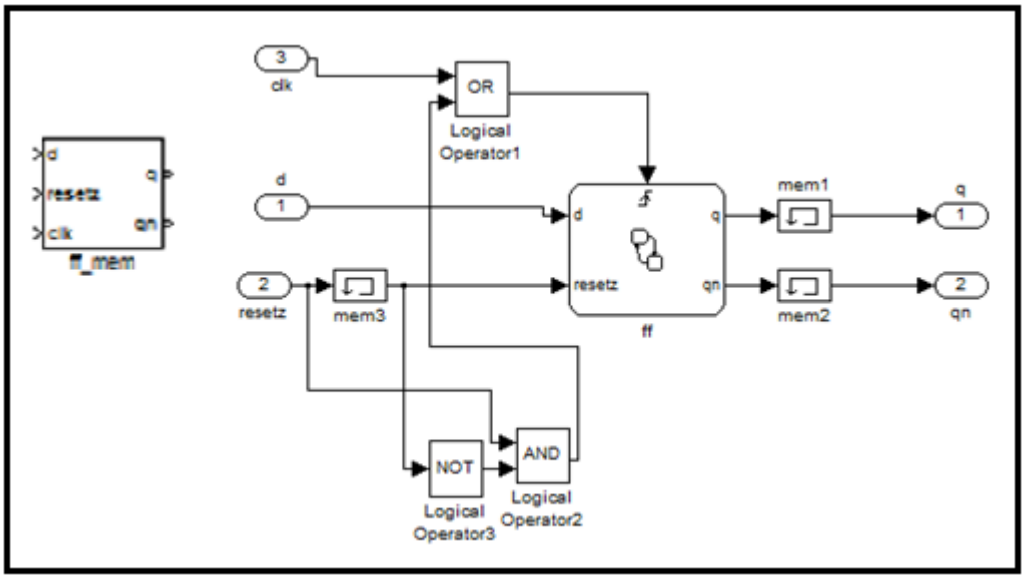

Figura 79. Máquina de estados. flip_flop D (ff_mem). Matlab/Simulink

A continuación se mostrara el diagrama de flujo del bloque *ff_mem* para implementar el flip-flop, de tal forma que se podrá observar que la funcionalidad es la misma que se implementaba en VHDL.

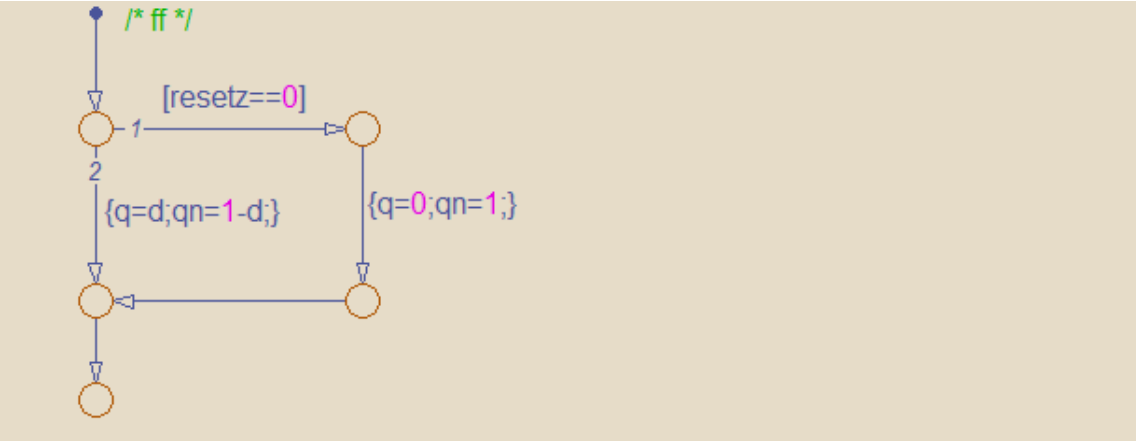

Figura 80. ff_mem. Stateflow

Una vez descritos los bloques necesarios para implementar la máquina, el aspecto que tiene la máquina de estados en Marlab/Simulink se muestra en la *Figura 81.*

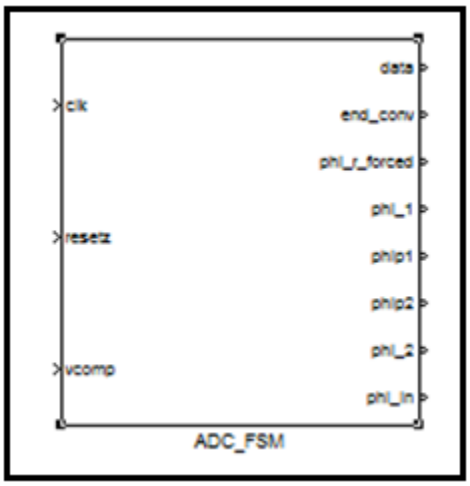

Figura 81. Máquina de estados. ADC_FSM. Matlab/Simulink

En el interior del bloque anterior se han conectado todos los elementos que forman la máquina de estados atendiendo al código implementado en VHDL. En el interior de este bloque se pueden observar las distintas conexiones entre las distintas señales internas, tanto como de entradas y salidas propias de la máquina de estados. Estas señales reciben el mismo nombre que en la implementación VHDL, de manera que se corresponde totalmente con el código VHDL. En la *Figura 82* se muestra el conexionado de los distintos bloques que forman la máquina de estados.

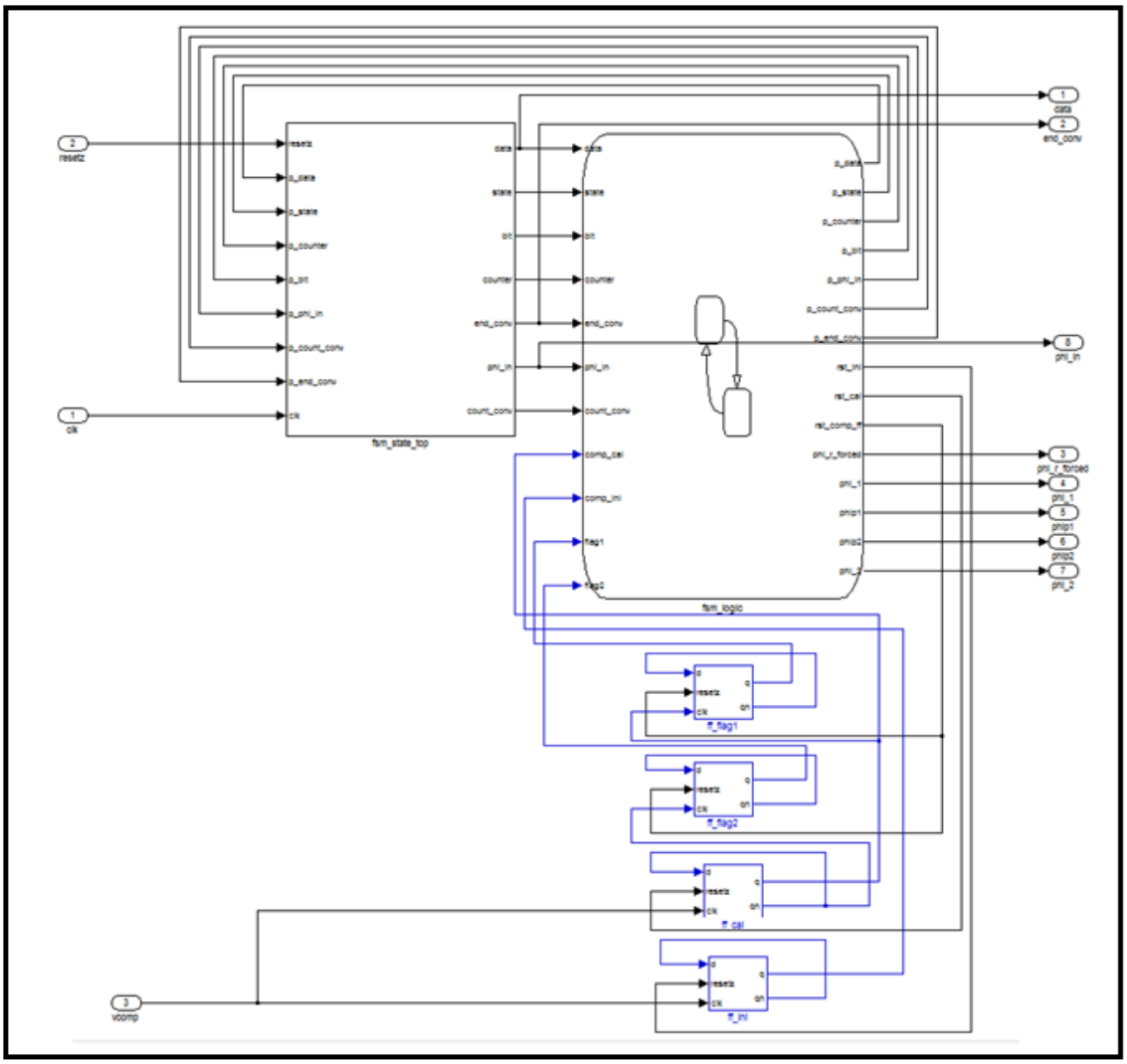

Figura 82. Conexión bloques de Máquina de estados. Matlab/Simulink

Finalmente, una vez se han implementado los bloques *Phases*, *Comparador* y *ADC_FSM*, estos se han conectado de de manera que el funcionamiento del modelo se corresponde con lo descrito en el capítulo 4, donde se ha descrito el funcionamiento de la arquitectura específica del ADC basado en tiempo que se tratado.

5.2.1.6 Modelo Completo:ADC basado en tiempo

Todos los bloques descritos anteriormente conectados debidamente se han introducido en un bloque denominado *Top_Model* que se corresponde con el diseño completo del ADC basado en tiempo. En la *Figura 83* se observa el aspecto de este bloque. Se puede observar que el ADC implementado en el modelo consta de cuatro puertos de entrada. Las señales de entrada son la señal de reloj *clk*, un reset asíncrono a nivel bajo *resetz*, la corriente de entrada I_{in} y finalmente la corriente de referencia I_{ref}. A la salida por un lado se tiene *data,* la cual se corresponde con el dato de salida digital, es decir el resultado tras realizar la conversión; y por otro lado la señal *end_conv*, la cual nos indica cuando finaliza la conversión, es decir, cuando se debe tomar el dato de salida (muestreo del dato).

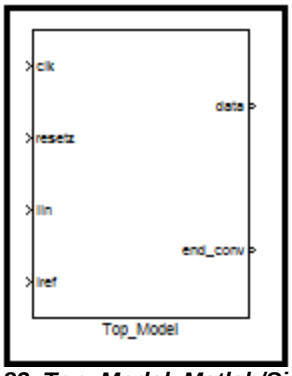

Figura 83. Top_Model. Matlab/Simulink

A la hora de conectar los distintos bloques para implementar el modelo completo, en primer lugar se han conectado tanto la señal de reloj como el reset a la entrada de la máquina de estados. Una vez hecho esto se han conectado tanto las salidas de la máquina de estados, como las corrientes a las entradas del bloque *Phases* para generar los valores de señal necesarios en la salida de los condensadores. Por otro lado se han conectado las salidas de los condensadores a los terminales del comparador, para lo cual se han utilizados switch propios de Matlab/Simulink controlados por salidas dedicadas a este efecto de la máquina de estados. Finalmente se ha conectado la salida del comparador a la tercera entrada de la máquina de estados, ya que como se ha explicado en capítulos anteriores esto es necesario para controlar la propia máquina de estados. Finalmente se introducido circuitería para controlar la habilitación del comparador. En la *Figura 84* se puede observar cómo se han conectados los distintos bloques para obtener modelo de ADC basado en tiempo de forma completa.

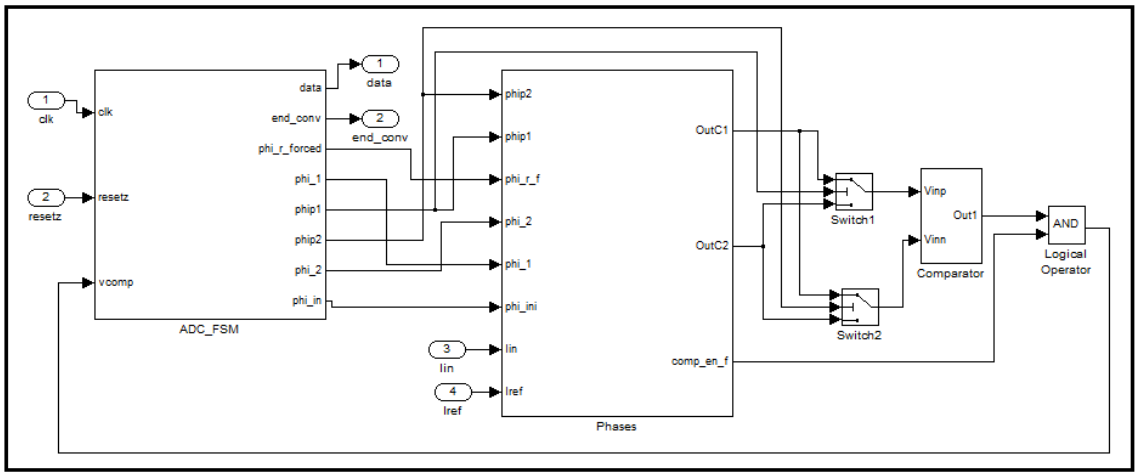

Figura 84. Conexión de Bloques: Modelo completo.Top_Model. Matlab/Simulink

A continuación, para comprobar el correcto funcionamiento del modelo, se muestra un resultado concreto en el que se podrá observar que se obtiene el dato de salida correctamente tras realizar la conversión. Para este ejemplo los valores de los parámetros son 5pF para ambas capacidades, 125nA para la corriente de referencia, y 50nA para la corriente de entrada. Se han tomado estos valores porque han sido recomendados en la bibliografía en la que se basa la arquitectura descrita para este ADC. Teniendo en cuenta que para la resolución del convertidor se han establecido 10 bits tanto por las recomendaciones de la bibliografía como por las necesidades de la aplicación, a la salida del convertidor el dato viene dado por

$$data = \frac{I_{in}}{FS} \bullet 2^{10}$$

donde *FS* es el fondo de escala, el cual es $4 \bullet I_{ref}$. Para los valores mencionados anteriormente a la salida se debe obtener un valor decimal de 102 como se puede ver en la primera gráfica de la *Figura 85.* En esta figura a parte del dato de salida, que se ve en la

primera gráfica, se pueden ver otras señales. En la segunda gráfica se muestra la señal *end_conv*, que como hemos mencionado anteriormente informa de cuando se ha terminado la conversión y por lo tanto es el momento de muestrear el dato. Finalmente en la tercera gráfica se muestran varias señales: en primer lugar la señal de color fucsia se corresponde con la señal de reloj, la señal en verde se corresponde con la salida del comparador, que como se puede observar se activa cuando las señales de color azul y rojo alcanzan el mismo valor, ya que se corresponden estas últimas con las salidas de los condensadores.

En la tercera gráfica de la figura se observa el funcionamiento del convertidor, donde se puede observar que se utiliza el tiempo como variable. Una vez que empieza la conversión, durante el primer ciclo de reloj, se produce la carga de un condensador a través de la corriente de entrada, que como se puede observar, al tener un valor menor que el de referencia (para este caso concreto), tiene una pendiente menor durante la carga (línea azul). Una vez cargado el primer condensador, se produce la carga del segundo con la corriente de referencia (línea roja, con una pendiente mayor), hasta que alcanza el mismo valor de señal que tiene el condensador anterior produciéndose la activación a la salida del comparador (línea verde). Esta fase se corresponde con la conversión de los dos primeros bits más significativos, que como se puede ver, unos ciclos de reloj más tarde, mientras que dura el procesado en la máquina de estados, se actualiza el dato. Una vez se ha cargado el segundo condensador se produce un residuo temporal, produciéndose las sucesivas cargas para obtener el siguiente bits más significativo como se ha explicado en el capítulo 4. Esta sucesión de cargas y descargas de los condensadores se va produciendo mientras el residuo resultante de cada conversión del bit correspondiente sea apreciable y no caiga demasiado cerca de los flancos de reloj. Cada vez que se producen las cargas y descargas de los condensadores en cada estado se calcula el residuo y se asigna el bit correspondiente, actualizándose el dato como se puede ver en la primera gráfica. Una vez que ya no se puede calcular el residuo por ser este inapreciable, se pasa a un estado de espera hasta que se llega a veintidós ciclos de reloj, tiempo que tarda como máximo una conversión. Una vez han pasado estos veintidós ciclos de reloj se activa la señal *end_conv,* que se puede observar en la segunda gráfica, donde se informa de que la conversión ha concluido. El paso de un estado ha otro se produce según se ha explicado en capítulos anteriores donde se describía el funcionamiento de la máquina de estados.

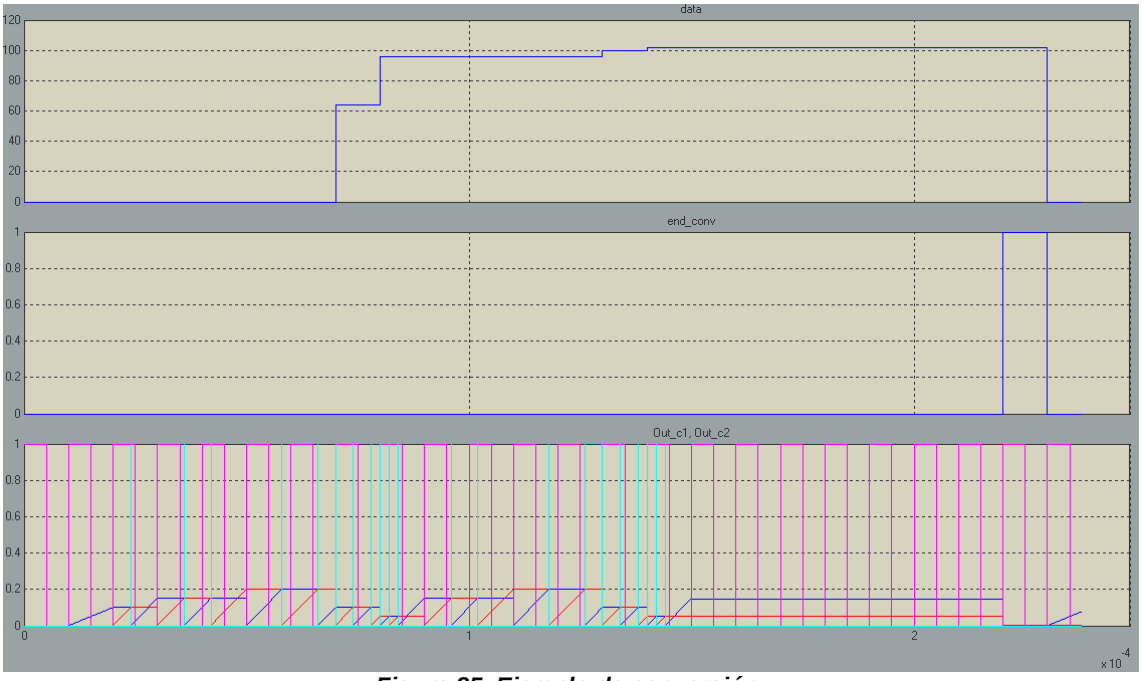

Figura 85. Ejemplo de conversión

Finalmente para ver que efectivamente se obtiene el valor de 102 tras la conversión con estos valores en el modelo se ha hecho un zoom en la primera gráfica de la figura anterior. Esto se puede ver en la *Figura 86*.

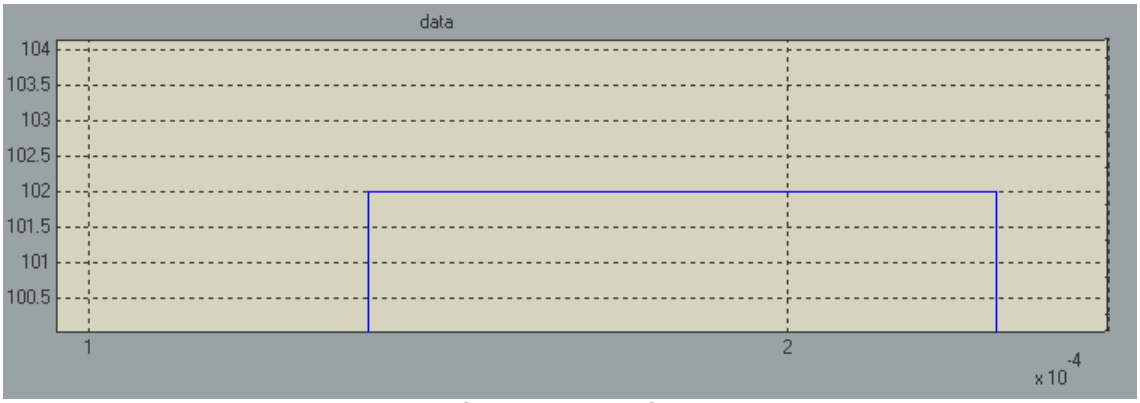

Figura 86. Zoom data

5.2.2 Test-Bench: Simulación

Una vez implementado el modelo completo del convertidor analógico-digital basado en tiempo y comprobado que funciona correctamente se ha de realizado un test-bench que consta del modelo simulink mas una serie de scripts de Matlab a través de los cuales, mediante programación, se llama al modelo Simulink para realizar distintos análisis paramétricos. A la hora de realizar los análisis paramétricos se han estudiado distintas especificaciones del convertidor analógico-digital que se implementa en el modelo.

En primer lugar se estudiara el error INL (No-linealidad integral), el cual describe la desviación de la curva de trasferencia lineal de un ADC ideal. Se estudia esta especificación por su importancia debido a que se puede interpretar como una medida del rendimiento del convertidor.

En segundo lugar se estudiará la ENOB (número efectivo de bits), la cual es una indicación global de la resolución del sistema. Se estudia esta especificación debido a que en ella quedan reflejadas todas las imperfecciones del sistema, es decir, engloba la totalidad de fuentes de ruido, siendo así la mejor manera de comprobar el rendimiento del convertidor.

En la *Figura 87* se observa el test-bench implementado en simulink en el cual se introducen por un lado la señal de reloj, el reset asíncrono, la corriente de entrada y de referencia a través de los puertos de entrada del bloque *Top_Model,* y los parámetros del convertidor se introducen a través de una interfaz que se puede ver en la *Figura 88.* A la hora de realizar las distintas simulaciones mediante Matlab, se calculará tanto la ENOB, como el error INL ejecutando una serie de scripts en la que se utilizará la función *sim()* de Matlab para ejecutar el modelo. Cabe destacar que tanto los parámetros del comparador como las entradas del convertidor son tratadas como variables globales para obtener los valores del workArea de Matlab. Por otro lado, a la salida del modelo tanto el dato digital *data,* la señal que indica que ha terminado la conversión *end_conv* y la corriente de entrada I_{in} se introducen en un bloque Mux de Simulink, el cual combina sus entradas en un vector de salida, es decir, actúa como un bus. Las entradas del bloque deben ser del mismo tipo, es por esto que se introducen dos bloques que cambian el tipo de la señales pasando a ser señales tipo *doublé.* De esta manera queda definido el test-bench sobre el que se trabajará para realizar distintos análisis paramétricos.

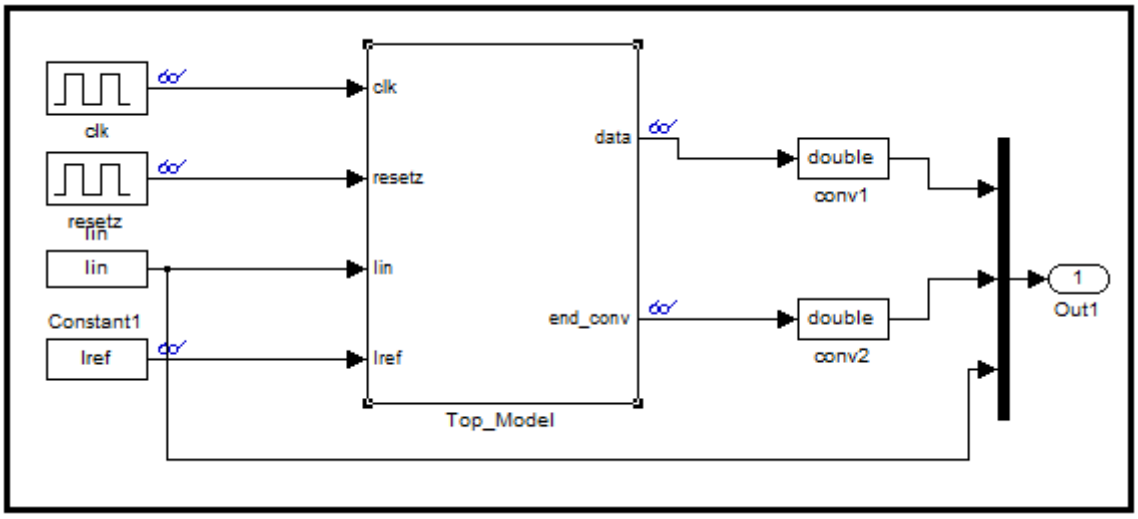

Figura 87. test-bench en simulink

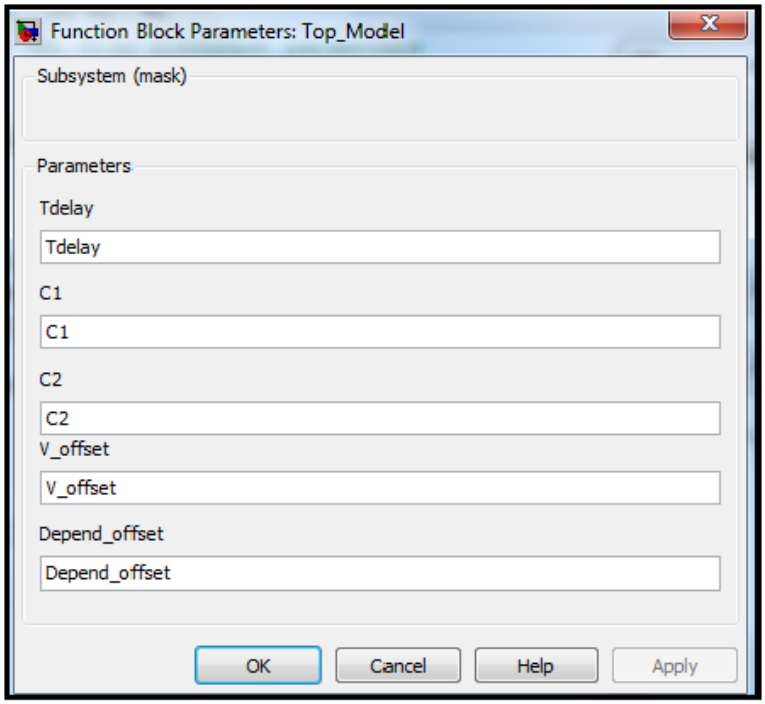

Figura 88. Interfaz: Parámetros Top_Model

Para el cálculo tanto del error INL como de la ENOB se han establecido unos valores para los parámetros que han venido dados tanto por las recomendaciones impuestas por la bibliografía en la que se explica la arquitectura del convertidor modelado, como por las necesidades de la aplicación. Los valores de los parámetros han sido:

- Frecuencia de reloj (f_{clk}) de 100kHz dado que por las necesidades de la aplicación no se necesita una velocidad excesivamente alta.
- Corriente de referencia de 125nA ya que los sensores resistivos que serán utilizados para la aplicación del ADC puede ser de hasta 5MΩ, de manera que se trabajará con una tensión acorde a la utilización con personas.
- Corriente de entrada entre 0 y 500nA, es decir, cuatro veces la corriente de referencia, lo cual viene impuesto por la arquitectura.
- Valor de capacidad de los condensadores de 5pF, lo cual viene impuesto por la arquitectura y por el diseño.

Tanto para analizar el error INL, como la ENOB se realizarán una serie de análisis paramétricos analizando como afecta al rendimiento del convertidor la posible variación de diferentes parámetros. Los parámetros que se variaran para comprobar el funcionamiento son:

- Retraso del comparador incluido en la arquitectura. Cuando se habla de retraso del comparador se refiere al tiempo que tarda el comparador en generar la activación de la salida a "uno" una vez los valores de señal que se encuentran en sus terminales se igualan. Este parámetro se denomina *Tdelay.*
- Mismatch de capacidades. Con mistmach de capacidades se refiere al posible desapareamiento de las capacidades incluidas en la arquitectura, las cuales deben ser del mismo valor para el correcto funcionamiento de este.
- Offset independiente del comparador incluido en la arquitectura. Los comparadores suelen tener un valor de offset independiente de las señales de entrada que se añade al valor de señal de estos. Este parámetro se denomina *V_offset.*
- Offset dependiente del comparador incluido en la arquitectura. A parte del offset de señal independiente en el comparador existe un offset dependiente de las señales de entrada, el cual también afecta a la salida del comparador. Este parámetro se denomina *Depend_offset.*

Se hacen estas consideraciones sobre el diseño porque que afectan al comportamiento del ADC, ya que la arquitectura básicamente consta del comparador, de las capacidades, de la máquina de estados y de una serie de lógica. Al ser estos dos últimos elementos digitales, serán los primeros los que pueden afectar al comportamiento del convertidor.

5.2.2.1 Cálculo de error INL (No-linealidad integral)

Para calcular el error INL se ha hecho uso del modelo implementado en Matlab/Simulink junto con una serie de scripts realizados en matlab.

En primer lugar se tiene un script en el que se ha elaborado una función general para obtener el error INL en el modelo simulink del ADC, la cual será llamada por otras funciones donde se variarán los distintos parámetros del convertidor. Mediante el uso de esta función se calcula el error INL propiamente dicho, para lo cual hay que proporcionarle como entrada a esta función el valor de la corriente de referencia (I_{ref}), el valor de las capacidades (C_1 y C_2), el valor del retraso del comparador (*Tdelay*), el valor del offset independiente de señal del comparador (*V_offset*) y el valor del offset dependiente del comparador (*Depend_offset*) y finalmente el número de muestras (*Nitems*). Estas variables de entrada, excepto el número de muestras, son globales, ya que son utilizadas por otras funciones que llamaran a esta y es necesario que se encuentren en el WorkArea de Matlab. En esta función se genera un vector de corrientes de entrada que es una rampa de 0 al 90% del fondo de escala con una longitud igual al número de muestras que en este caso es 64. Mediante la función *sim()* de Matlab se ejecuta el modelo implementado en Simulink para cada valor del vector de corrientes de entrada, obteniéndose un dato de salida digital para cada valor de corriente. Una vez generados los datos de salida se calcula el error Máximo INL mediante la resta de las rectas de regresión lineal tanto de la corriente de entrada (normalizada en LSB), como de los datos de salida en LSBs generadas ambas mediante el método de los mínimos cuadrados, para lo cual se han utilizado las funciones *polyfit()* y *polyval()* de Matlab. Tras la ejecución de esta función se genera una figura en la que se puede encontrar dos gráficas, por un lado una gráfica en la que se encuentran tanto la recta de regresión de la corriente de entrada normalizada en LSB (la recta se verá en color rojo), y por otro lado la recta de regresión generada por los datos de salida en LSBs (la recta se verá en color azul); y por otro lado en la segunda gráfica se verá el resultado de la resta entre ambas rectas de regresión, donde se el error INL se verá reflejado en la diferencia máxima entre estas dos rectas. Por último en el título de la gráfica se podrá ver es error INL y además superpuestos sobre las gráficas se podrán ver los valores de los distintos parámetros. El nombre del fichero donde se implementa esta función es *"test_bench_Par_INL".*

En segundo lugar se han implementado varios scripts para introducir los distintos parámetros variándolos dentro de unos valores adecuados acorde a las oscilaciones que se pueden dar en estos. Se han implementado los siguientes scripts:

- *test_Bench_Par_INL_Tdelay*. Función para obtener el error INL haciendo un análisis paramétrico del retraso del comparador del modelo del ADC de Matlab/Simulink (*Tdelay*), introduciendo un retraso de 0ns, 2ns, 10ns, 100ns y 1000ns. Llama a la función implementada en el fichero "*test_bench_Par_INL".*
- *test_bench_Par_INL_Cmistmath*. Función para obtener el error INL haciendo un análisis paramétrico del retraso del comparador (*Tdelay*) con los valores de retraso anteriores junto con del mistmach de capacidades del modelo del ADC de Matlab/Simulink variando unas de las capacidades de 4.75pF hasta 5.25pF. Llama a la función implementada en el fichero *"test_bench_Par_INL".*
- *test_bench_Par_INL_Tdelay_V_offset_Depend_offset*. Función para obtener el error INL haciendo un análisis paramétrico del retraso del comparador con los valores anteriores, offset independiente de señal del comparador con valores de 0.0005V, 0.0025V, 0.0050V, 0.0075V y 0.0100V junto con el offset dependiente de la señal en el comparador con valores de 0.2%, 0.4% y 0.6% en el modelo del ADC de Matlab/Simulink. Llama a la función implementada en el fichero "*test_bench_Par_INL*".
- *test_bench_Par_INL_T_Vo_Do_mstch.* Función para obtener el error INL haciendo un análisis paramétrico del retraso del comparador, el offset independiente del comparador, del offset dependiente y del mistmach capacidades, todos ellos con los valores anteriores, en el modelo del ADC de Matlab/Simulink. Llama a la función implementada en el fichero "*test_bench_Par_INL"*.

Todos estos scripts se podrán ver en el anexo III debidamente comentados para facilitar su comprensión.

5.2.2.1.1 Sistema Ideal

Al hablar de sistema ideal se refiere a un sistema en el que a los parámetros variables se le han impuesto los valores en los óptimos para que el convertidor funcione con un rendimiento máximo, es decir, con un comparador sin retraso, sin offset tanto dependiente como independiente de señal y con ambas capacidades fijadas a un valor de 5 pF. Para estos valores el resultado obtenido tras la simulación se puede observar en la *Figura 89.*

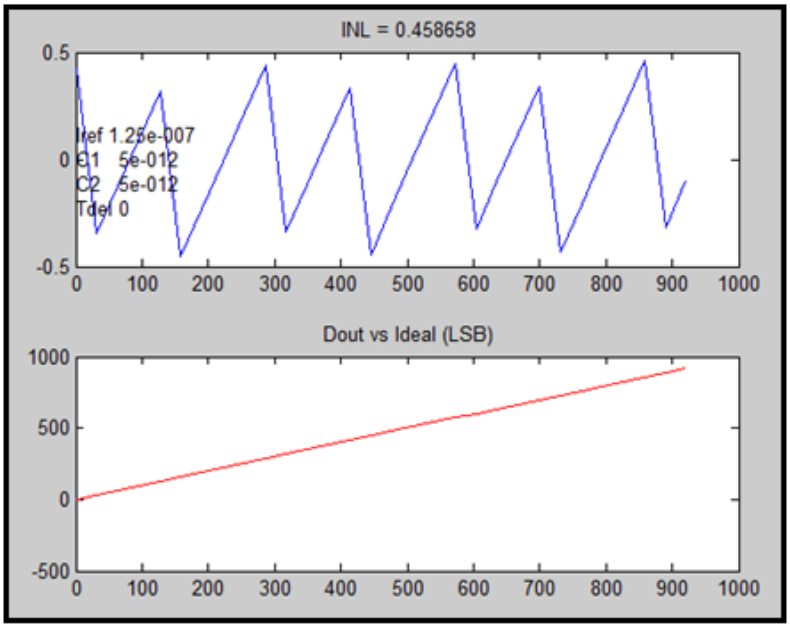

Figura 89. Resultado INL. Sistema Ideal

Si se analiza el resultado se observa que la recta de regresión generada con los puntos del obtenidos del dato de salida digital prácticamente se superpone con la recta de regresión obtenida con la entrada de corriente normalizada en LSBs, de manera que en el resultado, como era de esperar al no introducir ninguna no idealidad, se puede observar que el rendimiento del ADC en cuanto a INL se refiere es muy alto. Se puede observar que los errores del dato de salida con respecto al resultado ideal no sobrepasa un LSB, oscilando entre –0 y .5

y 0.5 LSBs dando como resultado un error máximo y por consiguiente una INL de 0.458658 LSBs.

5.2.2.1.2 Influencia del Retraso del Comparador

A continuación se va analizar la influencia del retraso del comparador en el rendimiento del ADC. Para ello se va a incluir en el modelo un retraso en el comparador comenzando en 2ns y se irá aumentando el orden de magnitud para ver cómo influye el retraso en cada caso.

En la *Figura 90* se pueden observar los resultados obtenidos tras la simulación para varios valores de retraso del comparador.

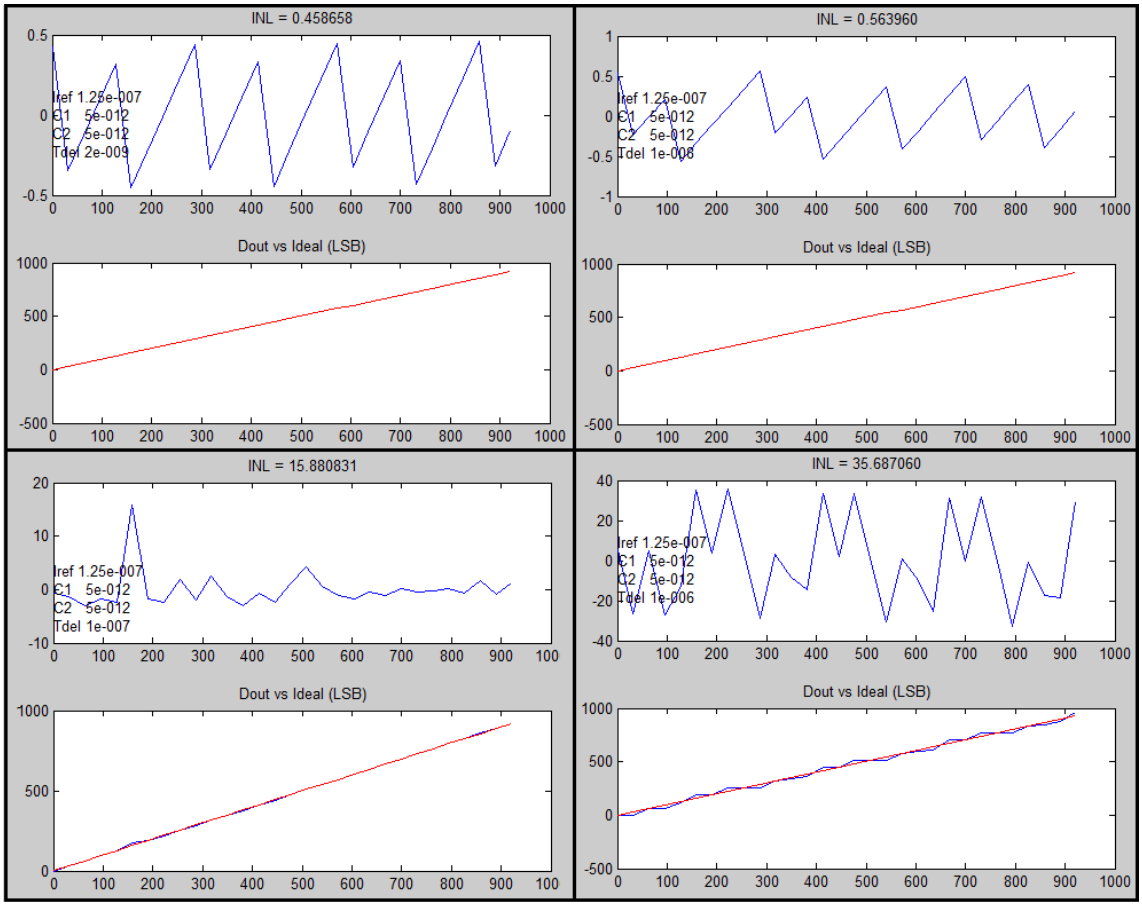

Figura 90. Resultado INL. Influencia del retraso del comparador

Si se analiza el resultado, en la primera gráfica, se observa que la recta de regresión generada con los puntos obtenidos del dato de salida digital prácticamente se superpone con la recta de regresión obtenida con la entrada de corriente normalizada en LSBs, de manera que en el resultado, al introducir un retraso de 2ns, se observar que el rendimiento del ADC en cuanto a INL se refiere es muy alto. Se puede observar que los errores del dato de salida con respecto al resultado ideal no sobrepasa un LSB, oscilando entre –0.5 y 0.5 LSBs dando como resultado un error máximo y por consiguiente una INL de 0.458658 LSBs, el cual es el mismo que se obtenía sin introducir retraso.

En el caso de que se aumente el retraso en un orden de magnitud, es decir si se introduce un retraso de 10ns, el rendimiento del ADC sigue siendo alto, ya que el error de INL no aumenta significativamente, como se observa en la segunda gráfica de la figura. En este caso los errores del dato de salida con respecto al resultado ideal tampoco sobrepasa un LSB, oscilando entre –0.6 y 0.6 LSBs dando como resultado un error máximo y por consiguiente una INL de 0.563960 LSBs

Para seguir analizando la influencia del retraso se aumenta en otro orden de magnitud, de manera que se introduce un retraso de 100ns. En este caso el ADC deja de funcionar

correctamente ofreciendo un rendimiento poco aceptable en cuanto a error INL se refiere. Como se puede observar en la tercera gráfica de la figura comienzan a darse diferencias significativas entre la recta de regresión generada con los puntos del obtenidos del dato de salida digital y la recta de regresión obtenida con la entrada de corriente normalizada en LSBs. En este caso los errores del dato de salida con respecto al resultado ideal sobrepasa en dos órdenes de magnitud un LSB, dando como resultado un error máximo y por consiguiente un error INL de 15.880831 LSBs, el cual es un error demasiado significativo.

Si se sigue aumentando el retraso en otro orden de magnitud llegando a 1000ns es de esperar que el resultado sea aún peor que en el caso anterior, aumentando significativamente el error. Este hecho se ve reflejado en el resultado que se puede observar en la cuarta gráfica de la figura, donde se puede observar una diferencia más significativa que en los caso anteriores entres las dos rectas de regresión, llegando a dar como resultado un error INL de 35.686070 LSBs, el cual es un error inaceptable en el rendimiento del convertidor.

En la *Figura 91* se observa el resultado tras simular el modelo para un retraso de 50ns. En esta se observa que el error INL que se obtiene como resultado es de 2.594512 LSBs, el cual comienza a ser significativo en cuanto al rendimiento del ADC, de manera que se puede concluir que el retraso máximo para el comparador sin tener en cuenta la influencia de las demás no idealidades de la arquitectura no debe superar los 50ns.

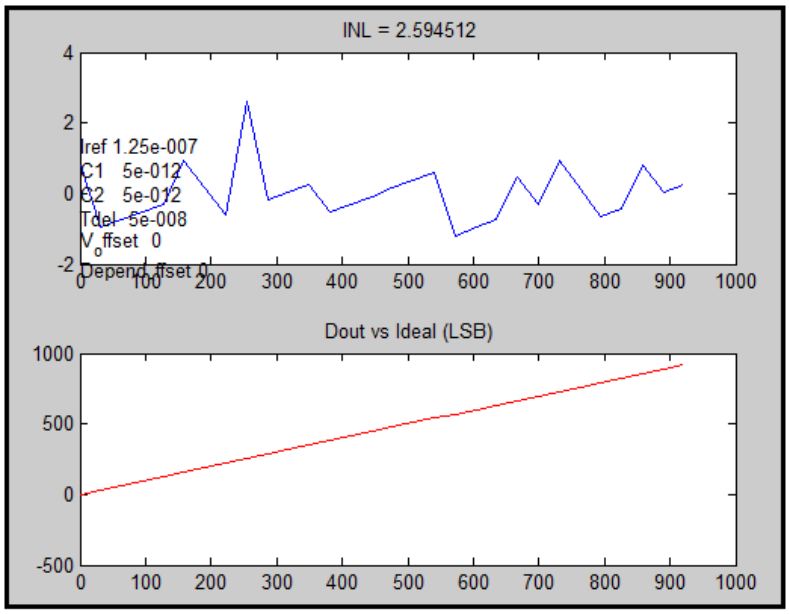

Figura 91. Resultado INL. Retraso 50ns

5.2.2.1.3 Influencia de la disparidad de Capacidades

En este apartado se va analizar la influencia de la disparidad de capacidades (mistmach) en el rendimiento del ADC. Para ello se variará una de las capacidades desde 4.75pF hasta 5.25pF en pasos de 10pF dejando fija la otra capacidad en 5pF y sin introducir retraso alguno.

En la *Figura 92* se pueden observar los resultados obtenidos tras la simulación para los distintos valores de capacidades.

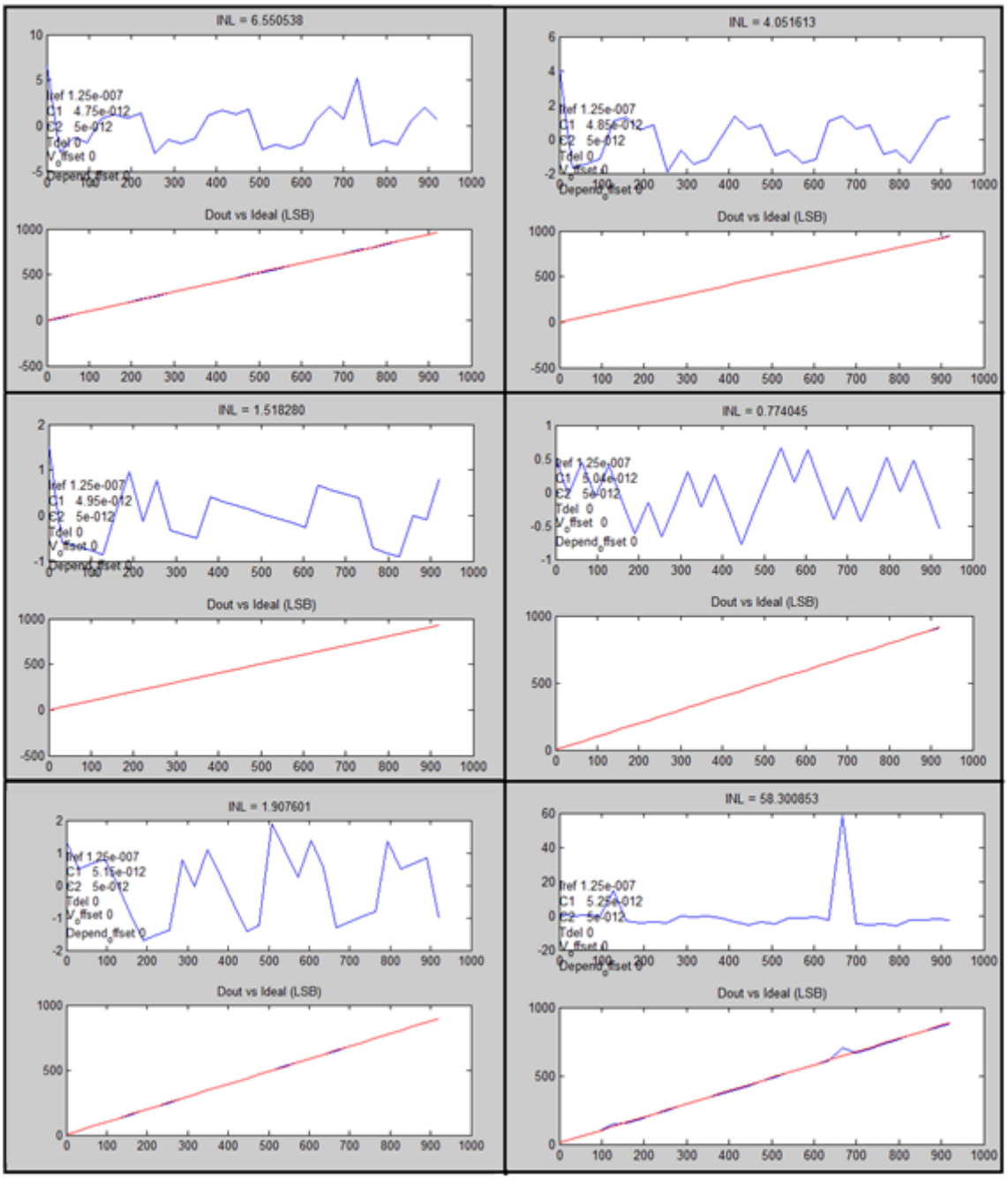

Figura 92. Resultado INL. Mistmach de capacidades

Si se observan los resultados de las distintas gráficas que se pueden ver en la figura se obtienen resultados aceptables para el rendimiento del ADC con los parámetros correspondientes a la tercera, cuarta y quinta gráfica, donde se obtiene un error INL de 1.518280 LSBs, 0.774045y 1.907601 LSBs respectivamente, los cuales se corresponden con un valor de capacidades de 4.95pF, 5.05 y 5.15pF. Para los demás valores de capacidades el error INL es demasiado elevado, aunque también se pude ver que para valores por debajo de 5pF el error es menor que para valores de capacidad por encima de este. Los resultados de error INL obtenidos han sido 6.550538 LSBs para una capacidad de 4.75pF y de 4.051613 LSBs para una capacidad de 4.85pF para valores por debajo de 4.95pF; y de 14.303152; y de 58.300853 LSBs para una capacidad de 5.25pF para valores por encima de 5.15pF. Ante estos resultados se puede concluir que el valor de las capacidades debe oscilar entre 4.95pF y 5.15pF sin tener en cuenta las demás no idealidades.

5.2.2.1.4 Influencia del offset independiente de señal en el comparador

Para completar el análisis paramétrico se debe tener en cuenta el offset en el comparador integrado en la arquitectura, tanto el offset independiente de la señal de entrada, como el offset dependiente. En el presente apartado se tratará el primero de ellos. Para analizarlo se han impuesto una serie de valores para los parámetros que se suelen dar habitualmente. Estos valores son 0.0005V, 0.0025V, 0.0050V y 0.0075V.

En la *Figura 93* se pueden observar los resultados obtenidos tras la simulación con los distintos valores de capacidades.

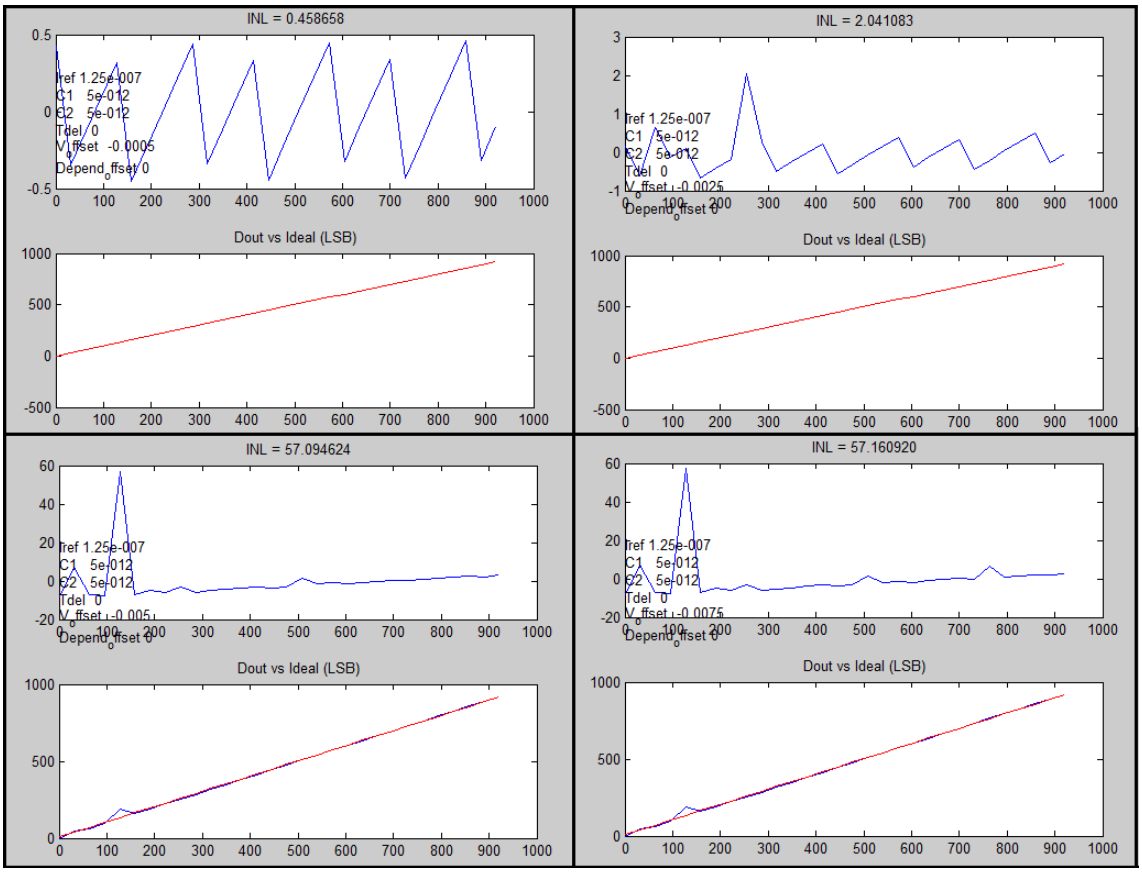

Figura 93. Resultado INL. Offset independiente de señal en el comparador

Si se observan las distintas gráficas de la figura, en la primera se puede ver que para un valor de offset de 0.0005V se obtiene un error INL de 0.458658 LSBs, lo cual da un buen rendimiento en el convertidor ya que no llega superar un LSB. Si se aumenta el offset a 0.00025V, cuyos resultados se corresponden con la segunda gráfica, se obtiene un error INL de 2.041083 LSBs, lo cual se puede entender como un decaimiento del rendimiento en el convertidor que empieza a ser importante. Como es de esperar para valores mayores de offset el error INL aumenta considerablemente, lo cual se observa en la tercera y cuarta gráfica de la figura donde se obtiene un error de 57.094624 LSBs y 57.160920 LSBs para unos valores de offset de 0.0050V y 0.0075V respectivamente. A parte de estos resultados en los que se ve como aumenta de manera importante el error cuando el valor de offset es superior a 0.0025V. Se han obtenido más resultados para mayores valores de offset en los que se han obtenido resultados similares en órdenes de magnitud a los obtenidos para los dos últimos. Estos resultados son coherentes con la arquitectura, ya que al tratarse de un convertidor basado en tiempo un offset en el comparador tiene el mismo efecto que un retraso, lo cual, como se ha visto anteriormente, influye considerablemente en el rendimiento del convertidor. Tras analizar los resultados de la gráfica se puede concluir que el offset en el comparador debe ser lo suficientemente pequeño, lo que podría parecer difícil de conseguir, y para lo que se deben buscar soluciones como el uso de un comparador con realimentación negativa.

Además de estas simulaciones se han obtenido los resultados para un valor de offset de 0.0100V, el cual no se ha incluido porque es del mismo orden de magnitud que el resultado obtenido para 0.0075, el cual es un resultado muy negativo de error de INL.

5.2.2.1.5 Influencia del offset dependiente de señal en el comparador

Para completar el análisis del offset del comparador, al efecto que tiene el offset independiente de señal del comparador se va a sumar el efecto de offset dependiente de señal de este. Se ha hecho este análisis de forma conjunta porque el efecto se da de esta manera, siendo el offset dependiente, analizado anteriormente, el que tiene un efecto más significativo en el rendimiento del ADC. Para añadir el efecto del offset dependiente, este parámetro añade un tanto por ciento de señal a la propia señal de entrada. Se han incluido unos valores para este parámetro de un 0.2%, 0.4% y 0.6% para cada valor de offset dependiente, sin mover los demás parámetros que pueden influir en el rendimiento del comparador.

En las siguientes figuras se verán los resultados más representativos obtenidos tras la simulación para distintos valores de offset dependiente e independiente conjuntamente.

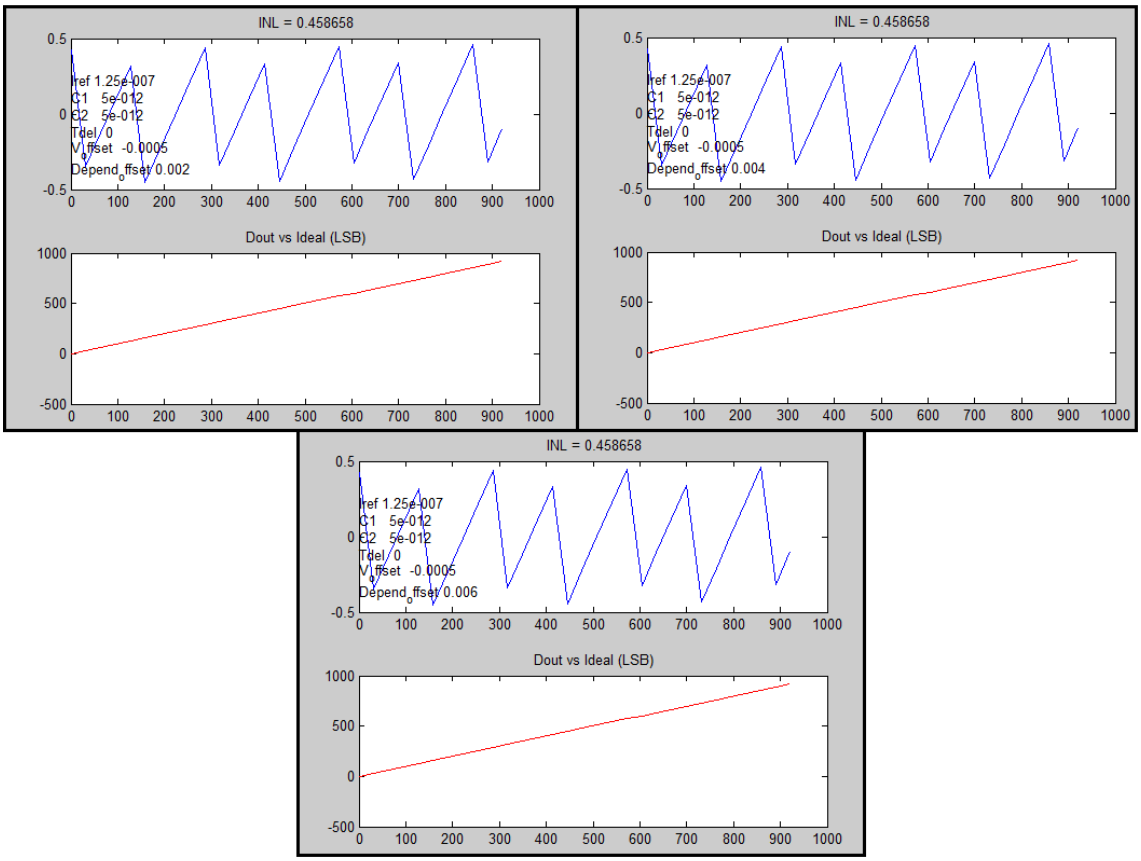

Figura 94. Resultado INL. Offset dependiente para V_offset=0.0005V

En la *Figura 94* se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0005V. Si se presta atención a los resultados obtenidos en el apartado anterior, el resultado es el mismo que se obtiene sin incluir offset dependiente para los tres valores de offset: 0.458658 LSBs. Tras observar estos resultados se puede concluir que el offset independiente de señal no influye significativamente en el rendimiento del convertidor.

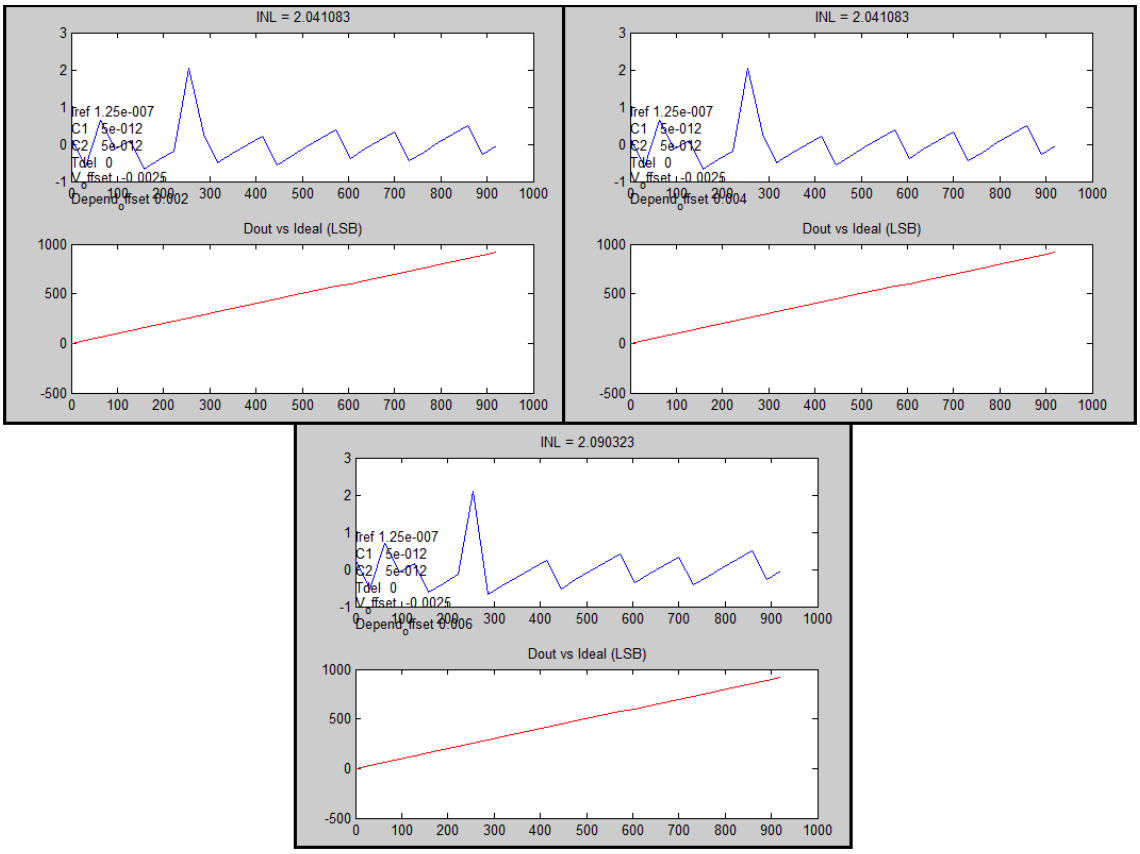

Figura 95. Resultado INL. Offset dependiente para V_offset=0.0025V

En la *Figura 95* se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0025V. Si se presta atención a los resultados obtenidos en el apartado anterior, el resultado es el mismo que se obtiene sin incluir offset dependiente para los valores del 0.2% y 0.4% de offset: 2.041083 LSBs. Para un valor del 0.6% de offset de este tipo el error aumenta, aunque en un valor no muy significativo, llegando a alcanzar un valor de 2.090323 LSBs. Como conclusión se dirá que el offset independiente no influye significativamente en el rendimiento de convertidor para estos valores, aunque ya se puede observar que empieza a influir aumentando el valor de error INL.

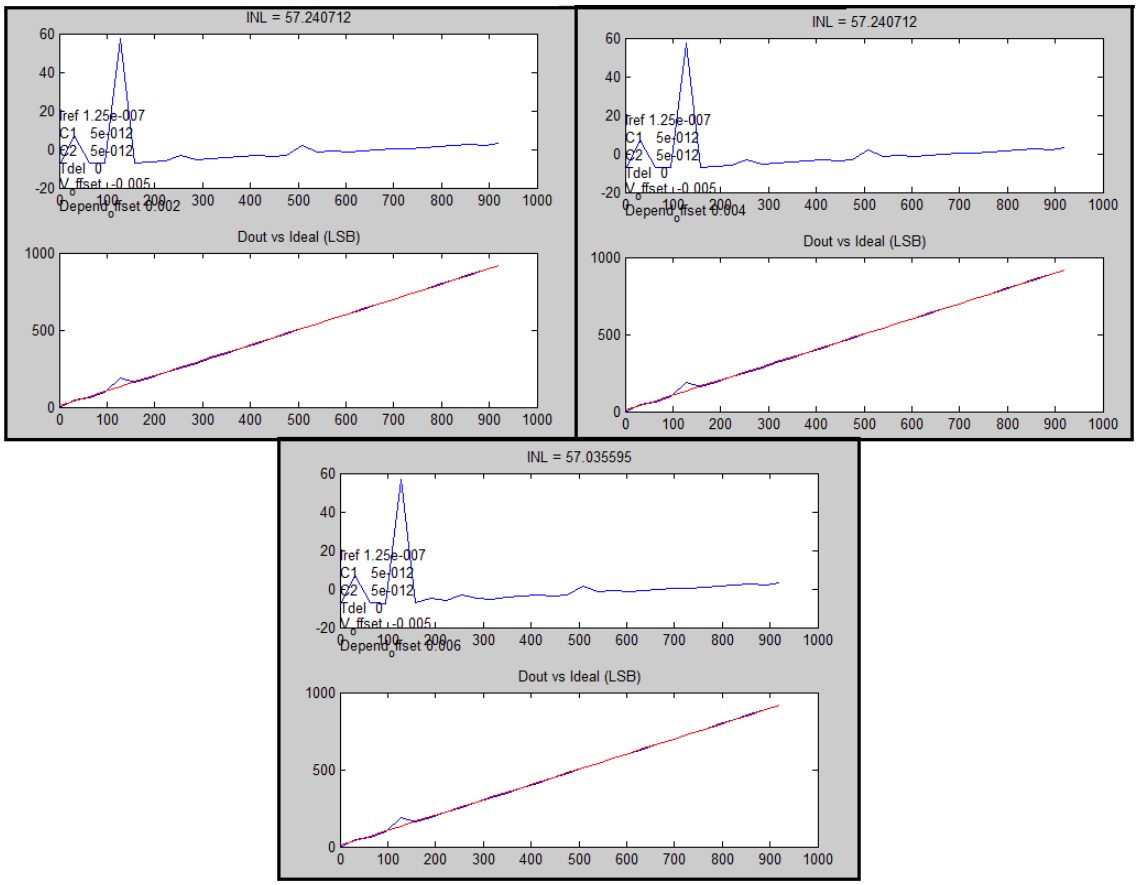

Figura 96. Resultado INL. Offset dependiente para V_offset=0.0050V

En la *Figura 96* se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0050V. Si se presta atención a los resultados obtenidos en el apartado anterior, el resultado es prácticamente el mismo que se obtiene sin incluir offset dependiente para todos los valores de offset, apareciendo alguna variación poco significativa, ya que el orden de magnitud de la variación es muy pequeño con respecto al resultado de los errores INL que se dan para estos parámetros. Concretamente se han obtenido unos valores de error INL de 57.240712 LSBs para valores de 0.2% y 0.4% y de 57.035595 LSBs para un valor de 0.6%, los cuales son demasiado significativos para el rendimiento del comparador, haciendo insignificante la variación debida al offset dependiente. Como conclusión se dirá que el offset independiente no influye significativamente en el rendimiento de convertidor para estos valores.

Se han obtenido más resultados para valores de offset independiente de señal de 0.0075V y 0.0100V, en los cuales el resultado ha seguido la misma tendencia que en el caso anterior, dando como resultado un error de INL del mismo orden de magnitud que el obtenido para 0.050V, el cual es muy negativo para el rendimiento del ADC.

5.2.2.1.6 Influencia de todo el conjunto de parámetros no Ideales

Hasta ahora se ha estudiado la influencia de todos los parámetros no ideales en el rendimiento del ADC por separado, sin tener en cuenta la influencia de los demás. En este apartado se va a realizar un estudio de todos los parámetros no ideales conjuntamente, es decir, se estudiarán la influencia de los distintos parámetros actuando al mismo tiempo. Para este estudio se van a mostrar los resultados más representativos obtenidos tras la simulación.

En primer lugar se va a estudiar la influencia del retraso del comparador (*Tdelay*), junto con el mistmach de capacidades. En las siguientes figuras se pueden observar los resultados con los distintos valores de capacidades y de retraso del comparador, dando al retraso del comparador valores de 2ns, 10ns y 100ns por un lado, y variando una de las capacidades desde 4.75pF hasta 5.25pF en pasos de 10pF por otro.

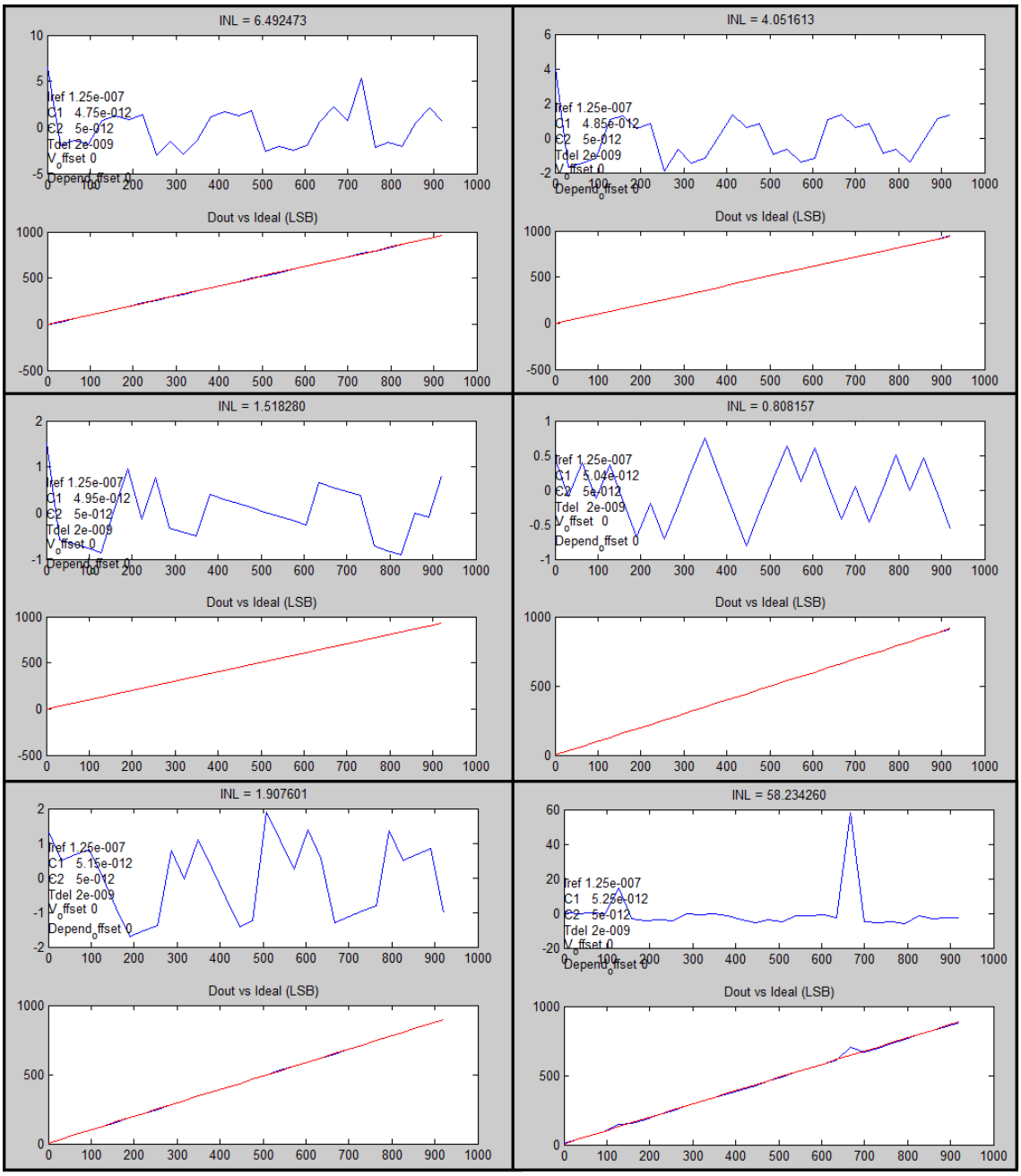

Figura 97. Resultado INL. Tdelay=2ns. Mistmach de capacidades

En la *Figura 97* se observan los resultados obtenidos tras incluir el efecto del retraso de 2ns en el comparador junto con el mistmach de capacidades para analizar el error INL. Si se atiende a los resultados en los que se estudia el mistmach de capacidades expuestos anteriormente en el apartado 5.2.2.1.3, solo difiere para tres de los valores de capacidad, siendo la diferencia poco significativa, variando en órdenes de magnitud por debajo de del orden más significativo del error. Ante este hecho, se puede concluir que predomina el error INL generado por el mistmach de capacidades para este valor de retraso, influyendo este último de manera poco significativa. Los valores del error INL en los que se difiere del error generado por el mistmach de capacidades son 6.492473 LSBs, 0.808157 LSBs y 58.234280 LSBs para unos valores de capacidad de 4.75pF, 5.05pF y 5.25pF respectivamente.

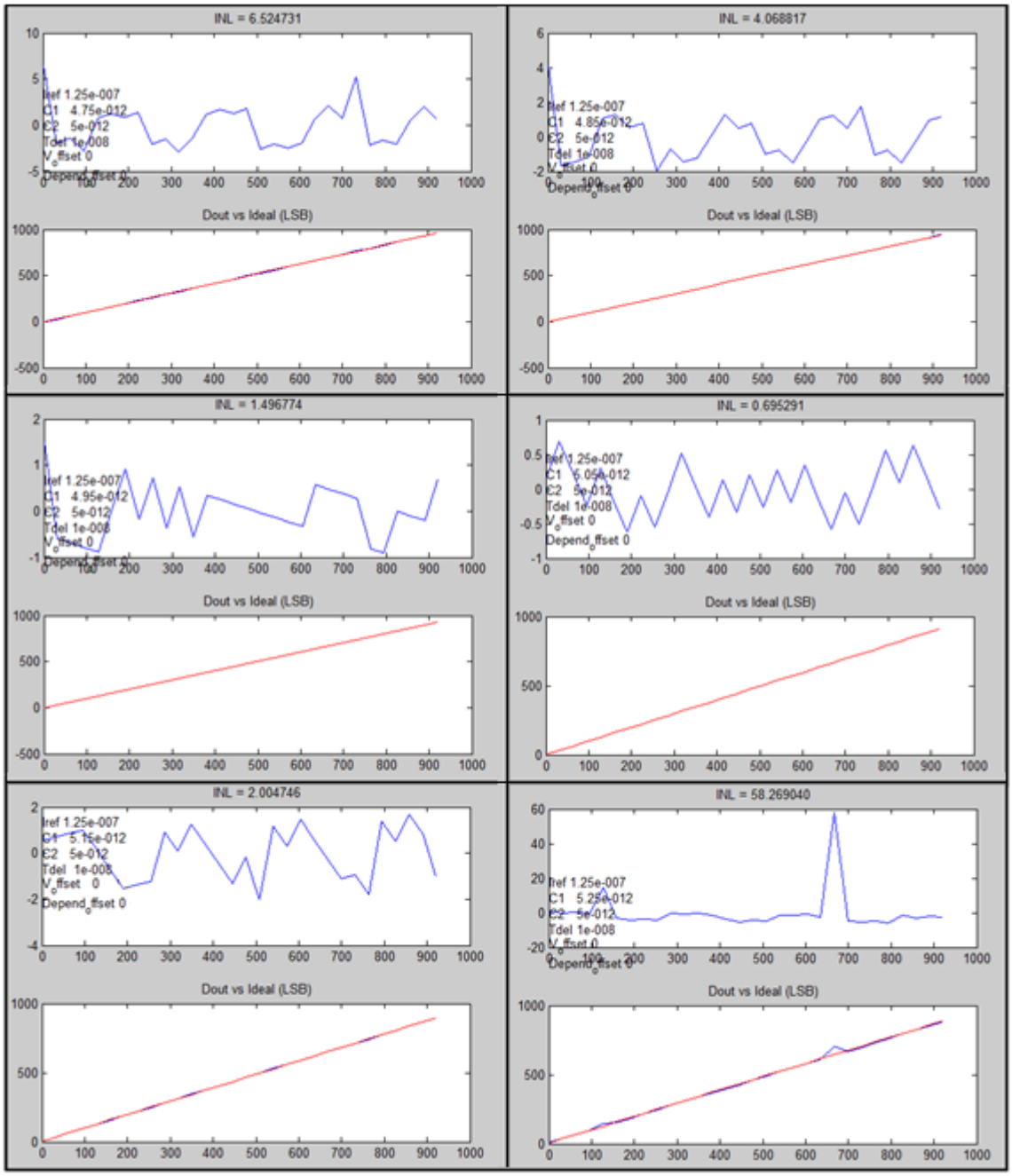

Figura 98. Resultado INL. Tdelay=10ns. Mistmach de capacidades

En la *Figura 98* se observan los resultados obtenidos tras incluir el efecto del retraso de 10ns en el comparador junto con el mistmach de capacidades para analizar el error INL. Si se observan los resultados en los que se estudia el mistmach de capacidades expuestos anteriormente en el apartado 5.2.2.1.3, difiere para todos los valores de capacidades, pero no significativamente. Por este motivo se puede concluir que predomina el error INL debido al mistmach para este valor de retraso, obteniendo unos resultados de 6.524731 LSBs, 4.068817 LSBs, 1.496774LSBs, 0.596291 LSBs, 2.004746 LSBs y 58.269040 LSBs de error INL para 4.75pF, 4.85pF, 4.95pF, 5.05pF, 5.15pF y 5.25pF respectivamente, los cuales son muy similares a los resultados obtenidos sin retraso. Ante estos resultados, se puede observar que si se consigue un retraso para al comparador igual o menor a 10ns y que no haya offset de ningún tipo en el comparador, se podría admitir un mistmach de capacidades oscilando estas entre 4.95pF y 5.15pF, ya que para estos valores de capacidad se produce un error INL admisible.

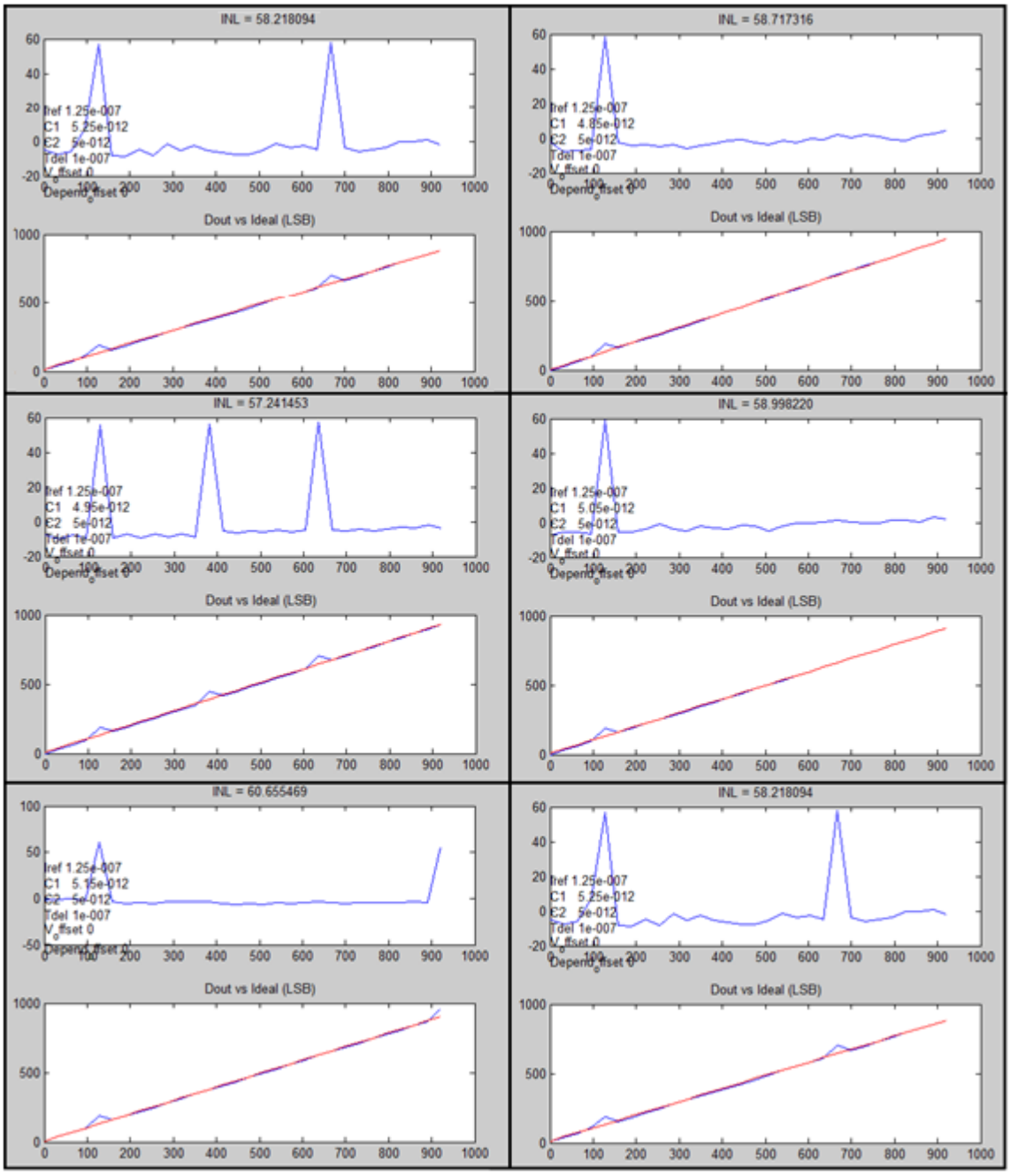

Figura 99. Resultado INL. Tdelay=100ns. Mistmach de capacidades

En la *Figura 99* se observan los resultados obtenidos tras incluir el efecto del retraso de 100ns en el comparador junto con el mistmach de capacidades para analizar el error. En el apartado 5.2.2.1.2, donde se estudiaba la influencia del retraso del comparador se comento que máximo se podría llegar un retraso del comparador de 50ns, por lo tanto en este caso, al analizar el error INL para un retraso de 100ns junto con el mistmach se espera que no se obtengan resultados admisibles para este error. No obstante, se han presentado los resultados ya que se podría pensar a priori que el error debido al retraso se compensa de alguna forma con el mistmach de capacidades. Si se observan los resultados, como se esperaba a priori, los resultados para los distintos valores de capacidades son muy negativos en lo que al error INL se refiere, obteniendo un error cercano a los 60 LSBs para todas las capacidades. Una vez visto esto, se concluye que no solo no se compensa el error producido por el retraso del comparador, sino que empeora significativamente.

Una vez estudiada la influencia del retraso del comparador, junto con el mistmach de capacidades, se va estudiar la influencia del retraso junto con la influencia del offset independiente de señal. En las siguientes figuras se pueden observar los resultados con los

distintos valores de offset independiente y de retraso del comparador, dando al retraso del comparador valores de 2ns, 10ns y 100ns por un lado, y para el offset dependiente 0.0005V 0.0025V, 0.0050V y 0.075V para cada valor de retraso por otro.

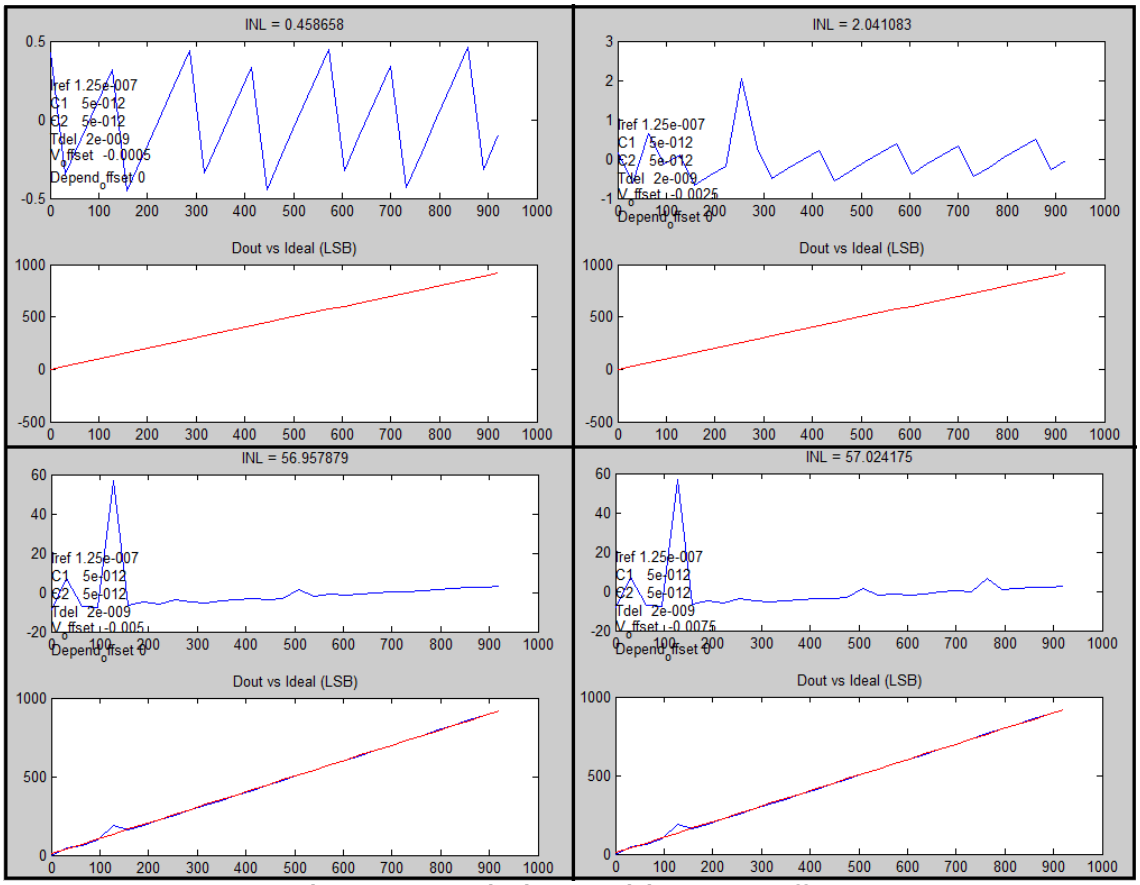

Figura 100. Resultado INL. Tdelay=2ns. V_offset

En la *Figura 100*, donde se presentan los resultados para un retraso de 2ns y distintos valores de offset independiente de señal del comparador, se observa que se obtienen prácticamente los mismos resultados que cuando se analizaba el offset independiente sin incluir ningún tipo de retraso, en el apartado 5.2.2.1.4. Solo varían en algunas décimas en el error INL para los valores de offset de 0.0050V y 0.0075V, donde el error INL es demasiado significativo, en torno a 60 LSBs, por lo que el aumento del error al incluir un retraso de 2ns no es importante. Por tanto, se puede concluir que un retraso tan pequeño no influye demasiado si se tiene en cuenta el efecto del offset independiente de señal del comparador, obteniendo resultados admisibles para unos valores de offset de 0.0005V y 0.0025V donde se obtiene un error INL de 0.458658 LSBs y 2.041083 LSBs respectivamente.

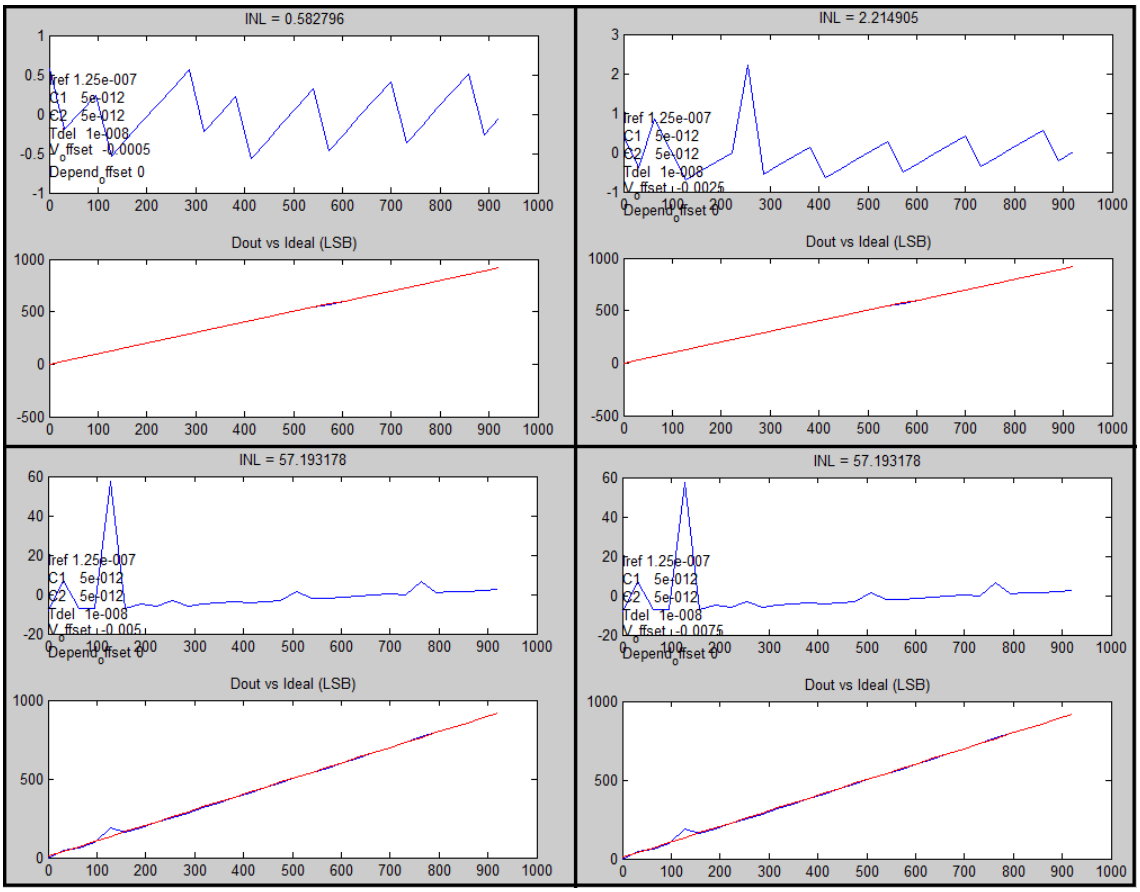

Figura 101. Resultado INL. Tdelay=10ns. V_offset

En la *Figura 101*, donde se presentan los resultados para un retraso de 10ns y distintos valores de offset independiente de señal del comparador, se observa que se obtienen resultados similares a los que se obtenían cuando se analizaba el offset independiente sin incluir ningún tipo de retraso, en el apartado 5.2.2.1.4. A diferencia de los resultados obtenidos para un retraso de 2ns, visto en la figura inmediatamente anterior, el error aumenta para todos los valores, aunque no lo hace significativamente, ya que para 0.0005V y 0.0025V de offset se obtiene un error INL de 0.582796 LSBs y 2.214905 LSBs respectivamente, los cuales siguen resultando admisibles, obteniendo para un offset mayor resultados demasiado altos de error INL como resultaba anteriormente.

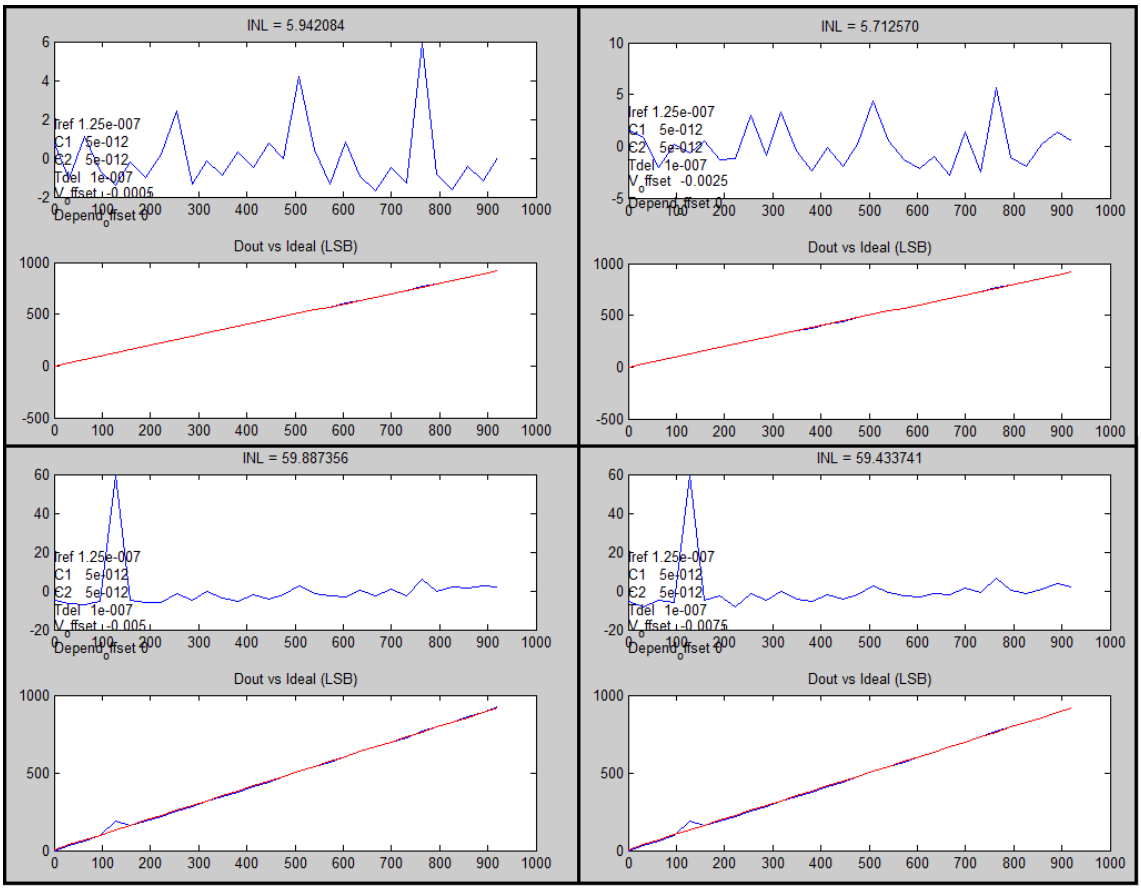

Figura 102. Resultado INL. Tdelay=100ns. V_offset

En la Figura 101, donde se presentan los resultados para un retraso de 10ns y distintos valores de offset independiente de señal del comparador, se puede observar que para ningún valor de offset se obtienen resultados admisibles de error INL, afectando significativamente al rendimiento del convertidor. Los resultados obtenidos han sido 5.942084 LSBs, 5.712570 LSBs, 59.887356 LSBs y 59.433741 LSBs para unos valores de offset de 0.0005V, 0.0025V, 0.0050V y 0.0075V respectivamente. Los resultados para los dos primeros valores de offset, pese a no ser demasiado elevados resultan demasiado altos, ya que un error en torno a 6 LSBs afectan negativamente al rendimiento del convertidor de una forma significativa. Cabe destacar que aunque el error INL es demasiado alto, para los dos primeros valores de offset el error es menor que el que se obtenía con solamente con un retraso de 100ns, por lo que se deduce que los errores se compensan de alguna forma. Concluyendo, para un retraso de 100ns y teniendo en cuenta el offset dependiente de señal, no se obtienen resultados satisfactorios para el rendimiento del convertidor.

A continuación, al análisis realizado para el estudio de la influencia del retraso y del offset independiente de señal del comparador en el rendimiento del convertidor, se van a estudiar los resultados añadiendo la influencia del offset dependiente de señal al efecto de los parámetros anteriores. Para ello se van a mostrar en las siguientes figuras los resultados con los distintos valores de offset dependiente, offset independiente y de retraso del comparador. Para ello se v a dar unos valores al retraso del comparador de 2ns, 10ns y 100ns por un lado, y un el offset independiente 0.0005V 0.0025V y 0.0050V para cada valor de retraso por otro, variando el porcentaje de offset dependiente de señal con valores del 0.2%, 0.4% y 0.6.

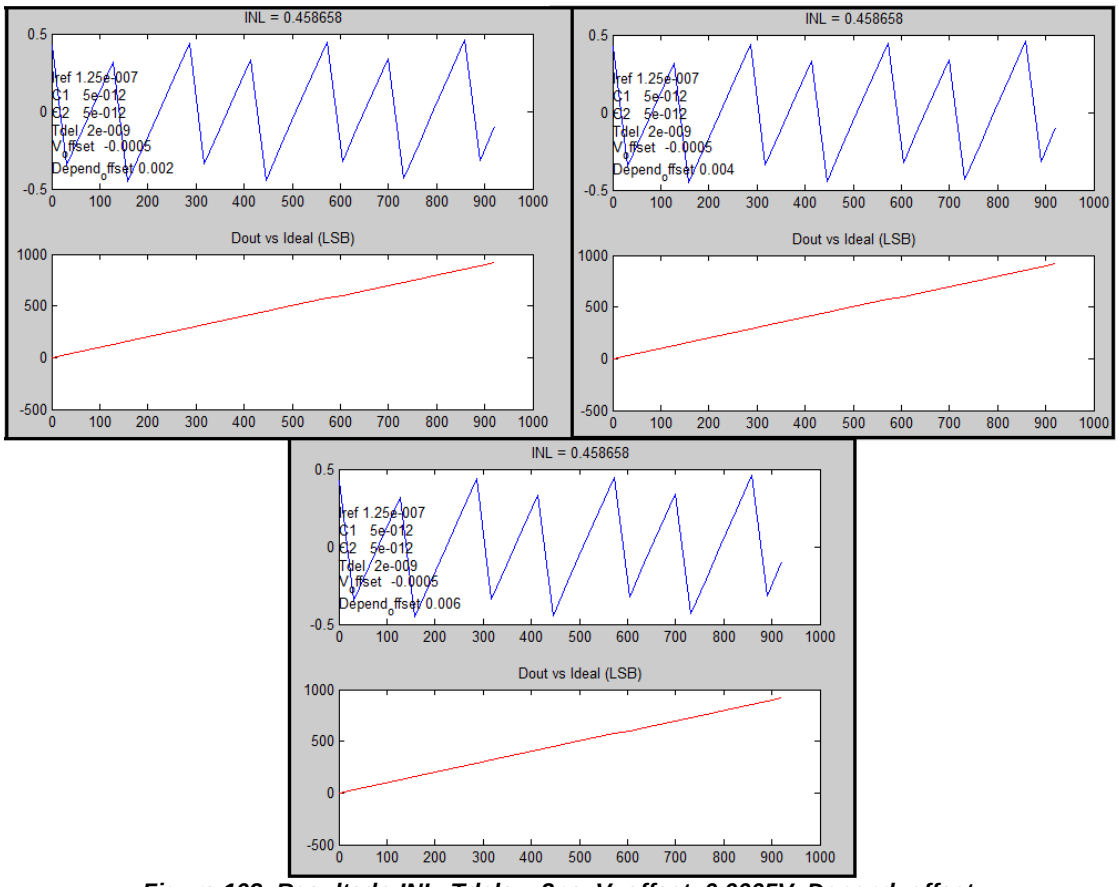

Figura 103. Resultado INL. Tdelay=2ns. V_offset=0.0005V. Depend_offset

En la *Figura 103* se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que, como ocurría en el apartado 5.2.2.1.5, el offset dependiente de señal no afecta significativamente, nada en este caso, en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 2ns y un offset independiente de 0.0005V, un error INL para todos los valores de offset dependiente de 0.458658 LSBs, el cual como se ha dicho anteriormente es un resultado satisfactorio para el rendimiento del convertidor.

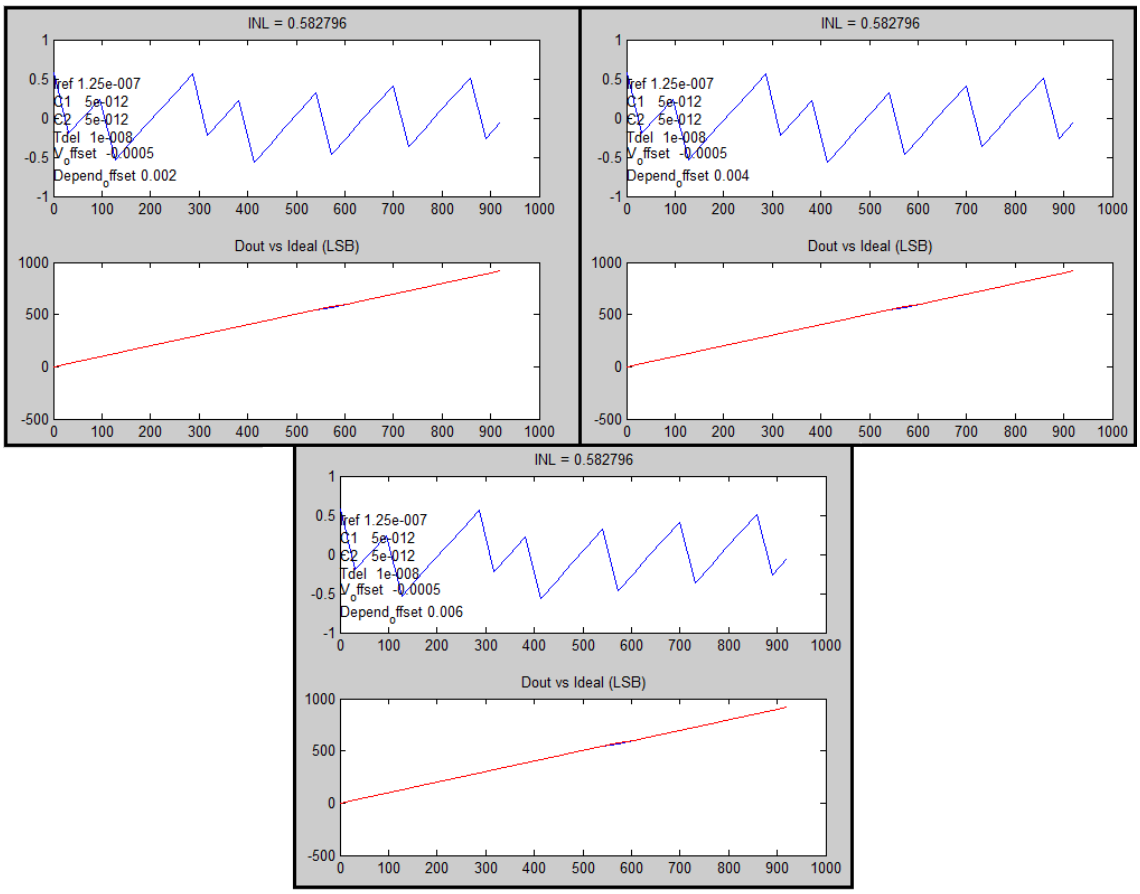

Figura 104. Resultado INL. Tdelay=10ns. V_offset=0.0005V. Depend_offset

En la *Figura 104* se observan los resultados obtenidos para un retraso de 10ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que se produce el mismo comportamiento anterior, ya que el offset dependiente de señal no afecta significativamente, nada en este caso, en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 10ns y un offset independiente de 0.0005V, un error INL para todos los valores de offset dependiente de 0.582796 LSBs, el cual como se ha dicho anteriormente es un resultado satisfactorio para el rendimiento del convertidor.

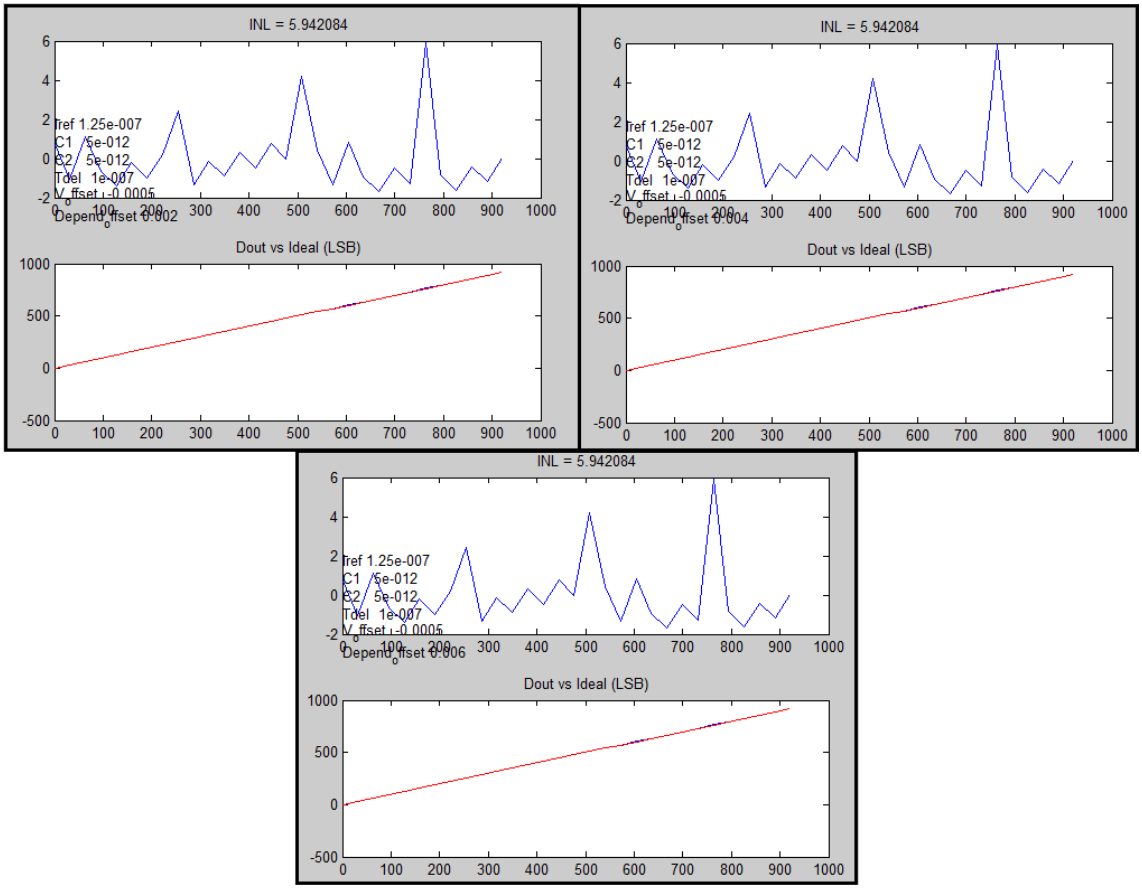

Figura 105. Resultado INL. Tdelay=100ns. V_offset=0.0005V. Depend_offset

En la Figura 104 se observan los resultados obtenidos para un retraso de 100ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que se produce el mismo comportamiento ya que el offset dependiente de señal no afecta significativamente, nada en este caso, en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 100ns y un offset independiente de 0.0005V, un error INL para todos los valores de offset dependiente de 5.942084 LSBs, el cual como se ha dicho anteriormente es un resultado no satisfactorio para el rendimiento del convertidor, como era de esperar ante los resultados obtenidos anteriormente.

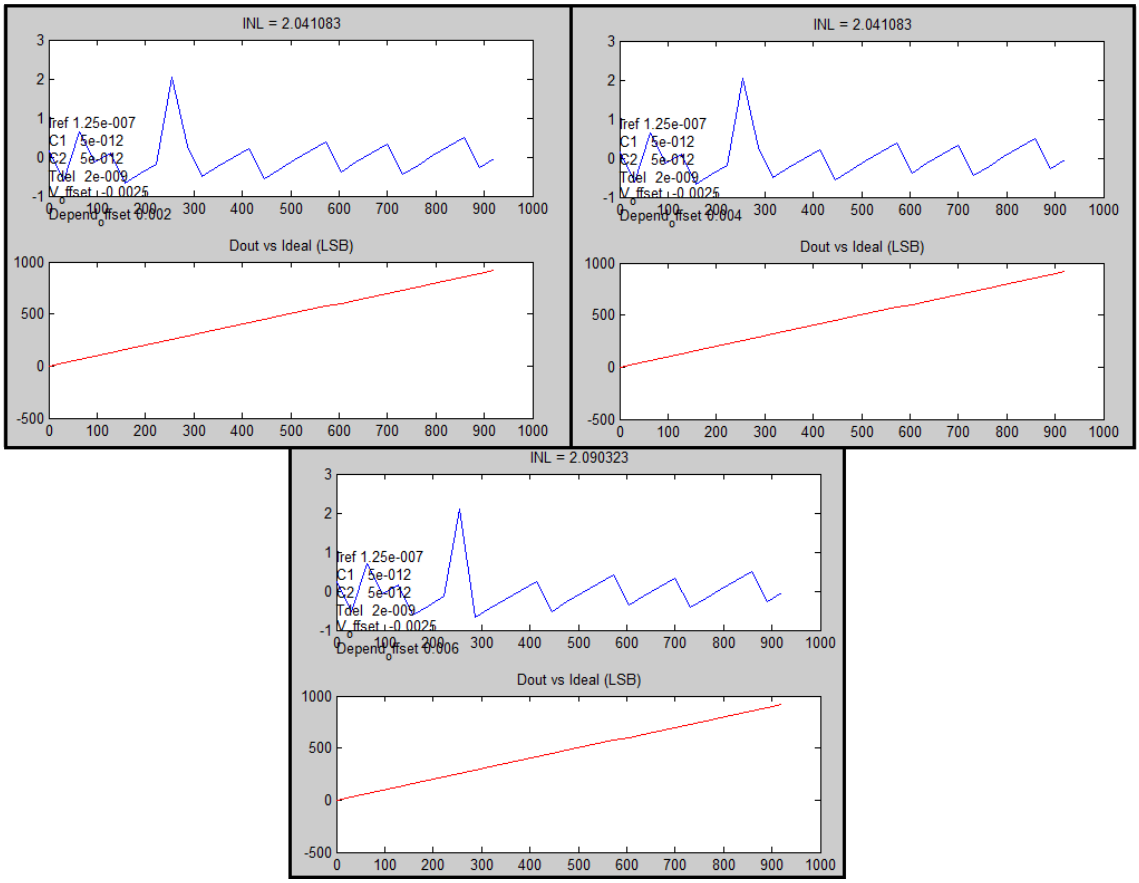

Figura 106. Resultado INL. Tdelay=2ns. V_offset=0.0025V. Depend_offset

En la *Figura 106* se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que se produce el mismo comportamiento ya que el offset dependiente de señal no afecta significativamente, en este caso solo varía y además de forma muy poco significativa para un 0.6% de offset dependiente, en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 2ns y un offset independiente de 0.0025V, un error INL para 0.2% y 0.4% de de offset dependiente de 2.041083 LSBs, y de 2.090323 LSBs para un offset dependiente de 0.6%. Esto supone un resultado satisfactorio para el rendimiento del convertidor.

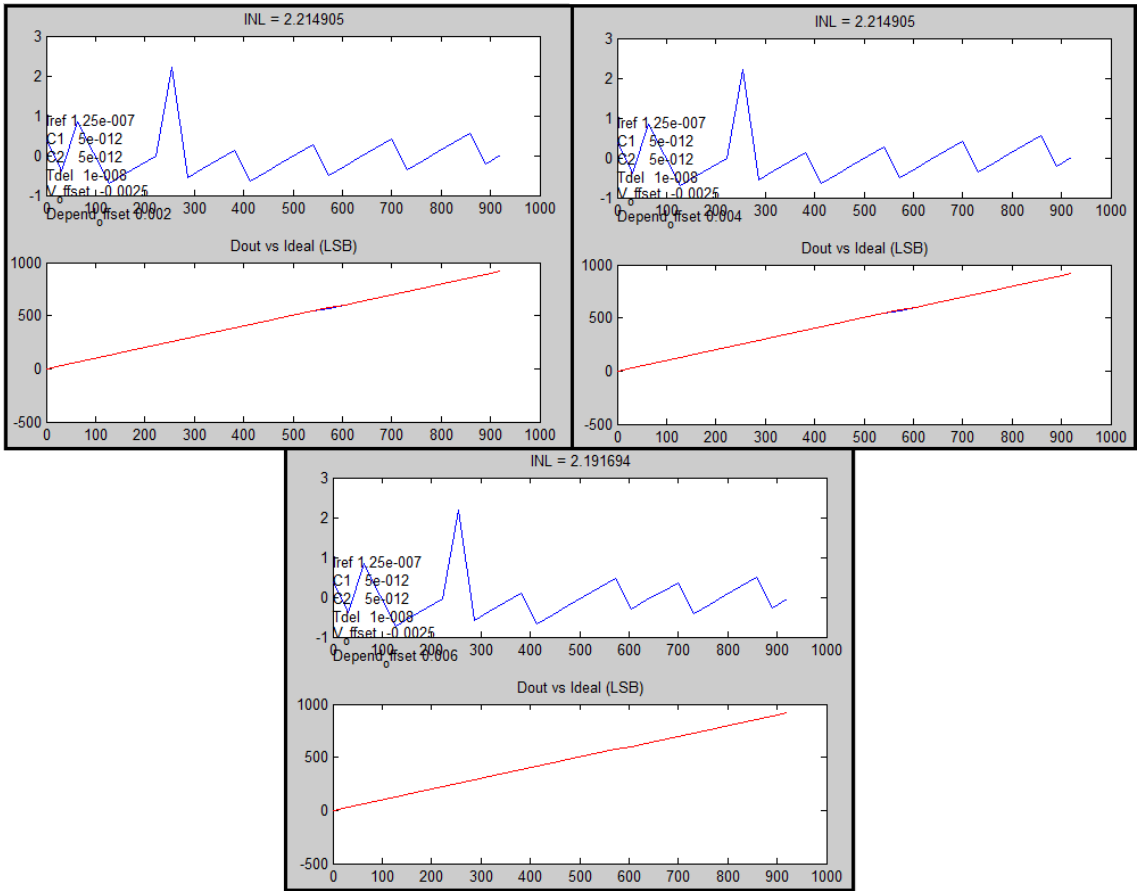

Figura 107. Resultado INL. Tdelay=10ns. V_offset=0.0025V. Depend_offset

En la *Figura 107* se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que se produce el mismo comportamiento ya que el offset dependiente de señal no afecta significativamente, en este caso solo varía y además de forma muy poco significativa para un 0.6% de offset dependiente, en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 10ns y un offset independiente de 0.0025V, un error INL para 0.2% y 0.4% de de offset dependiente de 2.041083 LSBs, y de 2.090323 LSBs para un offset dependiente de 0.6%. Esto supone un resultado satisfactorio para el rendimiento del convertidor.

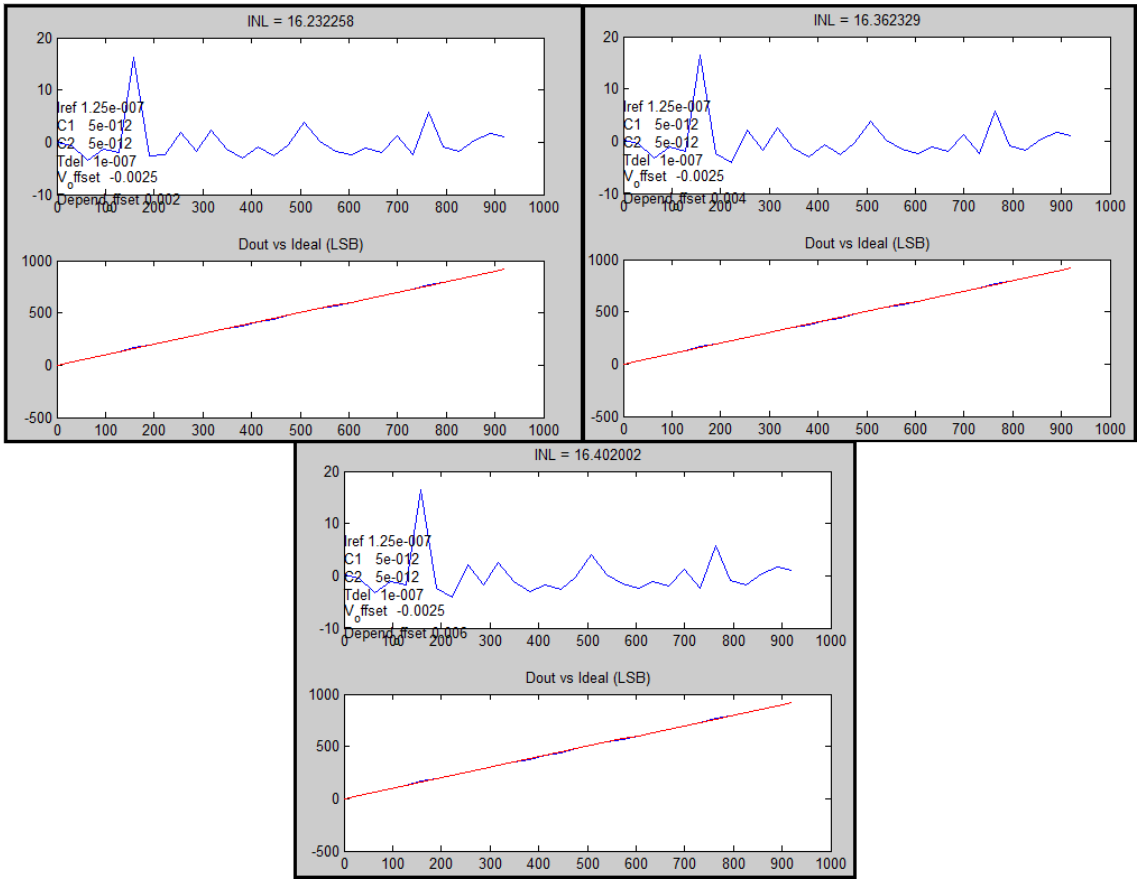

Figura 108. Resultado INL. Tdelay=100ns. V_offset=0.0025V. Depend_offset

En la *Figura 108* se observan los resultados obtenidos para un retraso de 100ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. En este caso se aprecia una influencia más significativa del offset dependiente del comparador, ya que para los tres valores de este se obtiene un resultado de error INL de 16.232258 LSBs, 16.362329 y 16.402002 LSBs para unos valores de offset dependiente del 0.2%, 0.4% y 0.6% respectivamente, frente a los 5.712570 LSBs que se obtenían cuando se analizaba la influencia para un retraso de 100ns y un offset independiente de 0.0025V. De esta manera se ve la influencia del offset dependiente, empeorando considerablemente el resultado. Si antes se decía que el resultado no era satisfactorio para el rendimiento del convertidor, con más razón, en este caso no se consigue un resultado satisfactorio.

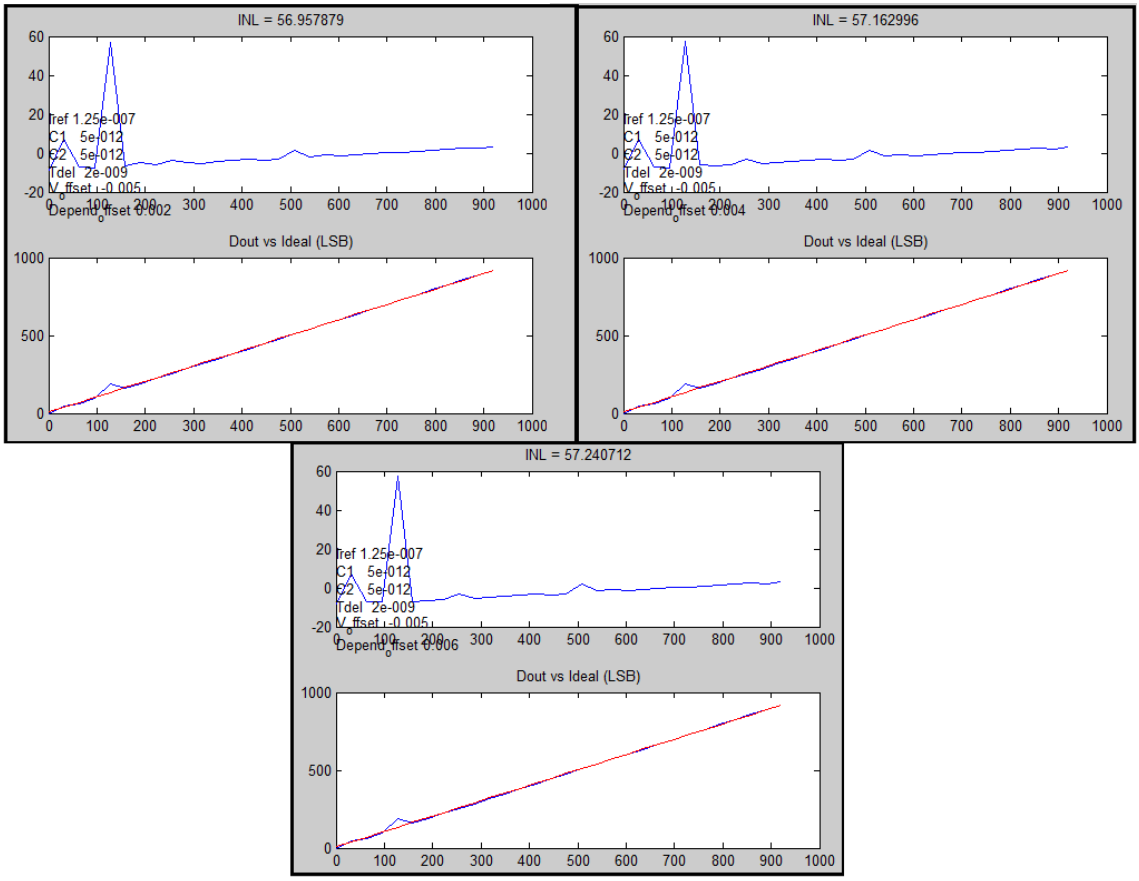

Figura 109. Resultado INL. Tdelay=2ns. V_offset=0.0050V. Depend_offset

En la *Figura 109* se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0050V y para los diferentes valores de offset dependiente de señal. En este caso se sigue la tendencia en la que el offset dependiente de señal no afecta significativamente, en este caso varía de forma muy poco significativa para un 0.4% y un 0.6% de offset dependiente, obteniendo unos valores de 57.162996 LSBs y 57.240712 LSBs de error INL, frente a los 56.957879 que se obtienen tanto para un 0.2% como cuando se analizaba la influencia para un retraso de 2ns y un offset independiente de 0.0025V. En el estudio anterior se concluía que el resultado no era satisfactorio ya que el error INL es muy alto, y en este caso, por comportarse de la misma forma se obtienen las mismas conclusiones, es decir, un resultado no satisfactorio para el rendimiento del convertidor.

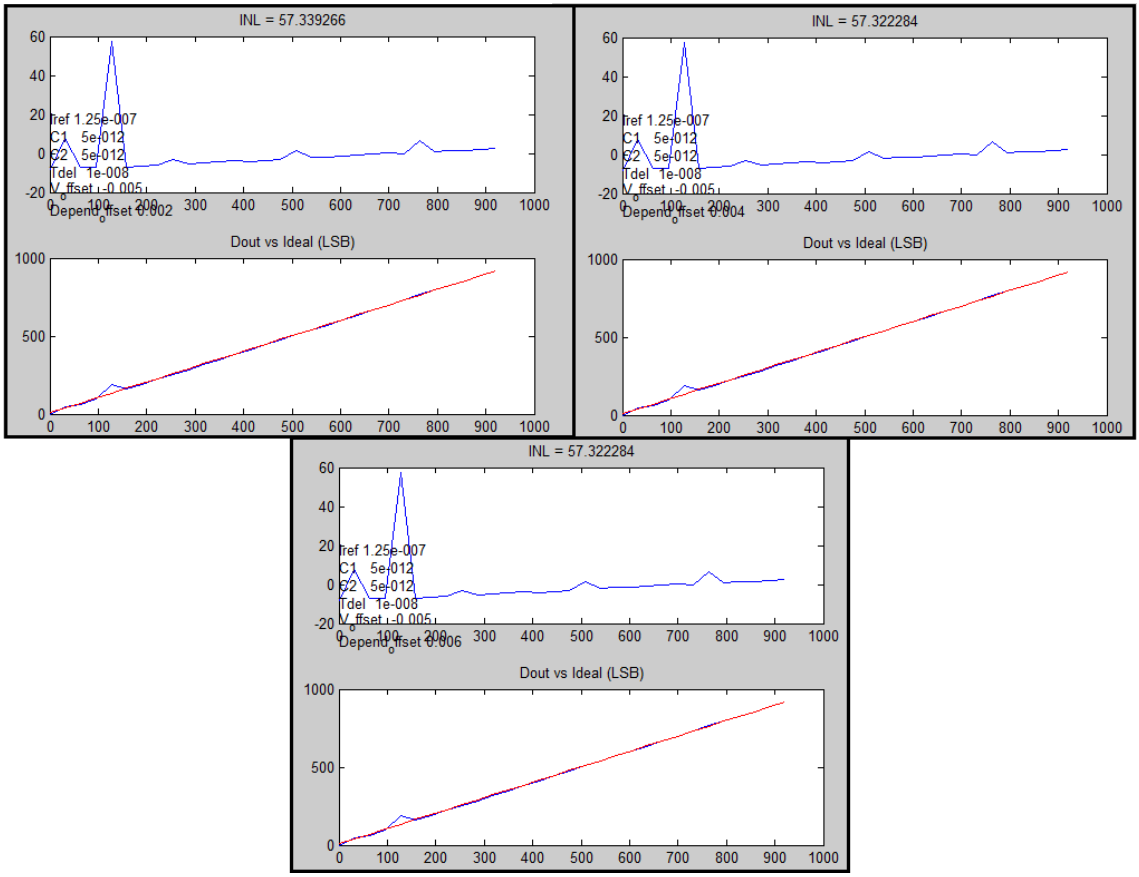

Figura 110. Resultado INL. Tdelay=10ns. V_offset=0.0050V. Depend_offset

En la *Figura 110* se observan los resultados obtenidos para un retraso de 10ns, un valor de offset independiente de señal de 0.0050V y para los diferentes valores de offset dependiente de señal. En este caso se produce el mismo comportamiento que en el caso anterior, ya que el offset dependiente de señal no afecta significativamente, en este caso varía de forma muy poco significativa para todos los valores de offset dependiente, obteniendo unos valores de 57.339266 LSBs de error INL para un offset dependiente de 0.2% y de 57.322284 LSBs de error INL para unos valores de offset dependiente de 0.4% y 0.6%, frente a los 57.193178 que se obtenían cuando se analizaba la influencia para un retraso de 10ns y un offset independiente de 0.0050V. En el estudio anterior se concluía que el resultado no era satisfactorio ya que el error INL es muy alto, y en este caso, por comportarse de la misma forma se obtienen las mismas conclusiones, es decir, un resultado no satisfactorio para el rendimiento del convertidor.

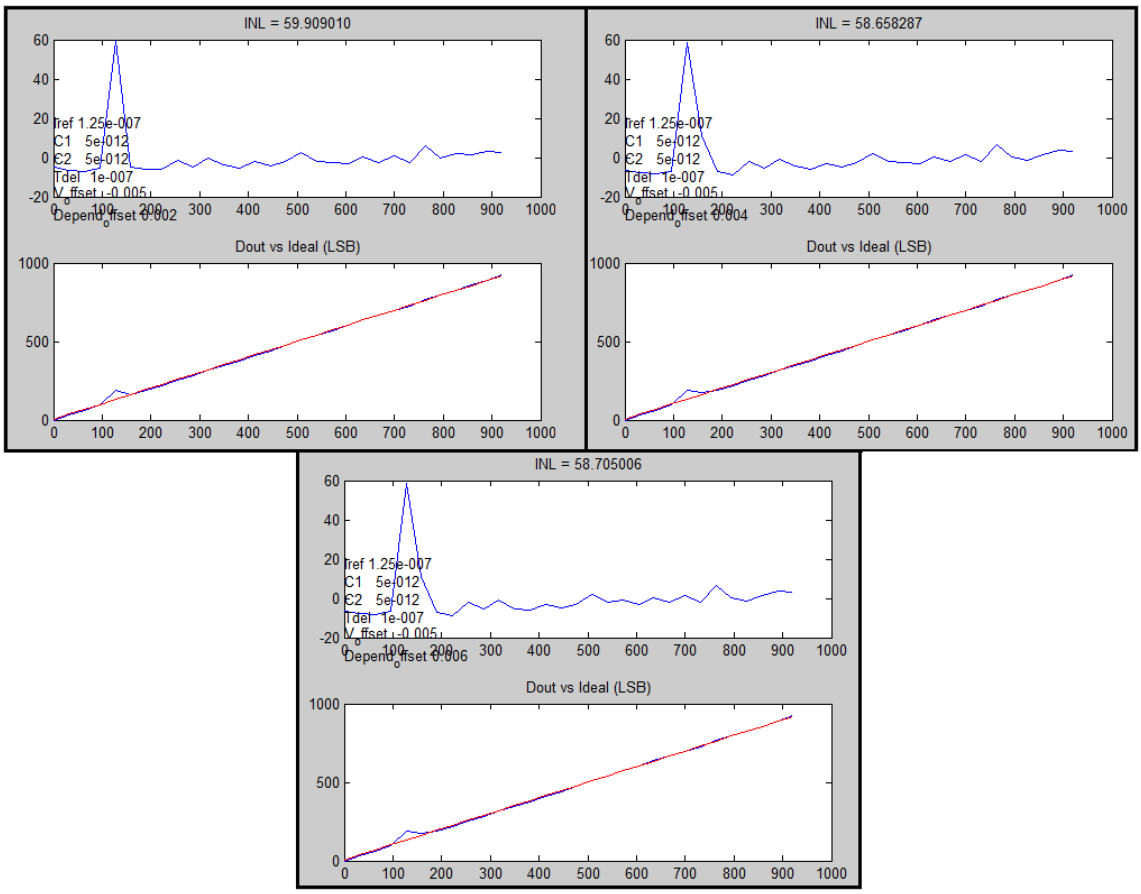

Figura 111. Resultado INL. Tdelay=100ns. V_offset=0.0050V. Depend_offset

En la *Figura 111* se observan los resultados obtenidos para un retraso de 100ns, un valor de offset independiente de señal de 0.0050V y para los diferentes valores de offset dependiente de señal. En este caso se produce el mismo comportamiento que en el caso anterior, ya que el offset dependiente de señal no afecta significativamente, variando de forma muy poco significativa para todos los valores de offset dependiente, obteniendo unos valores de error INL de 57.909010 LSBs, de 58.658287 LSBs y de 58.705006 para unos valores de offset dependiente de 0.2%, 0.4% y 0.6% respectivamente, frente a los 59.887356 LSBs que se obtenían cuando se analizaba la influencia para un retraso de 100ns y un offset independiente de 0.0050V. En el estudio anterior se concluía que el resultado no era satisfactorio ya que el error INL es muy alto, y en este caso, por comportarse de la misma forma se obtienen las mismas conclusiones, es decir, un resultado no satisfactorio para el rendimiento del convertidor.

Una vez realizados los estudios anteriores, se añade un análisis en el que intervienen todos los parámetros, es decir, el retraso y el offset tanto dependiente como independiente del comparador junto con el mistmach de capacidades. En este análisis solo se muestran los resultados más representativos, es decir, los resultados con los valores de los parámetros para los que los resultados son medianamente aceptables para el rendimiento del comparador y además el resultado con los valores de los parámetros para los que los resultados comienzan a ser negativos para el rendimiento del convertidor. En las siguientes figuras se muestran los resultados obtenidos.

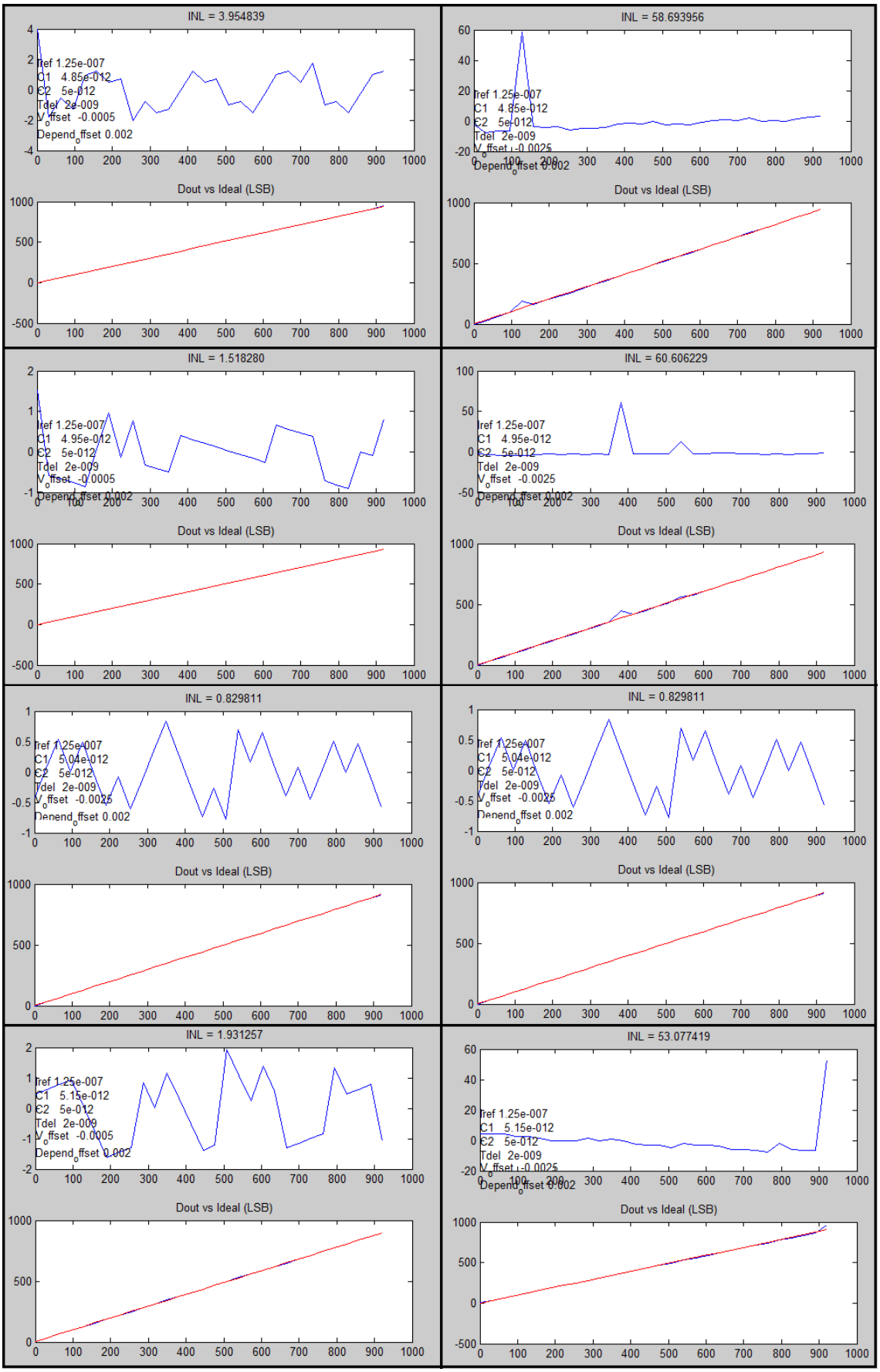

Figura 112. Resultado INL. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

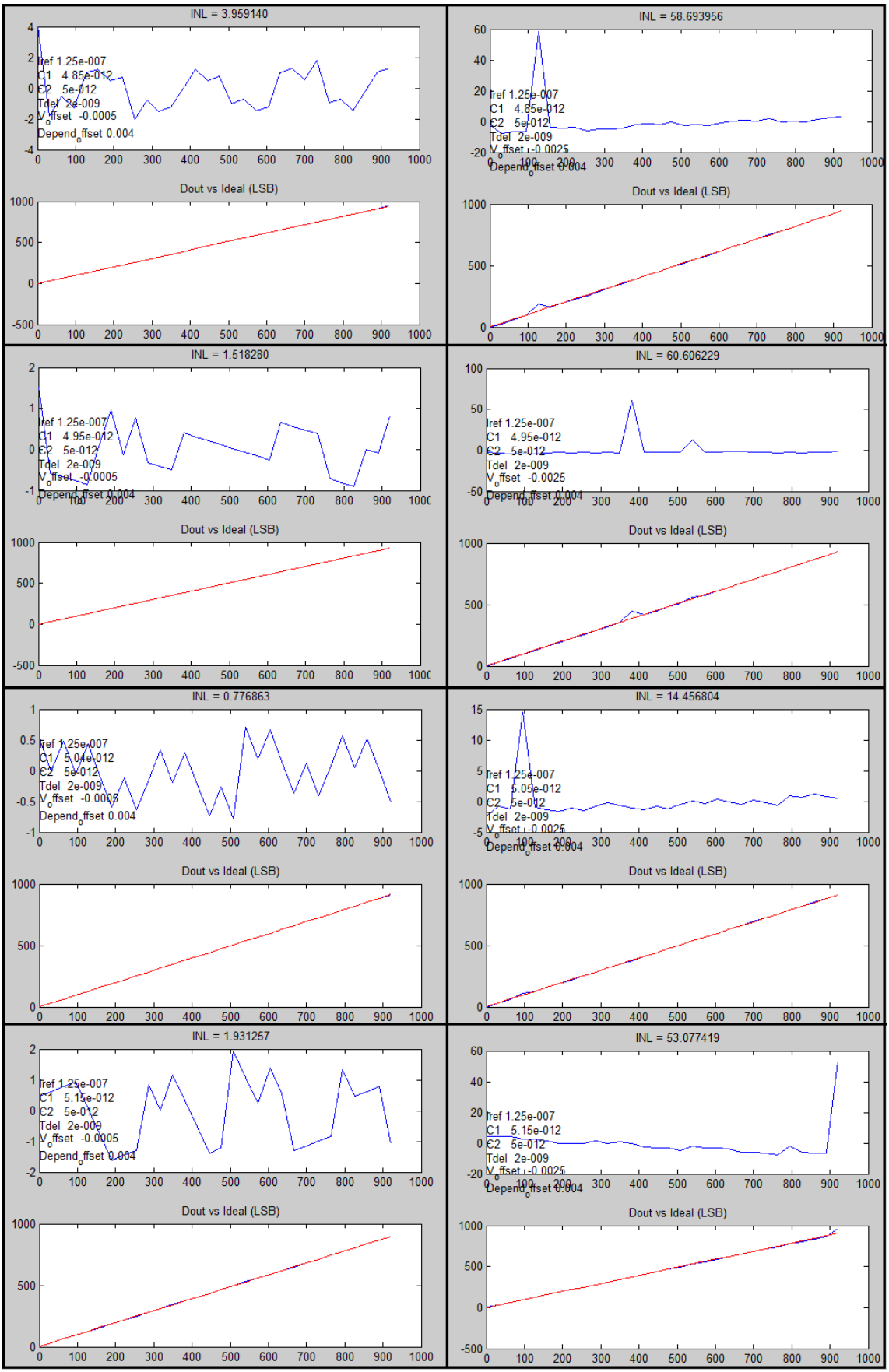

Figura 113. Resultado INL. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

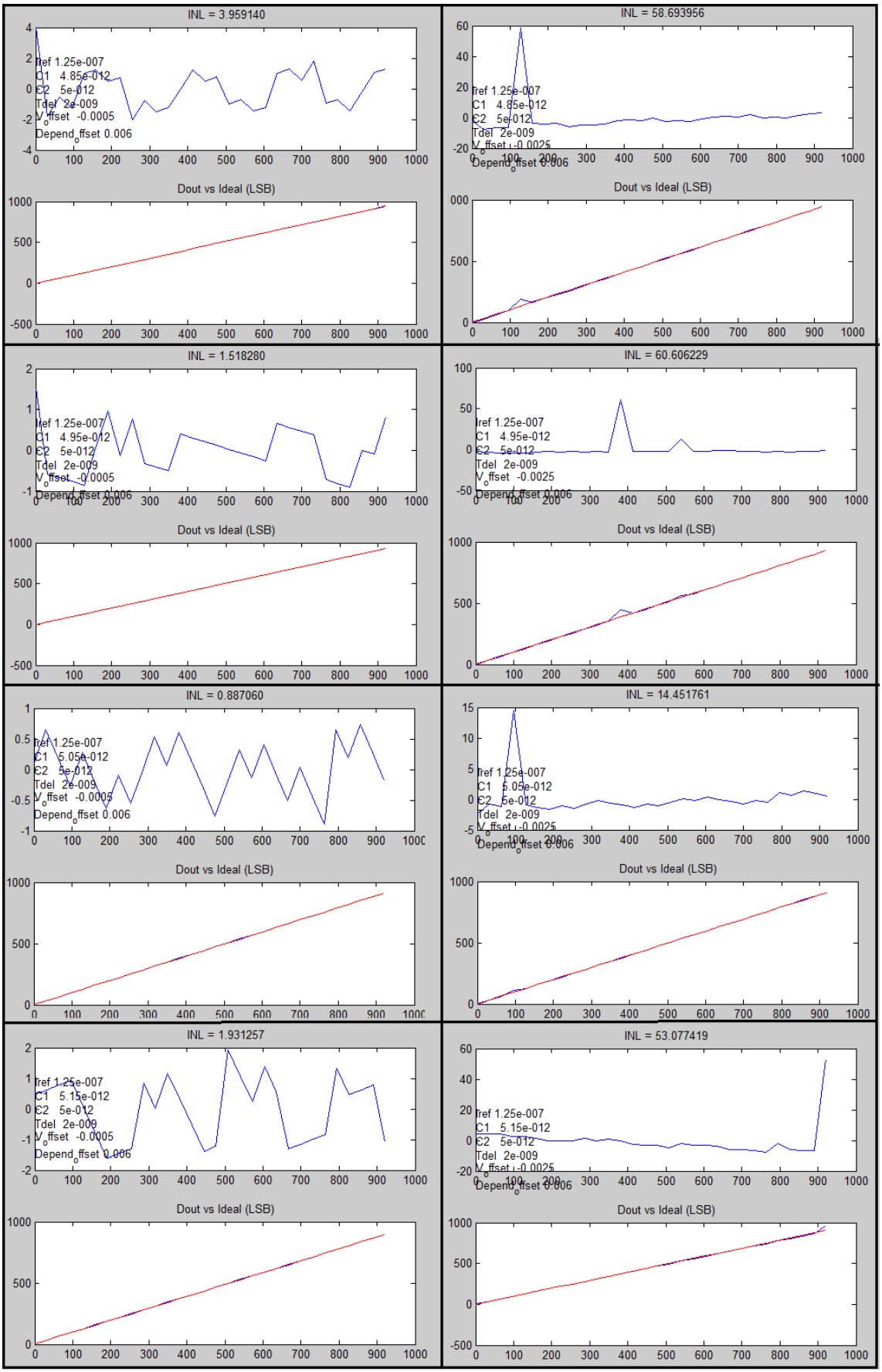

Figura 114. Resultado INL. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 112, 113 y 114 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 2ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente.

En la *Figura 112* se muestra el resultado para un offset dependiente del 0.2%, donde se puede observar que solo se dan resultados satisfactorios para el rendimiento del convertidor para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.95pF, 5.05pF y 5.15pF para los cuales se obtiene un error INL de 1.518280 LSBs, 0.829811 LSBs y 1.931257 LSBs respectivamente. Para un valor de offset independiente de 0.0025V no se obtienen resultados satisfactorios en ningún caso, ya que para la mayoría de los parámetros se obtienen errores INL en torno a los 60 LSBs, excepto para una capacidad de 5.05pF donde se obtiene un error de 14.456804 LSBs, el cual sigue siendo un error demasiado alto para obtener un buen rendimiento en el convertidor.

En la *Figura 113* se muestra el resultado para un offset dependiente del 0.4%, donde se puede observar que solo se dan resultados satisfactorios para el rendimiento del convertidor para un offset independiente de señal de 0.0005V para unos valores de capacidad de 4.95pF, 5.05pF y 5.15pF, como ocurría en el caso anterior, de hecho prácticamente se obtienen los mismos resultados. Solo difiere cuando se aplica un valor de 0.0005V de offset independiente y un valor de 5.05pF de capacidad donde se obtiene un error INL de 0.776863 LSBs, frente a los 0.829811 LSBs que se obtenían para el mismo caso en el análisis anterior. Ante estos resultados, se obtienen las mismas conclusiones que en el caso anterior, es decir, únicamente se obtienen resultados satisfactorios para un offset dependiente de 0.0005V y unas capacidad de entre 4.95 y 5.05pF.

En la *Figura 114* se muestra el resultado para un offset dependiente del 0.6%. En este caso, como en el anterior, se puede observar que solo se dan resultados satisfactorios para el rendimiento del convertidor para un offset independiente de señal de 0.0005V para unos valores de capacidad de 4.95pF, 5.05pF y 5.15pF, dándose únicamente una diferencia cuando se aplica un valor de 0.0005V de offset independiente y un valor de 5.05pF de capacidad donde se obtiene un error INL de 0.887060 LSBs, frente a los 0.829811 LSBs y 0.829811 LSBs que se obtenían para el mismo caso en los análisis anteriores. Ante estos resultados, se obtienen las mismas conclusiones que en los casos anteriores, es decir, únicamente se obtienen resultados satisfactorios para un offset dependiente de 0.0005V y unas capacidad de entre 4.95 y 5.05pF.

Con las tres figuras anteriores se han expuesto los resultados obtenidos para analizar el error INL variando todos los parámetros fijando el retraso del comparador en 2ns. Para todos los casos se han obtenido resultados satisfactorios para un valor de offset independiente de 0.0005V y una capacidad de entre 4.95pF y 5.15pF, resultando la influencia del offset dependiente poco significativa, como ocurría cuando se analizaba este offset por separado.

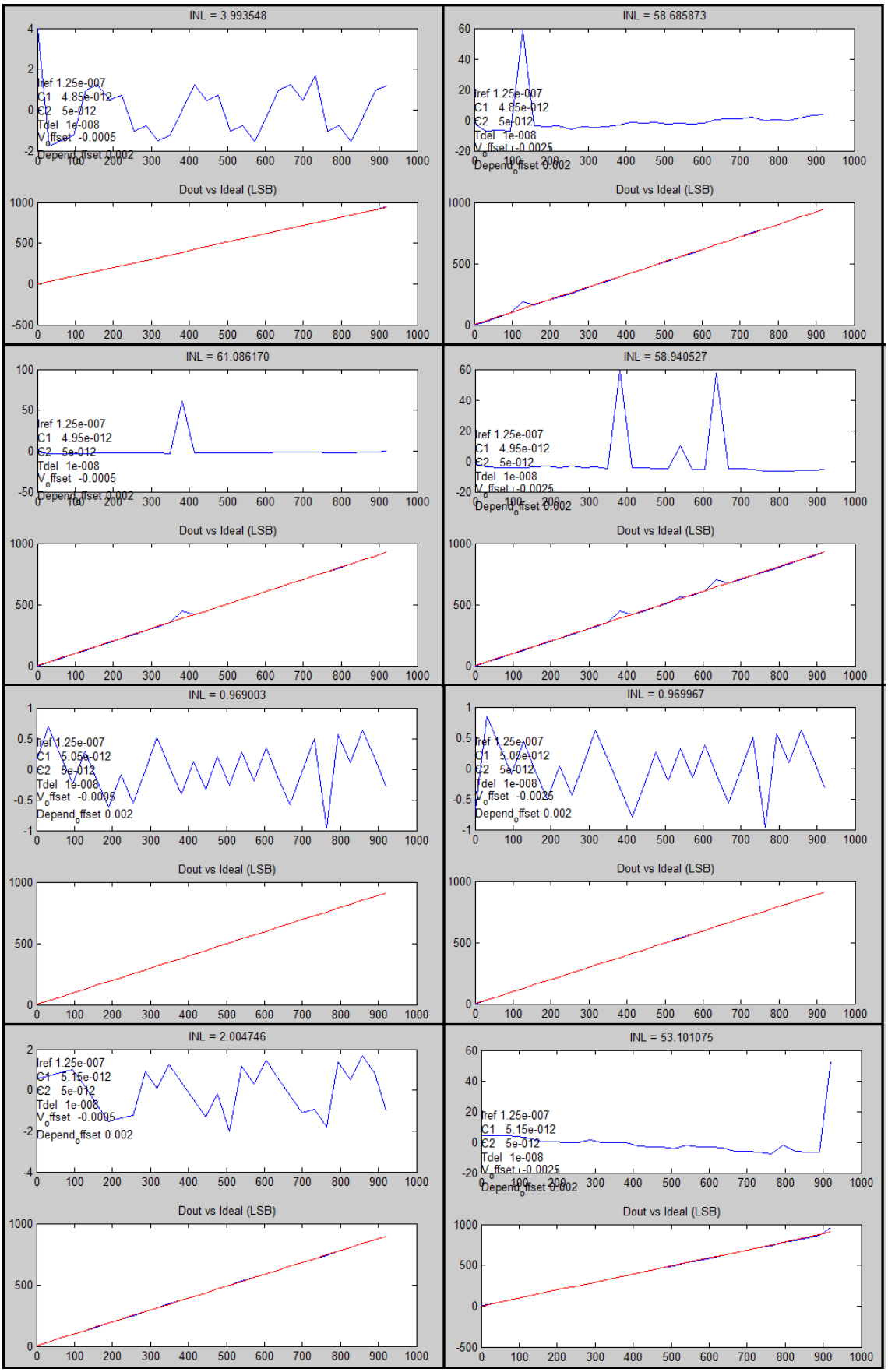

Figura 115. Resultado INL. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

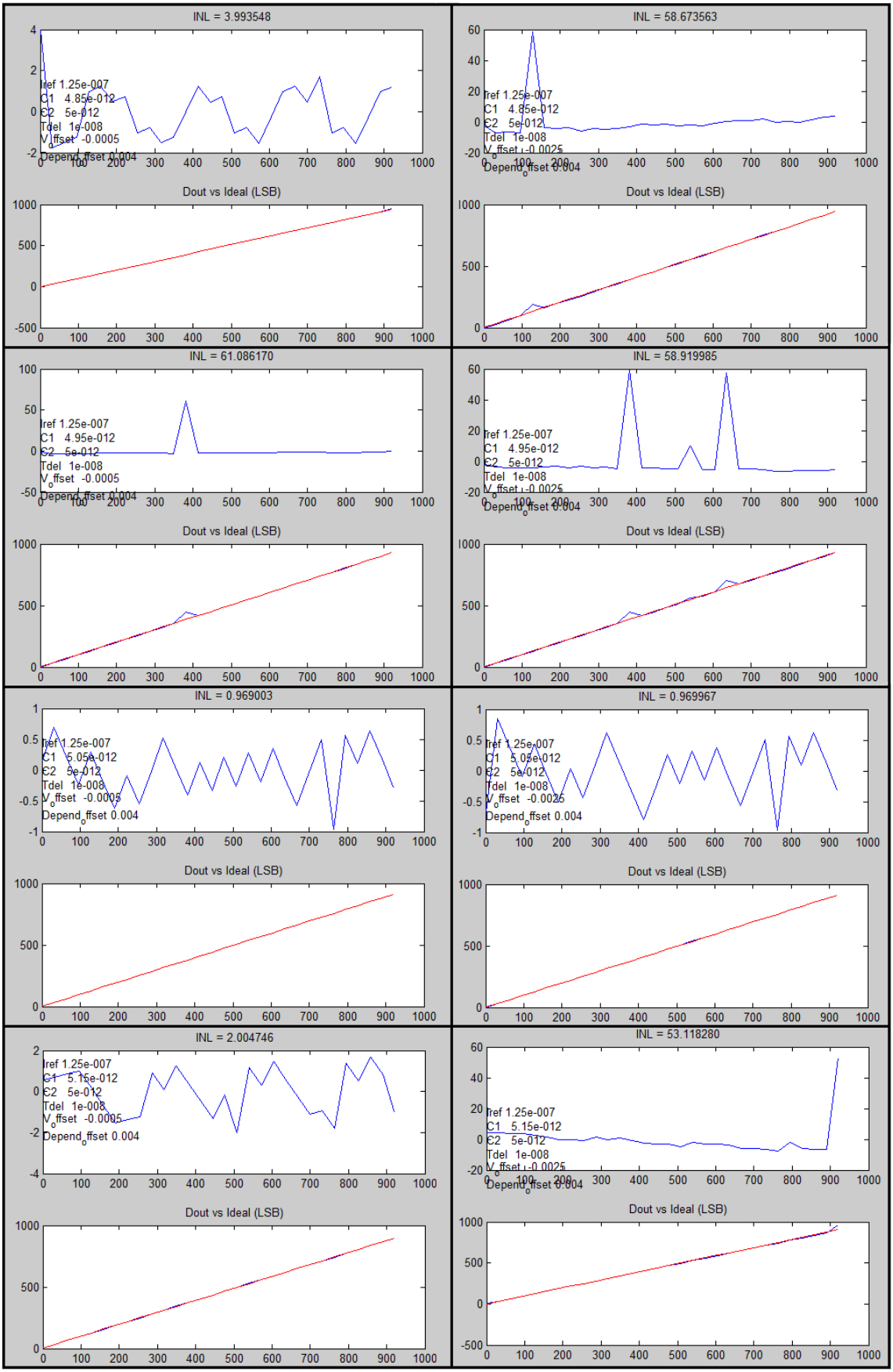

Figura 116. Resultado INL. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

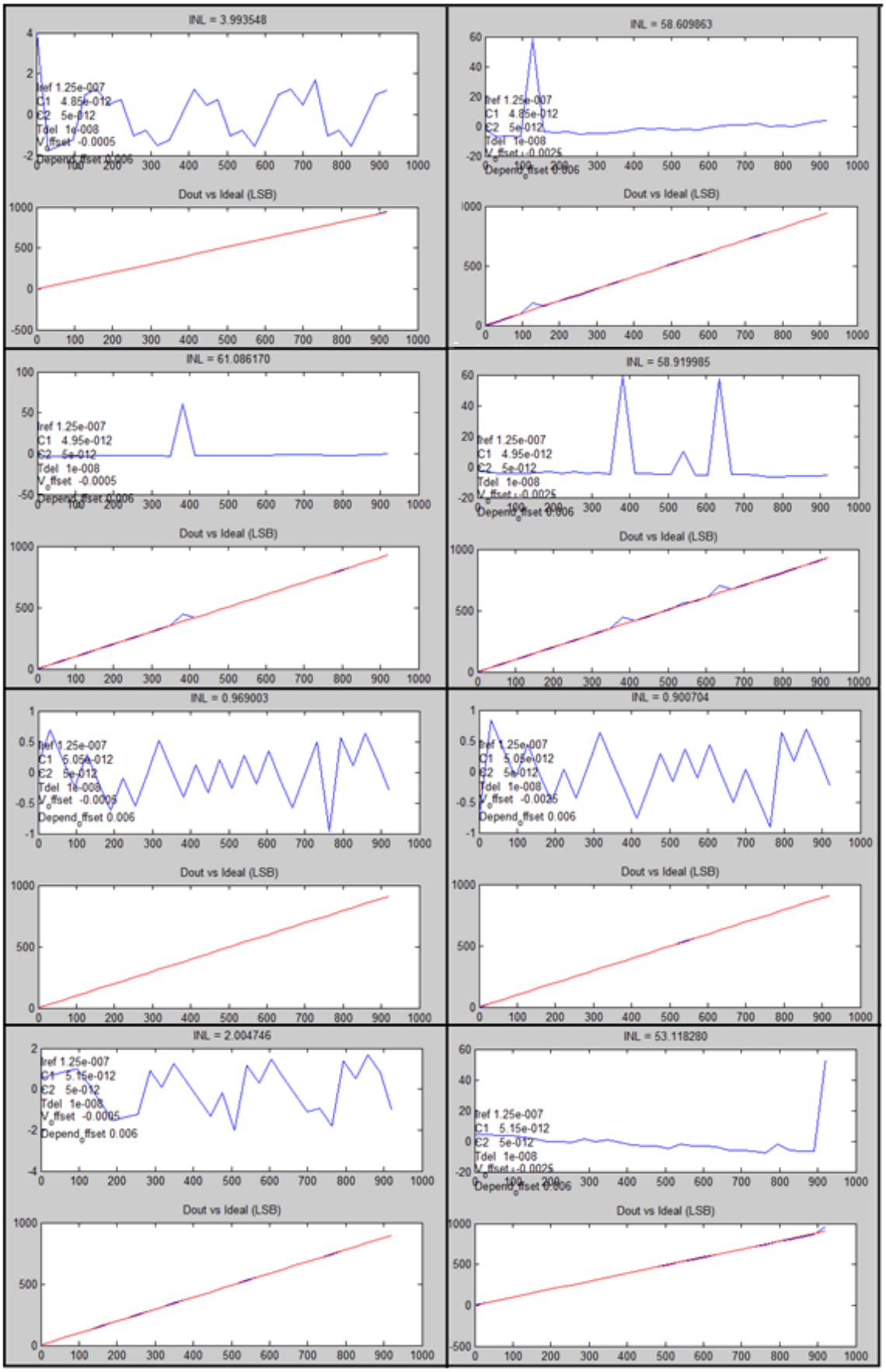

Figura 117. Resultado INL. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 115, 116 y 117 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 10ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente.

En la *Figura 115* se muestra el resultado para un offset dependiente del 0.2%, donde se puede observar que solo se dan resultados satisfactorios para el rendimiento del convertidor para un offset independiente de señal de 0.0005V para unos valores de capacidad de 5.05pF y 5.15pF para los cuales se obtiene un error INL de 0.969003 LSBs y 2.004746 LSBs respectivamente, junto con un resultado satisfactorio para un offset independiente de 0.0025V para un valor de capacidad de 5.05pF, obteniendo un error INL de 0.969967. Los demás valores de error INL obtenidos en los resultados para un offset dependiente de 0.0025V están en torno a los 60 LSBs, los cuales, como se ha dicho en otras ocasiones, son demasiado altos. En cuanto a los resultados obtenidos para un offset independiente de 0.0005V, además de los expuestos anteriormente, en concreto para una capacidad de 4.75pF se obtiene un error INL de 3.993548 LSBs, el cual sigue siendo un valor demasiado alto que afecta negativamente al rendimiento del convertidor, y para una capacidad de 4.95pF se obtiene un valor de error INL de 61.086170 LSBs, el cual es demasiado alto y afecta muy negativamente al rendimiento. Cabe destacar este último resultado, ya que por la tendencia de los resultados anteriores debería ser a un resultado satisfactorio, pero este no es el caso, por lo cual se podría pensar que el efecto del retraso junto con la disparidad de capacidad con este valor se debe descompensar el error de manera que afecta muy negativamente.

En la *Figura 116* se muestra el resultado para un offset dependiente del 0.4%. Los resultados obtenidos para este offset dependiente son prácticamente los mismos que los obtenidos en el caso anterior. De hecho, para todos los parámetros resultan ser los mismos, excepto para algunos valores de capacidad y un offset independiente de 0.0025V, los cuales siguen estando en torno a los 60 LSBs ya que el cambio en el error INL es muy poco significativo. Por lo explicado, se pueden obtener las mismas conclusiones que se obtenían en el caso inmediatamente anterior.

En la Figura 116 se muestra el resultado para un offset dependiente del 0.6%. Los resultados obtenidos para este offset dependiente son prácticamente los mismos que los obtenidos en los casos anteriores. De hecho, para todos los parámetros resultan ser los mismos en el caso de offset independiente de 0.0005V; y para el caso de offset dependiente de 0.0025V solo cambia para un valor de capacidad de 5.05pF, para el cual se obtiene un error INL de 0.900704 LSBs, el cual sigue siendo satisfactorio como resultaba en el caso anterior. Por lo explicado, se pueden obtener las mismas conclusiones que se obtenían en los casos inmediatamente anteriores.

Con las tres figuras anteriores se han expuesto los resultados obtenidos para analizar el error INL variando todos los parámetros fijando el retraso del comparador en 10ns. Para todos los casos se han obtenido resultados satisfactorios para un valor de offset independiente de 0.0005V y una capacidad de entre 5.05pF y 5.15pF, y para un valor de offset independiente de 0.0025V y una capacidad de 5.05pF, resultando la influencia del offset dependiente poco significativa, como ocurría cuando se analizaba este offset por separado.

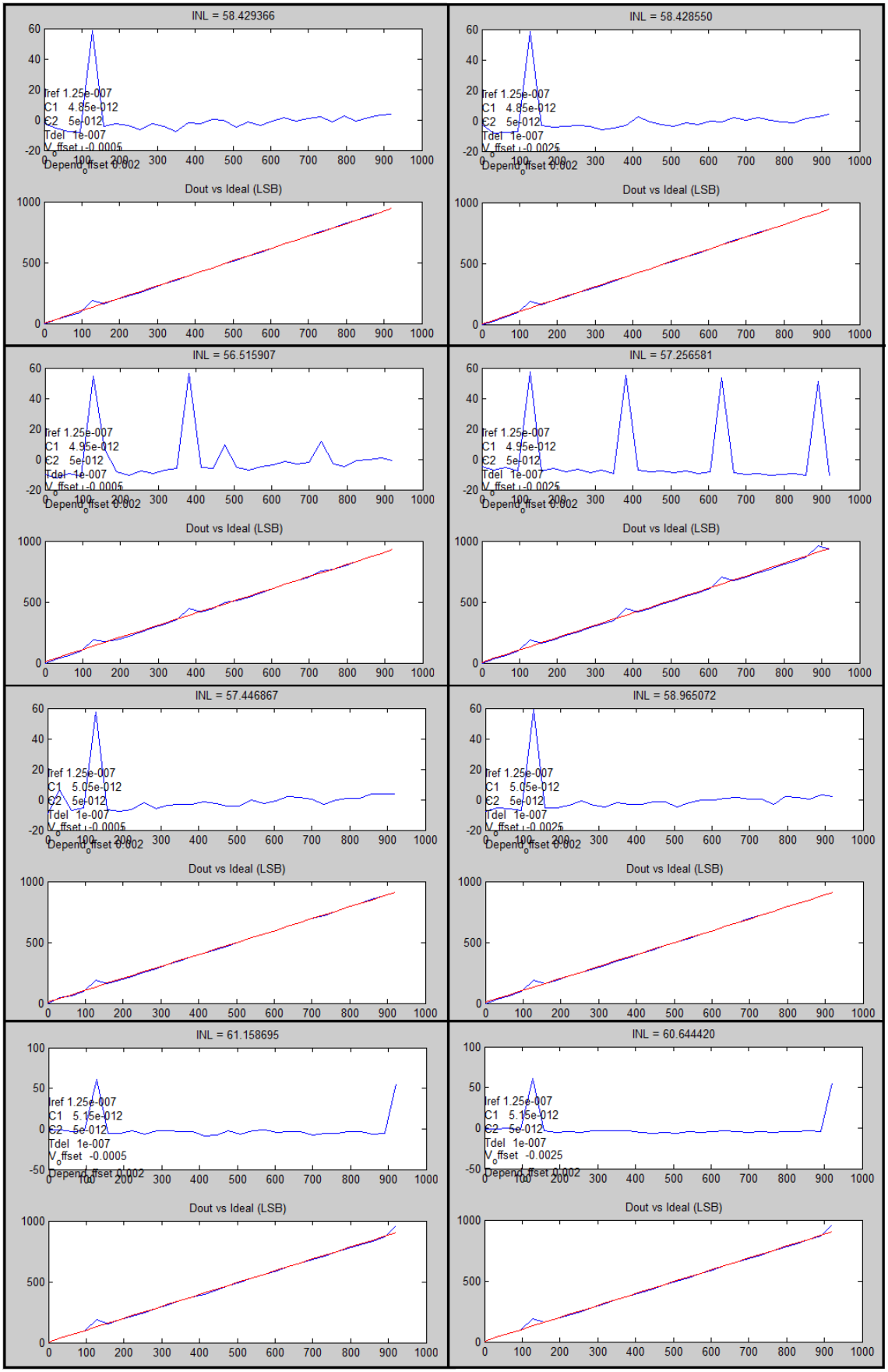

Figura 118. Resultado INL. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

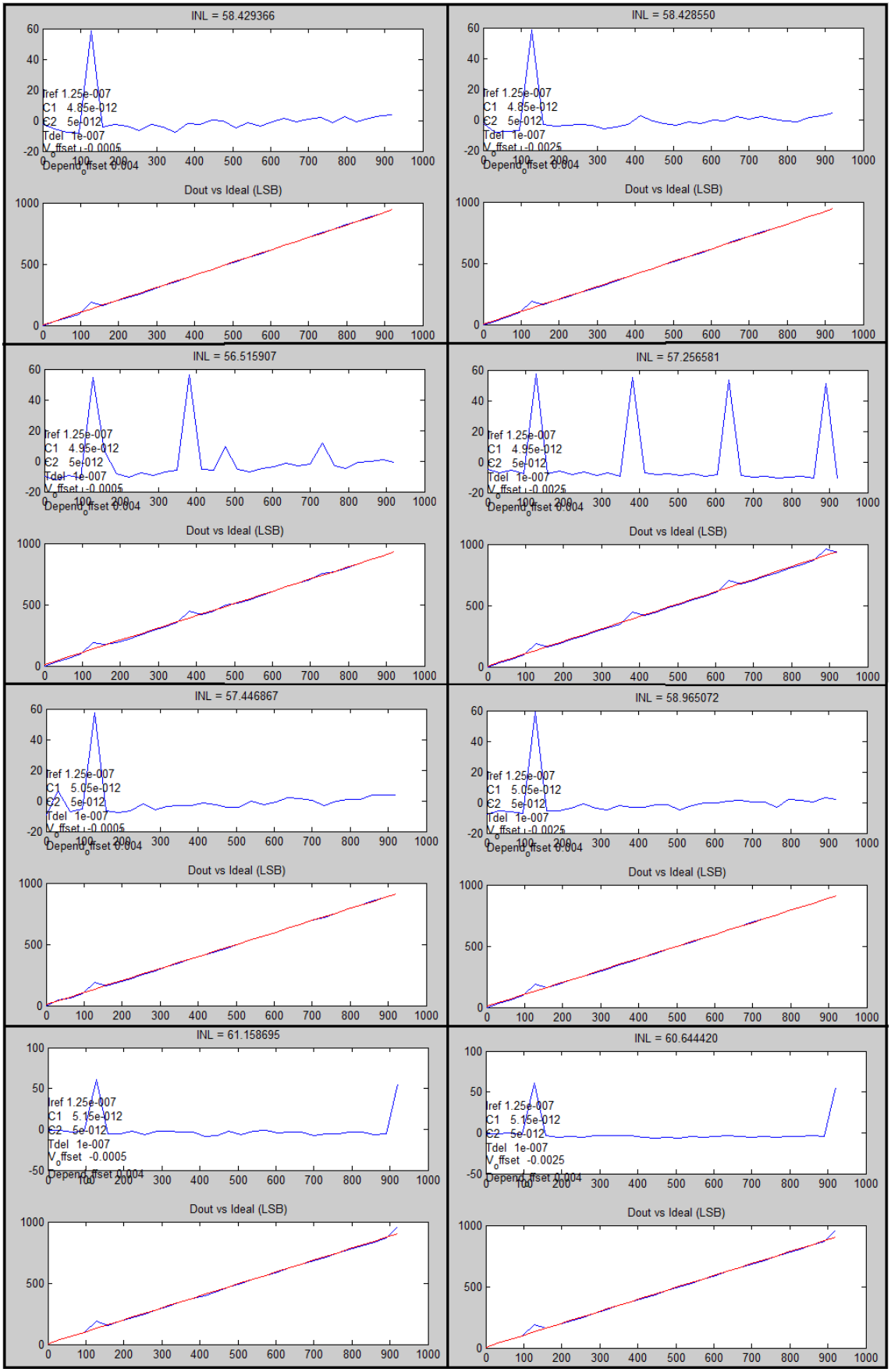

Figura 119. Resultado INL. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

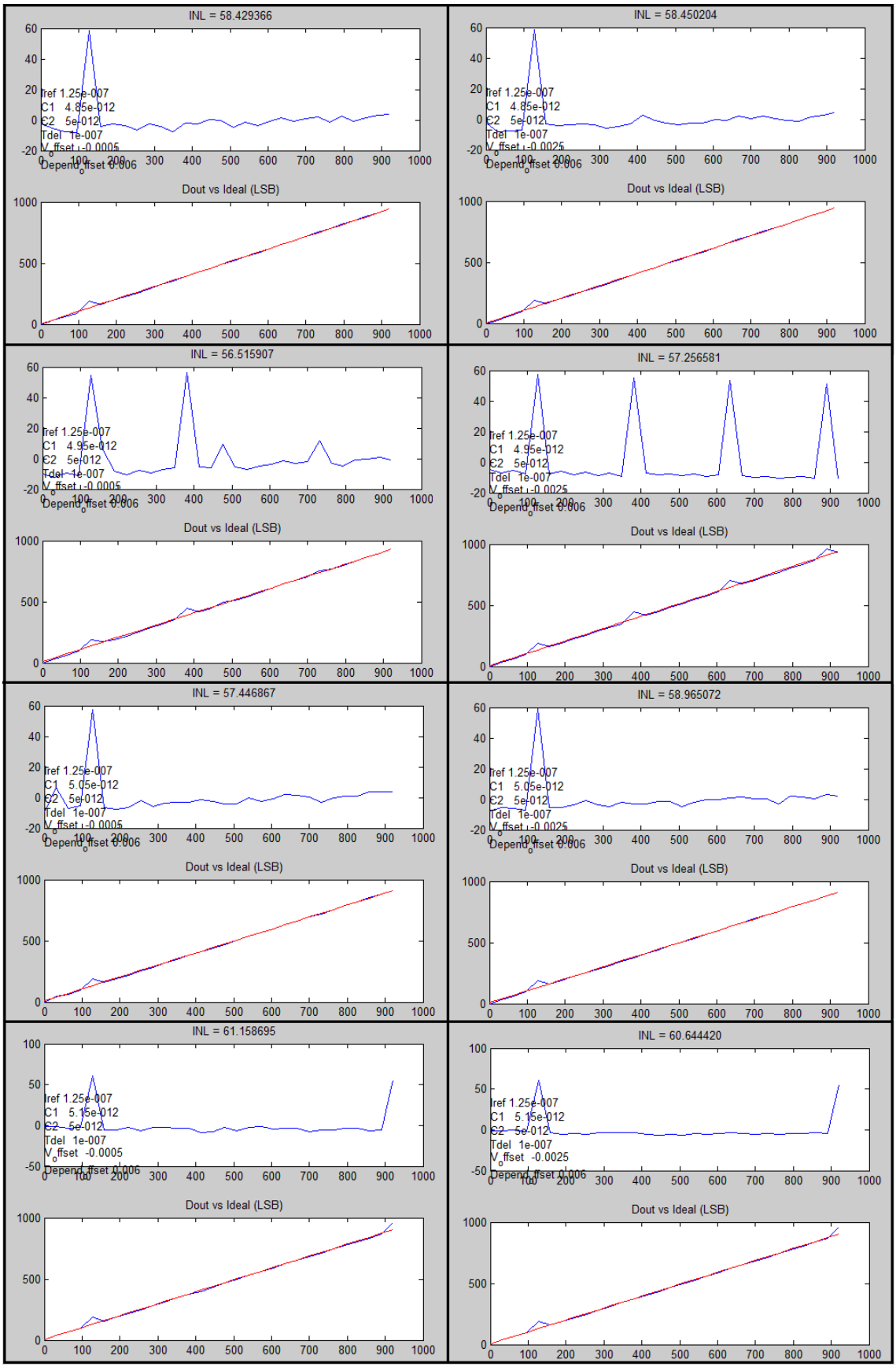

Figura 120. Resultado INL. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 118, 119 y 120 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 100ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente.

En las tres figuras se obtienen los mismos resultados ya que el offset dependiente de señal no influye en ninguno de los casos, como ha venido pasando a lo largo del análisis. Como era de esperar con un retraso en el comparador tan elevado los resultados obtenidos ofrecen un error INL demasiado alto, concretamente para todos los parámetros se ha obtenido un error en torno a los 60 LSBs, por lo que se da una influencia muy negativa en el rendimiento del comparador.

Con el estudio realizado para cada uno de los parámetros del modelo de Matlab/Simulink queda concluido el análisis del error INL para mostrar el rendimiento del convertidor. A modo de resumen, en el ANEXO IV se mostrará una tabla con los resultados obtenidos tras realizar todas las simulaciones, tanto los resultados mostrados a lo largo del análisis, como los que no se han mostrado por no resultar de relevancia atendiendo a resultados anteriores.

5.2.2.2 Cálculo de la Resolución Efectiva (enob)

Al igual que se hacía para calcular el error INL, para calcular la resolución efectiva o ENOB se ha hecho uso del modelo implementado en Matlab/Simulink junto con una serie de scripts realizados en matlab.

En primer lugar se tiene un script en el que se ha elaborado una función general para obtener la ENOB en el modelo simulink del ADC, la cual será llamada por otras funciones donde se variarán los distintos parámetros del convertidor. Mediante el uso de esta función se calcula la ENOB propiamente dicha, para lo cual hay que proporcionarle como entrada a esta función el valor de la corriente de referencia (I_{ref}), el valor de las capacidades (C_1 y C_2), el valor del retraso del comparador (*Tdelay*), el valor del offset independiente de señal del comparador (*V_offset*) y el valor del offset dependiente del comparador (*Depend_offset*) y finalmente el número de muestras (*Nitems*). Estas variables de entrada, excepto el número de muestras, son globales, ya que son utilizadas por otras funciones que llamaran a esta y es necesario que se encuentren en el WorkArea de Matlab. En esta función se genera un vector de corrientes de entrada con una longitud igual al número de muestras que en este caso es 64. Este vector genera un coseno con una frecuencia menor que 1/2 la frecuencia de muestreo (Teorema de Nyquist), que en este caso viene dada por la arquitectura, ya que como se ha dicho a lo largo de la descripción del modelo el muestreo se debe hacer cada veintidós ciclos de reloj, es decir, una frecuencia de $1/(220\text{x}10^{-6})$Hz y una amplitud pico-pico del 90% del fondo de escala, es decir, el 90% de cuatro veces la corriente de referencia (500nA). Mediante la función *sim()* de Matlab se ejecuta el modelo implementado en Simulink para cada valor del vector de corriente de entrada del vector, obteniéndose un dato de salida digital para cada valor de corriente. El nombre de la función que se genera en un fichero con el mismo nombre es "*test_bench_Par_ENOB",*la cual genera como salida tanto la ENOB, como la SNDR de la cual se obtiene el anterior. Para calcular la ENOB (SNDR) propiamente dicha, en esta función general, se llama a otra función denominada "enob2" donde se utiliza la función *pwelch()* proporcionada por Matlab. Esta función además de generar como salida la ENOB (SNDR) genera una gráfica en la que se puede ver la densidad espectral de potencia para cada valor de frecuencia. Por último, en el título de la gráfica se podrá ver la ENOB y además superpuestos sobre las gráficas se podrán ver los valores de los distintos parámetros.

En segundo lugar se han implementado varios scripts para introducir los distintos parámetros variándolos dentro de unos valores adecuados acorde a las oscilaciones que se pueden dar en estos. Se han implementado los siguientes scripts:

- *test_Bench_Par_ENOB_Tdelay*. Función para obtener la ENOB haciendo un análisis paramétrico del retraso del comparador del modelo del ADC de Matlab/Simulink (*Tdelay*), introduciendo un retraso de 0ns, 2ns, 10ns, 100ns y 1000ns. Llama a la función implementada en el fichero "*test_bench_Par_ENOB"*.

- *test_bench_Par_ENOB_Cmistmath*. Función para obtener la ENOB haciendo un análisis paramétrico del mistmach de capacidades del modelo del ADC de Matlab/Simulink variando unas de las capacidades de 4.75pF hasta 5.25pF. Llama a la función implementada en el fichero "*test_bench_Par_ENOB"*.
- *test_bench_Par_ENOB_Tdelay_V_offset_Depend_offset*. Función para obtener la ENOB haciendo un análisis paramétrico del retraso del comparador con los valores anteriores, offset del comparador con valores de 0.0005V, 0.0025V, 0.0050V, 0.0075V y 0.0100V junto con el offset dependiente de la señal en el comparador con valores de 0.2%, 0.4% y 0.6% en el modelo del ADC de Matlab/Simulink. Llama a la función implementada en el fichero "*test_bench_Par_ENOB"*.
- *test_bench_Par_ENOB_T_Vo_Do_mstch.* Función para obtener la INL haciendo un análisis paramétrico del retraso del comparador, el offset del comparador, del offset dependiente y del mistmach capacidades de la señal en el comparador, todos ellos con los valores anteriores, en el modelo del ADC de Matlab/Simulink. Llama a la función implementada en el fichero "*test_bench_Par_ENOB"*.

Todos estos scripts se podrán ver en el anexo III debidamente comentados para facilitar su comprensión.

5.2.2.2.1 Sistema Ideal

Como cuando se analizaba el error INL, en este apartado se trata un sistema ideal, es decir, con un comparador sin retraso, sin offset tanto dependiente como independiente de señal y con ambas capacidades fijadas a un valor de 5 pF para con esto obtener el máximo rendimiento del convertidor. Para estos valores el resultado obtenido tras la simulación se puede observar en la *Figura 121*.

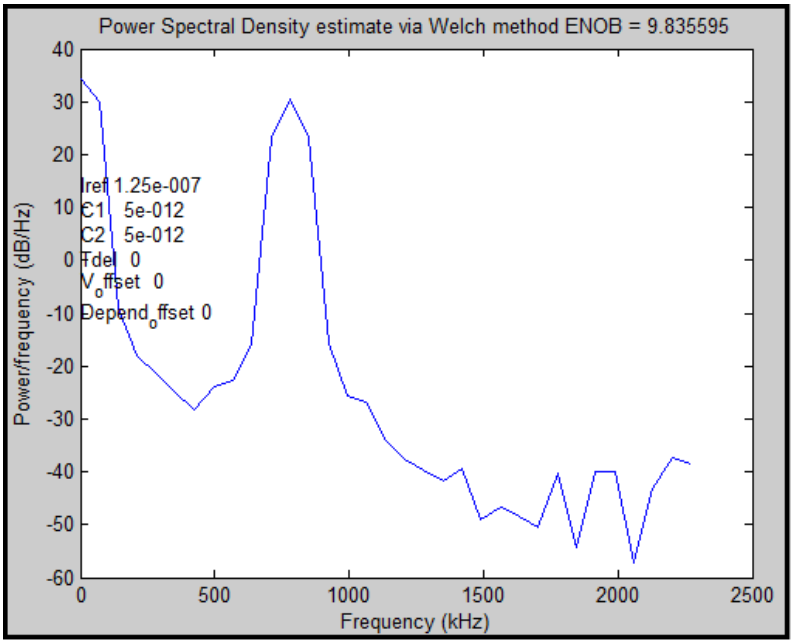

Figura 121. Resultado ENOB. Sistema ideal

Analizando el resultado, como era de esperar, se obtiene una resolución efectiva muy cercana a los 10 bits, ya que el ADC se ha diseñado con este número de bits. En concreto la ENOB obtenida sin la variación de ningún parámetro del convertidor es de 9.835595 bits, lo cual nos indica que la arquitectura diseñada en el modelo de Matlab/Simulink podría ser viable.

5.2.2.2.2 Influencia del Retraso del Comparador

A continuación se va analizar la influencia del retraso del comparador para calcular la resolución efectiva viendo así su efecto en el rendimiento del ADC. Para ello se va a incluir en el modelo un retraso en el comparador comenzando en 2ns y se irá aumentando el orden de magnitud para ver cómo influye el retraso en cada caso.

En la *Figura 122* se pueden observar los resultados obtenidos tras la simulación para varios valores de retraso del comparador.

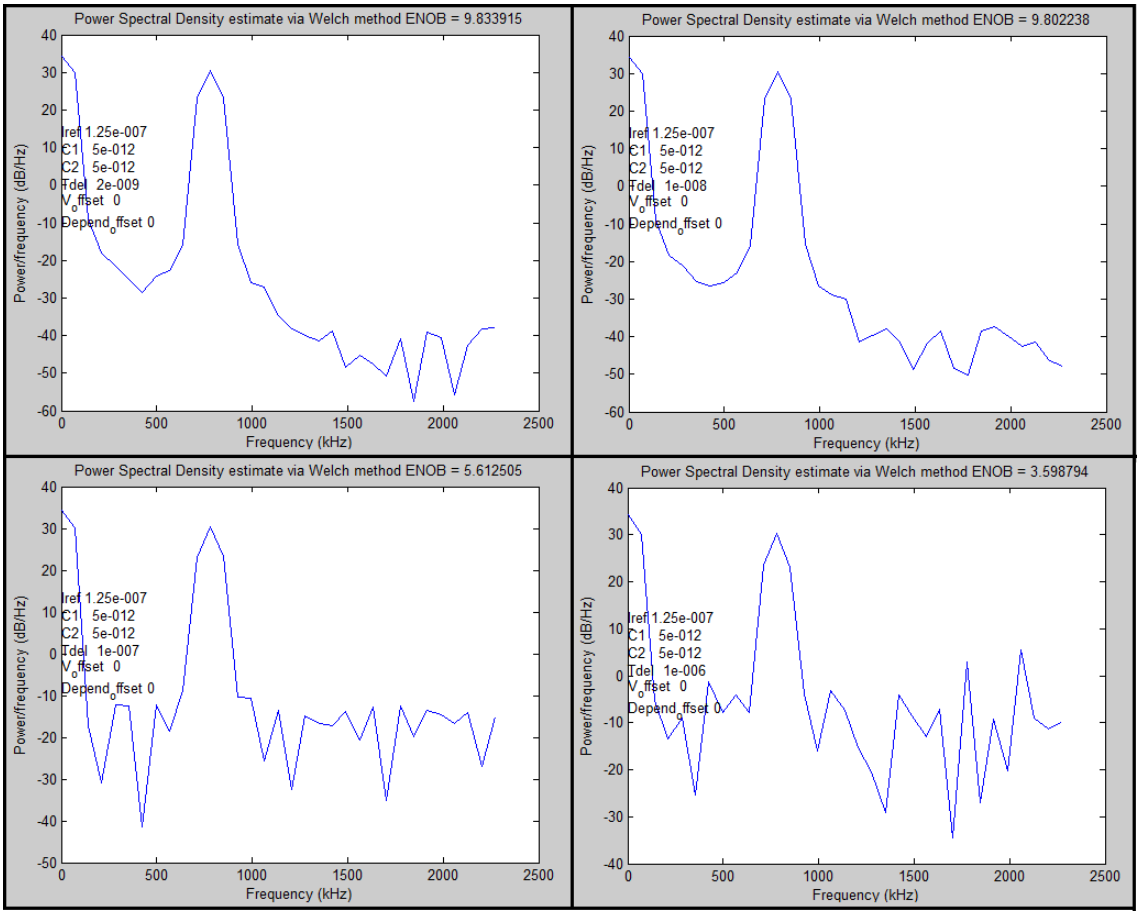

Figura 122. Resultado ENOB. Influencia del retraso del comparador

En la figura se puede observar como para valores de retraso de 2ns y 10ns se se siguen obteniendo resultados muy satisfactorios para el rendimiento del convertidor, cercanos a los 10 bits. Para un retraso en el comparador de 2ns se obtiene una ENOB de 9.833915 bits y para un retraso de 10ns se obtiene una ENOB de 9.802238 bits. Si se sube el orden de magnitud del retraso del comparador llegando a un retraso de 100ns y 1000ns se obtiene una resolución efectiva muy negativa para el rendimiento del convertidor 5.612505 bits y 3.598794 bits respectivamente. Se considerará de aquí en adelante que un resultado aceptable para un buen rendimiento en el convertidor por encima de 8 bits. Ante estos resultados se concluye que en ningún caso se debe llegar a un retraso en el comparador de 100ns para conseguir una resolución aceptable en el ADC.

5.2.2.2.3 Influencia de la disparidad de Capacidades

En este apartado se va analizar la influencia de la disparidad de capacidades (mistmach) en la ENOB para el rendimiento del ADC. Para ello se variará una de las capacidades desde 4.75pF hasta 5.25pF en pasos de 10pF dejando fija la otra capacidad en 5pF y sin introducir retraso alguno.

En la *Figura 123* se pueden observar los resultados obtenidos tras la simulación para los distintos valores de capacidades.

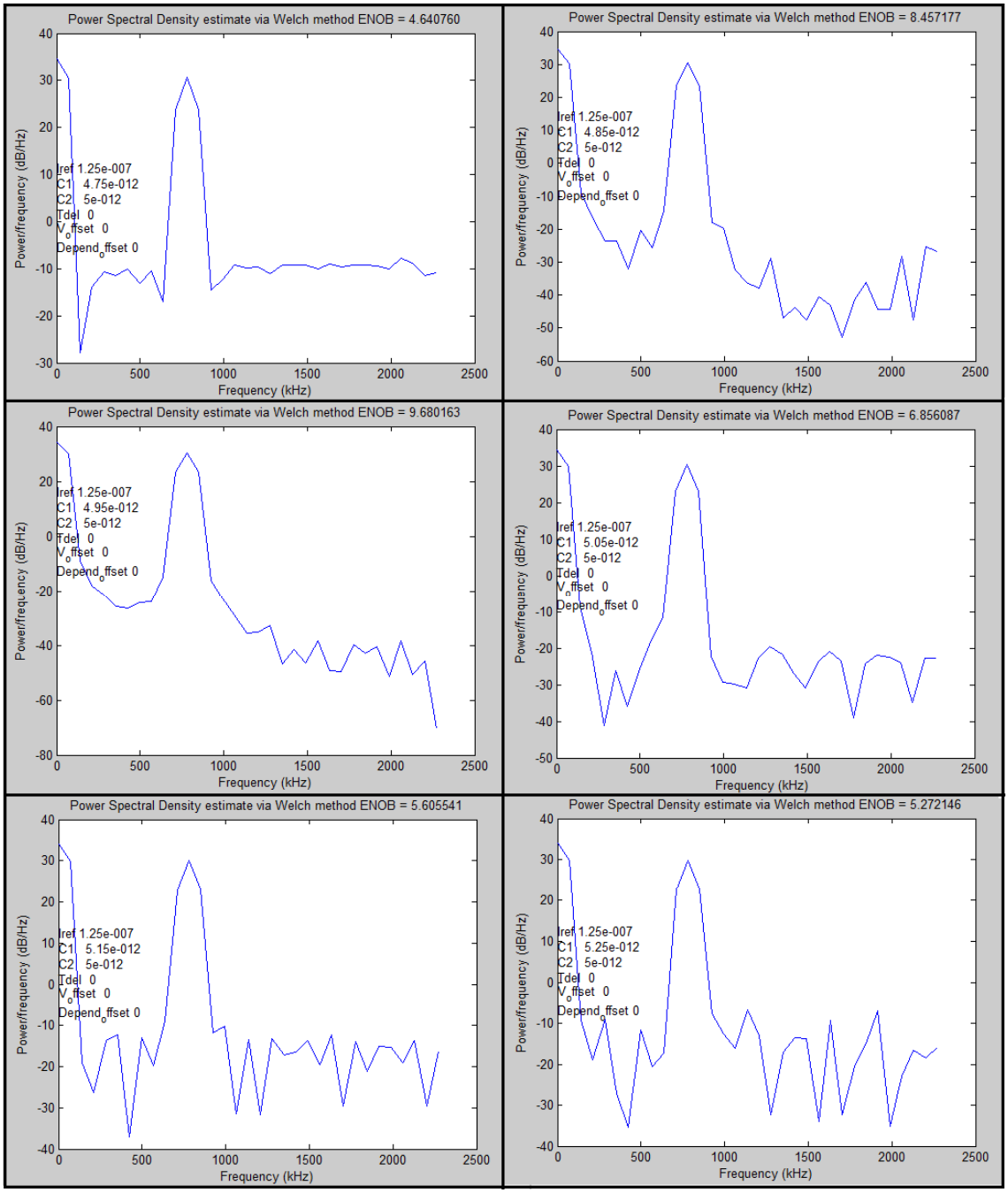

Figura 123. Resultado ENOB. Mistmach de capacidades

Si se observan los resultados de las distintas gráficas que se pueden ver en la figura se observa que se obtienen resultados aceptables para la ENOB del ADC con los parámetros correspondientes a la segunda y tercera gráfica, donde se obtiene una ENOB de 8.457177 bits y 9.680163 bits respectivamente, los cuales se corresponden con un valor de capacidades de 4.85pF y4.95pF. Para los demás valores de capacidades el resultado de la ENOB obtenido no llega a 8 bits, quedando bastante alejado de este, lo cual hace pensar que para un buen resultado el desapareamiento de capacidades debe oscilar entre 4.85pF y 5pF. Los resultados de ENOB obtenidos para los demás valores de capacidades han sido 4.640760 bits, de 6.856084 bits, 5.605541 bits y 5.272146 bits para unas capacidades de 4.75pF, 5.05pF, 5.15pF y 5.25pF respectivamente.

5.2.2.2.4 Influencia del offset independiente de señal en el comparador

Para completar el análisis paramétrico se debe tener en cuenta el offset en el comparador integrado en la arquitectura, tanto el offset independiente de la señal de entrada, como el offset dependiente. En el presente apartado se tratará el primero de ellos. Para analizarlo se han impuesto una serie de valores para los parámetros los valores que se suelen dar habitualmente. Estos valores son 0.0005V y 0.0025V.

En la *Figura 124* se pueden observar los resultados más representativos obtenidos tras la simulación con los distintos valores de capacidades.

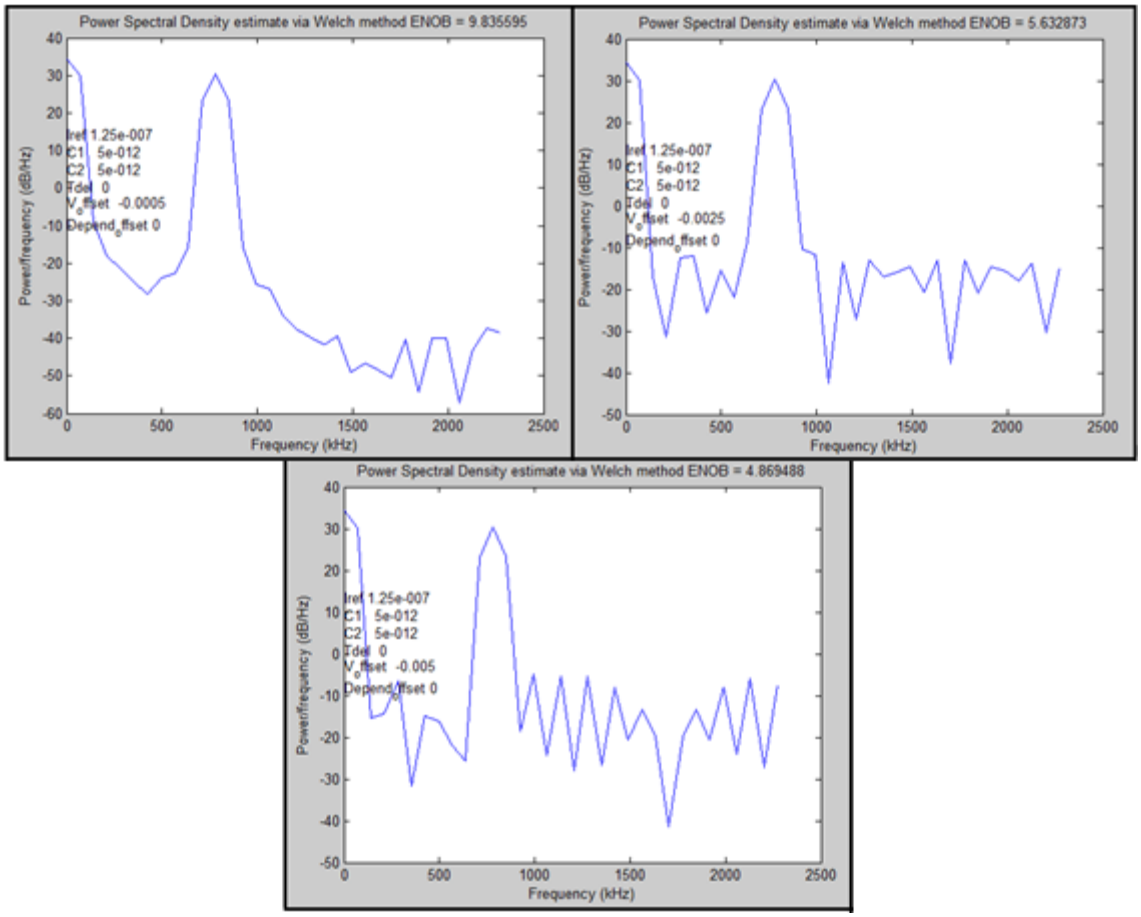

Figura 124. Resultado ENOB. Offset independiente de señal en el comparador

Si se observan las distintas gráficas de la figura, en la primera se puede ver que para un valor de offset de 0.0005V se obtiene una ENOB de 9.835595 bits, lo cual resulta un buen rendimiento en el convertidor ya que se encuentra por encima de los 8 bits. Si se aumenta el offset a 0.00025V, cuyos resultados se corresponden con la segunda gráfica, se obtiene un resultado de 5.632873 bits, lo cual se puede entender como un decaimiento del rendimiento en el convertidor suficientemente importante para no ser aceptable ya que está por debajo de 8 bits. Como es de esperar para valores mayores de offset la ENOB sigue empeorando, como se puede ver en la tercera gráfica donde se obtiene una resolución de 4.869488 bits para un offset de 0.0050V. Tras analizar los resultados de la gráfica se puede concluir que el offset en el comparador debe ser lo suficientemente pequeño, no muy lejano a los 0.0005V.

5.2.2.2.5 Influencia del offset dependiente de señal en el comparador

Para completar el análisis del offset del comparador, al efecto que tiene el offset independiente de señal del comparador se va a sumar el efecto de offset dependiente de señal de este. Se ha hecho este análisis de forma conjunta porque el efecto se da de esta manera, siendo el offset independiente, analizado anteriormente por separado, el que tiene un efecto más significativo en el rendimiento del ADC. Para añadir el efecto del offset dependiente, este parámetro añade un tanto por ciento de señal a la propia señal de entrada. Se han incluido unos valores para este parámetro de un 0.2%, 0.4% y 0.6% para cada valor de offset independiente.

En las siguientes figuras se verán los resultados más representativos obtenidos tras la simulación para distintos valores de offset dependiente e independiente conjuntamente.

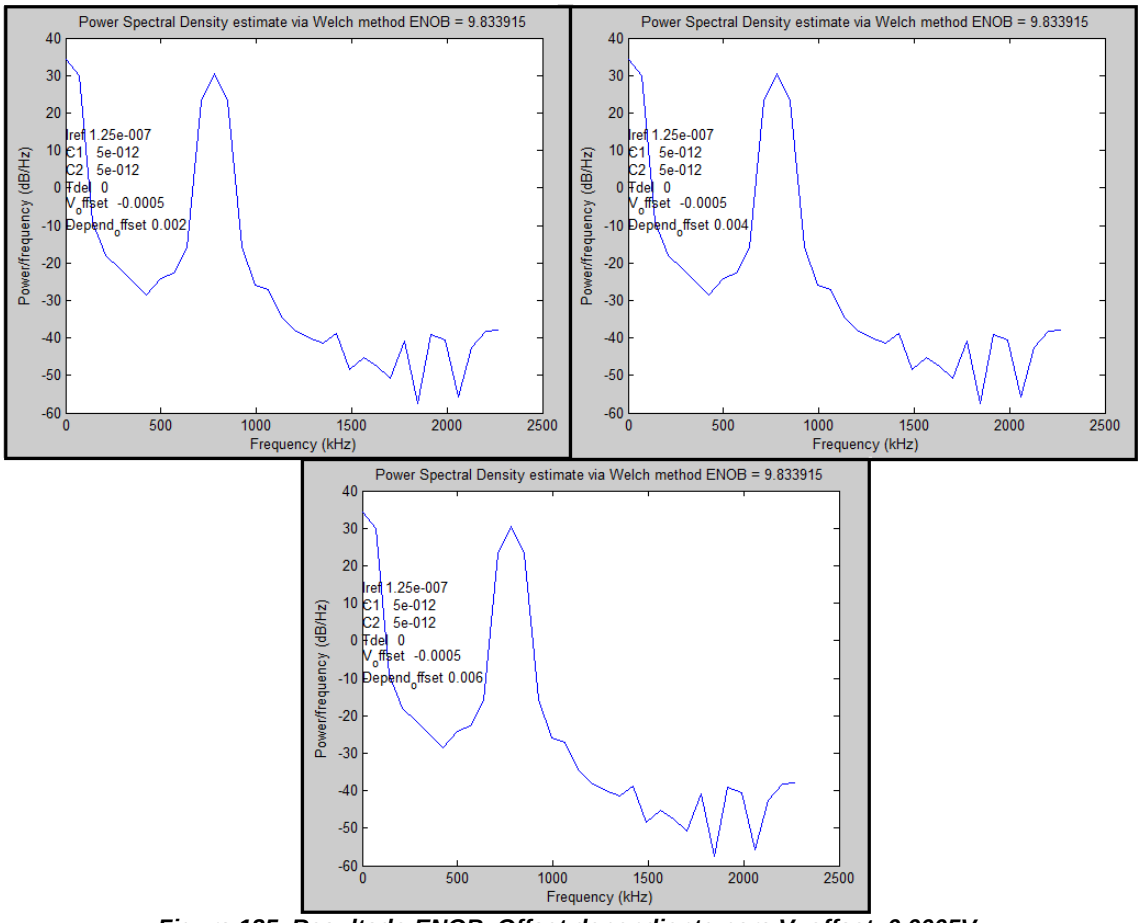

Figura 125. Resultado ENOB. Offset dependiente para V_offset=0.0005V

En la *Figura 125* se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0005V. Si se presta atención a los resultados obtenidos en el apartado anterior, el resultado es similar que se obtiene sin incluir offset dependiente para los tres valores de offset: 9.833915 bits. Tras observar estos resultados se puede concluir que el offset dependiente de señal no influye significativamente en el rendimiento del convertidor para un valor de offset independiente de 0.0005V.

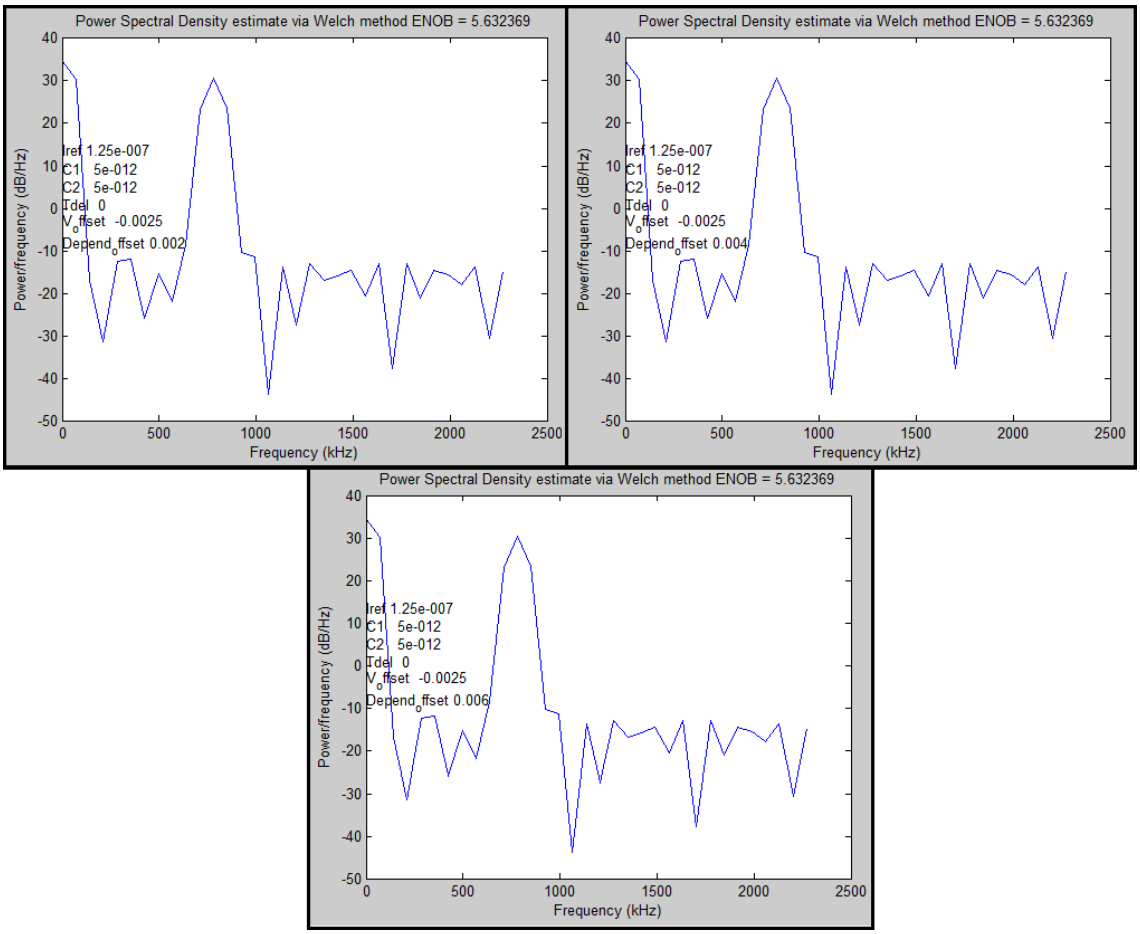

Figura 126. Resultado ENOB. Offset dependiente para V_offset=0.0025V

En la **¡Error! No se encuentra el origen de la referencia.** se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0025V. Si se presta atención a los resultados obtenidos en el apartado anterior, sigue siendo similar al que se obtiene sin incluir offset: 5.632369 bits. Se puede concluir que el offset dependiente de señal no influye significativamente en el rendimiento del convertidor para un valor de offset independiente de 0.0025V.

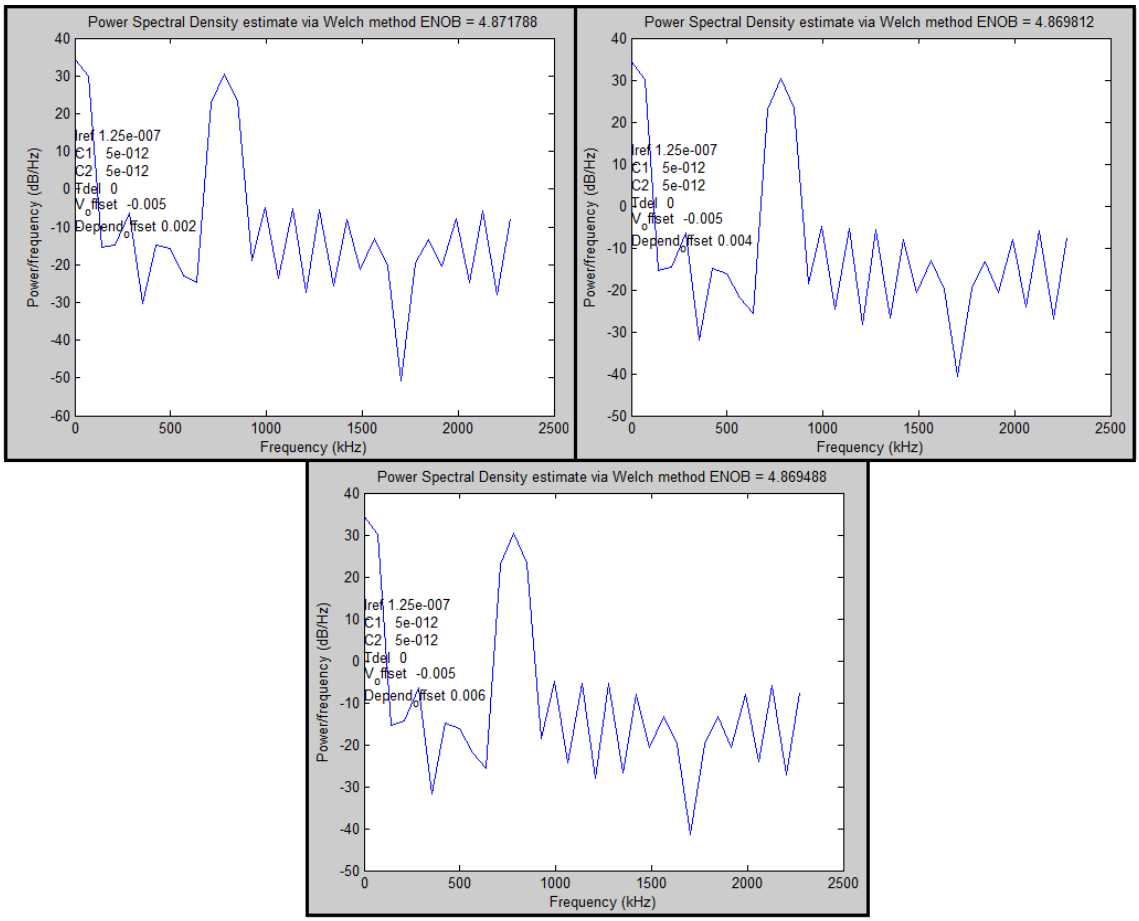

Figura 127. Resultado ENOB. Offset dependiente para V_offset=0.0050V

En la *Figura 127* se pueden observar los resultados obtenidos para los distintos parámetros de offset dependiente fijando el valor de offset independiente de 0.0050V. Prestando atención a los resultados obtenidos en el apartado anterior, sigue siendo similar al que se obtiene sin incluir offset: 4.859488 bits. Se puede concluir que el offset dependiente de señal no influye significativamente en el rendimiento del convertidor para un valor de offset independiente de 0.0025V.

5.2.2.2.6 Influencia de todo el conjunto de parámetros no Ideales

Hasta ahora se ha estudiado la influencia en el resultado de la resolución efectiva por separado, sin tener en cuenta la influencia de los demás. En este apartado se va a realizar un estudio de todos los parámetros no ideales conjuntamente, es decir, se estudiarán la influencia de los distintos parámetros actuando al mismo tiempo. Para este estudio se van a mostrar los resultados más representativos obtenidos tras la simulación.

En primer lugar se va a estudiar la influencia del retraso del comparador (*Tdelay*), junto con el mistmach de capacidades. En las siguientes figuras se pueden observar los resultados con los distintos valores de capacidades y de retraso del comparador, dando al retraso del comparador valores de 2ns, 10ns y 100ns por un lado, y variando una de las capacidades desde 4.75pF hasta 5.25pF en pasos de 10pF por otro.

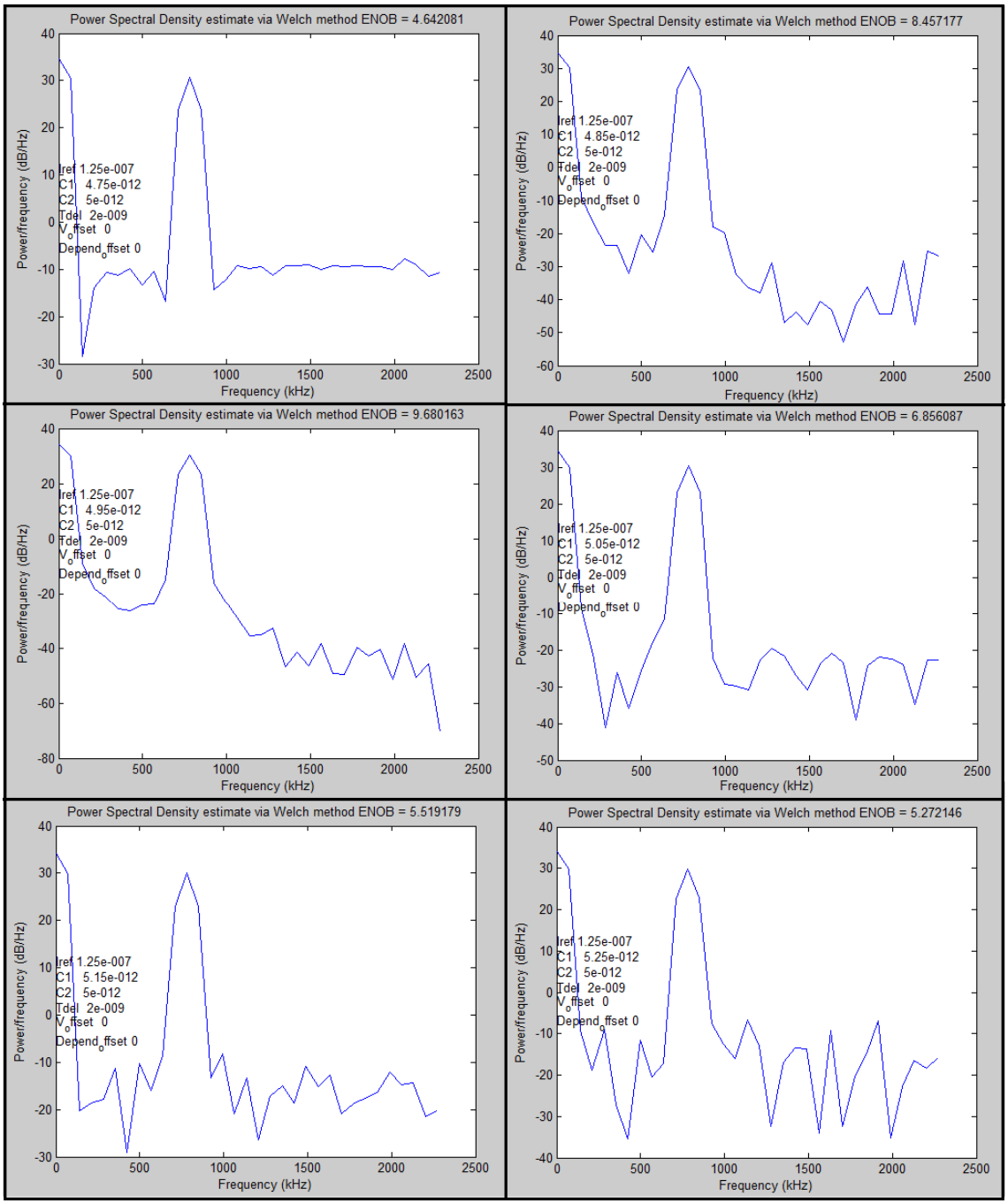

Figura 128. Resultado ENOB. Tdelay=2ns. Mistmach de capacidades

En la *Figura 128* se observan los resultados obtenidos tras incluir el efecto del retraso de 2ns en el comparador junto con el mistmach de capacidades. Si se recuerdan los resultados en los que se estudia el mistmach de capacidades expuestos anteriormente en el apartado 5.2.2.2.3, solo difiere para dos de los valores de capacidad, siendo la diferencia my poco significativa. Ante este hecho, se puede concluir en la influencia de la ENOB predomina el efecto del mistmach de capacidades para este valor de retraso, influyendo este último de manera poco significativa. Los valores de ENOB en los que se difiere a la obtenida analizando el mistmach de capacidades son 4.640760 bits y 5.510179 bits para unos valores de capacidad de 4.75pF y 5.15pF respectivamente, frente a los 4.640760 bits y 5.605541bits obtenidos anteriormente. Ante lo explicado, solo se obtienen resultados positivos para el rendimiento del ADC para unos valores de capacidad de 4.85pF y 4.95pF con los mismos valores de ENOB que resultaban en el apartado 5.2.2.2.3.

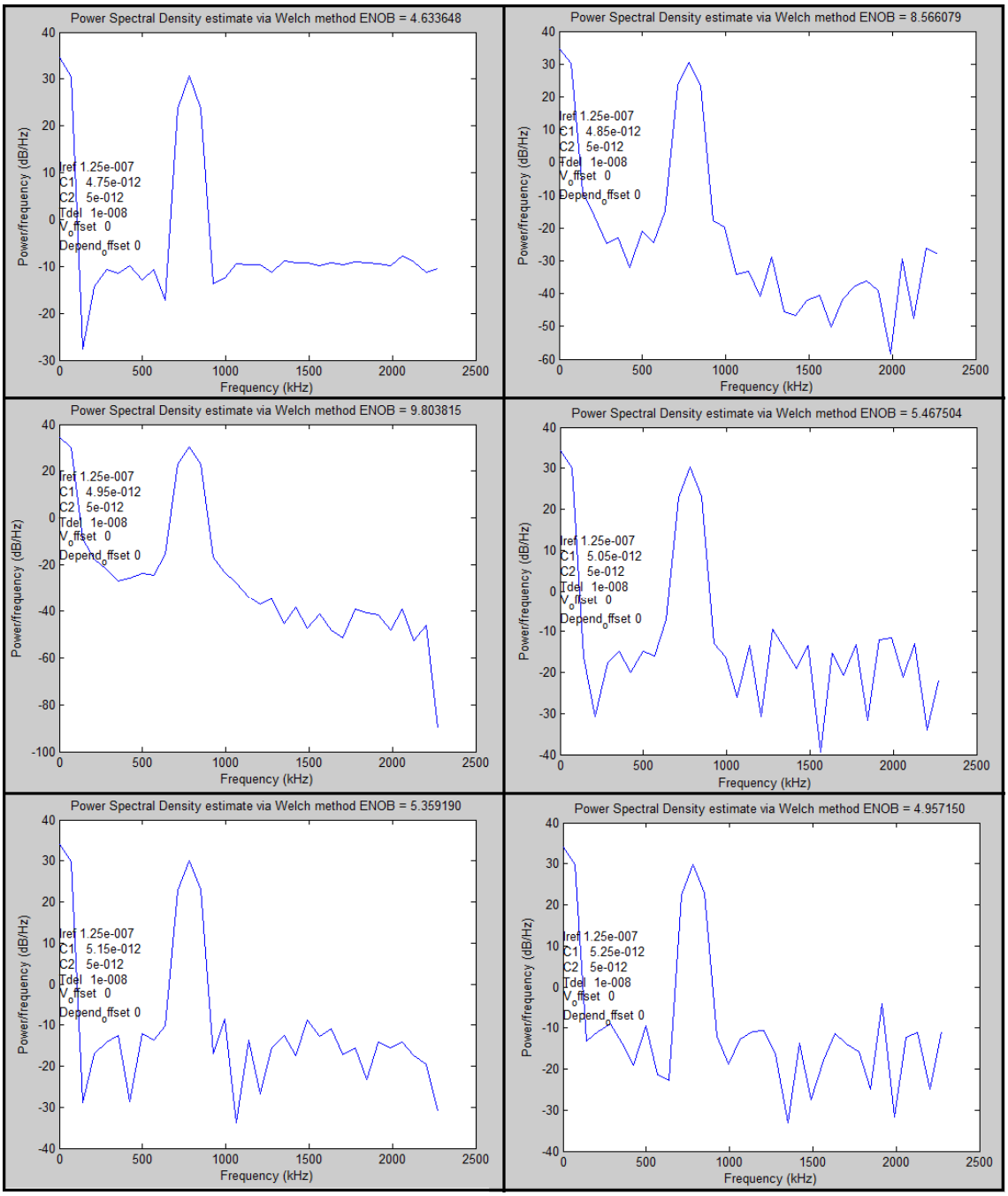

Figura 129. Resultado ENOB. Tdelay=10ns. Mistmach de capacidades

En la *Figura 129* se observan los resultados obtenidos tras incluir el efecto del retraso de 10ns en el comparador junto con el mistmach de capacidades. Como pasaba en el caso anterior, los resultados son muy similares a los obtenidos cuando se estudiaba únicamente el mistmach de capacidades expuestos anteriormente en el apartado 5.2.2.2.3.En este caso, a diferencia del anterior ningún resultado coincide, pero todos son muy cercanos a los obtenidos en el apartado mencionado. Como se comentaba, al analizar el mistmach de capacidades únicamente se obtienen dos resultados en los que la ENOB resulta satisfactoria en los que se obtienen 8.566079 bits y 9.803815 bits para unos valores de capacidad de 4.85pF y 4.95pF respectivamente.

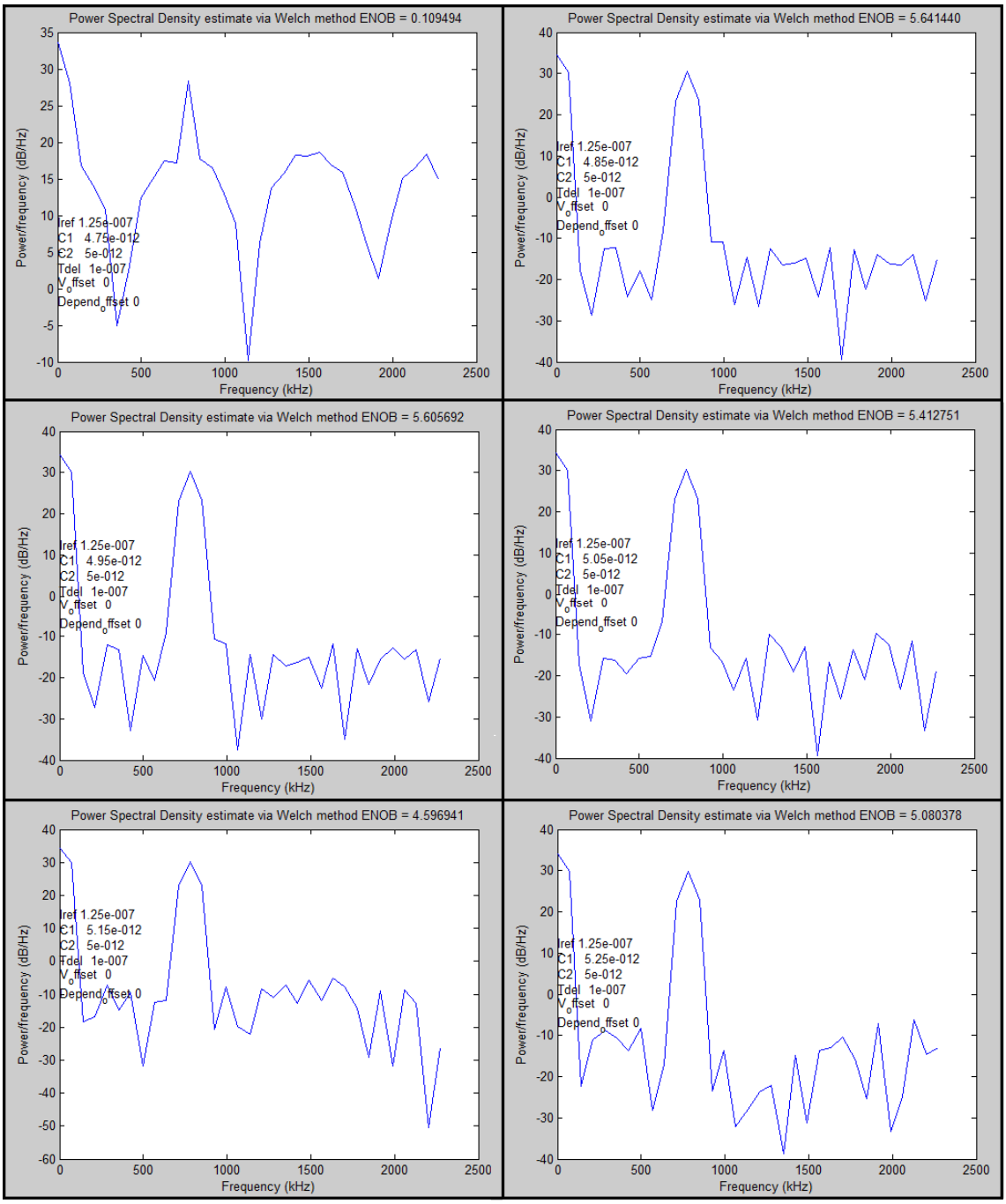

Figura 130. Resultado ENOB. Tdelay=100ns. Mistmach de capacidades

En la *Figura 130* se observan los resultados obtenidos tras incluir el efecto del retraso de 100ns en el comparador junto con el mistmach de capacidades para analizar el error. En este caso como era de esperar, no se obtiene ningún resultado en el que la ENOB sea aceptable. Esto es debido a que al efecto que tenía sobre la ENOB un retraso de 100ns visto en el apartado 5.2.2.2.2, se ha unido el efecto del mistmach de capacidades obteniendo resultados aún perores. Los resultados obtenidos para todos los valores de capacidades para la ENOB se encuentran en torno a los 5 bits, excepto para una capacidad de 4.75pF, estando éste último a 0 bits de resolución. Esto reafirma la conclusión de que en ningún caso se debe llegar a un retraso del comparador de 100ns.

Una vez estudiada la influencia del retraso del comparador, junto con el mistmach de capacidades, se va estudiar la influencia del retraso junto con la influencia del offset independiente de señal para el cálculo de la ENOB. En las siguientes figuras se pueden observar los resultados con los distintos valores de offset independiente y de retraso del

comparador, dando al retraso del comparador valores de 2ns, 10ns y 100ns por un lado, y para el offset independiente 0.0005V 0.0025V y 0.0050V para cada valor de retraso por otro.

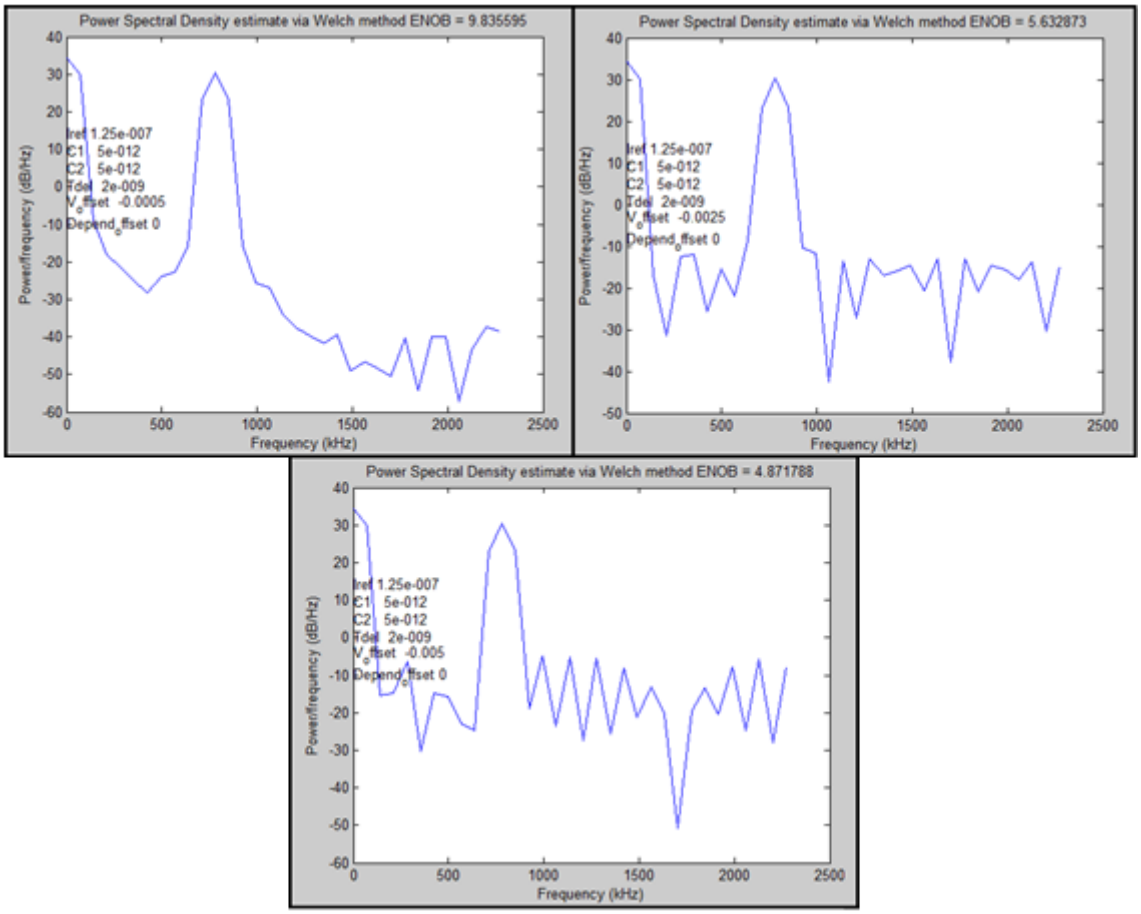

Figura 131. Resultado ENOB. Tdelay=2ns. V_offset

En la Figura 131, donde se presentan los resultados para un retraso de 2ns y distintos valores de offset independiente de señal del comparador, se observa que se obtienen prácticamente los mismos resultados cuando se analizaba el offset independiente sin incluir ningún tipo de retraso, en el apartado 5.2.2.2.4. Solo varían en algunas décimas la ENOB para un valor de offset de 0.0050V, donde la ENOB obtenida no resulta ser satisfactoria, por lo que la influencia de un retraso de 2ns no es importante. Por tanto, se puede concluir que un retraso tan pequeño no influye demasiado si se tiene en cuenta el efecto del offset independiente de señal del comparador, obteniendo resultados admisibles únicamente para un valor de offset de 0.0005V donde se obtiene una ENOB de 9.835595 bits.

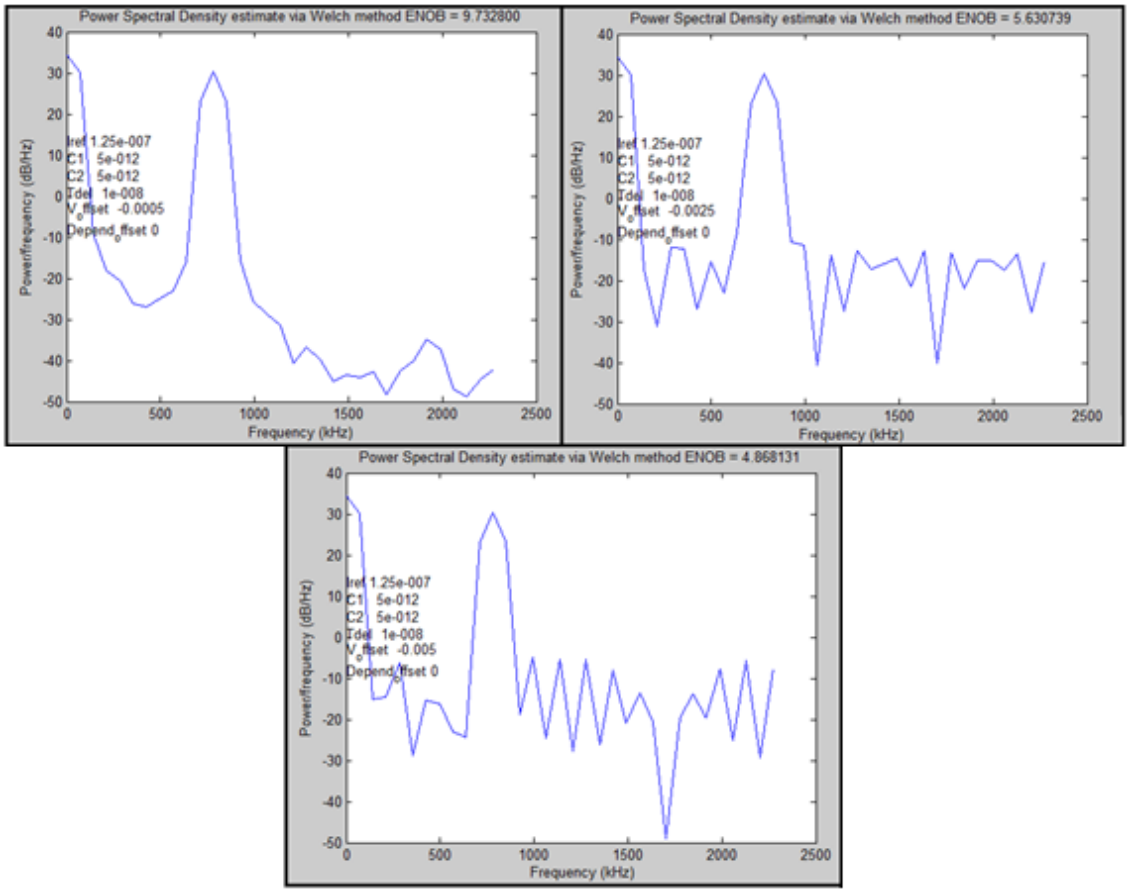

Figura 132. Resultado ENOB. Tdelay=10ns. V_offset

En la *Figura 132*, donde se presentan los resultados para un retraso de 10ns y distintos valores de offset independiente de señal del comparador. Los resultados siguen siendo similares a los que se obtenían en el apartado 5.2.2.2.4, variando en algunas décimas para cada valor de offset. Ante estos resultados, se deduce que la influencia de un retraso de 10ns no es demasiado importante. Por lo tanto, sólo se obtienen resultados admisibles para un valor de offset de 0.0005V donde se obtiene una ENOB de 9.732800 bits.

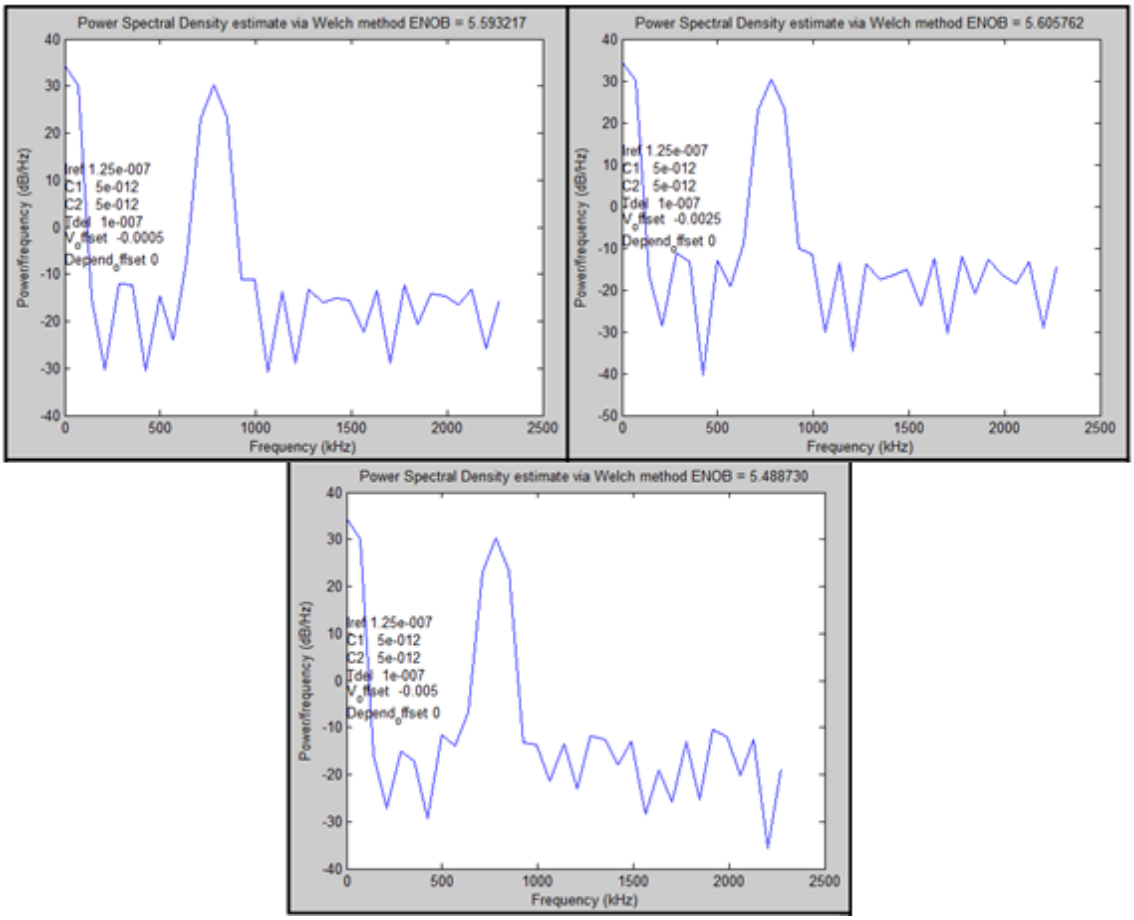

Figura 133. Resultado ENOB. Tdelay=100ns. V_offset

En la Figura 133 se observan los resultados obtenidos para un retraso de 10ns y distintos valores de offset independiente de señal del comparador. En este caso como era de esperar, no se obtiene ningún resultado en el que la ENOB sea aceptable. Esto es debido a que al efecto que tenía sobre la ENOB un retraso de 100ns visto en el apartado 5.2.2.2.2, se ha unido el efecto del offset independiente de señal en el comparador. Los resultados obtenidos para la ENOB se encuentran en torno a los 5 bits, lo cual está bastante por debajo de 8 bits, donde se podrían aceptar los resultados como aceptables.

A continuación, al análisis realizado para el estudio de la influencia del retraso y del offset independiente de señal del comparador en el rendimiento del convertidor calculando la ENOB, se va a añadir la influencia del offset dependiente. Para ello se van a mostrar en las siguientes figuras los resultados con los distintos valores de offset dependiente, offset independiente y de retraso del comparador. Para ello se va dar unos valores al retraso del comparador de 2ns, 10ns y 100ns por un lado, y un el offset independiente 0.0005V y 0.0025V, para cada valor de retraso por otro, variando el porcentaje de offset dependiente de señal con valores del 0.2%, 0.4% y 0.6.

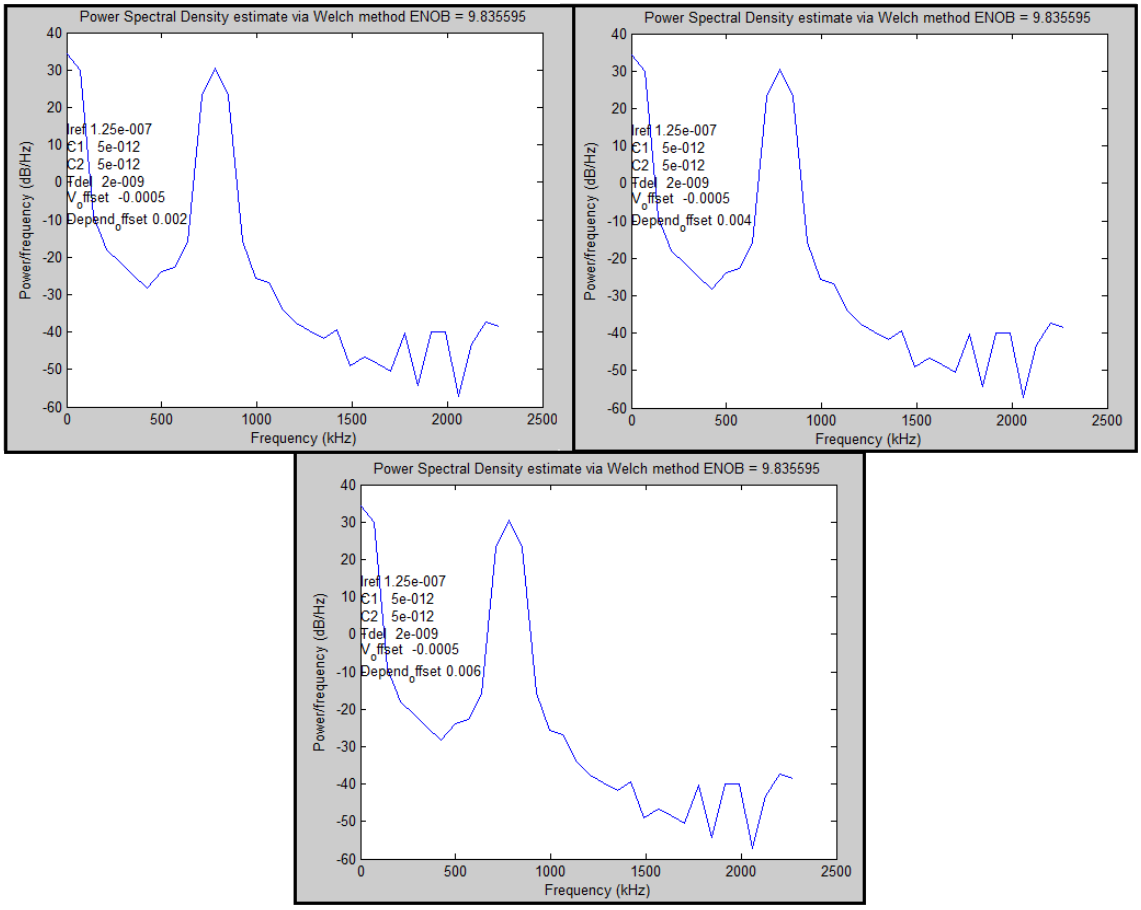

Figura 134. Resultado ENOB. Tdelay=2ns. V_offset=0.0005V. Depend_offset

En la *Figura 134* se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que, como ocurría en el apartado 5.2.2.2.5, el offset dependiente de señal no afecta significativamente en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 2ns y un offset independiente de 0.0005V, una ENOB para todos los valores de offset dependiente de 9.835595 bits, que como se ha comentado anteriormente se trata de una resolución efectiva satisfactoria para el rendimiento del convertidor.

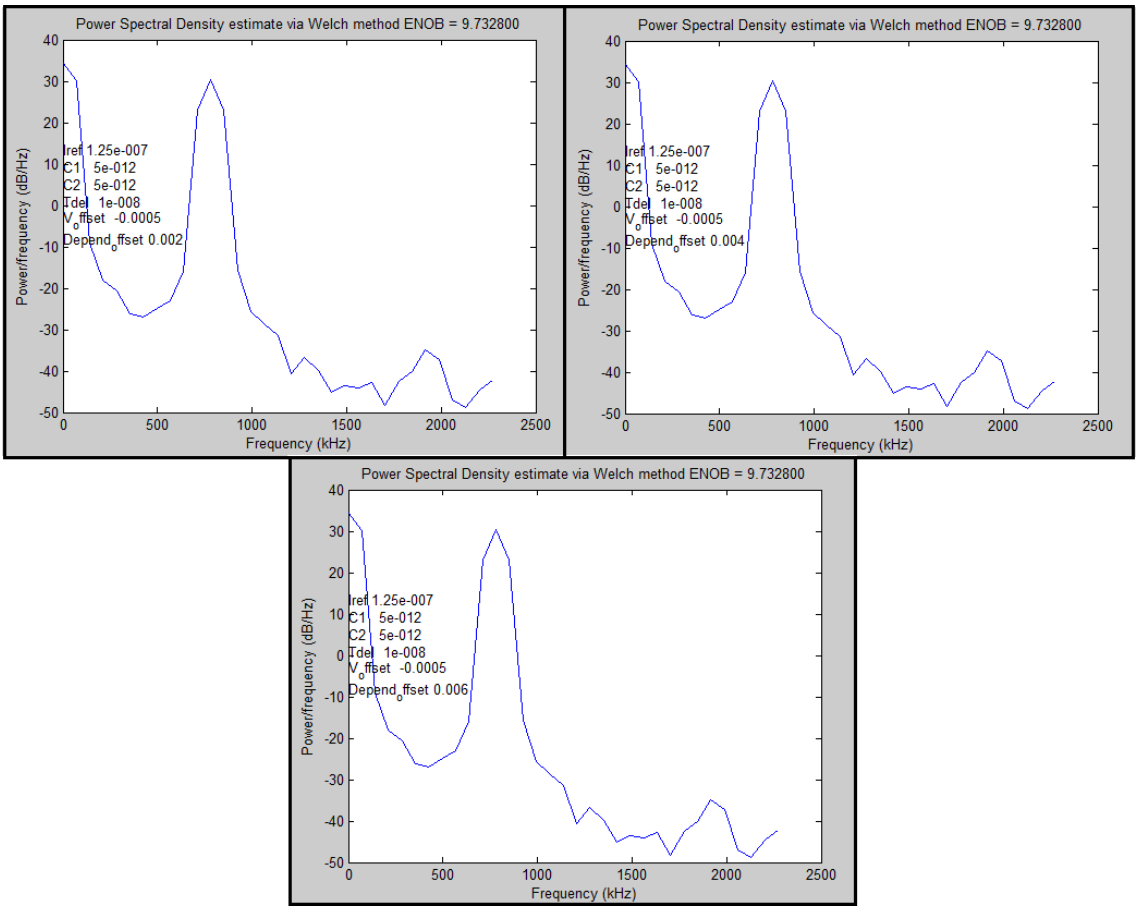

Figura 135. Resultado ENOB. Tdelay=10ns. V_offset=0.0005V. Depend_offset

En la *Figura 135* se observan los resultados obtenidos para un retraso de 10ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que el offset dependiente de señal sigue sin afectar en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 10ns y un offset independiente de 0.0005V, una ENOB para todos los valores de offset dependiente de 9.732800 bits, que como se ha comentado anteriormente se trata de una resolución efectiva satisfactoria para el rendimiento del convertidor.

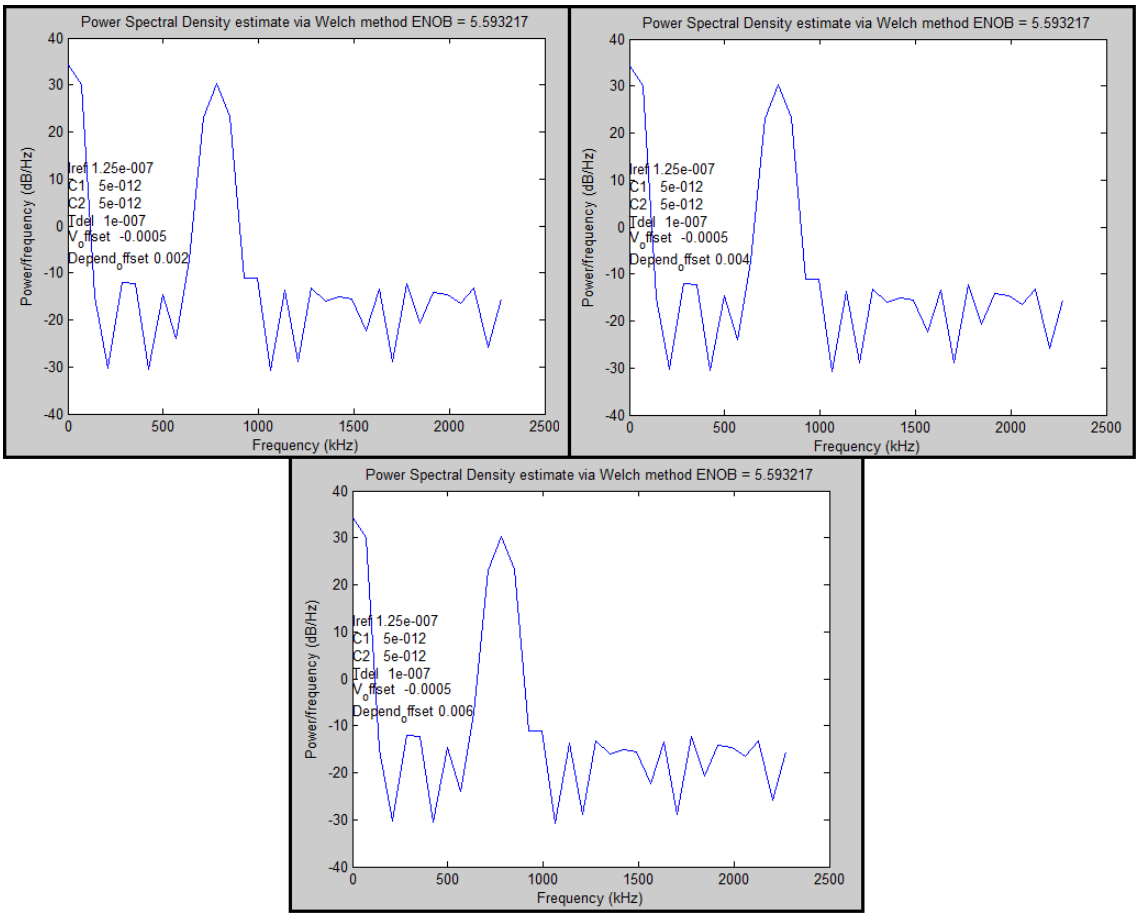

Figura 136. Resultado ENOB. Tdelay=100ns. V_offset=0.0005V. Depend_offset

En la *Figura 136* se observan los resultados obtenidos para un retraso de 100ns, un valor de offset independiente de señal de 0.0005V y para los diferentes valores de offset dependiente de señal. En estos se puede observar que el offset dependiente de señal sigue sin afectar en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 100ns y un offset independiente de 0.0005V, una ENOB para todos los valores de offset dependiente de 5.593217 bits, que como se ha comentado anteriormente se trata de una resolución efectiva que no llega a 8 bits dándose un resultado negativo para la resolución del convertidor.

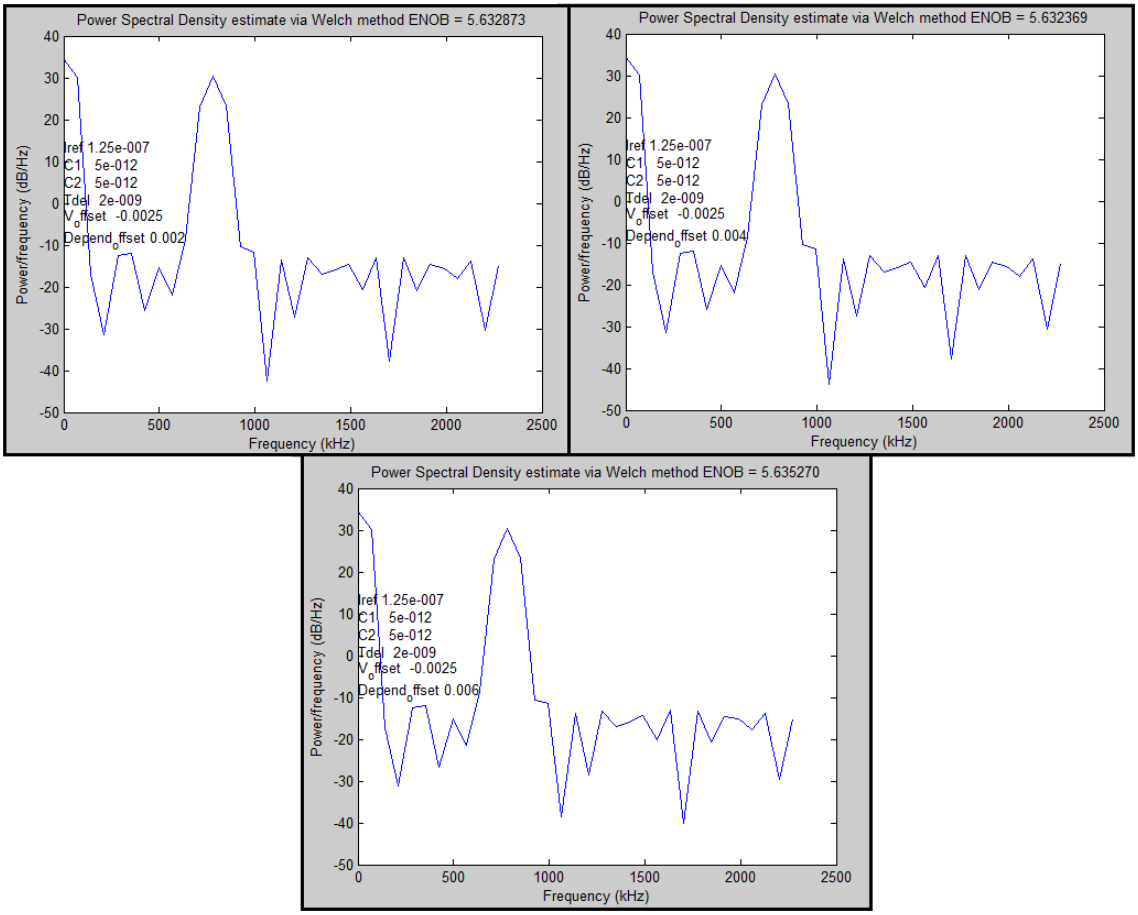

Figura 137. Resultado ENOB. Tdelay=2ns. V_offset=0.0025V. Depend_offset

En la *Figura 137*Figura 136 se observan los resultados obtenidos para un retraso de 2ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. Siguiendo la tendencia de resultados anteriores se puede observar que el offset dependiente de señal sigue sin afectar en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 2ns y un offset independiente de 0.0025V, una ENOB para todos los valores de offset dependiente de 5.632875, que como se ha comentado anteriormente se trata de una resolución efectiva que no llega a 8 bits, dándose un resultado negativo para la resolución del convertidor.

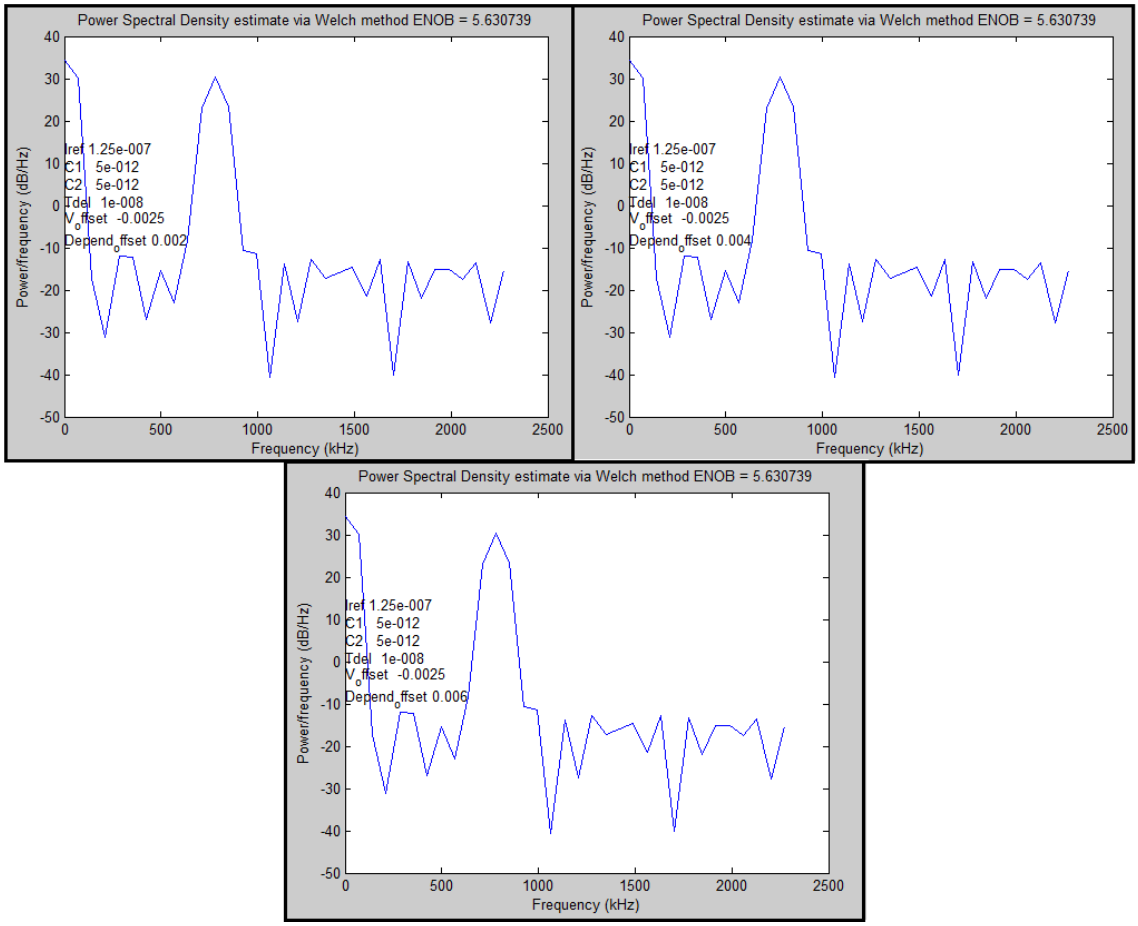

Figura 138. Resultado ENOB. Tdelay=10ns. V_offset=0.0025V. Depend_offset

En la *Figura 138*Figura 136 se observan los resultados obtenidos para un retraso de 10ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. Como pasaba anteriormente en los resultados se puede observar que el offset dependiente de señal sigue sin afectar en el rendimiento del convertidor obteniéndose, como en el estudio anterior en el que se analizaba la influencia para un retraso de 10ns y un offset independiente de 0.0025V, una ENOB para todos los valores de offset dependiente de 5.630739, que como se ha comentado anteriormente se trata de una resolución efectiva que no llega a 8 bits dándose un resultado negativo para la resolución del convertidor.

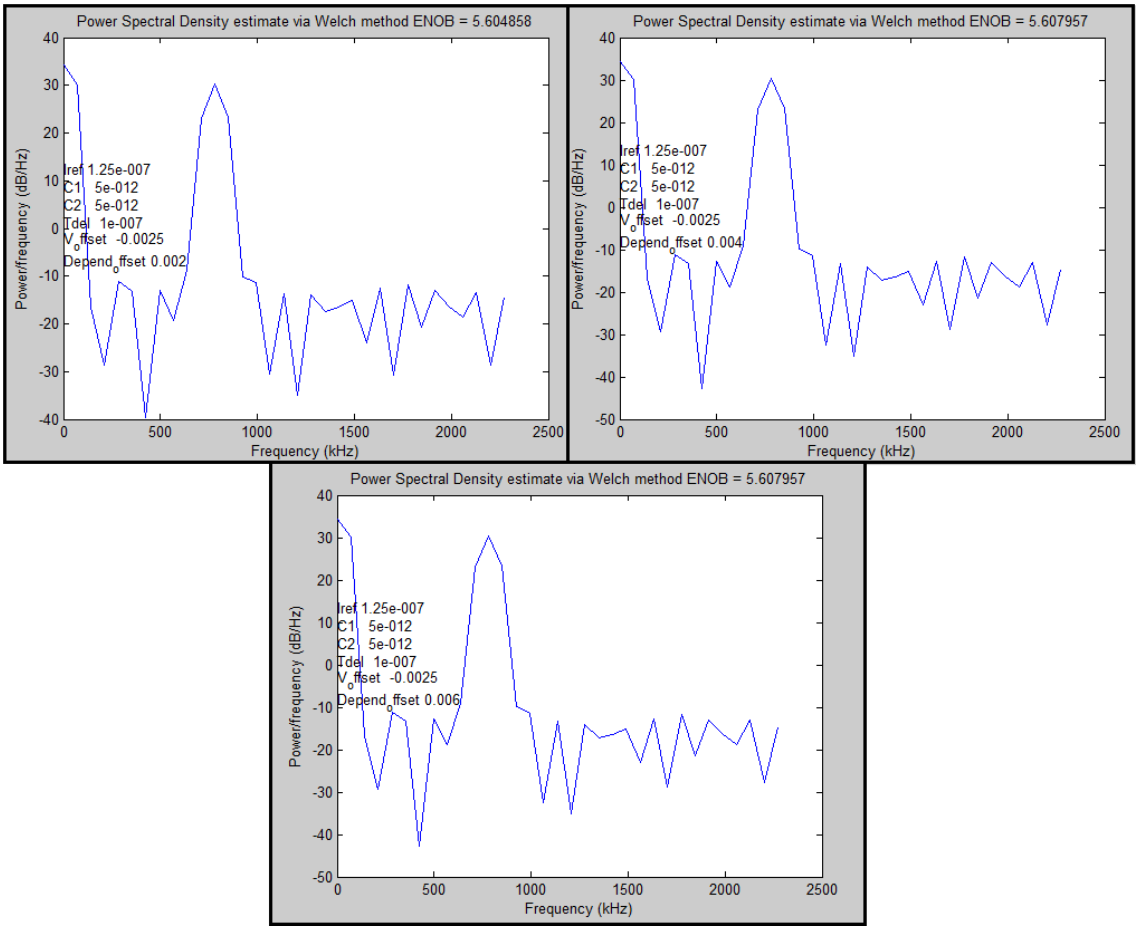

Figura 139. Resultado ENOB. Tdelay=100ns. V_offset=0.0025V. Depend_offset

En la *Figura 139*Figura 136 se observan los resultados obtenidos para un retraso de 100ns, un valor de offset independiente de señal de 0.0025V y para los diferentes valores de offset dependiente de señal. En este caso tampoco resulta una influencia significativa, pero a diferencia que en casos anteriores sí que hay alguna variación de décimas de bits. En cualquier caso, como en el estudio anterior en el que se analizaba la influencia para un retraso de 100ns y un offset independiente de 0.0025V, se obtiene una ENOB para todos los valores de offset dependiente en torno a los 5 bits, que como se ha comentado anteriormente se trata de una resolución efectiva que no llega a 8 bits dándose un resultado negativo para la resolución del convertidor.

Una vez realizados los estudios anteriores, se añade un análisis en el que intervienen todos los parámetros es decir, el retraso y el offset tanto dependiente como independiente del comparador junto con el mistmach de capacidades, mostrando los resultados más representativos, es decir, los resultados con los valores de los parámetros para los que los resultados son medianamente aceptables para el rendimiento del comparador y además el resultado con los valores de los parámetros para los que los resultados comienzan a ser negativos para el rendimiento del covertidor. En las siguientes figuras se muestran los resultados obtenidos.

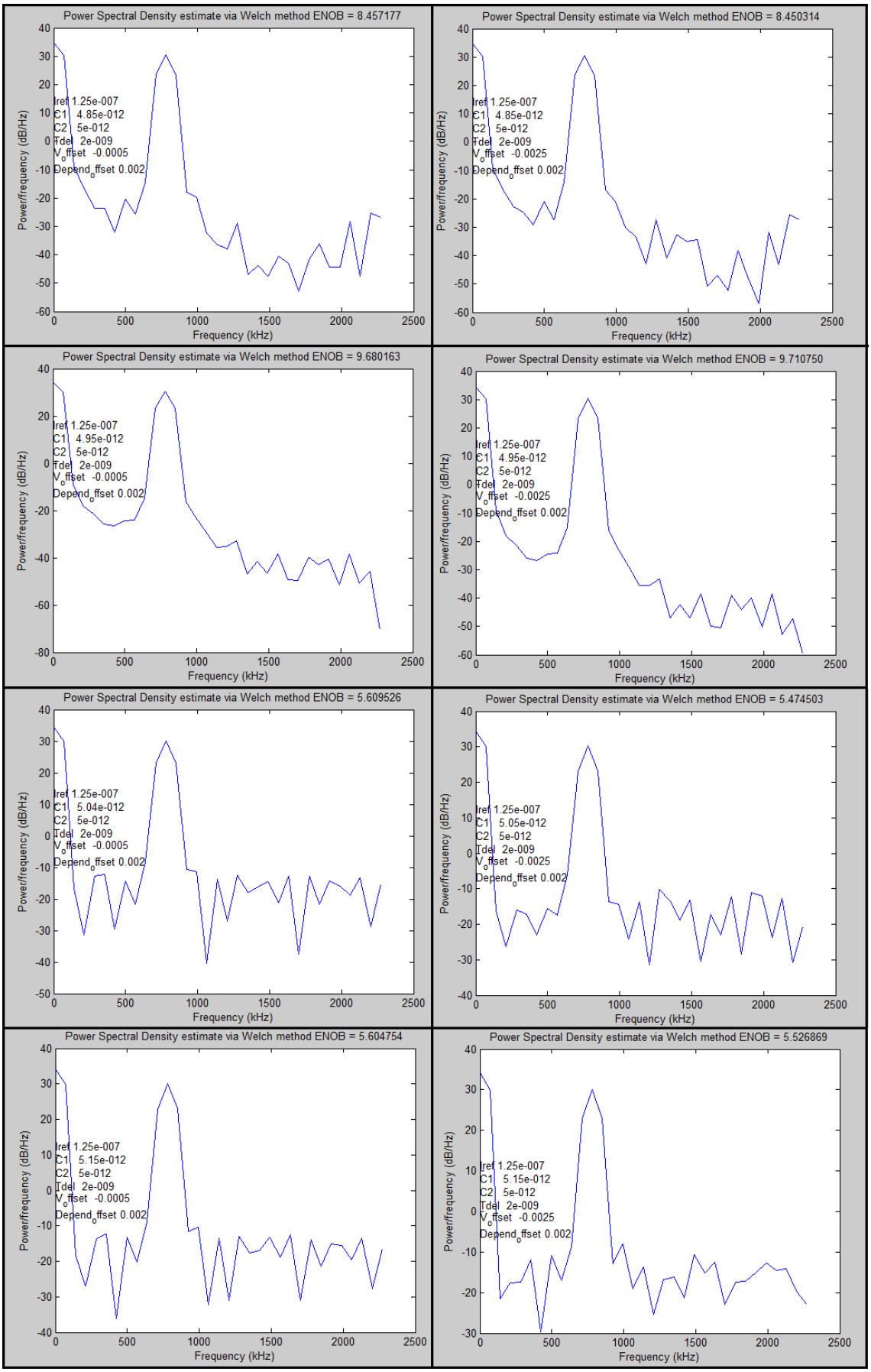

Figura 140. Resultado ENOB. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

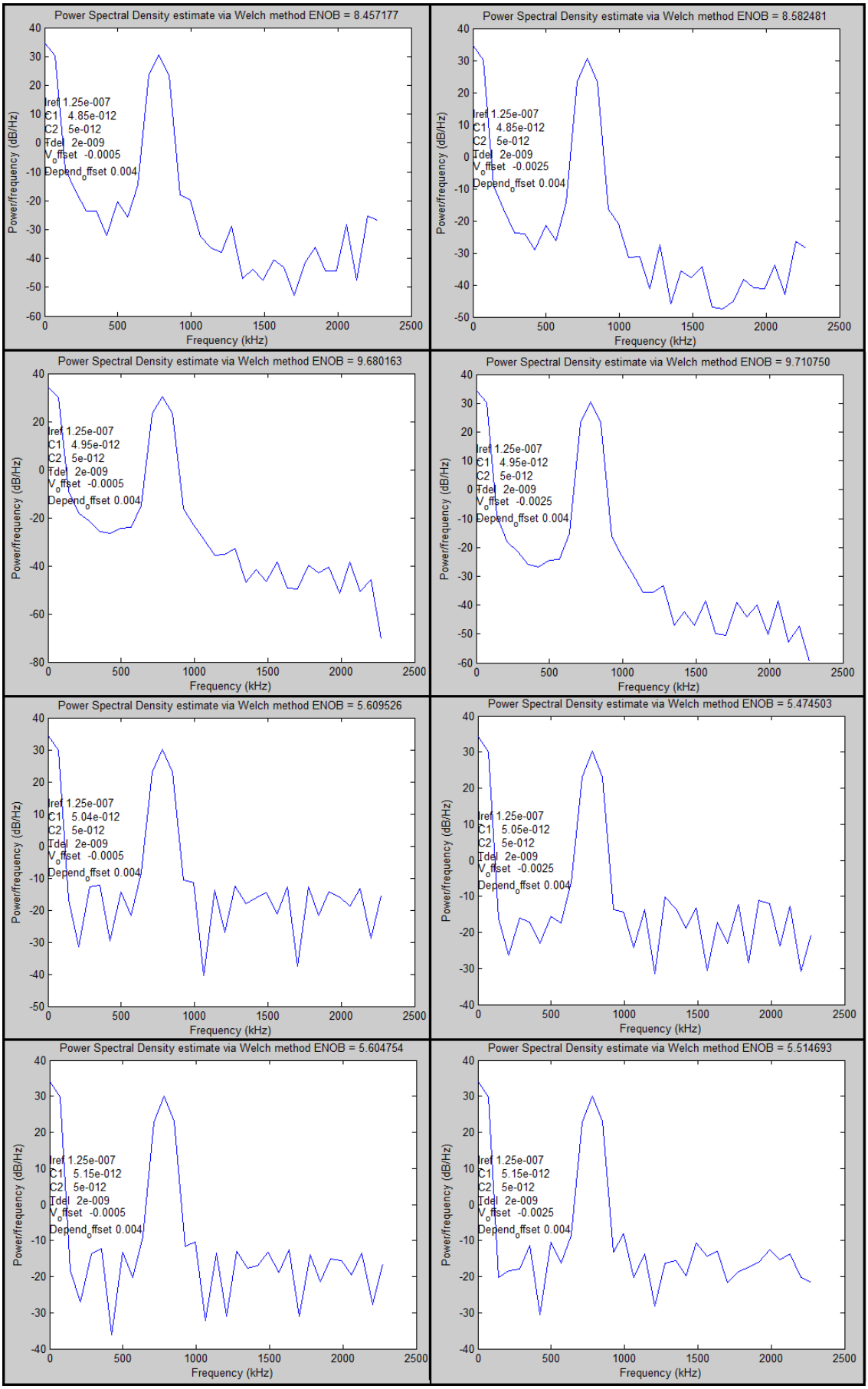

Figura 141. Resultado ENOB. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

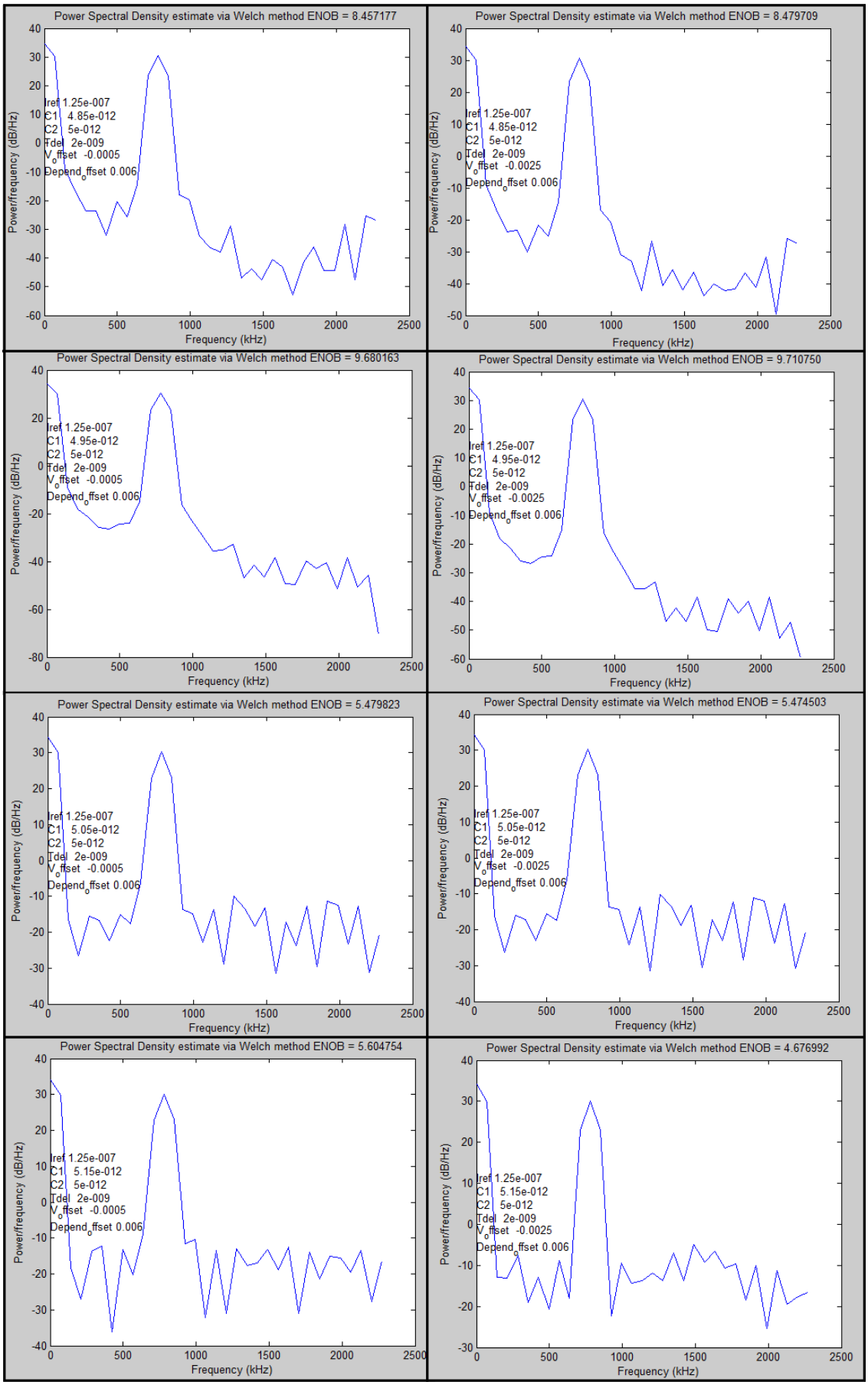

Figura 142. Resultado ENOB. Tdelay=2ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 140, 141 y 142 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 2ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente.

En la *Figura 140* se muestra el resultado para un offset dependiente del 0.2%, donde se puede observar que solo se obtiene una ENOB satisfactoria para el rendimiento del convertidor para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.457177 bits y 9.680163 bits respectivamente, junto con un offset independiente de señal de 0.0025V para unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.479709 bits y 9.710750 bits respectivamente. Para los demás valores de los parámetros se obtiene una resolución específica en torno a 5 bits, lo cual, como se ha repetido en varias ocasiones, no es un buen resultado para el rendimiento del convertidor.

En la *Figura 141* se muestra el resultado para un offset dependiente del 0.4%. Como ha venido pasando en estudios anteriores el offset dependiente de señal sigue sin influir significativamente en el rendimiento del convertidor variando sólo algunos de los resultados y de forma poco significativa. Al igual que en el caso inmediatamente anterior se obtiene una ENOB satisfactoria para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.457177 bits y 9.680163 bits respectivamente, junto con un offset independiente de señal de 0.0025V para unos valores de capacidad de 4.85pF y 4.95pF resultando 8.582481 bits y 9.710750 bits respectivamente.

En la *Figura 142* se muestra el resultado para un offset dependiente del 0.6%. El offset dependiente de señal tiene el mismo comportamiento que en los casos anteriores, de manera que varían sólo algunos de los resultados y de forma muy significativa. Al igual que en el caso inmediatamente anterior se obtiene una ENOB satisfactoria para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.457177 bits y 9.680163 bits respectivamente, junto con un offset independiente de señal de 0.0025V para unos valores de capacidad de 4.85pF y 4.95pF resultando 8.479769 bits y 9.710750 bits respectivamente.

Con las tres figuras anteriores se han expuesto los resultados obtenidos para analizar la ENOB variando todos los parámetros fijando el retraso del comparador en 2ns. Para todos los casos se han obtenido resultados satisfactorios para un valor de offset independiente de 0.0005V y de 0.0025V con unas capacidades para ambos de 4.85pF y 4.95pF, resultando la influencia del offset dependiente poco significativa, como ocurría cuando se analizaba este offset por separado.

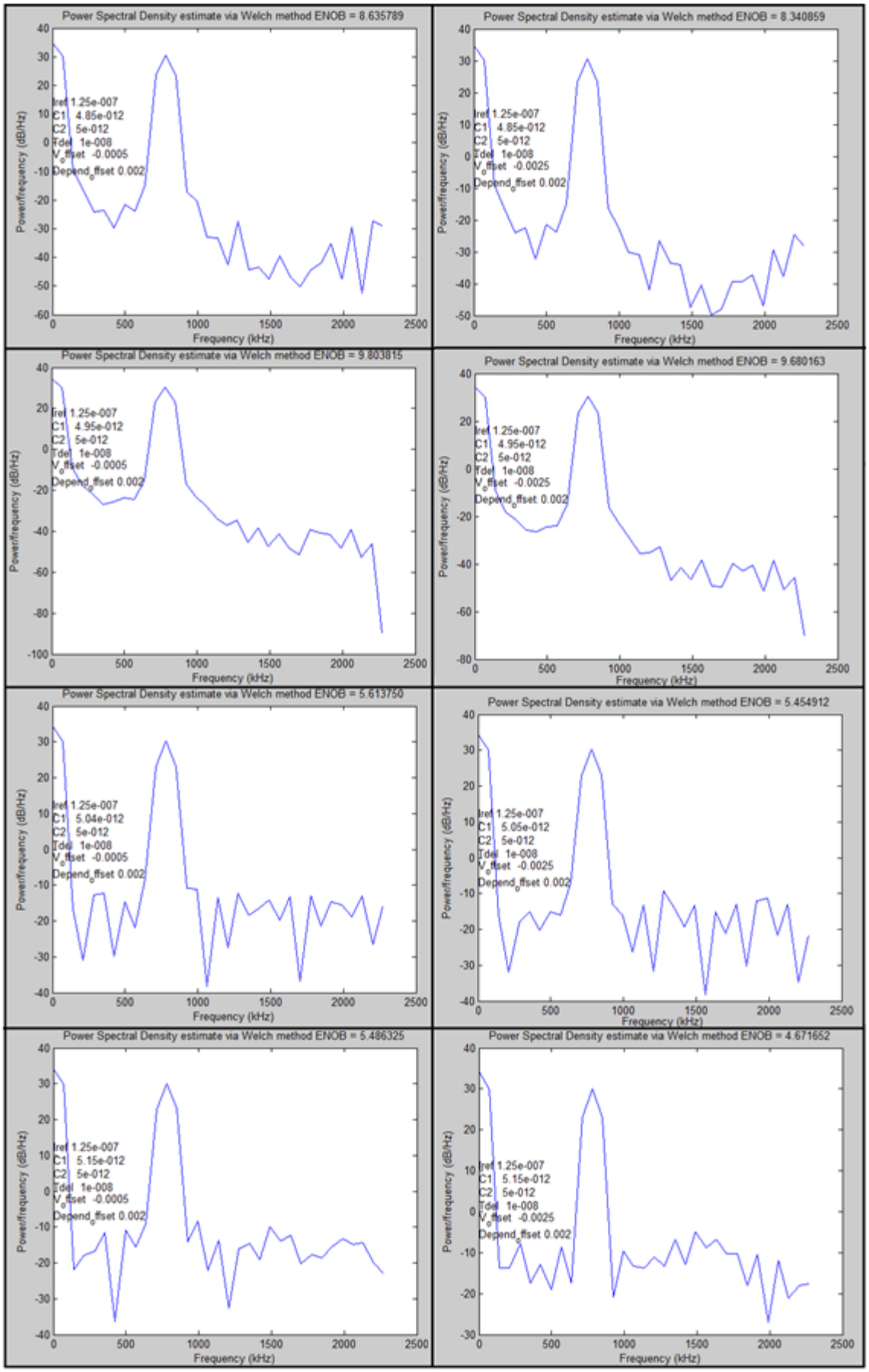

Figura 143. Resultado ENOB. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

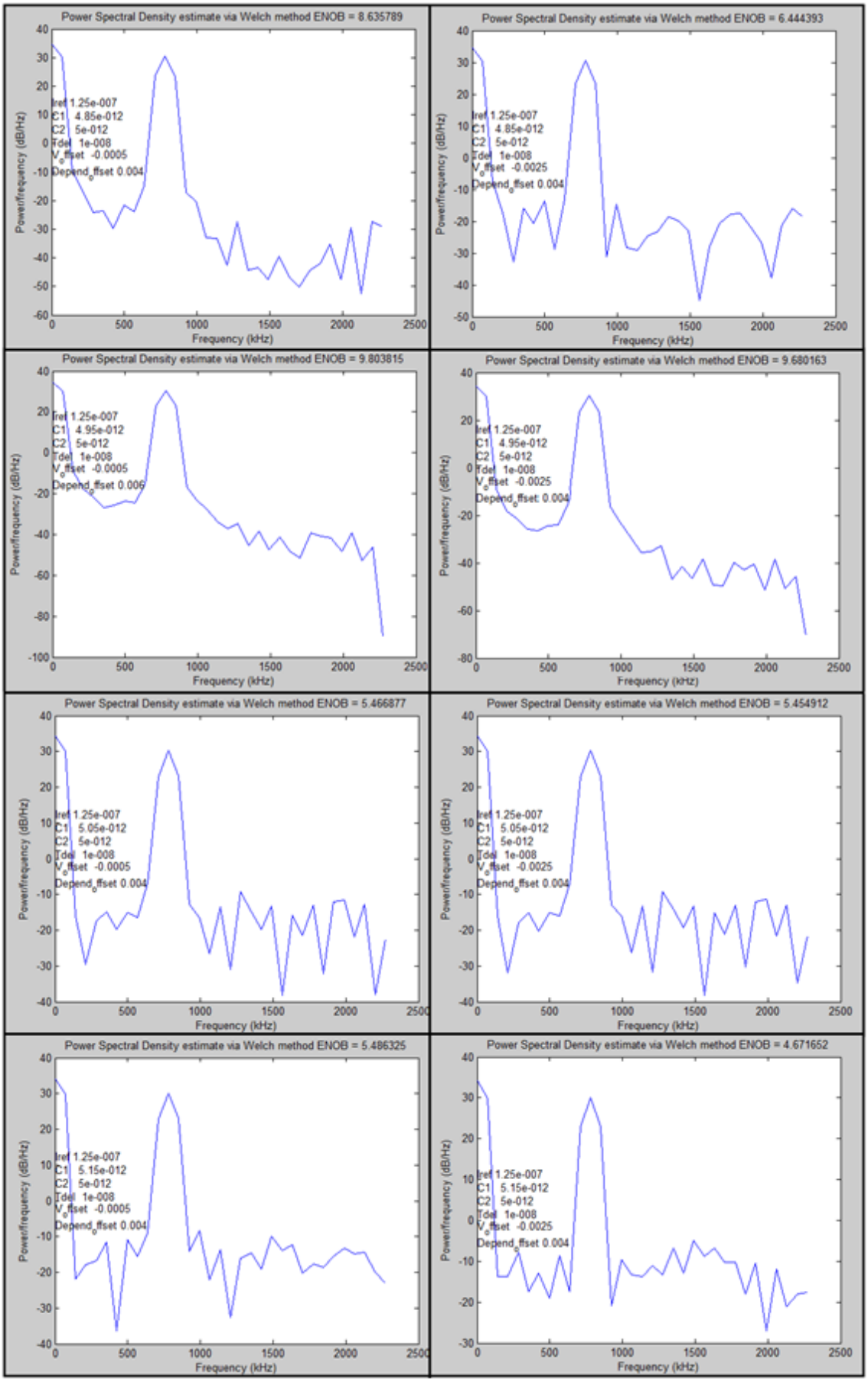

Figura 144. Resultado ENOB. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

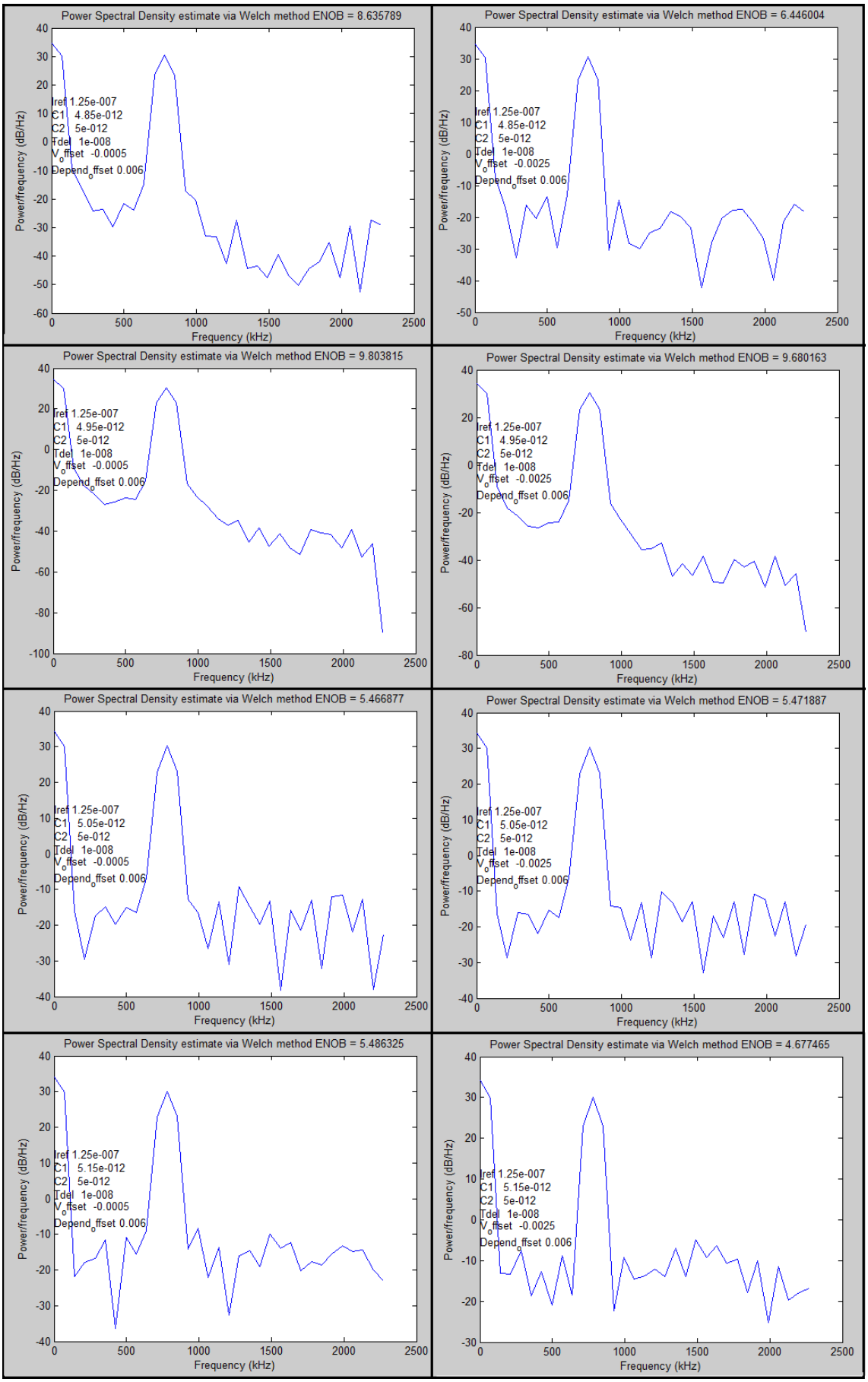

Figura 145. Resultado ENOB. Tdelay=10ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 143, 144 y 145 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 10ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente. Los resultados obtenidos para este retraso son muy similares a los obtenidos para un retraso de 2ns.

En la *Figura 143* se muestra el resultado para un offset dependiente del 0.2%, donde se puede observar que solo se obtiene una ENOB satisfactoria para el rendimiento del convertidor para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.635789 bits y 9.803815 bits respectivamente, junto con un offset independiente de señal de 0.0025V para unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.340859 bits y 9.680163 bits respectivamente. Para los demás valores de los parámetros se obtiene una resolución específica en torno a 5 bits, lo cual, como se ha repetido en varias ocasiones, no es un buen resultado para el rendimiento del convertidor.

En la *Figura 144* se muestra el resultado para un offset dependiente del 0.4%. En este caso el offset dependiente influye de manera más significativa que en casos anteriores, pero sigue siendo una influencia no demasiado grande, en concreto, para un offset independiente de 0.0025V y una capacidad de 4.85pF ya no se obtiene una ENOB satisfactoria, siendo esta de 6.444393 bits. Para los demás parámetros se siguen obteniendo una ENOB positiva para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.635789 bits y 9.803815 bits respectivamente, junto con un offset independiente de señal de 0.0025V para un valor de capacidad de 4.95pF resultando 9.680163 bits.

En la *Figura 145* se muestra el resultado para un offset dependiente del 0.6%. El offset dependiente de señal tiene el mismo comportamiento que en el caso inmediatamente anterior, de manera que varían sólo algunos de los resultados y de forma poco significativa. Al igual que en el caso inmediatamente anterior se obtiene una ENOB satisfactoria para un offset independiente de señal de 0.0005V con unos valores de capacidad de 4.85pF y 4.95pF para los cuales el resultado ha sido 8.457177 bits y 9.680163 bits respectivamente, junto con un offset independiente de señal de 0.0025V para un valor de capacidad de 4.95pF resultando 9.680163 bits.

Con las tres figuras anteriores se han expuesto los resultados obtenidos para analizar la ENOB variando todos los parámetros fijando el retraso del comparador en 10ns. Para todos los casos se han obtenido resultados satisfactorios para un valor de offset independiente de 0.0005V con una capacidad de 4.85pF y 4.95pF y para un offset independiente de 0.0025V con una capacidad de 4.95pF.

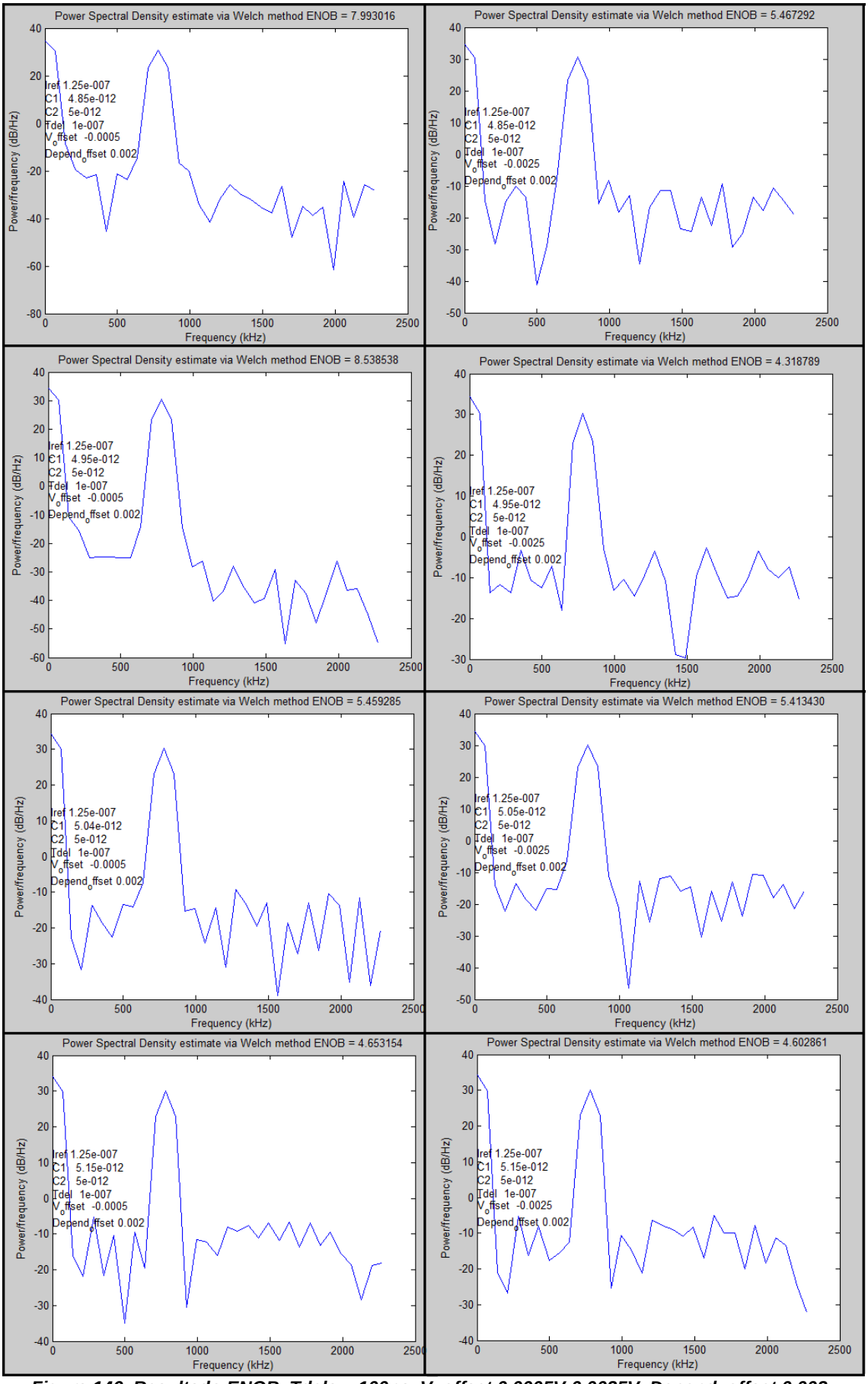

Figura 146. Resultado ENOB. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.002. Cmistmach

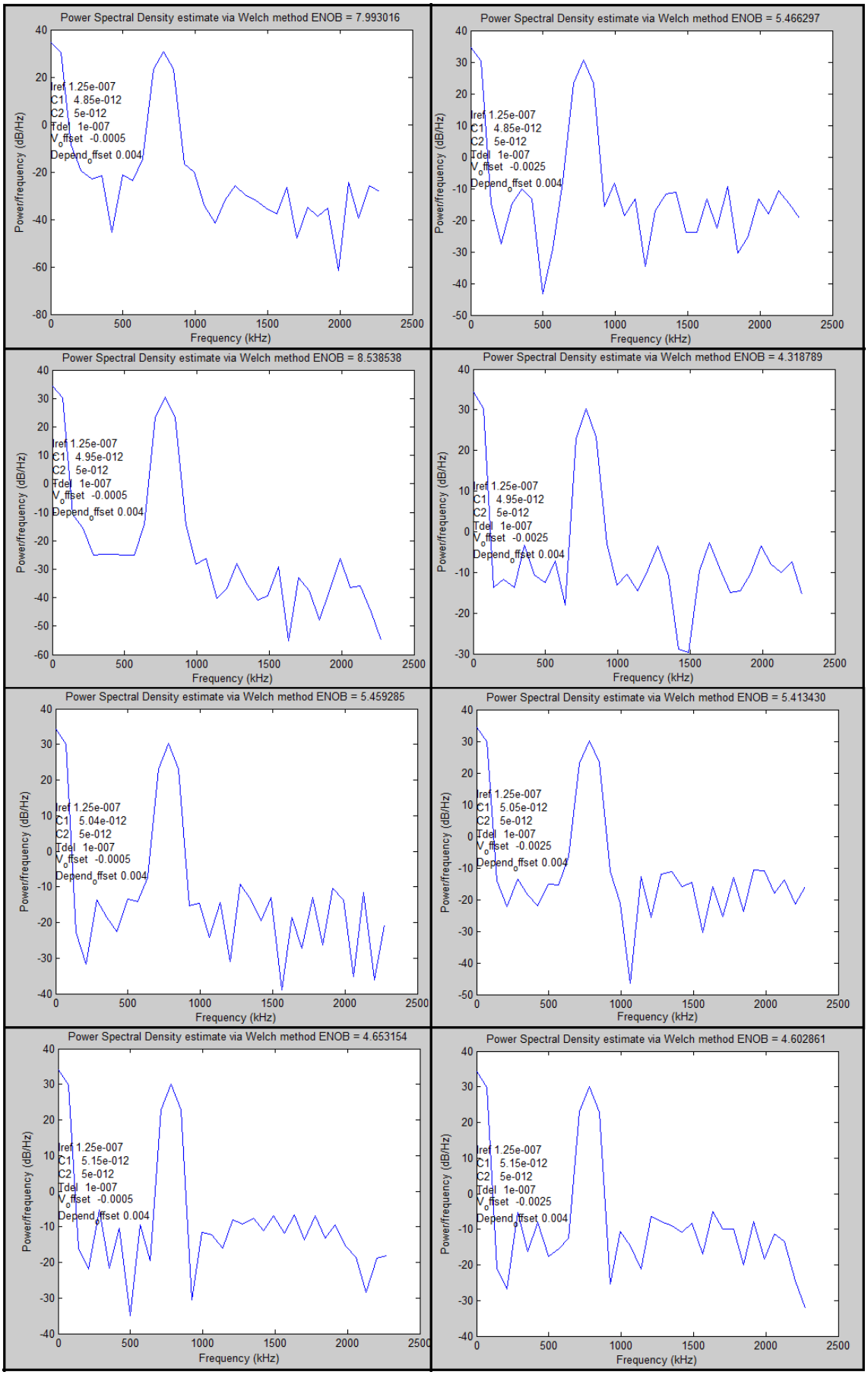

Figura 147. Resultado ENOB. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.004. Cmistmach

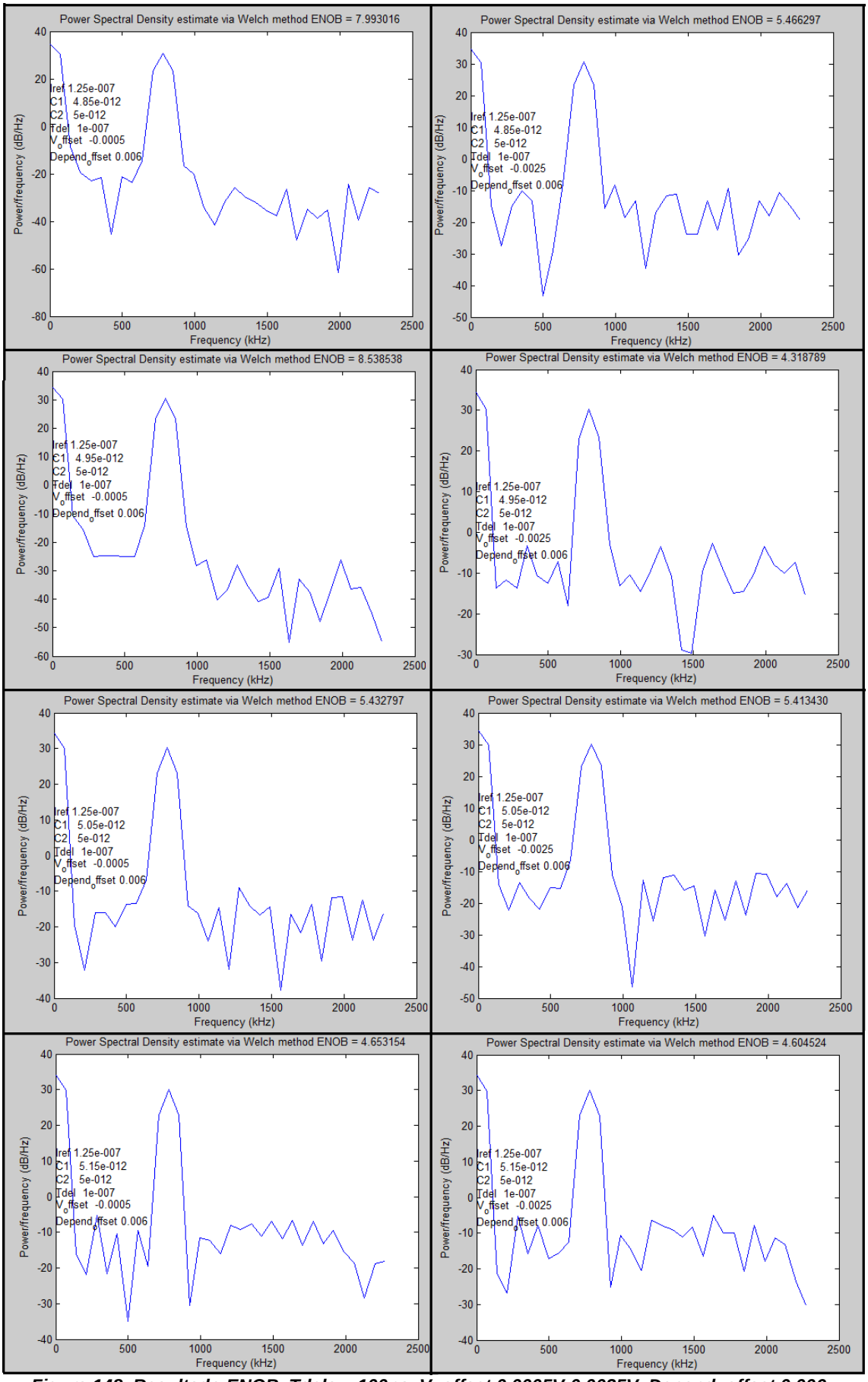

Figura 148. Resultado ENOB. Tdelay=100ns. V_offset 0.0005V-0.0025V. Depend_offset 0.006. Cmistmach

En las figuras 146, 147 y 148 se muestran los resultados obtenidos tras la simulación para un valor de retraso en el comparador de 100ns, un valor de capacidades de 4.85pF, 4. 95pF, 5.05pF y 5.15pF, un valor de offset independiente de señal de 0.0005V (gráficas de la columna izquierda) y de 0.0025V (gráficas de la comuna derecha) y finalmente para un offset dependiente de señal de un 0.2%, 0.4% y 0.6% para cada una de las figuras respectivamente.

En las tres figuras se obtienen los mismos resultados ya que el offset dependiente de señal no influye en ninguno de los casos, como ha venido pasando a lo largo del análisis. Como era de esperar con un retraso en el comparador tan elevado los resultados obtenidos ofrecen una ENOB muy negativa para el rendimiento del convertidor, concretamente, para todos los parámetros se ha obtenido una ENOB en torno a los 5 bits.

Con el estudio realizado para cada uno de los parámetros del modelo de Matlab/Simulink queda concluido el análisis de la ENOB para mostrar el rendimiento del convertidor. A modo de resumen, en el ANEXO V se mostrará una tabla con los resultados obtenidos tras realizar todas las simulaciones, tanto los resultados mostrados a lo largo del análisis, como los que no se han mostrado por no resultar de relevancia atendiendo a resultados anteriores.

Una vez analizado tanto el error INL como la ENOB, se está en disposición de abordar el diseño en Cadence introduciendo elementos reales, teniendo en cuenta las distintas consideraciones obtenidas tras estos estudios.

5.3 Implementación del diseño en verilog: Modelo eléctrico

Para continuar con el análisis de la arquitectura y terminar diseñando el convertidor analógico-digital basado en tiempo para llevarlo al silicio se ha implementado un modelo verilog de este. La idea es partir de un modelo similar al implementado en Malab/Simulink, el cual es muy simple, e ir introduciendo elementos reales para ver su comportamiento. En este proyecto únicamente se introducido el modelo para poder añadir elementos reales y seguir analizando distintas especificaciones del convertidor como el error INL y la ENOB. Una vez que se hayan introducido todos los elementos reales y se haya comprobado la viabilidad de la arquitectura se podrá llevar al silicio.

Para implementar el modelo verilog se ha actuado de forma similar a como se actuaba para diseñar el modelo en Matlab/Simulink. Se han implementado por separado los distintos bloques y posteriormente se han conectado para completar el diseño completo del convertidor.

5.3.1 Diseño de los distintos bloques del sistema

5.3.1.1 Capacidad

En el caso de las capacidades del sistema no se ha implementado ningún bloque, ya que estas se pueden encontrar como elementos de librería, pues se está trabajando con un programa de diseño de circuitos integrados. Por otro lado, en Cadence también se tienen switches controlados por tensión ideales de librería, por lo que únicamente se han conectado las capacidades y los switches de manera adecuada para conseguir el efecto que se busca en la arquitectura. Esto se puede ver en la *Figura 149.*

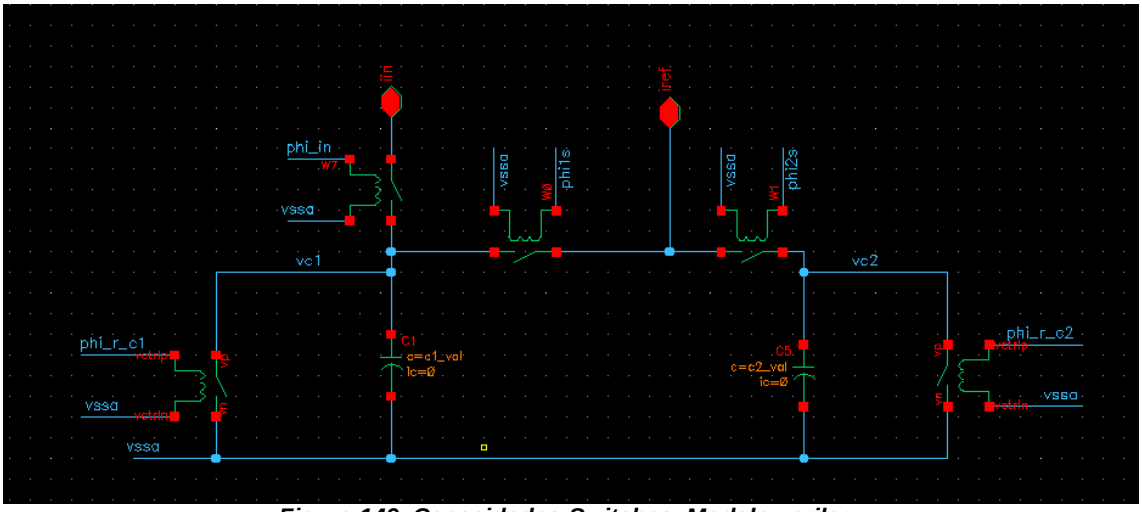

Figura 149. Capacidades-Switches. Modelo verilog

5.3.1.2 Lógica

Al igual que en el modelo Matlab/Simulink, se hace necesaria una serie de lógica para generar fases no solapadas y producir un funcionamiento correcto en el convertidor. El aspecto de este bloque se puede ver en la *Figura 150.*

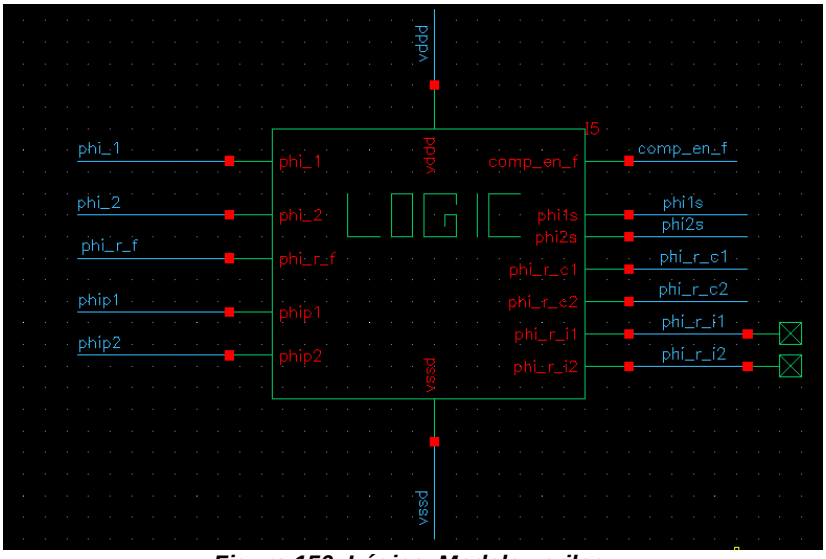

Figura 150. Lógica. Modelo verilog

En la *Figura 151* se puede observar como se ha implementado el bloque de lógica denominado *Logic_seg*. Se han utilizado diferentes puertas lógicas NOT, NAND y NOR, las cuales a su vez están implementadas en veriloga. Tanto las señales de entrada y salida, como las puertas lógicas están implementadas y tiene la misma función que en el modelo Matlab/Simulink.

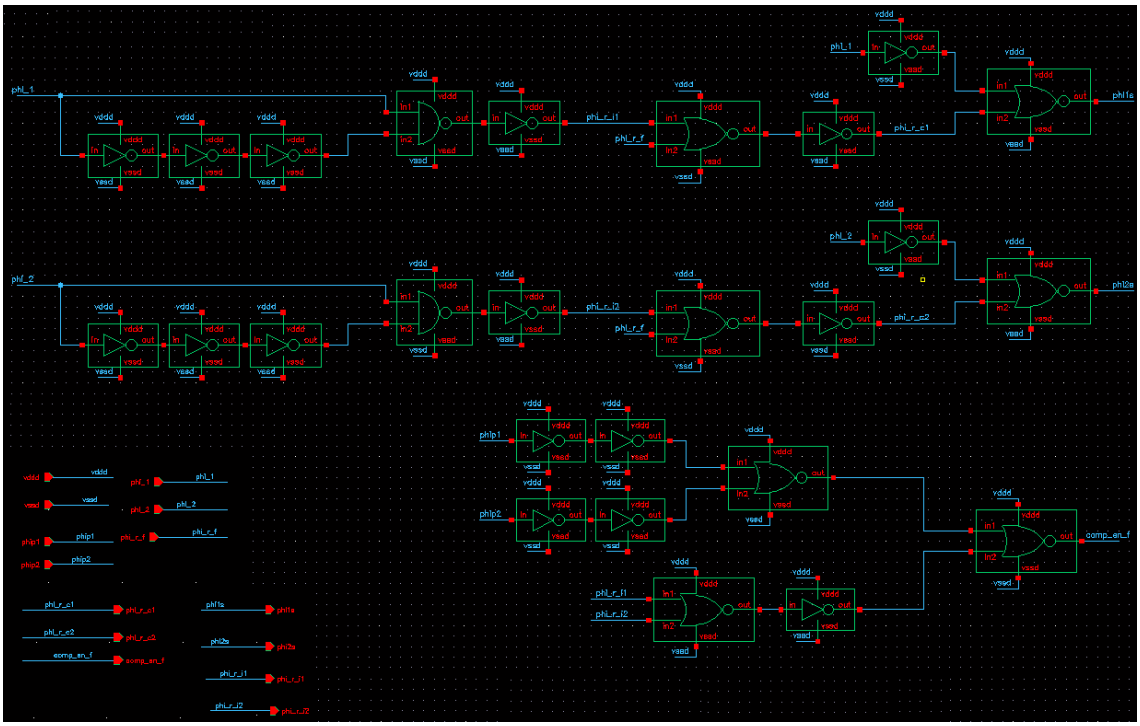

Figura 151. Implementación de Lógica: Logic_seg. Modelo verilog

5.3.1.3 Fuente de corriente

En el modelo Matlab/Simulink no se implementaban las fuentes de corriente, simplemente, para emular su efecto, se utilizaba una constante. En Cadence, al tratarse de un modelo eléctrico no se puede actuar de la misma forma. Por este motivo se ha implementado un bloque que actúa como fuente de corriente. Tanto el aspecto del bloque, como la implementación de este se pueden ver en la *Figura 152.*

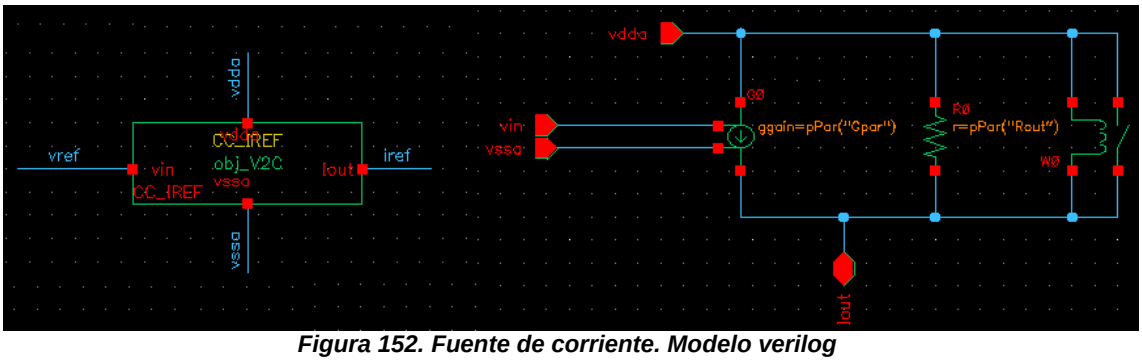

Figura 152. Fuente de corriente. Modelo verilog

El nombre del este bloque es *Obj_V2C*. Este modelo de fuente de corriente incluye un interruptor que es un limitador para acotar la salida de la fuente a la tensión *vdda*.

5.3.1.4 Comparador

Se ha implementado un bloque que incluye el comparador propiamente dicho y también los switches que controlan la conexión y desconexión de la salida de los condensadores al comparador. Por este motivo como entradas de este bloque se encuentran tanto las salidas de los condensadores y la señal que habilita al comparador, como las señales que controlan los switches para lo conexión al terminal positivo y negativo del comparador. El aspecto de este bloque denominado *comparator_sh* se puede ver en la *Figura 153*.

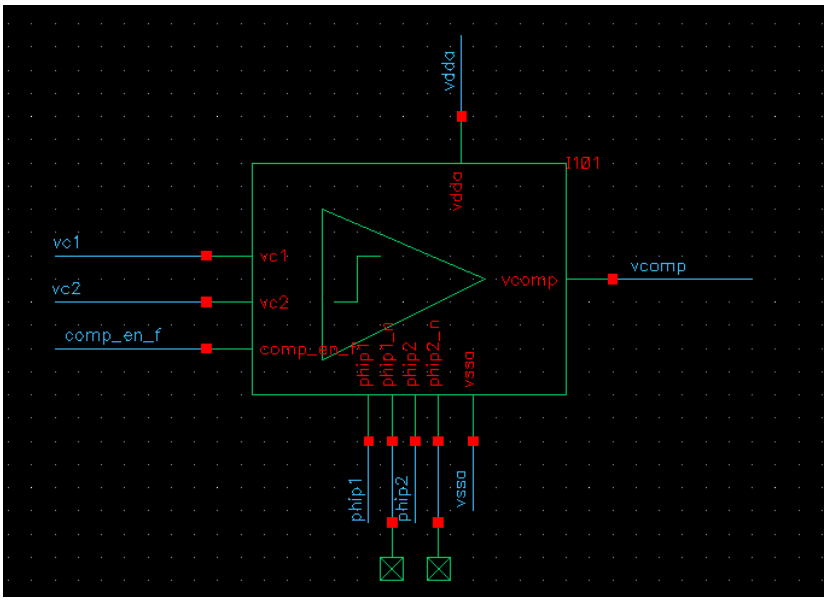

Figura 153. Comparador: comparator_sh. Modelo verilog

La implementación de este bloque se ve en la *Figura 154*.

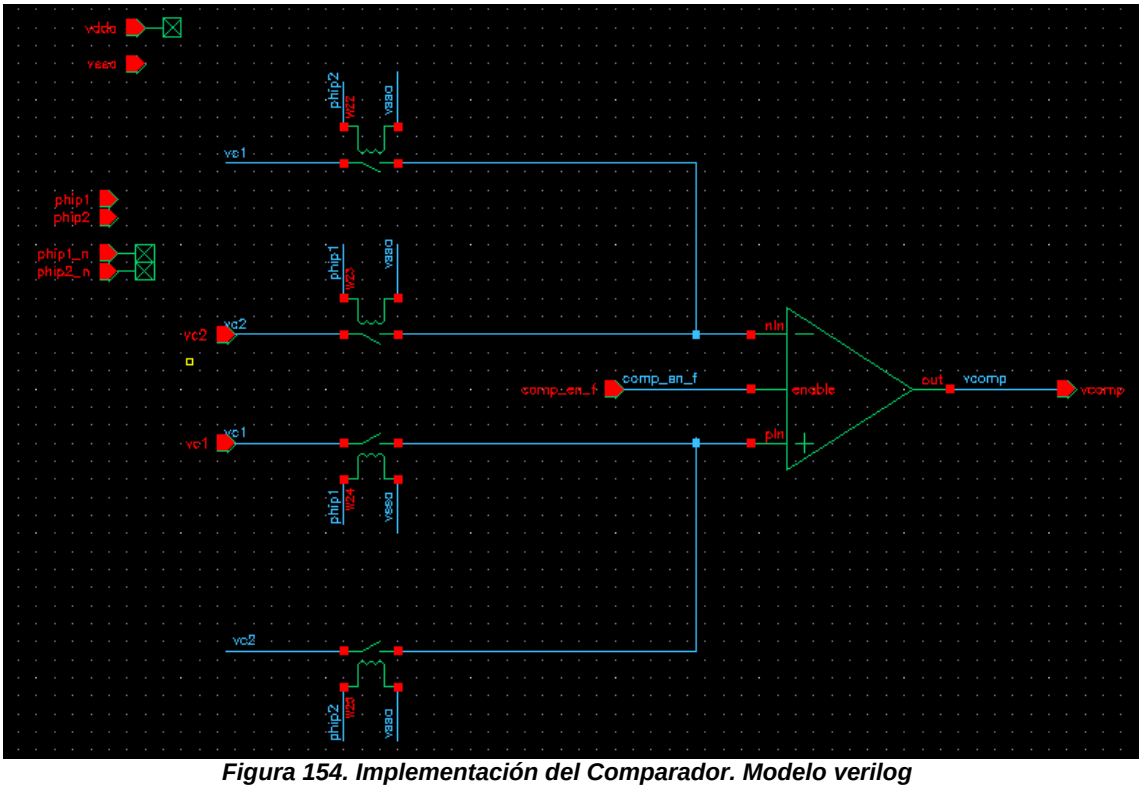

Figura 154. Implementación del Comparador. Modelo verilog

Como se ha comentado anteriormente, se pueden observar tanto los switches, como un comparador ideal. La implementación de este comparador ha sido realizada en veriloga, de manera que este se comporta de una manera ideal. El código veriloga que implementa el comparador se puede ver en la *Figura 155*.

```
// VerilogA for ADC_SAR_sensor, comparator, veriloga

`include "constants.vams"
`include "disciplines.vams"

module comparator(out, pin, nin, enable);
        parameter real td = 2n; //time delay (s)
        parameter real tr = 1n; //output transition time (s)
        parameter real tf = 1n;
        output out;
        input pin, nin, enable;
        electrical out, pin, nin, enable;
        integer x;
        parameter real vtrans_clk=0.6;
        parameter real vlogic_high = 1.2;
        parameter real vlogic_low = 0;

        analog begin
                @(cross(V(pin)-V(nin), +1))
                        x=1;
                if (V(enable)<vlogic_high) begin
                        x=0;
                end
                V(out) <+ transition (vlogic_high*x + vlogic_low*!x, td, tr, tf);
        end

endmodule
```

Figura 155. Código veriloga-Comparador

5.3.1.5 Máquina de estados: Síntesis digital con PKS

Para exportar la máquina de estados diseñada en código VHDL y llegar hasta el circuito digital sintetizado que implementa dicha máquina describiendo la funcionalidad se ha utilizado la herramienta PKS del entorno Cadence. El aspecto del bloque denominado *cur_based adc _sm* se puede ver en la *Figura 156.*

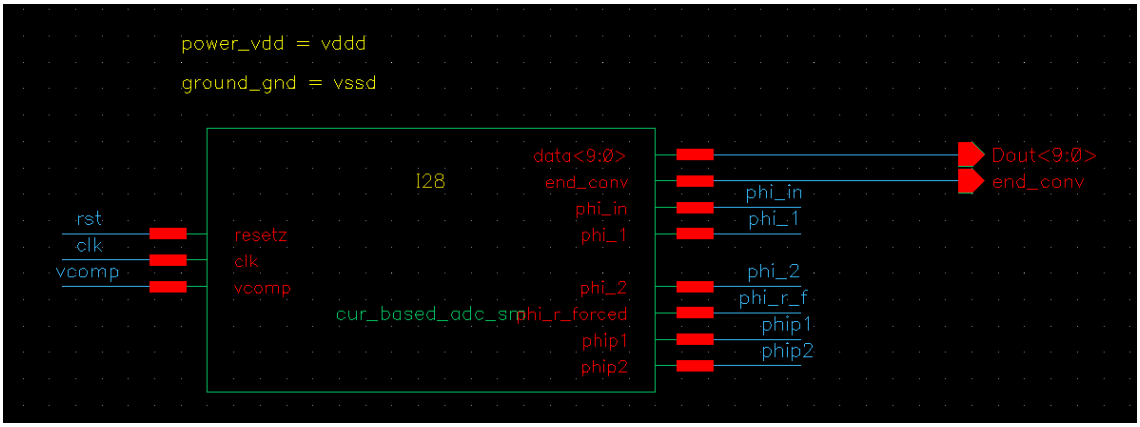

Figura 156. Máquina de estados: cur_based_adc_sm. Modelo verilog

La implementación de este bloque lleva consigo un gran número de puertas lógicas y flip-flops. Pese a que no se verán de forma clara, se mostrará la implementación para ver su complejidad en la **¡Error! No se encuentra el origen de la referencia.**.

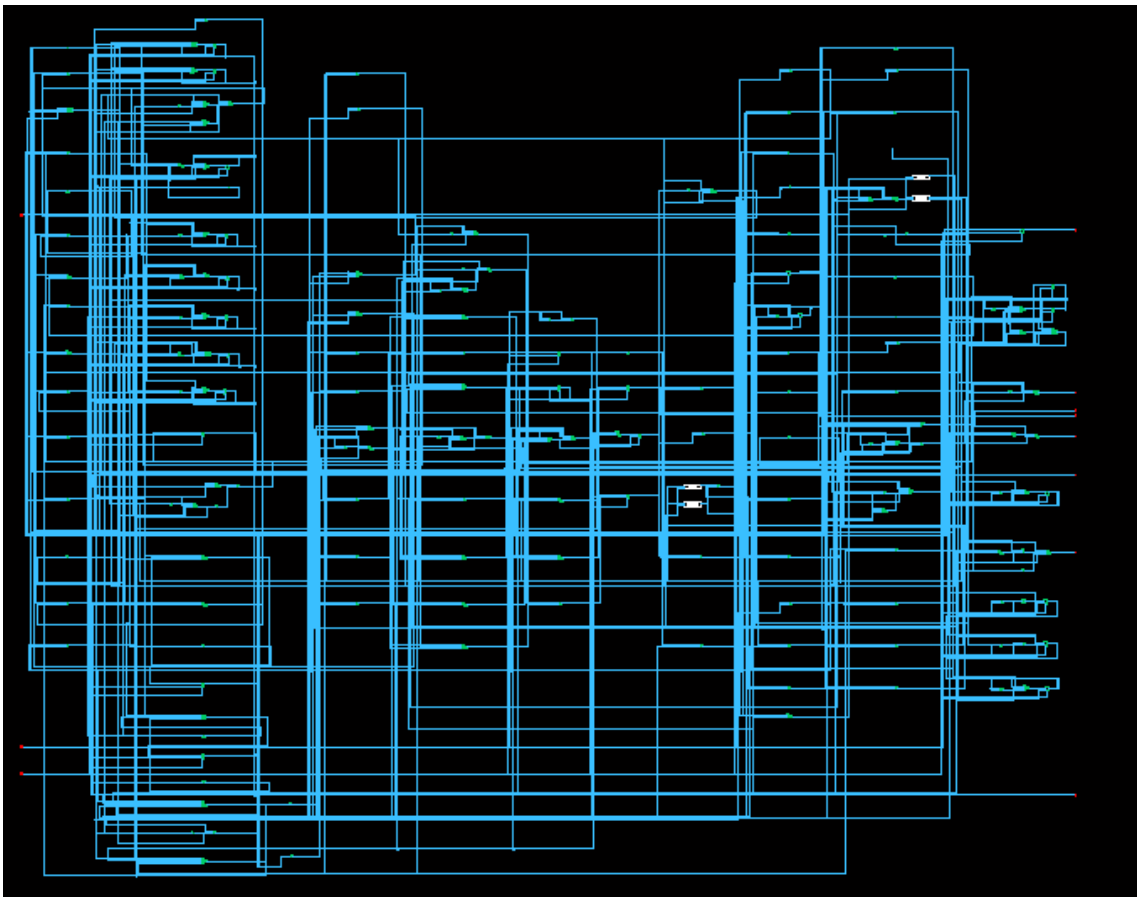

Figura 157. Implementación Máquina de estados. Modelo verilog

Con el diseño de la máquina de estados quedan todos los bloques diseñados para formar la arquitectura del ADC basado en tiempo..

5.3.1.6 Modelo Completo:ADC basado en tiempo

Una vez diseñados todos los bloques, se han conectado para formar la arquitectura y así el convertidor analógico-digital basado en tiempo en un bloque denominado *current_adc_top.* El aspecto de este bloque se puede ver en la *Figura 158*.

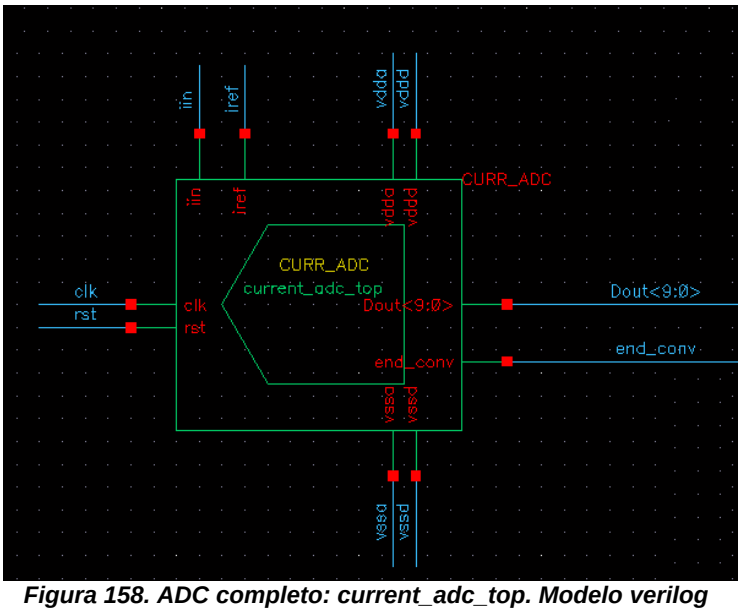

Figura 158. ADC completo: current_adc_top. Modelo verilog

El ADC basado en un convertidor Time-To-Digital, se implementa mediante los cuatro grandes bloques que se han descrito. Uno de carga y generación de señales dependientes de tiempo (rampas en condensadores). Un bloque que compara la tensión de los dos condensadores. Un bloque lógica para la generación de las fases necesarias. Y finalmente, la máquina de estados. La implementación del bloque *current_adc_top* se puede ver en la *Figura 159.*

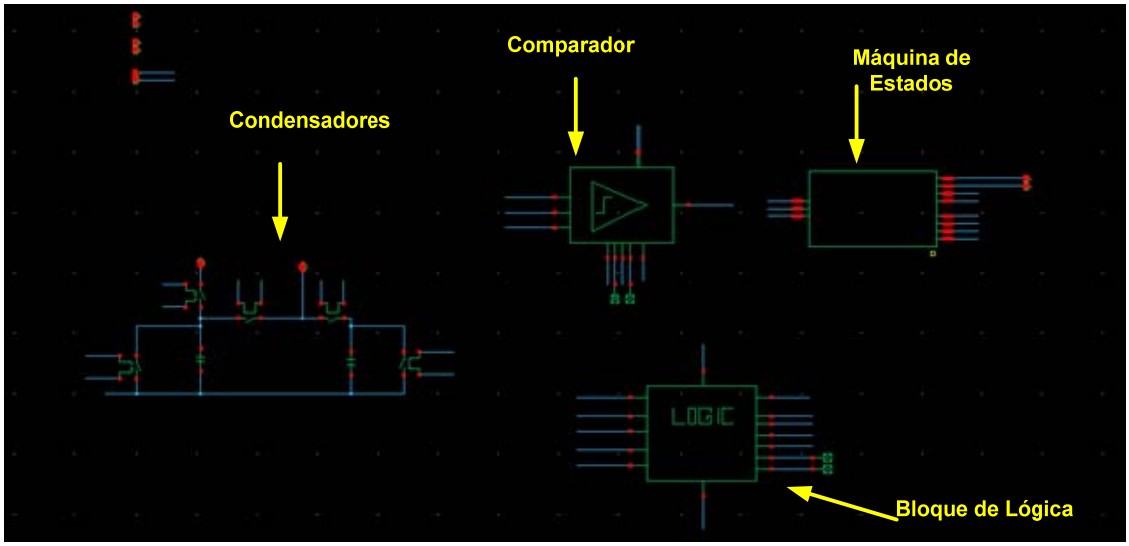

Figura 159. Implementación del ADC completo. Modelo verilog

5.3.2 Test-Bench: Simulación

Una vez implementado el modelo completo del convertidor analógico-digital basado en tiempo, se ha de realizado un test-bench para seguir estudiando la viabilidad de la arquitectura y completar el diseño introduciendo elementos reales, y poder llegar al silicio. La idea es ir incluyendo elementos reales en el modelo, ya que hasta el momento únicamente se han utilizado componentes ideales. Para ello se ha realizado un test-bench denominado *Sim_current_adc_tran_02.* Este test-bench consta de una señal de entrada sinusoidal muestreada cada veintidós ciclos de clk (10us). En el test-bench se encuentra el modelo del ADC bajo estudio, convertidor analógico digital basado en tiempo (comparaciones de tiempo). Además en el banco de pruebas se incluye un ADC ideal para comparar los resultados. Adicionalmente, se incluyen dos DAC ideales que nos permitirán evaluar la calidad del ADC. El test-bench se muestra en la *Figura 160.*

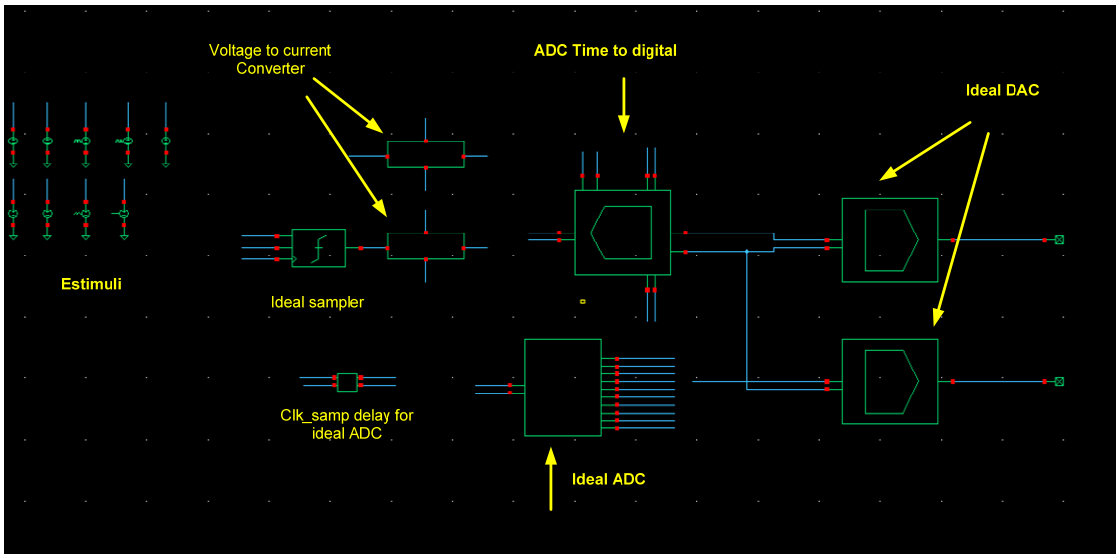

Figura 160. Test_bench: Sim_current_adc_tran_02

Con este test-bench se obtendrá la ENOB del convertidor analógico digital basado en tiempo, ya que esta es una indicación global de la resolución del sistema. Se estudia esta especificación debido a que en ella quedan reflejadas todas las imperfecciones del sistema, es decir, engloba la totalidad de fuentes de ruido, siendo así la mejor manera de comprobar el rendimiento del convertidor.

Hasta ahora, como se ha comentado anteriormente, solo se incluyen elementos ideales en el modelo, por lo que se espera una resolución efectiva muy buena. De hecho se una ENOB de 10.093 bits, como se muestra a continuación en la *Figura 161*

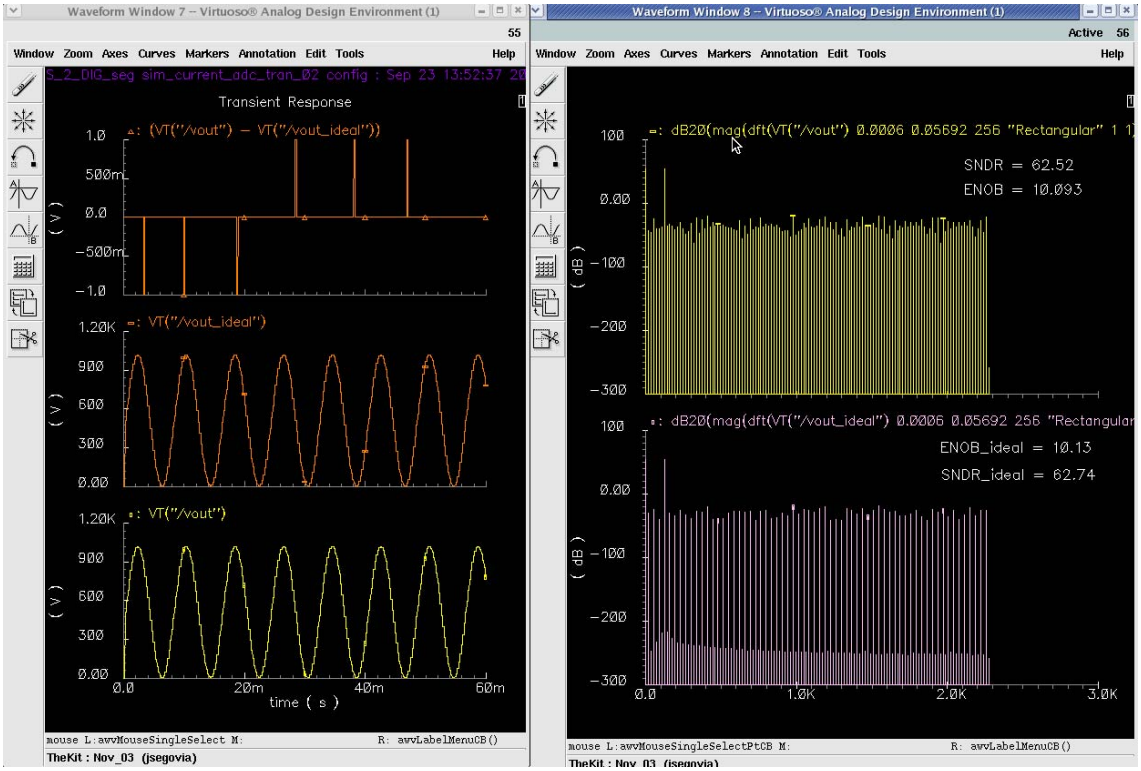

Figura 161. ENOB. Diseño ideal

En este proyecto se ha llegado hasta este punto, de manera que se está en disposición de abordar el diseño completo introduciendo los diferentes elementos reales llegando a implementar el ADC basado en tiempo en caso de que la arquitectura sea viable.