Ejercicios resueltos de procesado de señal en FPGA

Hugo Moreno Avilés
Jhon Jairo Cevallos

ESPOCH
2021

Ejercicios resueltos de procesado de señal en FPGA

Ejercicios resueltos de procesado de señal en FPGA

Jhon Jairo Cevallos
Hugo Moreno Avilés

DIRECCIÓN DE PUBLICACIONES

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

Ejercicios resueltos de procesado de señal en FPGAs

Panamericana Sur, kilómetro 1 ½
Instituto de Investigaciones
Dirección de Publicaciones Científicas
Riobamba, Ecuador
Teléfono: 593 (03) 2 998-200
Código Postal: EC0600155

Aval ESPOCH
Este libro se sometió a arbitraje bajo el sistema de doble ciego
(*peer review*)

Corrección y diseño:
La Caracola Editores

Impreso en Ecuador

CDU: 004 + 004.7
Ejercicios resueltos de procesado de señal en FPGAs
Riobamba: Escuela Superior Politécnica de Chimborazo
Dirección de Publicaciones, año 2020
146 pp. vol: 17,6 x 25 cm
ISBN: 978-9942-40-450-3
1. Informática
2. *Software*

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

PRÓLOGO

Querido Lector:

El libro que tienes en tus manos contiene una colección de problemas resueltos de Procesado de Señal en FPGA (Field Programmabe Gate Array). Estos problemas han sido propuestos a lo largo de la Asignatura de Procesado Digital de Señales en FPGA, dictada en el Master Universitario en Ingeniería de Sistemas Electrónicos en la Especialidad de Sistemas Electrónicos Digitales de la Universitat Politécnica de Valencia para su resolución por parte de los alumnos.

El objetivo principal que se persigue con este libro es ofrecer una herramienta a los estudiantes, docentes, investigadores y profesionales en general para que profundicen su conocimiento en el manejo y programación de FPGA, tomando en cuenta que el uso de estos dispositivos en el desarrollo de nueva tecnología ha ido incrementando en los últimos años, y es muy importante hoy en día tener el conocimiento suficiente y actualizado para poder implementar las nuevas soluciones tecnológicas sobre estos dispositivos de última gama y altas prestaciones.

INTRODUCCIÓN

El presente libro, Ejercicios Resueltos de Procesado de Señal en FPGA (Field Programmable Gate Array), está dirigido a estudiantes, docentes, investigadores y profesionales en general que tengan conocimiento del manejo de FPGA, programación del lenguaje HDL como el modelado de precisión finita, circuitos aritméticos y las diferentes arquitecturas que se pueden implementar sobre estos dispositivos. Este conocimiento es esencial para el aprendizaje, ya que se constituye en una herramienta complementaria para fortalecer el diseño y verificación de sistemas digitales.

Los ejercicios resueltos en el presente libro son el resultado de la investigación desarrollada en el Master Universitario en Sistemas Electrónicos, Especialidad de Sistemas Electrónicos Digitales; cursado en la Universitat Politécnica de Valencia.

Consta de seis capítulos. El Capítulo I comprende la resolución de ejercicios de DDS (Direct Digital Synthesis) y Mezcladores para el desarrollo de Sintetizadores Digitales de Frecuencia.

En el Capítulo II, se desarrollan ejercicios de Modelado en Precisión Finita para codificar diferentes formatos numéricos.

En el Capítulo III, se desarrollan ejercicios de Circuitos Aritméticos, su segmentación y el calculo de la Frecuencia Máxima de Funcionamiento.

En el Capítulo IV, se desarrolla un ejercicio del Filtro CIC (Cascaded Integrator Comb), su Implementación y Cuantificación.

En el Capítulo V, se desarrollan ejercicios de Arquitecturas Hardware Secuencia-

les, su Implementación, Cuantificación, Segmentación y Frecuencia Máxima de Funcionamiento.

Finalmente el Capítulo VI, comprende una Miscelánea de Ejercicios, donde se combinan todas las temáticas anteriores.

CAPÍTULO 1
DDS (DIGITAL DIRECT SYNTHESIS) Y MEZCLADORES

Síntesis Digital Directa (Direct Digital Synthesis, DDS) es el proceso de generar directamente en el dominio digital las portadoras u otras señales de referencia requeridas en comunicaciones.

Para el diseño basado en tablas se debe considerar:

- Las muestras de un periodo de la sinusoide se almacenan en una tabla.
- Un contador genera una rampa y selecciona las posiciones de la tabla, obteniendo a la salida de ésta el valor correspondiente de la señal sinusoidal.
- La frecuencia de reloj del circuito es fija.

1.1. EJERCICIO 1

Un DDS está diseñado con los siguientes parámetros:

- Frecuencia de Reloj: f_{clk}= 200 MHz
- Ancho del acumulador: $M = 28\ bits$
- Número de direcciones de la tabla: $L = 14\ bits$
- Ancho de palabra de la tabla: $W = 13\ bits$

Calcule:

1. Frecuencia máxima sintetizable f_{max}.

$$f_{max} = \frac{f_{clk}}{2} = \frac{200\ MHz}{2} = 100\ MHz$$

2. Resolución de frecuencia Δf.

$$\Delta f = f_{min} = \frac{f_{clk}}{2^M} = \frac{200MHz}{2^{28}}$$

$$\Delta f = 0.745058Hz$$

3. Rango Dinámico Libre de Espurios **SFDR**.

$$SFDR = 6.02L - 3.92dB$$

$$SFDR = 6.02(14) - 3.92dB$$

$$SFDR = 80.36dB$$

4. Relación Señal Ruido **SNR**.

$$SNR = 6.02W + 1.76dB$$

$$SNR = 6.02(13) + 1.76dB = 80.02dB$$

5. El paso del acumulador **P** para generar una oscilación de $23.5MHz$. Para realizar este cálculo tenga en cuenta que **M** tiene que ser un valor codificable con **P** bits, por ello utilice la función round de Matlab para redondear el valor que obtenga a su valor entero más cercano. $f_o = 23.5MHz$

$$P = \frac{f_o * 2^M}{f_{clk}} = \frac{23.5 \times 10^6\ Hz}{200 \times 10^6\ Hz}$$

$$P = 31541166.08$$

Matlab$\gg$

$$round(P)(2^{-M})$$

$$round(31541166.08)(2^{-28}) = 0.1175$$

6. Frecuencia que se genera cuando se utiliza el paso calculado en el numeral 5.

$$f = \frac{P * f_{clk}}{2^M} = \frac{31541166.08 * (200 \times 10^6)}{2^{28}}$$

$$f = 23499999.94\ Hz$$

7. Error que se comete al generar la frecuencia con el paso calculado en el numeral 5. **¿Es menor que la resolución de frecuencia?**

$$\textbf{Resolución} \gg \ = 0.745058 Hz$$

$$\textbf{Error} \gg \ = 0.0596047 Hz$$

$$8\,\%\ \Delta f\ menor$$

8. El paso que hay que utilizar para generar una frecuencia mayor y lo más cercana a la calculada en el numeral 6. ¿Cuál es la frecuencia que se genera con este nuevo paso? ¿Qué relación existe entre la frecuencia calculada en el numeral 6, la calculada en esta sección y la resolución de frecuencia?

$$P = 31541166 + 1 = 31541167$$

$$f_o = \frac{P * f_{clk}}{2^M} = \frac{31541167 * (200 \times 10^6)}{2^{28}}$$

$$f_o = 23500000.69\ Hz$$

$$\textbf{Resolución} \gg \ = 0.745058\ Hz$$

$$\textbf{Error} \gg 0.6854534\ Hz$$

$$Error\ <\ Resolucion$$

1.2. EJERCICIO 2

Se pretende diseñar un DDS para alcanzar las siguientes especificaciones:

- Frecuencia de Reloj: f_{clk}= 100 MHz
- Resolución de la frecuencia: $\Delta f = 1\ Hz$
- Rango dinámico libre de espurios: **SFDR** $> 85\ dB$
- Relación Señal Ruido **SNR** $> 85\ dB$

1. Determine los parámetros del DDS: ancho del acumulador, **M**, número de direcciones de la tabla, **L**, y ancho de palabra de la tabla, **W**.

$$Ancho\ Acumulador \gg M$$

$$\Delta f = \frac{f_{clk}}{2^M}$$

$$2^M = \frac{f_{clk}}{\Delta f}$$

$$log 2^M = log_2 \frac{f_{clk}}{\Delta f}$$

$$log 2^M = log_2 \frac{100 \times 10^6\ Hz}{1\ Hz}$$

$$M = 26.5754\ bits$$

$$\#Direccion\ Tabla \gg L$$

$$SFDR = 6.02L - 3.92dB$$

$$85 < 6.02L - 3.92dB$$

$$L < 6.02L - 3.92dB$$

$$L > 14.771 \approx 15\ bits$$

$$\#Ancho\ Palabra \gg W$$

$$SNR = 6.02W + 1.76dB$$

$$85 < 6.02W + 1.76dB$$

$$W > 13.827 \approx 14\ bits$$

2. Calcule el paso del acumulador, **P**, para generar una oscilación de **10.7 MHz.**

$$P = \frac{f_o * 2^M}{f_{clk}} = \frac{10.7x * 10^6 Hz * 2^{27}}{100x * 10^6 Hz}$$

$$P = 14361296.9$$

Matlab≫

$$round(P)(2^{-M})$$

$$round(14361296.9)(2^{-27}) = 0.1070$$

3. Indique qué frecuencia se genera cuando se utiliza el paso calculado en la sección anterior.

$$f = \frac{P * f_{clk}}{2^M} = \frac{14361297 - (100x * 10^6)}{2^{27}}$$

$$P = 10700000.08\ Hz$$

1.3. EJERCICIO 3

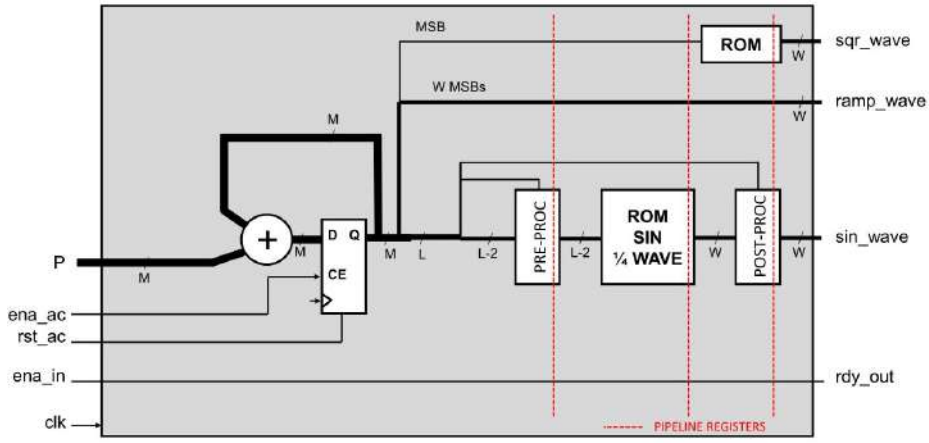
Se pretende modelar un sintetizador de frecuencias DDS con lenguaje Verilog parametrizable, que genere 3 formas de onda: rampa, cuadrada y sinusoidal **(Figura 1.1)**. El modelo del DDS deberá ser sintetizable y podrá ser sintetizable e implementado en un dispositivo FPGA funcionando a una frecuencia de reloj mayor de **125 MHz.**

Las **especificaciones** del DDS son:

- Acumulador parametrizable de **M bits.**
- Truncado de fase parametrizable a **L bits** para direccionar la etapa de generación de la forma de onda sinusoidal.
- Salidas de las diferentes formas de onda parametrizadas con **W bits.**
- El acumulador dispone de una señal de clock *enable* **ena_ac** para indicar en qué ciclos está activado y una señal de *reset* síncrono **rst_ac**, ambas activas a nivel alto.

- Señal de entrada **ena_in** (binaria) para indicar el ciclo en el que se introduce un valor válido en la entrada **P.**
- Señal de salida **rdy_out** (binaria) indicando que por las **salidas_wave** están saliendo datos válidos.
- El DDS deberá alcanzar una frecuencia de funcionamiento mínima de **125 MHz** al implementarse en el dispositivo FPGA Cyclone IV EP4CE115F29C7, cuando se configura con los parámetros **M=27**, **L=15** y **W = 16 bits.**

Figura 1.1. Esquema de Implementación del DDS



El diseño debe construir las señales cuadrada, rampa y seno (a partir de tablas dadas almacenadas en ROM). La solución planteada es un diseño modular dividiendo funciones en bloques que posteriormente se instanciarán en un módulo denominado DDS_test.

Se empieza con el diseño del bloque del **acumulador** de entrada **P** de **M** bits con *enable* **ena_ac** y *reset* **rst_ac** a nivel alto, cuyo código se presenta a continuación:

```
module acumulador
#(parameter M=16)
```

```
(
input rst_ac, ena_ac,
input clk,
input [M-1:0] p,
output reg [M-1:0] q
);

always @ (posedge clk)
if (rst_ac)
q=0;
else if (ena_ac)
q=q+p;
endmodule
```

Se continua con diseño de la señal seno que se divide en 3 bloques:

Preprocesado: evaluación del segundo MSB de la salida del acumulador (M bits) para el direccionamiento de la memoria **ROM** que almacena el primer cuarto del periodo de la señal seno a construir. La salida serán los **L** MSB de **M** restando los **2 MSB** de los mismos, invirtiendo los bits en la dirección en caso de que el bit evaluado sea **1**.

```
module pre_proc
#(parameter M=16,
parameter L=6)
(
input clk,
input [M-1:0] l,
output reg [L-3:0] rom_addr
);

always @ (posedge clk)
if (!l[M-2])
```

```
rom_addr=l[M-3:M-L]
else
rom_addr= ~ l[M-3:M-L];
endmodule
```

Lectura de memoria ROM: a partir de los parámetros **W** y **L** se leen distintas memorias **.txt** en función a la dirección dada por el preprocesado. La salida será la entrada del siguiente módulo, el posprocesado. Código proporcionado por el profesor.

Posprocesado: evaluación del **MSB** de la salida del acumulador para la construcción de la señal seno. La Salida será el valor de la señal seno, haciendo el Complemento a **2** en caso de que el bit evaluado sea **1**.

```
module post_proc
#(parameter M=16,
parameter W=16)
(
input clk,
input [M-1:0] l,
input [W-1:0] w,
output reg signed [W-1:0] sin_wave
);

always @ (posedge clk)
if(!l[M-1])
sin_wave=w;
else
sin_wave=~ w+1;
endmodule
```

Posteriormente, se construye el módulo de la **señal rampa**, que será gene-

rada a partir de los **W** de **MSB** de los **M** bits del acumulador.

```
module ramp_wave
#(parameter M=20,
parameter W=16)
(
input clk,
input [M-1:0] q,
output reg signed [W-1:0] ramp_wave
);

always @ (posedge clk)
ramp_wave=q[M-1:M-W];
endmodule
```

Luego se construye el módulo de la **señal cuadrada**, que será generada evaluando el **MSB** de **M** y asignando el valor máximo o mínimo del rango de **W** en función de si el bit evaluado vale 0 o 1.

```
module sqr_wave
#(parameter M=16,
parameter W=16)
(
input clk,
input [M-1:0] l,
output reg signed [W-1:0] sqr_wave
);
//assign val_sqr_wave=rom_val;

always @ (posedge clk)
if(!l[M-1]) //d3 = 0?
sqr_wave=2**(W-1)-1;
```

```
else
sqr_wave=-2**(W-1);
endmodule
```

Finalmente se instanciaron todos los módulos en el **DDS_test** teniendo en cuenta que, para que las 3 señales generadas no tengan ningún desfase entre sí, se consideran los retardos que tiene cada una de ellas (añadiendo registros previos a la conformación de las señales rampa y cuadrada, además de sobre la entrada de habilitación **ena_in** y la salida **rdy_out**.

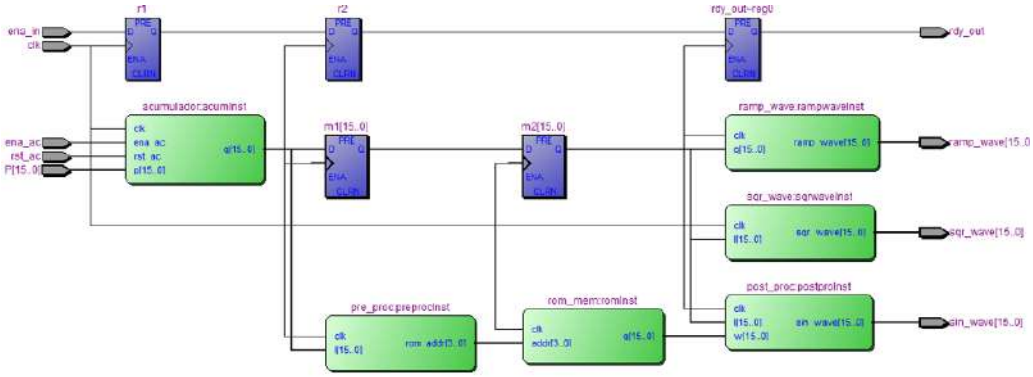
```
module DDS_test
#(parameter M=16,
parameter L=6,
parameter W=16)
(
input [M-1:0] P,
input rst_ac, ena_ac, ena_in,
input clk,
output signed [W-1:0] sqr_wave,
output signed [W-1:0]
ramp_wave,
output signed [W-1:0] sin_wave,
output reg rdy_out
);

reg [M-1:0] m1,m2;
reg r1,r2;
wire [M-1:0] m;
wire [L-3:0] rom_addr;
wire [W-1:0] rom_val;
```

```
acumulador #(.M(M)) acumInst
(
.rst_ac(rst_ac),
.ena_ac(ena_ac),
.clk(clk),
.p(P),
.q(m)
);

pre_proc #(.M(M),.L(L))
preprocInst
(
.clk(clk),
.l(m),
.rom_addr(rom_addr)
);

rom_mem
#(.DATA_WIDTH(W),.ADDR_WIDTH(L-2)) romInst(
.addr(rom_addr),
.clk(clk),
.q(rom_val)
);

post_proc #(.M(M),.W(W))
postproInst
(
.clk(clk),
.l(m2),
.w(rom_val),
```

```
.sin_wave(sin_wave)
);

sqr_wave #(.M(M),.W(W))
sqrwaveInst
(
.clk(clk),
.l(m2),
.sqr_wave(sqr_wave)
);

ramp_wave #(.M(M),.W(W))
rampwaveInst
(
.clk(clk),
.q(m2),
.ramp_wave(ramp_wave)
);

always @ (posedge clk)
begin
m1=m;
m2=m1;
r1=ena_in;
r2=r1;
rdy_out=r2;
end
endmodule
```

Como se muestra en la **Figura 1.2**, se aprecia todo el sistema construido, los módulos empleados y los registros añadidos previo a la conformación de las señales para obtener a la salida las señales sin ningún retraso temporal.

Figura 1.2. Diagrama de Bloques DDS



Se procede a la simulación del diseño parametrizado con **M=16**, **L=6** y **W=16** en Modelsim, tal como se muestra en la **Figura 1.3** donde se verifica que las señales generadas no tienen ningún desfase entre sí y se compara con las señales generadas en el Simulink **Figura 1.4** con los mismos parámetros.

Figura 1.3. Señales generadas Modelsim (M=16, L=6 y W=16)

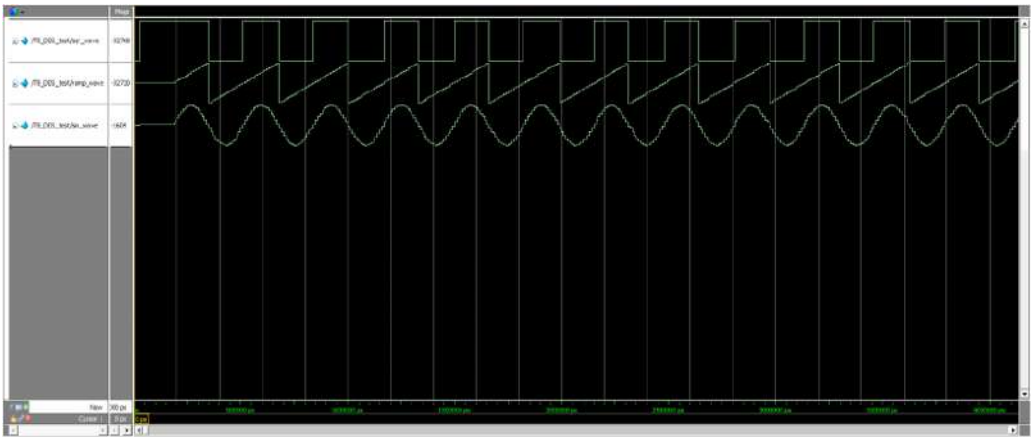
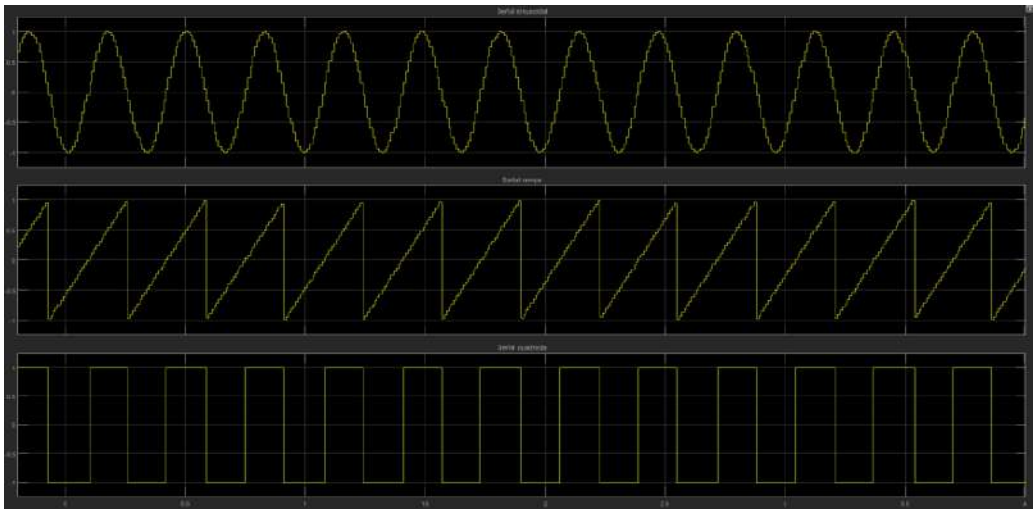


Figura 1.4. Señales generadas Simulink (M=16, L=6 y W=16)



Como se puede apreciar, las señales generadas son similares, además se observa que el complemento a 2 de los valores de la sinusoide se ha realizado co-

rrectamente.

CAPÍTULO 2
MODELADO DE PRECISIÓN FINITA

Un correcto Modelado de Precisión Finita permite:

1. Computar algoritmos en un dispositivo FPGA.
   - Implementación de operadores con tamaños de palabra fijos.
2. Reducir el tamaño de los operadores afecta a la implementación HW:
   - Reducción área.
   - Mejorar la frecuencia de operación.
   - Reducir el consumo de potencia.
3. Es necesario evaluar el comportamiento con precisión finita.
   - El algoritmo debe mantener las prestaciones deseadas.
4. Disponer de un modelo de precisión finita facilita la verificación del *hardware*:
   - Proporciona vectores de prueba con los cuales comparar los modelos HDL con los impementados.

2.1. EJERCICIO 1

¿Qué formato numérico se necesita para representar la señal $x(n) = 10\,\text{sin}(2\pi n\frac{f_o}{f_s})$ con una resolución de al menos 0.01?

Resolución$\gg Q = 0.01 \;\; Q = 2^b$

$$log 2^b = log\, 0.01$$

$$b = log_2\, 0.01$$

$$b = 6.64 \approx 7\, bits$$

$$v_{max} = 10\, V \qquad v_{min} = -10\, V$$

$$\textbf{Rango Dinámico} \gg \frac{v_{max} - v_{min}}{2^{-b}} = 2^N - 1$$

$$\frac{10 - (-10)}{2^{-7}} + 1 = 2^N$$

$$N = log_2 \left(\frac{20}{2^{-7}} + 1 \right)$$

$$N = 11.32 \approx 12\, bits$$

$$\textbf{Formato} \gg [12 : 7]$$

2.2. EJERCICIO 2

Dado el formato numérico **[10,8]** $([N, b])$ con signo representado en complemento a dos:

1. ¿Cuál es el rango de valores representables?

$$\textbf{Resolución} \gg Q = 2^{-b} \;\; Q = 2^{-8}$$

$$\textbf{Rango:}\; [-2^{N-1} * Q : (2^{N-1} - 1) * Q]$$

$$[-2^9 * 2^{-8} : (2^9 - 1) * 2^{-8}]$$

$$\textbf{Rango} \gg [-2 : 1.90609375]$$

2. ¿Qué resolución se puede alcanzar?

$$\textbf{Resolución} \gg Q = 2^{-b} \;\; Q = 2^{-8}$$

$$3.90625 \times 10^{-3}$$

3. ¿Qué error se comete al codificar el número **A=9/7** con dicho formato?

   # Entero$\gg$ $\frac{7}{7}$ $sin\ error$

   # Decimal$\gg$ $\frac{2}{7}$

   # Decimal Codificado$\gg$ $\#Decimal * 2^8$

   $$\frac{2}{7} * 2^8 = 73.142857$$

   # Codificado$\gg$ $73 * 2^{-8} = 0.28515625$

   Error$\gg$ $\frac{2}{7} - \#\,Codificado = 0.000558$

4. ¿Existe algún número dentro del rango cuyo error al codificarlo con dicho formato sea mayor que su resolución?

   No es posible, porque, de ser así, el número estaría codificado mal. El error máximo sería la resolución.

5. ¿Existe algún número dentro del rango que se pueda codificar de forma exacta (sin error)? Ponga algún ejemplo.

   Todos los números que se encuentren dentro de la rejilla se los puede codificar sin error.

2.3. EJERCICIO 3

¿Cuántos bits se necesitan para cuantificar una señal de 1 Vpp con una resolución de 1mV?

$$v_{max} = 0.5\ V \qquad v_{min} = -0.5\ V$$

Resolución$\gg Q = 0.001 \;\; Q = 2^{-b}$

b= $-\, log_2\, 0.001 = 9.9657 \;\approx 10\; bits$

Rango$\gg \frac{v_{max} - v_{min}}{2^{-b}} = 2^N - 1$

$$\frac{1}{2^{-10}} + 1 = 2^N$$

$$N = log_2 \left(\frac{1}{2^{-10}} + 1 \right) = 10.001\; bits$$

Formato$\gg [10 : 10]$

2.4. EJERCICIO 4

¿Qué valor se obtiene si se aplica el formato numérico [10,8] al valor 6.625?

$$\#Base\ 2 = 0110.1010$$

1. Con signo y desbordamiento (overflow) con envoltura (wrap) **(Valor máximo representable)**

$$Se\ elimina\ los\ bits\ superiores$$

$$6.625_{(0110.1010)} = -1.375_{(10.1010)}$$

2. ¿Y si el formato numérico realiza el desbordamiento con saturación?

$$Se\ asigna\ el\ valor\ mximo\ representable$$

$$(2^{N-1} * 2^b$$

$$(2^9 - 1) * 2^{-8} = 1.9960375$$

2.5. EJERCICIO 5

Se requiere realizar la suma de 100 datos codificados con formato [12,11] con signo. ¿Qué formato debe tener el valor resultante para que se pueda codificar sin que se produzca desbordamiento ni pérdida de precisión?

$S[N_s, s] \gg Sumatorio$
$D[N_d, d] \gg Datos$
$N = 100 \gg \#Datos$
$s = d \gg 11$

$$N_s = N_d + ceil(log_2\ 100) \gg\ 19$$

Formato$\gg$ **S**$[19 : 11]$

2.6. EJERCICIO 6

¿Con qué formato numérico habría que codificar el resultado de la multiplicación de dos datos con formato [18,17] con signo para que no haya pérdida de precisión en la operación?

$N_1[N_{n1}, n_1]$
$N_2[N_{n2}, n_2]$
$P[N_p, p]$

$$N_P = N_{n1} + N_{n2} = 18 + 18 \gg\ 36$$

$$p = n_1 + n_2 = 17 + 17 \gg\ 34$$

Formato$\gg$ **P**$[36 : 34]$

2.7. EJERCICIO 7

Se requiere realizar la operación multiplica y acumula para evaluar la suma de 500 productos $(Y = \sum A_i * B_i)$. Si los formatos numéricos de A_i y B_i son de $[8, 7]$ y $[10, 7]$ respectivamente, ¿Cuál es el formato numérico de la salida (Y) para que se calcule sin pérdida de precisión en las operaciones intermedias?

$A_i[N_a, a] \gg [8, 7]$
$B_i[N_b, b] \gg [10, 7]$
$S[N_s, s] \gg Sumatorio$
$N = 500 \gg \#Productos$
$s = a + b \gg 14$

$$N_s = N_a + N_b + ceil(log_2\ 500) \gg\ 27$$

Formato$\gg$ **S**$[27 : 14]$

2.8. EJERCICIO 8

Supóngase que un operador genera una salida A_{out} con formato numérico [25,15] con signo. La salida de dicho operador se quiere conectar a otro operador cuya entrada B_{in} dispone de 16 bits con formato fraccionario [16,15].

1. ¿Qué 16 bits de A_{out} hay que conectar al operando de entrada B_{in} ?

   Se conectan los 16 bits más significativos para no perder información relevante.

2. Indique cómo se modela con Matlab la obtención del operando de entrada B_{in} con formato [16,15] a partir de la salida A_{out}.

   $A_{out}[N_a, a] \gg [25, 15]$
   $B_{in}[N_b, b] \gg [16, 15]$
   $d = N_a - N_b \gg 9$

   $$A_{out_1} = A_{out} * 2^{-d} \gg A_{out} * 2^{-9}$$

   $Matlab \gg$

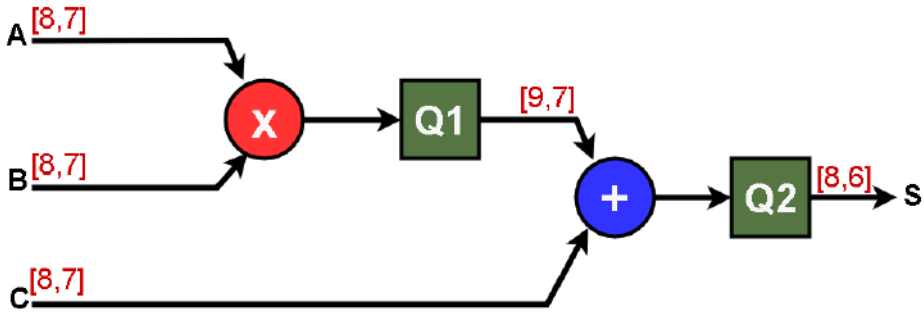
   $$B = floor(A_{out_1} * 2^{15}) * 2^{-15}$$

   $$B = floor(A_{out} * 2^{6}) * 2^{-15}$$

2.9. EJERCICIO 9

Realice el modelo de precisión finita del operador S=A·B+C, para los formatos numéricos con signo indicados en la **Figura 2.1**. Q1 y Q2 indican los truncados a la salida del multiplicador y del sumador, respectivamente.

Figura 2.1. Esquema Operador S=A*B+C



1. Utilizando la función floor de Matlab

$$M = A \times B\ [16, 14]$$

$$qm = quantizer([9.7], \text{“}fixe\text{”}, \text{“}wrap\text{”}, \text{“}floor\text{”})$$

$$Q1 = quantizer(qm, M)$$

$$S1 = Q1 + C[10, 7]$$

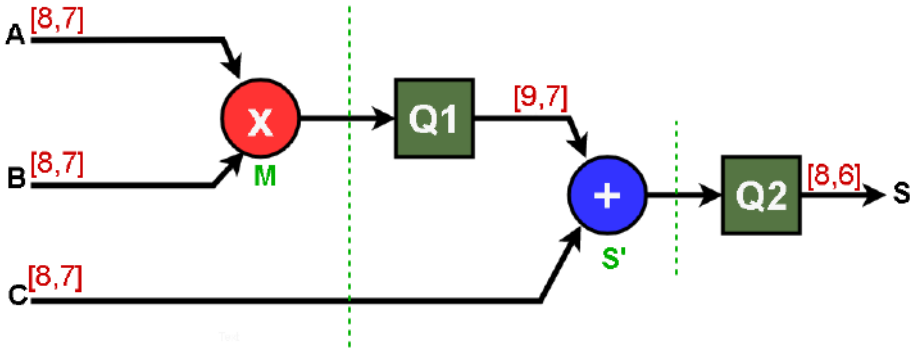
$$qs = quantizer([8.6], \text{“}fixed\text{”}, \text{“}wrap\text{”}, \text{“}floor\text{”})$$

$$Q1 = quantizer(qs, S1)$$

2. Código Verilog

```
module mult_add(
input signed [7:0] a, b, c,
input clk,
```

Figura 2.2. Modelo Precisión Finita Operador S=A*B+C




```
output signed [7 : 0] s,
);

wire signed [15:0] m;
reg signed [8:0] m_t;
reg signed [8:0] s_t;
reg signed [7:0] c_r;

assign m=a*b;
assign s=s_t[8:1];

always@(posedge clk)
begin
m_t<=m[15:7]
c_r<=c;
s_t<=m_t+c_r;
end
endmodule
```

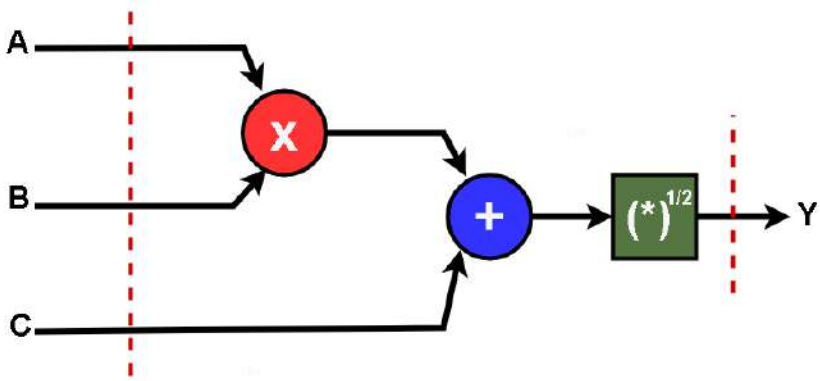
CAPÍTULO 3
CIRCUITOS ARITMÉTICOS

Los circuitos digitales síncronos siempre transfieren datos entre registros, los cuales trabajan a una señal de reloj determinada. Esto hace necesario e indispensable un correcto diseño de Circuitos Artiméticos, con el objetivo de llegar a la mayor frecuencia de trabajo, considerando como herramientas claves la segmentación del circuito para aumentar su rendimiento y reducir la latencia, lo que permitira implementar soluciones eficientes con la menor cantidad de recursos empleados.

3.1. EJERCICIO 1

Se dispone de un circuito para computar $Y = sqrt(A * B + C)$ implementado con el esquema que se muestra en la **Figura 3.1**. Los tiempos de propagación de los operadores aritméticos son $t_{mul} = 3.5ns$, $t_{add} = 1.5ns$ y $t_{sqrt} = 5.5ns$, y el tiempo de propagación de los *flip-flops* (FF) y de *set-up* son $t_{co} = 0.2ns$, $t_{su} = 0.05ns$, respectivamente.

Figura 3.1. Esquema Operador Y=sqrt(A*B+C)



1. Calcule la frecuencia máxima de funcionamiento del circuito.

$$Tp_{min} = t_{co} + t_p LCR + t_{su}$$

$$Tp_{min} = 0.2ns + (3.5ns + 1.5ns + 5.5ns) + 0.05ns$$

$$Tp_{min} = 10.75\ ns$$

$$F_{max} = \frac{1}{Tp_{min}} \gg \frac{1}{10.75 \times 10^{-9}} \gg 93.05\ MHz$$

2. Indique cuáles son los puntos óptimos de segmentación y la frecuencia máxima de funcionamiento del circuito segmentado.

$$Tp_1 = 3.5ns + 0.2ns + 0.05ns \gg 3.75\ ns$$

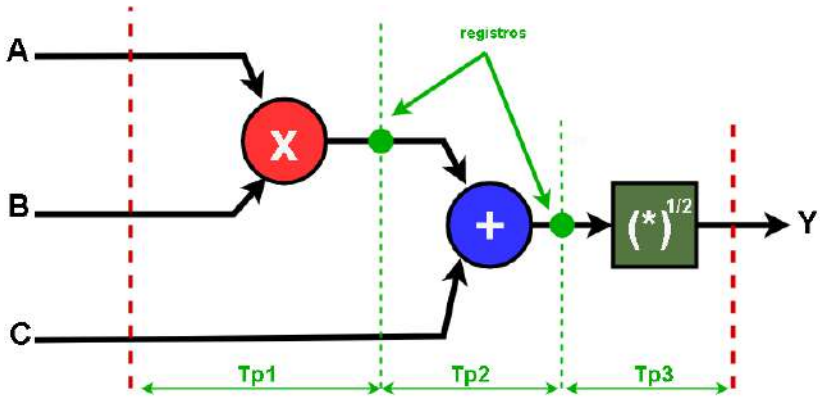
$$Tp_2 = 1.5ns + 0.2ns + 0.05ns \gg 1.75\ ns$$

$$Tp_3 = 5.5ns + 0.2ns + 0.05ns \gg 5.75\ ns$$

$$T = max(Tp_1, Tp_2, Tp_3) \gg 5.75\ ns$$

$$F_{max} = \frac{1}{T} \gg \frac{1}{5.75 \times 10^{-9} s} \gg 173.91\ MHz$$

Figura 3.2. Segmentación Operador Y=sqrt(A*B+C)



3. ¿Qué latencia, dada en ciclos de reloj, tiene el circuito segmentado?
   El primer dato es calculado después de 3 ciclos de reloj (latencia).

3.2. EJERCICIO 2

Se pretende conectar la señal A con formato numérico [5,0] a la señal B con formato [3,0], ambas con signo.

1. Indique qué valores numéricos se obtienen (en formato decimal y binario) si se utiliza saturación y envoltura (*wrapping*) en la conversión de formatos.**(Tabla 3.1)**

Tabla 3.1. Conversión de Formatos

A [5:0]	B [3:0]	
	Saturación	Envoltura*
$0\ (00000)_{2C}$	$0\ (000)_{2C}$	$0\ (000)_{2C}$
$3\ (00011)_{2C}$	$3\ (011)_{2C}$	$3\ (011)_{2C}$
$5\ (00101)_{2C}$	$3\ (011)_{2C}$	$-3\ (101)_{2C}$
$11\ (01011)_{2C}$	$3\ (011)_{2C}$	$3\ (011)_{2C}$
$13\ (01100)_{2C}$	$3\ (011)_{2C}$	$-4\ (100)_{2C}$
$-3\ (11101)_{2C}$	$-3\ (101)_{2C}$	$-3\ (101)_{2C}$
$-4\ (11101)_{2C}$	$-4\ (100)_{2C}$	$-4\ (100)_{2C}$
$-5\ (11011)_{2C}$	$-4\ (100)_{2C}$	$3\ (011)_{2C}$
$-11\ (10101)_{2C}$	$-4\ (100)_{2C}$	$-3\ (101)_{2C}$
$-13\ (10011)_{2C}$	$-4\ (100)_{2C}$	$3\ (011)_{2C}$
$-16\ (10000)_{2C}$	$-4\ (100)_{2C}$	$0\ (000)_{2C}$

* elimina los bits superiores

Como se puede observar en la **Tabla 3.1**, si se aplica *wrapping* (envoltura), se pierde información al eliminar los bits superiores de la señal.

2. En la **Figura 3.3** se muestra un esquema del circuito de conexión entre A y B aplicando saturación. ¿Qué bits de la señal A hay que utilizar para seleccionar el valor a conectar en la señal B?
   Como se muestra en la **Figura 3.3**, se conectan los **3 bits** superiores de la señal **A** para conectar con la señal **B.**

3. Escriba la tabla de verdad de la función Lógica de selección **Tabla 3.2**, que tiene como entradas los bits de A indicados en el numeral 2 y como salida los dos bits de selección del multiplexor: SEL [1:0].

$$SAT+ = 011 \qquad SAT- = 100$$

Considerando una saturacion de bits positiva **SAT+** y negativa **SAT-**, se emplea el seleccionador **SEL** o directamente se muestran los **3 bits** a la salida.

Figura 3.3. Esquema de conexión

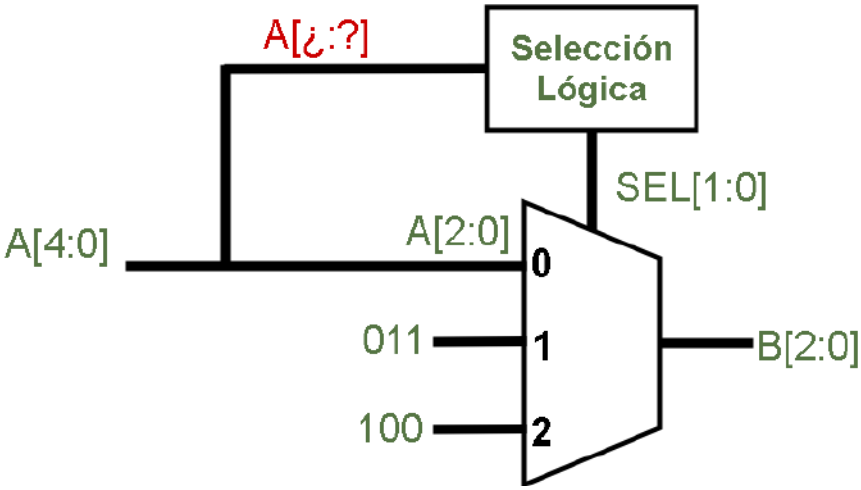


Tabla 3.2. Tabla de Verdad-Función SEL

A [4:2]	SEL [1:0]
# > SAT+	1
# < SAT-	2
SAT- > # >SAT+	A [2:0]

3.3. EJERCICIO 3

Se requiere realizar la suma de dos operandos, uno con formato [8,7] y el otro con formato [5,2], ambos con signo.

1. ¿Qué formato tendrá el resultado?
   $N_1[N_a, a] \gg [8, 7] \gg 1\ bit\ entero$
   $N_2[N_b, b] \gg [5, 2] \gg 3\ bits\ enteros$

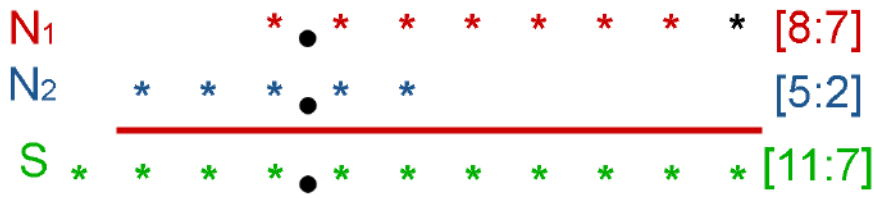
   Se necesitan 4 bits enteros con el mismo número de bits decimales.

   $$S = N_1 + N_2 \gg [11 : 7]$$

2. ¿Qué tamaño tiene el sumador? ¿Cuántos LE (Logic Elements) se requieren para su implementación en un dispositivo FPGA Cyclone IV? **(Figura 3.4)**

   El sumador tiene un tamaño S $\gg [11 : 7]$

Figura 3.4. Cuantificación Sumador [11:7]



Se requieren 11 LE para implementar el sumador en un dispositivo FPGA Cyclone IV.

3. Escriba el código Verilog del sumador.

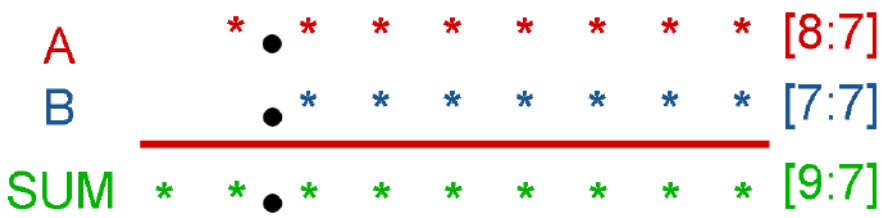
```
module sumador (
input signed [7:0] N1,
input signed [4:0] N2,
input clk,
output signed [10:0] S
);

assign S= N1+(N2<<<5);

always @(posedge clk)
S<= N1+(N2<<<5);
endmodule
```

3.4. EJERCICIO 4

Escriba el código Verilog de un sumador SUM=A+B que opera con un operando con signo en complemento a dos S y el otro sin signo U según los formatos **(Figura 3.5)** y **(Figura 3.6).**

CASO A: **A** $\gg S[8,7]$ **B** $\gg U[7,7]$ **SUM** $\gg S[9,7]$

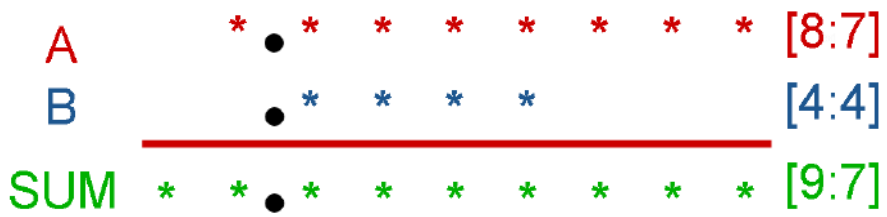
Figura 3.5. Cuantificación Sumador [9:7] Caso A



```
module caso_A (
input signed [7:0] A,
input [6,0] B,
input clk,
output signed [8:0] SUM
);
assign SUM=A+B;
endmodule
```

CASO B: A $\gg S[8,7]$ **B** $\gg U[4,4]$ **SUM** $\gg S[9,7]$

Figura 3.6. Cuantificación Sumador [9:7] Caso B



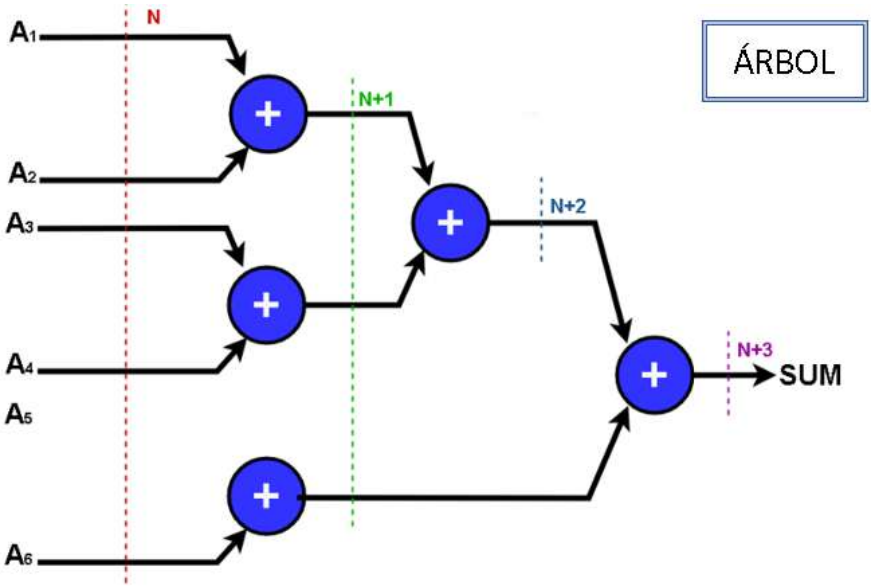

```
module caso_B (
input signed [7:0] A,
input [3,0] B,
input clk,
output signed [8:0] SUM
);

always @(posedge clk)
SUM <= A+(B<<<3);
endmodule
```

3.5. EJERCICIO 5

Dibuje el esquema de un sumador en árbol y en cascada de 6 operandos SUM=A1+A2+A3+
A4+A5+A6.

1. Suponga que todos los operandos son de N= 8 bits indique el crecimiento de los datos a lo largo del circuito.**(Figura 3.7)** y **(Figura 3.8).**

Figura 3.7. Sumador en Árbol - 6 operandos

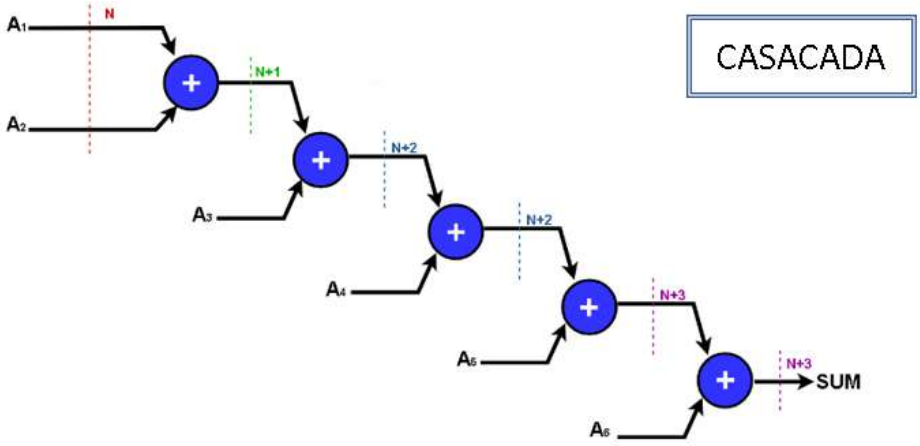


2. Suponga que el formato de los operandos es S[8,7] ¿cuál es el formato numérico de la salida SUM?
   En los dos casos, se ve un crecimiento de 3 bits en la parte entera por lo que el formato numérico de la salida sería **SUM [11,7].**

3. ¿Cuántos LE de un dispositivo Cyclone IV se requieren para implementar el sumador?
   LE=$\#\ sumadores(n\ bits) + \#\ bits\ acarreo$

$$\textbf{LE=}5(8) + 8 = 48LE$$

Figura 3.8. Sumador en Cascada - 6 operandos



Se requieren 48 LE para implementar el sumador en la Cyclone IV.

3.6. EJERCICIO 6

Escriba el código Verilog de un sumador en árbol del ejercicio anterior en las siguientes condiciones:

1. Salida SUM computada sin pérdida de precisión.

```
module sum_arbol (
input signed [7:0] A1, A2,A3, A4, A5, A6,
input clk,
output signed [10:0] SUM
);

assign SUM=((A1+A2)+(A3+A4)+(A5+A6));

endmodule
```

2. Operandos con formato S[8,7] y salida SUM con formato S[8,4]

```
module sum_arbol (
input signed [7:0] A1, A2,A3, A4, A5, A6,
```

```
input clk,
output signed [7:0] SUM
);
wire signed [10:0] sum_t;
assign sum_t=((A1+A2)+(A3+A4)+(A5+A6));

always @ (posedge clk)
SUM<= sum_t[10:3];
endmodule
```

3. ¿Cómo se modela con Matlab el sumador del caso anterior?
   sum_t=((A1+A2)+(A3+A4)+(A5+A6));

$$\textbf{SUM} = sum\_t * 2^{-3}$$

$$\textbf{SUM} = floor(sum\_t * 2) * 2^{-4}$$

3.7. EJERCICIO 7

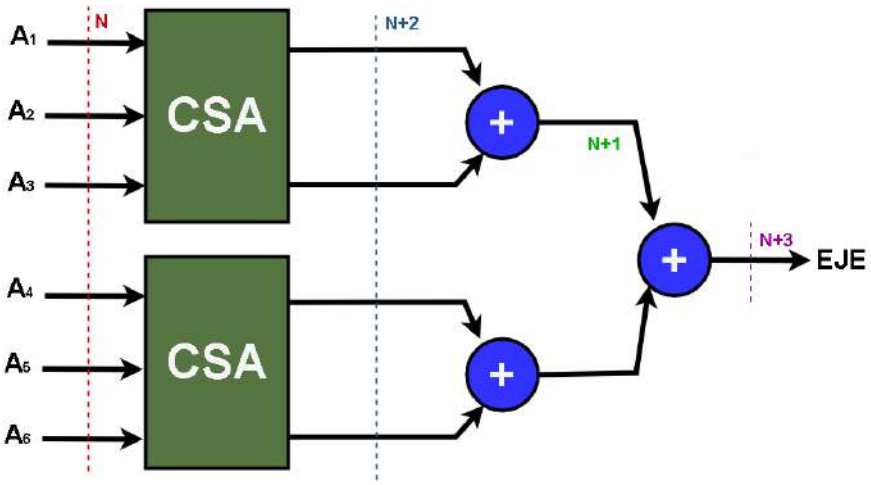
Suponga que dispone de un dispositivo Cyclone V que tiene la capacidad de implementar sumadores de 3 operandos (ternary adders) utilizando la estrategia CSA+RCA. Dibuje el esquema de un sumador en árbol de 6 operandos (SUM=A1 + A2 + A3 + A4 + A5 + A6).

1. Suponga que todos los operandos son de 8 bits e indique el crecimiento de los datos a lo largo del circuito.**(Figura 3.9).**

2. Suponga que el formato de los operandos es S[8,7] ¿Cuál es el formato numérico de la salida **SUM**?
   Se ve un crecimiento de 3 bits en la parte entera por lo que el formato numérico de la salida sería **SUM [11,7].**

3. ¿Cuántas LUT de un dispositivo **Cyclone V** se requieren para implementar el sumador?
   Se requieren 11 LUT para implementar el sumador en la Cyclone V.

Figura 3.9. Sumador 6 operandos – Cyclone V



3.8. EJERCICIO 8

El dispositivo FPGA Cyclone IV dispone de multiplicadores de 18x18 bits que se pueden configurar para que operen sin signo y con signo en complemento a dos. Dibuje el esquema de implementación de un multiplicador de 25x18 bits obtenido a partir de los multiplicadores de 18x18 disponibles en el dispositivo. Indique los tamaños de palabra en cada punto del circuito.

$$\mathbf{A}=[A_H : A_L] \qquad A_H, B_H \gg 7\ bits$$

$$\mathbf{B}=[B_H : B_L] \qquad A_L, B_L \gg 18\ bits$$

$$A_L, B_L \gg 18\ LSBs\ (unsigned)$$

$$A_H, B_H \gg 7\ MSBs\ (signed)$$

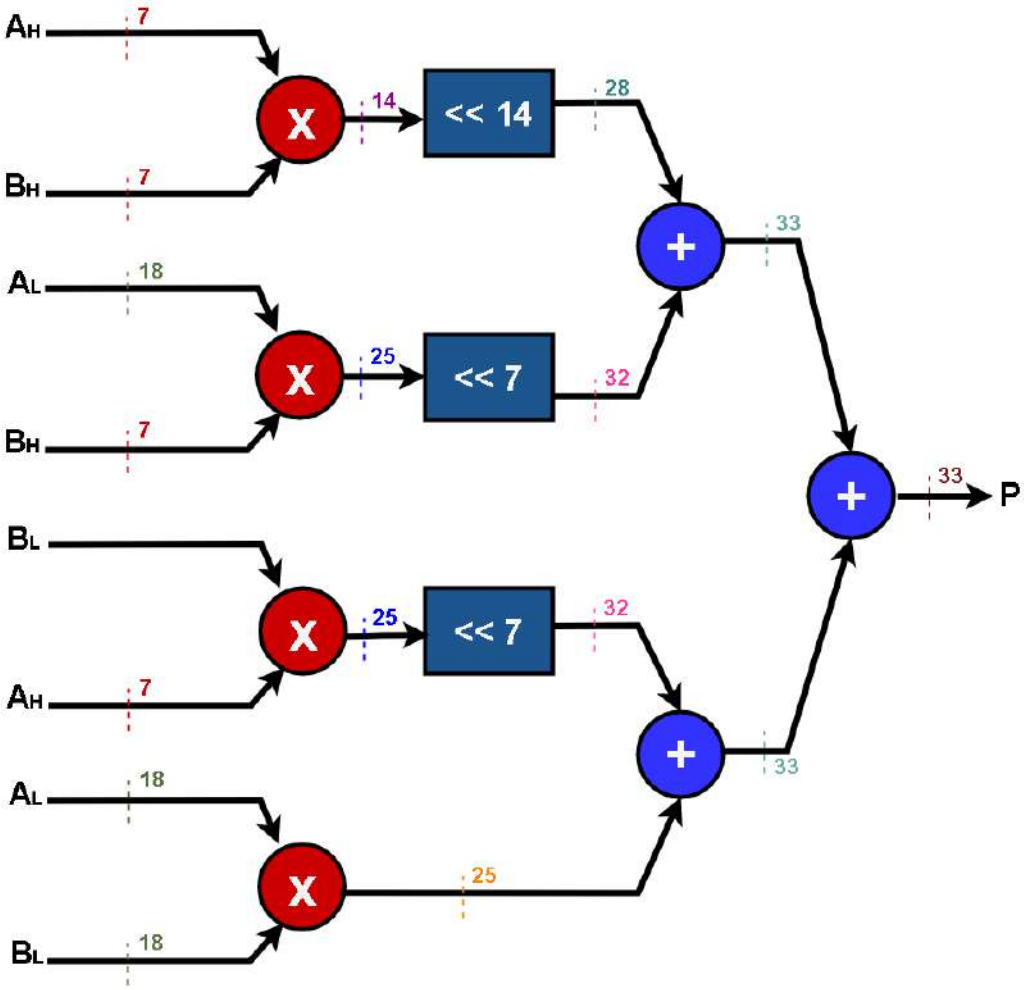
$$P = A * B \gg (A_H * 2^7 + A_L) * (B_H * 2^7 + B_L)$$

$$P = A_H * B_H * 2^{14} + A_L * B_H * 2^7 +$$

$$A_H * B_L * 2^7 + A_L * B_L$$

En la **Figura 3.10**, se puede observar que los multiplicadores disponibles en el dispositivo son utilizados para operar los numeros divididos en los 18 bits menos significativos sin signo LSB y los 7 bits mas significativos con signo MSB.

Figura 3.10. Esquema Multiplicador 25x18 bits

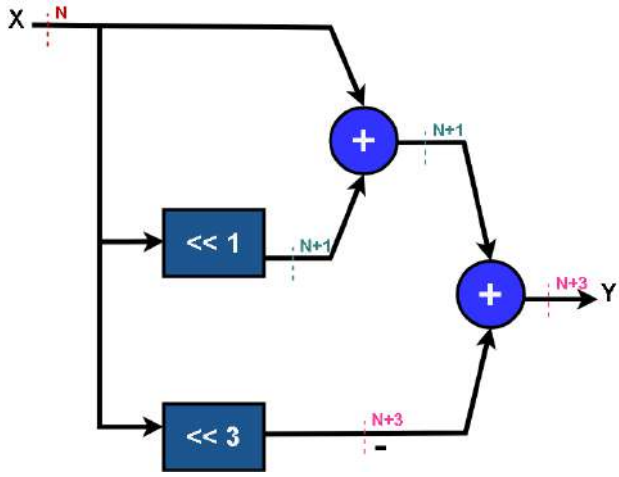


3.9. EJERCICIO 9

Se desea implementar un circuito que multiplique el dato de entrada **X** con formato [8,7] con signo por un valor constante $K = -5 \gg (1011)_{2C}$.

1. Dibuje su esquema de implementación utilizando sumadores e indique los tamaños de palabra en las conexiones intermedias del circuito **(Figura 3.11)**.

   Figura 3.11. Esquema multiplicador constante K entera



2. Teniendo en cuenta el formato de la entrada y el valor de la constante indique el formato de salida del multiplicador.
   El crecimiento de los datos depende del valor de la constante $K (log_2 5 = 3)$, el formato a la salida será $Y \gg S[11, 7]$.

3. Escriba el código Verilog del multiplicador implementado.

```
module mult_const(
input signed [7:0] X,
input clk,
output signed [10:0] Y
);
```

```
assign Y=(X+(X<<<2)- (X<<<3));
endmodule
```

3.10. EJERCICIO 10

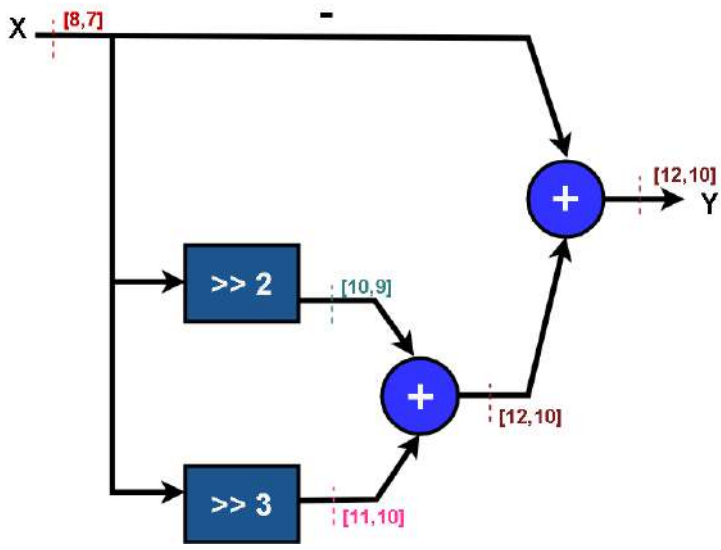
Se desea implementar un circuito que multiplique el dato de entrada X con formato [8,7] con signo por un valor constante $K = -0.625 \gg (1.011)_{2C.}$

1. Dibuje su esquema de implementación utilizando sumadores e indique los tamaños de palabra en las conexiones intermedias del circuito **(Figura 3.12)**.

$$K = (0.625)_{10} \gg (1.011)_{2C} \gg 2^{-3} + 2^{-2} - 1$$

$$Y = K * X \gg 2^{-3} * X + 2^{-2} * X - X$$

Figura 3.12. Esquema multiplicador constante K decimal



2. Teniendo en cuenta el formato de la entrada y el valor de la constante indique el formato de salida del multiplicador.
   Los datos crecen en la parte fraccionaria por lo que el formato a la salida será $Y \gg S[12, 10]$.

3. Escriba el código Verilog del multiplicador implementado.

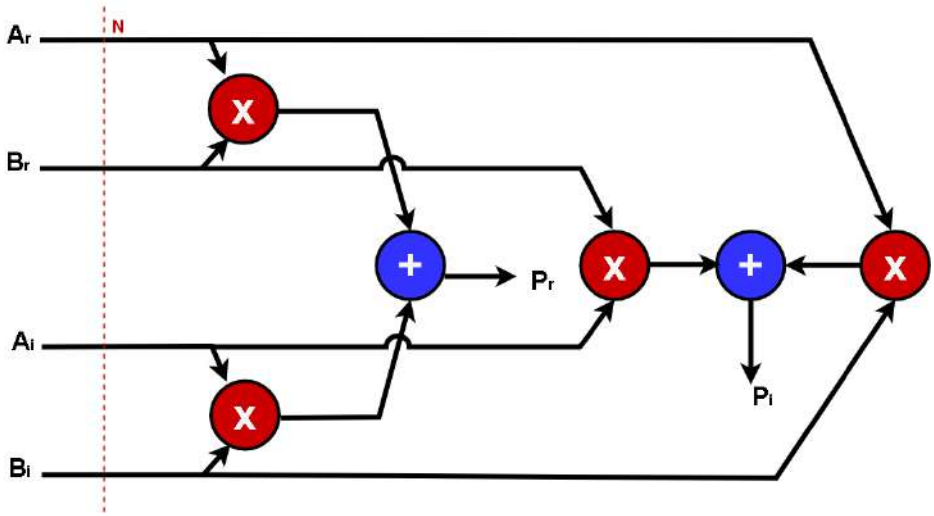
```
module mult_const_2(
input signed [7:0] X,
input clk,
output signed [10:0] Y
);
assign Y=((X>>>3)+ (X>>>2)-X);
endmodule
```

3.11. EJERCICIO 11

Se dispone de un dispositivo FPGA con multiplicadores embebidos y se requiere realizar la operación de multiplicación de números complejos $A = A_r + jA_i$ por $B = B_r + jB_i$ para dar como resultado $P = P_r + jP_i$.

1. Dibuje el esquema del circuito multiplicador complejo con entradas $A_r,\ A_i,\ B_r,\ B_i$, y salidas $P_r,\ P_i$. **(Figura 3.13)**

Figura 3.13. Esquema Multiplicador Números Complejos



A=$A_r + jA_i$

B=$B_r + jB_i$ $\qquad\qquad$ $j^2 = 1$

P=$P_r + jP_i$

$$\mathbf{P}=(A_r + jA_i) * (B_r + jB_i)$$

$$\mathbf{P}=A_r * B_r + jA_r * B_i + jA_i * B_r + j^2 A_i * B_i$$

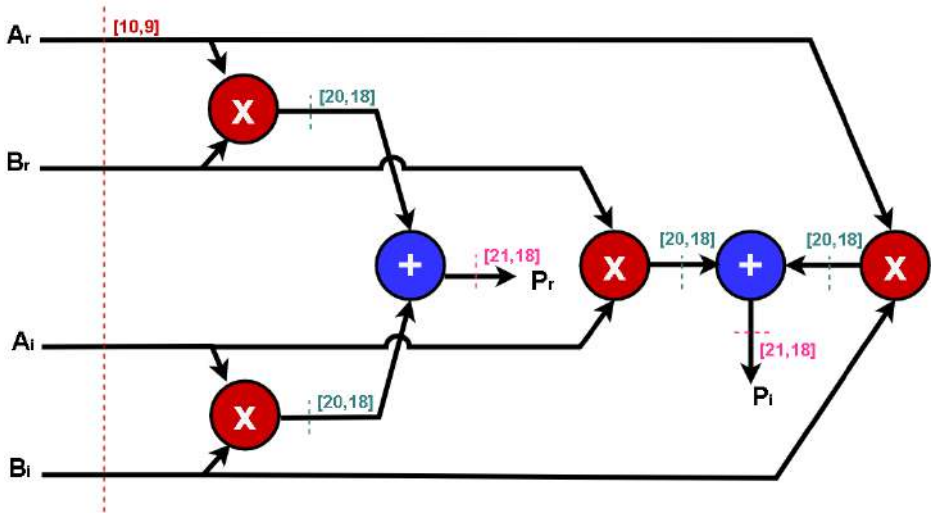
$$\mathbf{P}=A_r * B_r - A_i * B_i + j(A_r * B_i + A_i * B_r)$$

$$P_r = A_r * B_r - A_i * B_i$$

$$P_i = j(A_r * B_i + A_i * B_r)$$

2. Suponga que las entradas A_r, A_i, B_r y B_i tienen un formato numérico [10,9] con signo. Indique los formatos numéricos en todos los puntos del multiplicador complejo para que no haya pérdida de precisión en el resultado **(Figura 3.14)**.

Figura 3.14. Cuantificación Multiplicador Números Complejos



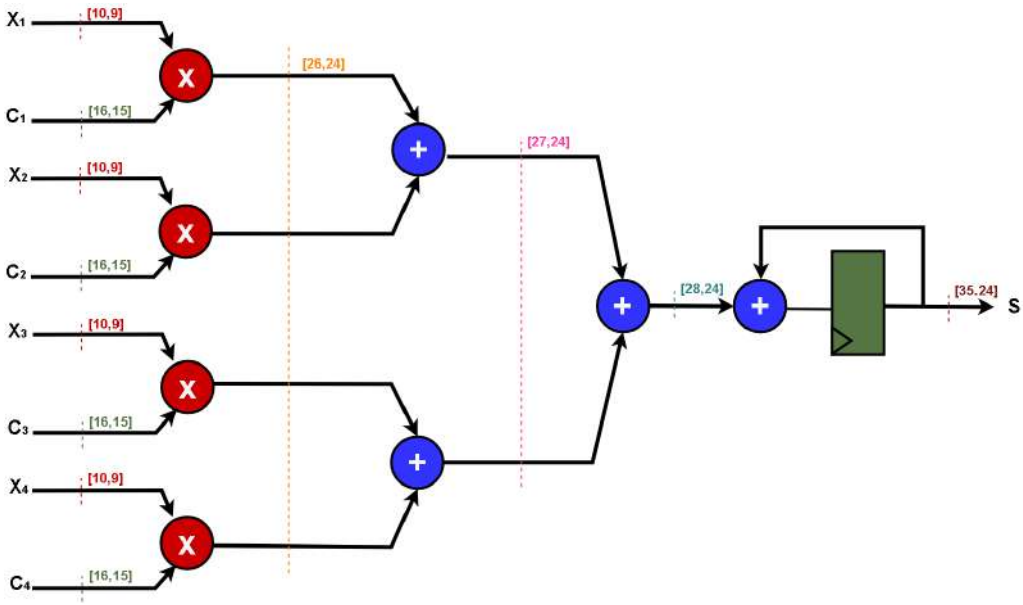
3.12. EJERCICIO 12

Se pretende computar la suma de 500 productos $X_i * C_i$ con una unidad multiplicadora-acumuladora paralelizada con 4 multiplicadores. Los datos X_i tie-

nen formato numérico [10,9] y los coeficientes [16,15], ambos con signo en complemento a dos.

1. Indique los tamaños de palabra a lo largo del circuito para que pueda implementarse la operación $S = \sum X_i * C_i$ sin pérdida de precisión **(Figura 3.15)**.

Figura 3.15. Esquema - Suma 500 productos

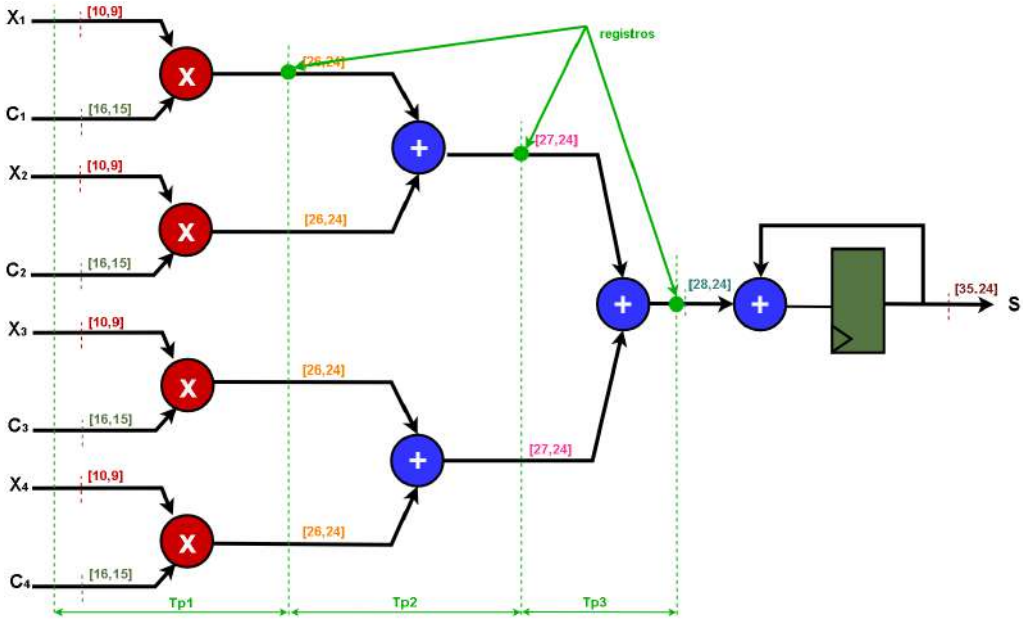


2. ¿Cuántos ciclos de reloj se requieren para completar la computación de la suma de los 500 productos?

   En cada ciclo de reloj se realizan 4 productos por lo que para computar los 500 productos serán necesarios 125 ciclos de reloj.

3. Suponga que se dispone de un dispositivo FPGA cuyos multiplicadores y sumadores tienen tiempos de propagación $t_{mul} = 3.5ns$ y $t_{add} = 1.5ns$ respectivamente. ¿Cuál es la frecuencia máxima de funcionamiento del circuito? NOTA: considere nulos los tiempos de *set-up* y propagación de los registros para resolver este numeral **(Figura 3.16)**.

Figura 3.16. Cuantificación Suma 500 productos



$$Tp_{min} = t_{su} + t_{mul} + 3t_{sum} + t_{FF}$$

$$Tp_{min} = 3.5ns + 3(15ns)$$

$$Tp_{min} = 8ns$$

$$F_{max} = \frac{1}{Tp_{min}} \gg \frac{1}{8 \times 10^{-9}s} \gg 125\ MHZ$$

4. Calcule la frecuencia máxima de funcionamiento del circuito segmentado.

$$Tp_1 = t_{mul} \qquad Tp_2 = t_{sum} \qquad Tp_3 = t_{sum}$$

$$T = max(Tp_1, Tp_2, Tp_3) \gg 3.5\ ns$$

$$F_{max} = \frac{1}{T} \gg \frac{1}{3.5 \times 10^{-9}s} \gg 285.71\ MHZ$$

5. ¿Cuánto tiempo se requiere para computar los 500 productos con el circuito sin segmentar y el circuito segmentado?

$\gg$ Sin Segmentar

$$t_{ss} = \#ciclos * Tp_{min}$$

$$t_{ss} = 128(8\ ns) \gg 1\ us$$

$\gg$Segmentado

$$t_s = \#ciclos * Tp_{max}$$

$$t_s = 125(3.75\ ns) \gg 0.4375\ us$$

$$4\ ciclos\ de\ latencia \gg 4 * (3.5ns) = 14us$$

$$t_t = 14ns + t_s \gg 0.4515ns$$

CAPÍTULO 4
CIC (CASCADE INTEGRATOR-COMB FILTER)

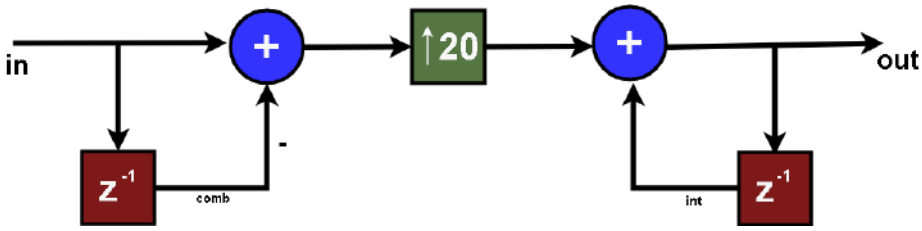
El Filtro CIC es muy empleado en el Procesado Digital de Señales, concretamente en el filtrado multitasa, pues es muy útil para el muestreo a altas frecuencias. Se construye con etapas peine e integradoras, además de un interpolador o un diezmador.

4.1. EJERCICIO 1

Dado un filtro CIC de 2 etapas interpolador por 20, formado a partir de uno de una etapa y orden M=1.

1. Dibuje su diagrama de bloques. **(Figura 4.1)**

Figura 4.1. Esquema Filtro CIC 2 etapas



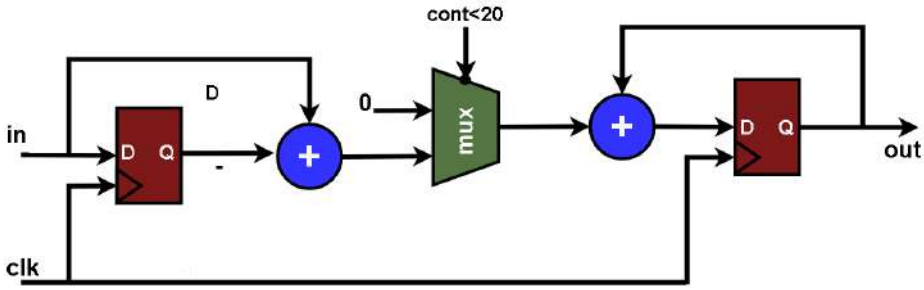
2. Dibuje el esquema de la implementación *hardware* (no utilice el operador z-1, utilice registros) **(Figura 4.2)**.

3. Calcule la ganancia del filtro.

$N = \#\ etapas \gg 2$
$M = Orden\ Filtro \gg 1$
$R =$Interpolación$\gg 20$

$$G = \frac{(R * M)^N}{R} \gg \frac{(1 * 20)^2}{20} = 20$$

Figura 4.2. Filtro CIC (registros)



4. Indique los tamaños de los sumadores. Suponga que los datos de entrada son de 10 bits.

   ≫ Etapa Peine.

   $W_{in} = 10\ bits \gg aumenta\ 1\ bit\ por\ cada\ sumador$

   $W_{out} = w_{in} + 2 \gg salida\ segundo\ sumador$

   $$W_{out} = 17\ bits$$

   ≫ Etapa Integradora.

   $W_{in} = 12bits$

   Se dimensionan los sumadores de acuerdo a la ganancia del filtro.

   $W_g = log_2 G \gg log_2 20 \approx 5\ bits\ de\ ganancia$

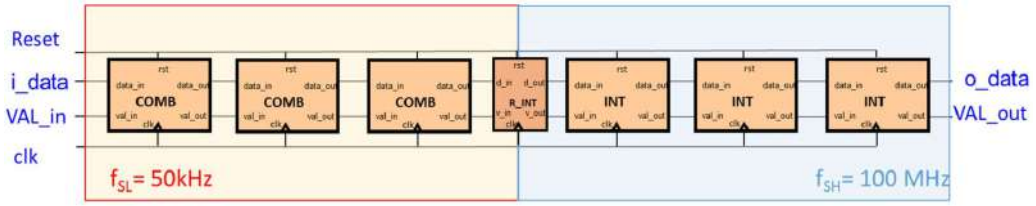
   $W_{out} = W_{in} + W_g \gg salida\ de\ cada\ sumador$

   $$W_{out} = 17\ bits$$

4.2. EJERCICIO 2

Se pretende modelar con lenguaje Verilog un Filtro CIC Interpolador por **2000** **(Figura 4.3)**, con entradas y salidas parametrizables, capaz de cambiar de tasa de muestreo de la señal de **50 KHz** a **100 KHz.**

Figura 4.3. Esquema Filtro Interpolador CIC



El módulo CIC dispone de los siguientes puertos:

- **i_data:** entrada del dato cuantificado con formato **signed [Win, Win-1]**, siendo **Win=16 bits**
- **VAL_in:** entrada binaria que informa de que existe una muestra válida en **i_data**
- **Reset:** *reset* del sistema activo a nivel alto
- **clk:** entrada de reloj
- **o_data:** salida del dato interpolado, con formato **signed [Wout, Wout-1]**siendo **Wout=16 bits**
- **VAL_out:** salida binaria que informa de que existe una muestra válida en **o_data**

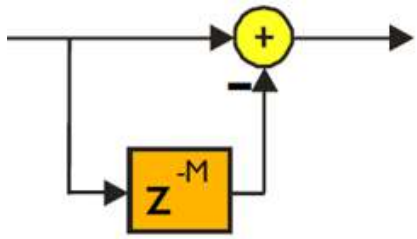
Las especificaciones del filtro CIC son:

- Número de etapas: **N=3**
- Retardo de las etapas peine (*comb*) **M=1**
- Factor de interpolación: **R=2000**
- Tamaño de la entrada: **Win = 16 bits**
- Crecimiento de los datos **Wg** (a calcular)

- El modelo del CIC deberá ser sintetizable y podrá ser implementado en un dispositivo FPGA Cyclone IV EP4CE115F29C7 funcionando a una frecuencia de reloj de **125 MHz.**
- El ancho de banda de la señal de entrada será de **15kHz.**

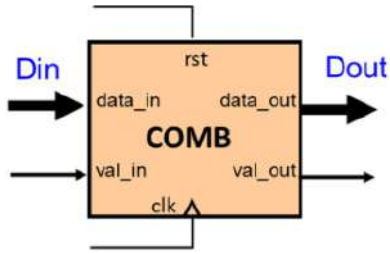
COMB: El filtro *comb* o peine **Figura 4.4** consiste en un restador de la señal original y la misma retrasada en un factor **M**, en nuestro caso igual a **01** por lo indicado en las especificaciones:

Figura 4.4. Diagrama Etapa COMB



El módulo a diseñar **Figura 4.5** será únicamente parametrizable con el tamaño de palabra de entrada, **Win=16 bits**:

Figura 4.5. Módulo Etapa COMB



El código del módulo COMB.v programado es:

```
module COMB
#( parameter Win=16)
```

```
( input signed [Win+1:0] data_in,
input clk,
input rst,
input val_in,
output reg va_out,
output reg signed [Win+1:0] data_out
);
reg signed [Win+1:0] data_in_r=0;

always @ (posedge clk)
if (rst)
begin
data_out = 0;
val_out = 0;
end
else if (val_in)
begin
data_in_r = data_in;
data_out = data_in - data_in_r;
val_out = val_in;
end
else
val_out = 0;
endmodule
```

INT: El filtro interpolador **Figura 4.6** consiste en un acumulador de la señal de entrada y la salida retrasada un periodo:

El módulo a diseñar **Figura 4.7** será parametrizable con el tamaño de palabra de entrada, **Win=16 bits**, y con el crecimiento de la salida, **Wg=22 bits**:

El código del módulo INT.v programado es:

Figura 4.6. Diagrama Etapa INT

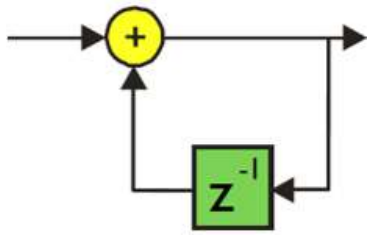
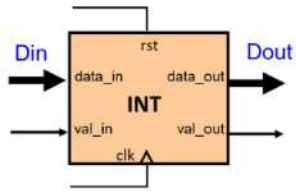


Figura 4.7. Módulo Etapa INT

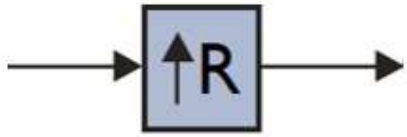



```
module INT
#(parameter Win=16,
parameter Wg=22)
(input signed [Win+Wg-1:0] data_in,
input clk,
input rst,
input val_in,
output reg val_out,
output signed [Win+Wg-1:0] data_out
);

reg signed [Win+Wg-1:0] data_out_r=0;
assign data_out = data_out_r;

always@(posedge clk)
if (rst)
begin
```

```
val_out = 0;
end
else if (val_in == 1)
begin
data_out_r = data_in + data_out_r;
val_out = val_in;
end
endmodule
```

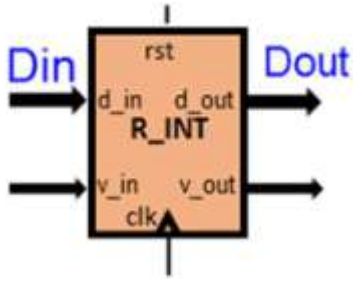
R_INT: El interpolador **Figura 4.8** introduce **R** ceros (factor de interpolación) entre cada muestra de la señal recibida. Con esto se aumenta la frecuencia de muestreo virtualmente, aunque en realidad sea la misma:

Figura 4.8. Diagrama Etapa R_INT



El módulo a diseñar **Figura 4.9** expande la salida (introduciendo **R-1** ceros entre muestras) y genera la señal de habilitación de los filtros integradores para conseguir un factor de interpolación **R=2000**. Será únicamente parametrizable con el tamaño de palabra de entrada, **Win=16 bits**:

Figura 4.9. Módulo Etapa R_INT



El código del módulo R_INT.v programado es:

```
module R_INT
#( parameter Win=16,
parameter FC=2000)
(input signed [Win+1:0] data_in,
input clk,
input rst,
input val_in,
output reg val_out,
output reg signed [Win+1:0] data_out
);

include "MathFun.vh"
parameter n=CLogB2(FC-1);
reg [15:0] count;
integer flag=0;

always@(posedge clk)
begin
if (rst)
data_out = 0;
else if (val_in ==1)
data_out = data_in;
else
data_out = 0;
end

always@(posedge clk)
begin
if (count == FC-1)
```

```
begin
val_out = 0;
count = 0;
flag = 0;
end
else if (flag == 1)
begin
val_out = 1;
count = count+1;
end
if (val_in ==1)
begin
flag = 1;
count = 0;
val_out = 1;
end
end
endmodule
```

A partir de los módulos antes realizados **(COMB.v, INT.v y R_INT.v)** y usando la instrucción *generate*, se programa el módulo **CIC_pc.v**. Se concatenarán **3** etapas **COMB** y **3** etapas **INT** como muestra la **Figura 4.3**.

Aclarar que debido a cómo funciona el comando *generate* se ha tenido que ampliar el tamaño de entrada y salida del bloque **COMB.v** en tantos bits como necesitaba el último bloque **COMB** debido al crecimiento de la señal (en nuestro caso **M-1 bits**, siendo **M** el número de etapas **COMB**).

El código del módulo **CIC_pc.v** programado es:

```
module CIC_pc
```

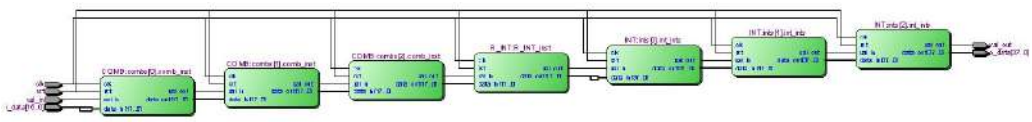
```
#( parameter Win=16,
parameter Wg=22,
parameter M=3,
parameter FC=2000)
(input signed [Win-1:0] i_data,
input clk,
input rst,
input val_in,
output val_out,
output signed [(Win+Wg-1):0] o_data
);

wire signed [Win+1:0] o_data_comb;
wire signed [Win+1:0] o_data_r_int;

wire signed [Win+1:0] data [M:0];
wire val [M:0];
wire val_r_int;

wire signed [(Win+Wg-1):0] data_int [M:0];
wire val_int [M:0];

assign data[0] = i_data;
assign val[0] = val_in;
assign o_data_comb = data[M];

genvar i;
generate
for(i=0;iM;i=i+1)
begin:combs
```

```
COMB #(.Win(Win)) comb_inst
(
.data_in(data[i]),
.clk(clk),
.rst(rst),
.val_in(val [i]),
.val_out(val [i+1]),
.data_out(data[i+1])
);
end
endgenerate

R_INT #(.Win(Win), .FC(FC)) R_INT_inst
(
.data_in(data [M]),
.clk(clk),
.rst(rst),
.val_in(val [M]),
.val_out(val_r_int),
.data_out(o_data_r_int)
);

assign data_int[0][Wg+Win-1:Win+2] = Wg o_data_r_int [Win+1];
assign data_int[0][Win+1:0] = o_data_r_int[Win+1:0];

assign val_int[0] = val_r_int;
assign val_out = val_int[M];

genvar j;
generate
```

```
for(j=0;jM;j=j+1)
begin: ints
INT #(.Win(Win),.Wg(Wg)) int_ints
(
.data_in(data_int[j]),
.clk(clk),
.rst(rst),
.val_in(val_int [j]),
.val_out(val_int [j+1]),
.data_out(data_int[j+1])
);
end
endgenerate

assign o_data = data_int[M];
endmodule
```

La **Figura 4.10** muestra cómo se conectan en cascada los módulos previamente diseñados:

Figura 4.10. Diagrama de Bloques CIC.pc



Para poder realizar el Test Bench es necesario excitar la entrada **val_in** correctamente. Como muestra la **Figura 4.11**, se necesita un **val_in** con un pulso alto de un periodo cada **2000** ciclos de reloj. Para ello se diseña un módulo **VAL.v** que únicamente se usará en el Test Bench:

Como se muestra en la **Figura 4.12**, se comprueba el correcto funcionamiento del módulo excitando con una señal impulso:

Figura 4.11. Señales Entrada y Salida Filtro CIC

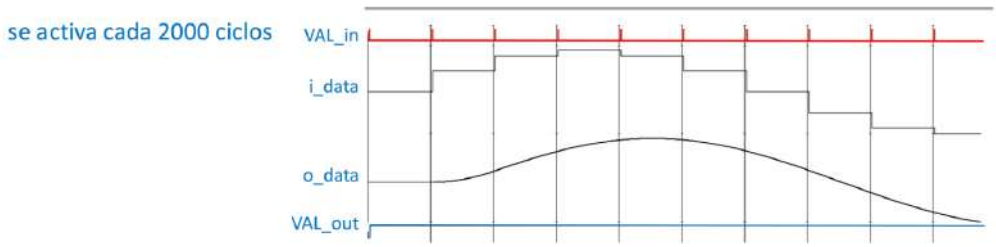
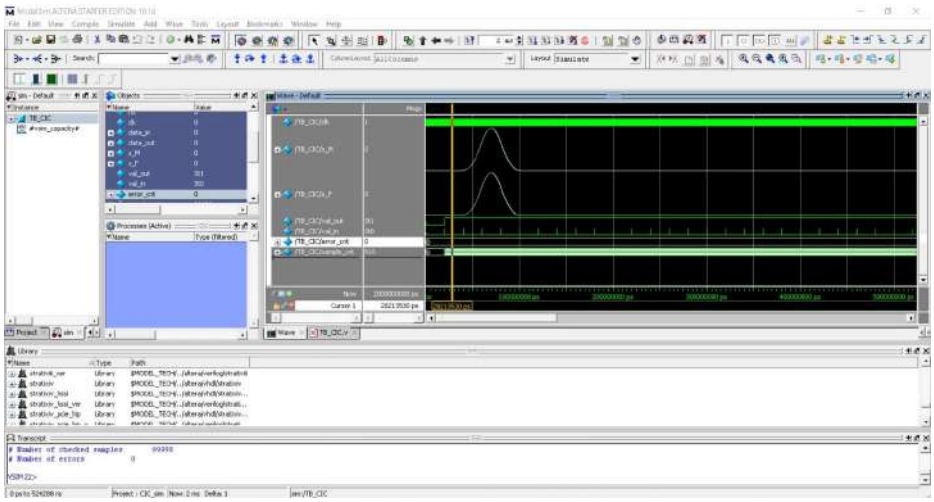
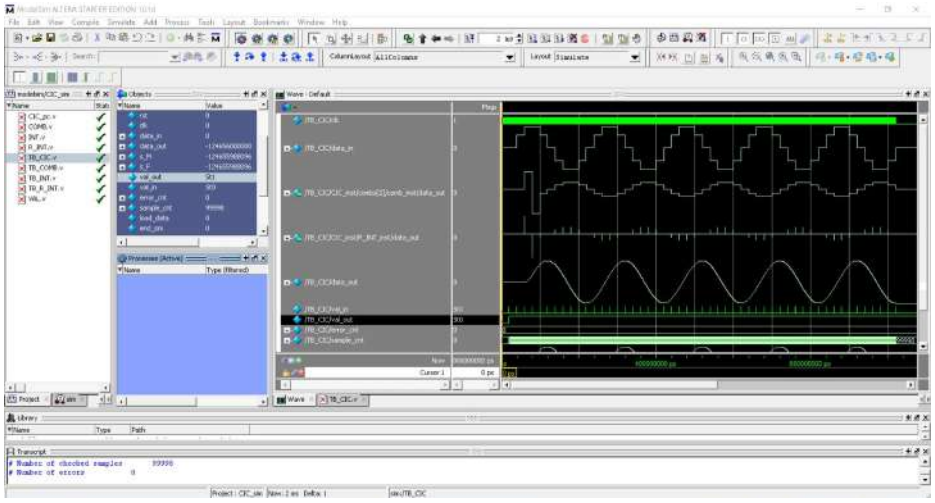


Figura 4.12. Test Bench Filtro CIC - Señal Impulso



También se excita con una señal sinusoidal y se comprueba que funciona correctamente, como se ve en la **Figura 4.13**:

Figura 4.13. Test Bench Filtro CIC - Señal Sinusoidal



Se realiza el módulo **CIC.v** a partir del **CIC_pc.v**, escalando la salida para realizar un truncado a **16 bits** aprovechando todo el rango dinámico de amplitud entre **[-1, 1]**, quedando un formato **[16,15]**.

El código del módulo **CIC.v** programado es:

```
module CIC
#(parameter Win=16,
parameter Wg=22,
parameter M=3,
parameter FC=2000)

(input signed [Win-1:0] i_data,
input clk,
input rst,
input val_in,
output val_out,
```

```
output signed [(Win-1):0] o_data
);

wire signed [(Win+Wg-1):0] data_pc;
assign o_data = data_pc [(Win+Wg-1):Wg];

CIC_pc #(.Win(Win), .Wg(Wg), .M(M),.FC(FC)) CIC_inst
(
.i_data(i_data),
.clk(clk),
.rst(rst),
.val_in(val_in),
.val_out(val_out),
.o_data(data_pc)
);

endmodule
```

Se comprueba que no hay pérdidas en la señal generada y su correcto funcionamiento, como se muestra en la **Figura 4.14**:

Se puede verificar la correcta implementacion de los FIltros CIC, verificando que los recursos empleados solo son **Logic Elements** y no se emplean memorias **M9K**, debido a que este filtro tiene un bajo coste computacional.

Finalmente, se comprueba que la frecuencia máxima obtenida es **231.91 MHz**, como se ve en la **Figura 4.15**:

Figura 4.14. Test Bench Filtro CIC Truncado - Señal Sinusoidal

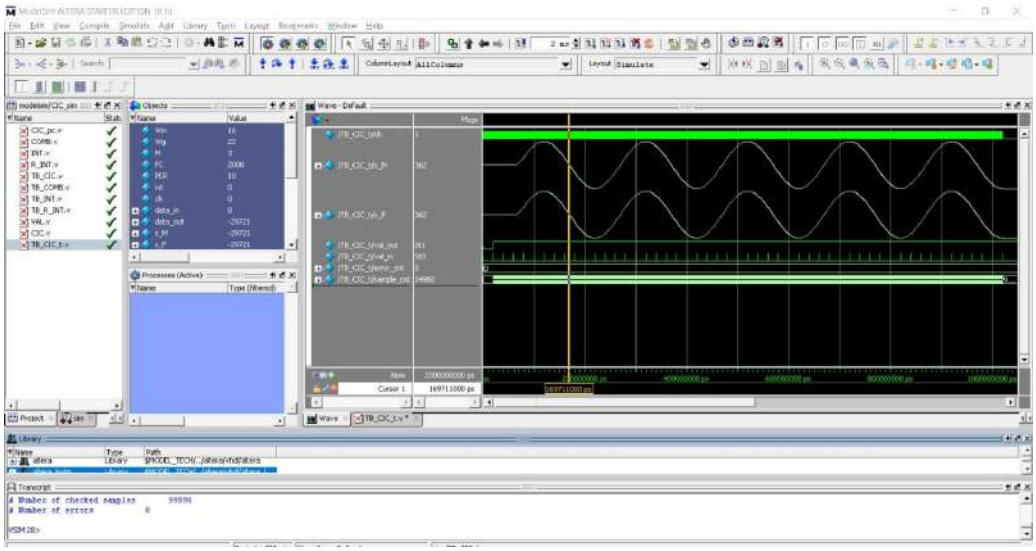


Figura 4.15. Frecuencia Máxima de Funcionamiento - Filtro CIC

Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	231.91 MHz	231.91 MHz	clk	

CAPÍTULO 5
ARQUITECTURAS *HARDWARE* SECUENCIALES

El empleo de arquitecturas secuenciales permite:

- Frecuencia de muestreo (f_s) < Frecuencia de reloj (f_{clk}).
- Si $f_s = \frac{f_{clk}}{c}$ hay c ciclos de reloj para realizar todas las operaciones del algoritmo.
- La arquitectura necesita menos recursos que operaciones tiene el algoritmo.
- Se necesitan los recursos suficientes para implementar el algoritmo en c ciclos. Ej. Un algoritmo que tiene que hacer M=100 multiplicaciones siendo $f_s = \frac{f_{clk}}{c}$, con c=10 podrá implementarse con $\frac{M}{c} = 10$ multiplicadores.
- Se genera un Nuevo resultado cada c ciclos de reloj.
- Cada registro implementa un retardo fracional de valor $\frac{T_s}{c}$.

5.1. EJERCICIO 1

Se pretende implementar un filtro FIR de 50 coeficientes para operar con una frecuencia de muestreo $f_S = 10\ MHz$ y se dispone de un dispositivo FPGA con multiplicadores que pueden funcionar a una frecuencia de reloj máxima de $300\ MHz$.

1. ¿Se puede implementar el filtro utilizando una arquitectura secuencial con un único multiplicador? Justifique la respuesta.

$$\#max_{iter} = \frac{f_{clk}}{f_s} \gg \frac{300\ MHz}{10\ MHz} = 30\ iteraciones$$

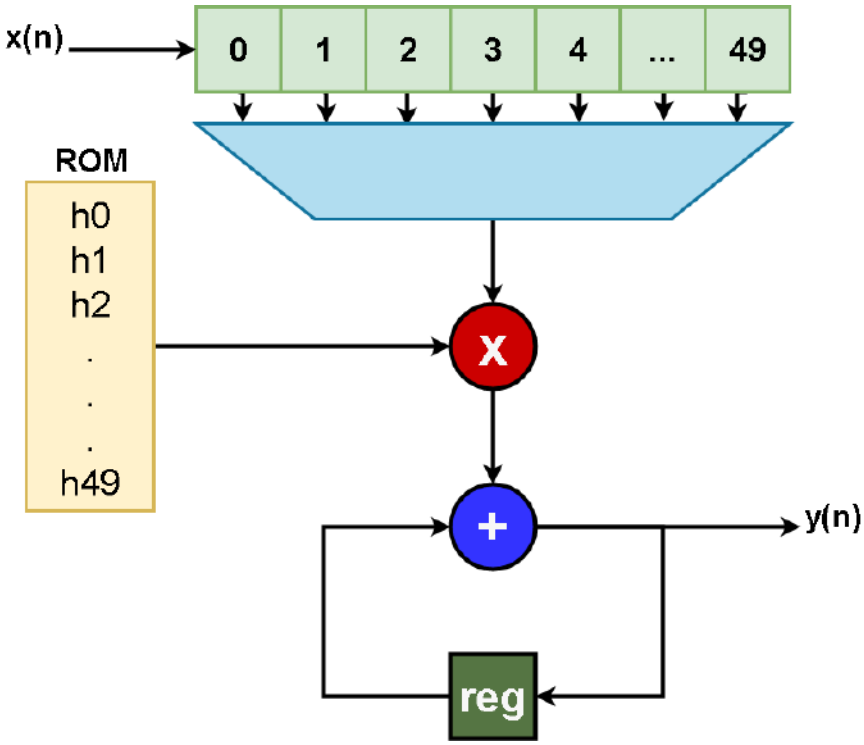
Se pueden realizar 30 iteraciones máximas por ciclo de reloj, por lo que no se puede implementar el filtro FIR con un solo multiplicador.

2. ¿Cuál es la frecuencia máxima de muestreo que se podría alcanzar utilizando un único multiplicador en la arquitectura? Justifique la respuesta.

$$fs_{max} = \frac{f_{clk}}{\#\ coef} \gg \frac{300\ MHz}{50} = 6\ MHz$$

3. Dibuje el esquema de implementación de la arquitectura con un único multiplicador con la que se alcanzaría la máxima frecuencia de muestreo.**(Figura 5.1)**

Figura 5.1. Esquema Filtro FIR de 50 coeficientes



5.2. EJERCICIO 2

Se quiere implementar un filtro FIR simétrico de 50 coeficientes para operar con una frecuencia de muestreo de $10\ MHz$ y se dispone de un dispositi-

vo FPGA con multiplicadores y sumadores cuyos tiempos de propagación son $t_{mul} = 3.5\ ns$ y $t_{add} = 1.5\ ns$, respectivamente. NOTA: considere nulos los tiempos de *set-up* y propagación de los registros para resolver este ejercicio.

1. ¿Se puede implementar el filtro utilizando una arquitectura secuencial con un único multiplicador? Justifique la respuesta.
   Filtro FIR Simétrico $\gg$ 25 coeficientes.
   Camino Crítico $\gg$ 1 mult y 2 sumadores.

$$f_{max} = \frac{1}{t_{mult} + 2 * t_{add}} \gg \frac{1}{3.5\ ns + 2(1.5\ ns)}$$

$$f_{max} = 153.84\ MHz$$

   Se necesitan 25 ciclos para obtener el resultado.

$$f_s = \frac{153.84\ MHz}{25} \gg 6.154\ MHz$$

   No se puede implementar el filtro FIR con un solo multiplicador.

2. ¿Cuál es la frecuencia máxima de muestreo que se podrá alcanzar utilizando un único multiplicador en la arquitectura? Justifique la respuesta.
   $t_{mult} = 3.05\ ns$

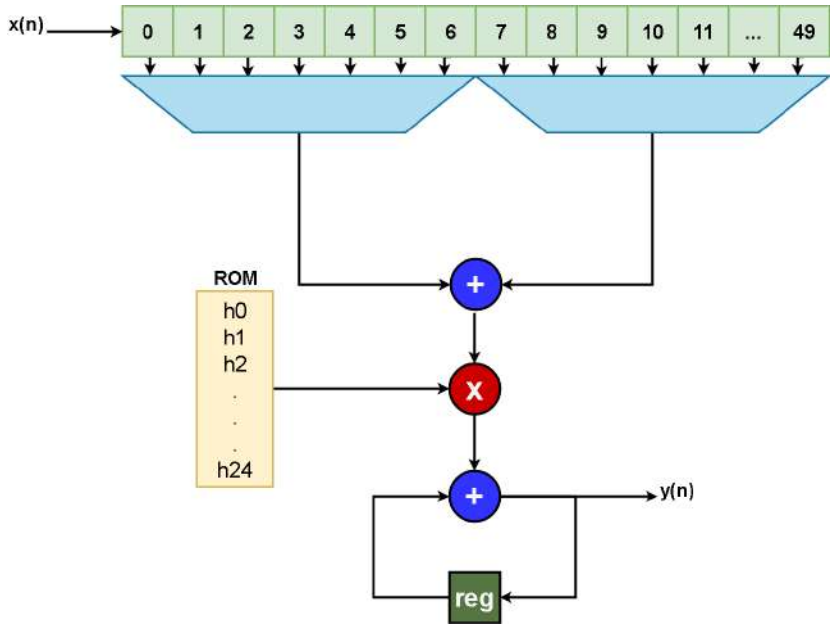
$$fs_{min} = \frac{1}{25 * t_{mult}} \gg \frac{1}{25(3.5\ ns)} = 11.423\ MHz$$

3. Dibuje el esquema de implementación de la arquitectura con un único multiplicador con la que se alcanzaría la máxima frecuencia de muestreo.(Figura 5.2)

5.3. EJERCICIO 3

Estime los recursos *hardware* (indique si son memorias RAM, ROM, multiplicadores, sumadores o registros y sus dimensiones en bits) necesarios para implementar un filtro FIR simétrico de 200 coeficientes con una arquitectura secuencial que dispone un único multiplicador **(Figura 5.2)**. El filtro tiene una ganancia

Figura 5.2. Esquema FIR Simétrico 50 coeficientes



de 7.5, y el formato numérico de los datos y los coeficientes es $[12, 11]$ y $[16, 15]$, respectivamente, ambos con signo en complemento a dos. ¿Cuál es la frecuencia máxima de muestreo a la que puede operar el filtro? Indique el resultado en función de la frecuencia de reloj f_{clk}.

Filtro FIR Simétrico $\gg$ 100 coeficientes
$D[N_d, d] \gg D[12, 11]$ Datos
$C[N_c, c] \gg C[16, 15]$ Coeficientes
$M[N_m, m]$ $G = 7.5$

$\gg$ *Dimensionamiento*

- Sumador Datos Simétricos [13,11]

- Multiplicador $\gg$ M=D*C $\gg$ M[28,26]

- Sumador Acumulador

$$\#bits_{mult} + log_2\, G = [30, 26]$$

- Registros $\gg$ [30,26]

- Memoria ROM

$$coeficientes = 100x16\ bits$$

- Memoria RAM

$$datos = 200x12\ bits$$

$\gg$ *Dimensionamiento*

- 1 multiplicador
- 2 sumadores
- 30 registros
- 1 ROM (128x12 bits)
- 1 RAM (256x16 bits)

$\gg$ Frecuencia Máxima

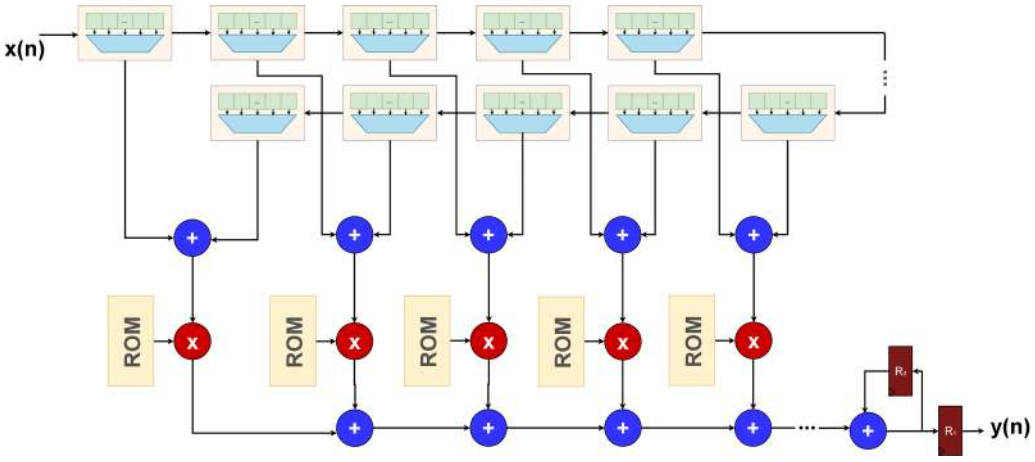
$$f_{max} = \frac{f_{clk}}{\#coef\ o\ ciclos} \gg \frac{f_{clk}}{100}$$

5.4. EJERCICIO 4

Se quiere implementar un filtro FIR simétrico de 200 coeficientes para operar con una frecuencia de muestreo de 20 MHz y se dispone de un dispositivo FPGA que soporta una frecuencia máxima de reloj de 250 MHz.

1. Dibuje el esquema de implementación del filtro. **(Figura 5.3)**

Figura 5.3. FIR 200 Coeficientes – Arq. Paralela



2. Indique cuántos multiplicadores se requieren para su implementación. Justifique la respuesta.

Filtro FIR Simétrico $\gg$ 100 coeficientes.

$$\#_{max}iter = \frac{f_{clk}}{f_s} \gg \frac{250\ MHz}{20\ MHz} = 12.5\ ciclos$$

$$\frac{100\ mult}{12.5\ ciclos} = 8\ multiplicaciones\ x\ ciclo$$

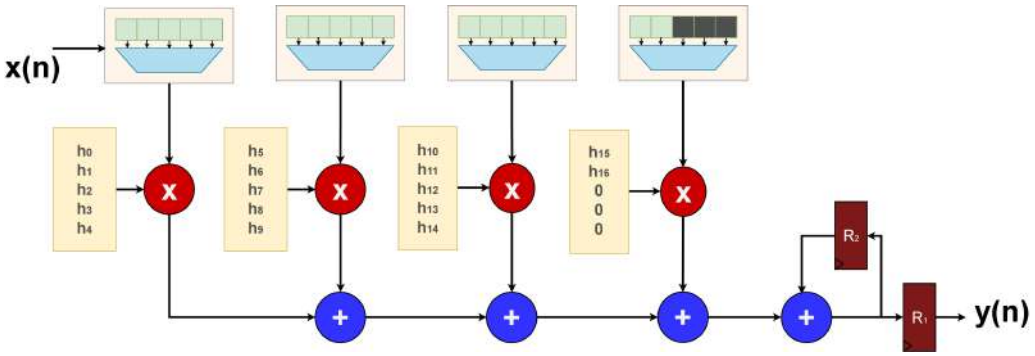
Con 10 multiplicadores, se puede obtener el dato luego de 10 iteraciones.

5.5. EJERCICIO 5

Dibuje el esquema de implementación de un filtro FIR de orden 16 (17 coeficientes) con una arquitectura semiparalela que utiliza 4 multiplicadores **(Figura 5.4)**. Dimensione la arquitectura indicando el número de registros y tamaño de las memorias e indique qué coeficientes se almacenan en cada memoria ROM. Suponga que los coeficientes del filtro son: $h_0,\ h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7,\ h_8,\ h_9,\ h_{10},\ h_{11},\ h_{12},\ h_{13},\ h_{14},\ h_{15},\ h_{16}$. ¿Cuál es la frecuencia máxima de muestreo a

la que puede operar el filtro? Indique el resultado en función de la frecuencia de reloj f_{clk}.

Figura 5.4. FIR Orden 16 – Arq. Semi-Paralela



$$\#ciclos = ceil\left(\frac{\#coef}{\#mult}\right) \gg ceil\left(\frac{17}{4}\right) \approx 5\ ciclos$$

$$fs_{max} = \frac{f_{clk}}{\#ciclos} \gg \frac{f_{clk}}{5}$$

$\gg$ Recursos

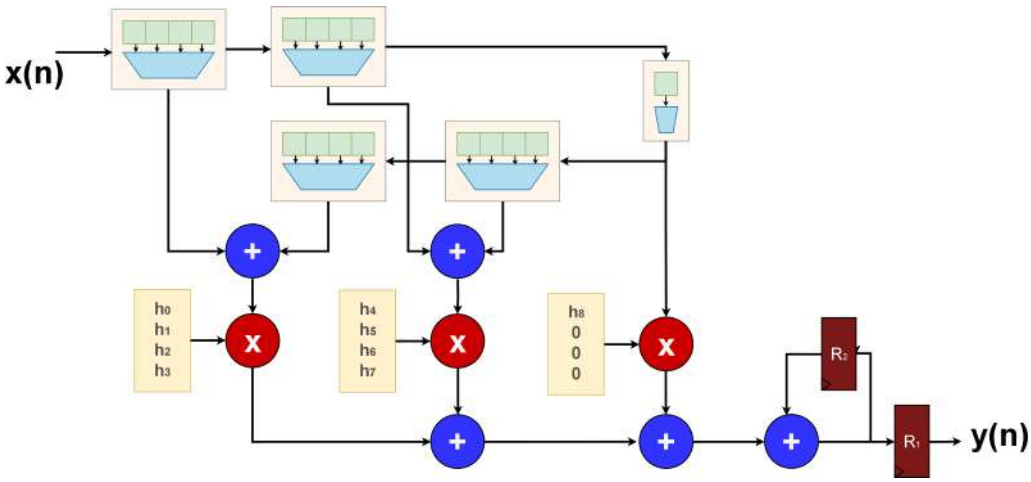
- 4 memorias ROM (8 $\times$ n bits)
- 4 multiplicadores
- 3 sumadores
- 1 acumulador
- 2 registros

5.6. EJERCICIO 6

Dibuje el esquema de implementación de un filtro FIR simétrico de orden 16 (17 coeficientes) con una arquitectura semiparalela que utiliza 3 multiplicadores **(Figura 5.5)**. Dimensione la arquitectura indicando el número de registros y

tamaño de las memorias e indique qué coeficientes se almacenan en cada memoria ROM. Suponga que los coeficientes del filtro son: $h_0,\ h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7,\ h_8,\ h_9,\ h_{10},\ h_{11},\ h_{12},\ h_{13},\ h_{14},\ h_{15}, h_{16}$. ¿Cuál es la frecuencia máxima de muestreo a la que puede operar el filtro? Indique el resultado en función de la frecuencia de reloj f_{clk}.

Figura 5.5. FIR Simétrico – Arq. Semiparalela



$$h_0 = h_{16} \qquad h_5 = h_{11}$$

$$h_1 = h_{15} \qquad h_6 = h_{10}$$

$$h_2 = h_{14} \qquad h_7 = h_9$$

$$h_3 = h_{13} \qquad h_8$$

$$h_4 = h_{12}$$

$$\#ciclos = ceil\left(\frac{\#coef}{\#mult}\right) \gg ceil\left(\frac{9}{3}\right) \approx 3\ ciclos$$

$$fs_{max} = \frac{f_{clk}}{\#ciclos} \gg \frac{f_{clk}}{3}$$

$\gg$ Recursos

- 3 memorias ROM ($4 \times$ n bits)

- 3 multiplicadores
- 4 sumadores
- 1 acumulador
- 2 registros

CAPÍTULO 6
ARQUITECTURAS PARALELAS

El empleo de arquitecturas paralelas permite:

- Frecuencia de muestreo(f_s) = Frequencia de reloj (f_{clk}).
- La arquitectura necesita tantos recursos como operaciones tiene el algoritmo.
- Se obtiene un resultado cada ciclo de reloj.
- Un registro equivale a un retardo de una muestra temporal.

6.1. EJERCICIO 1

Se desea implementar un filtro FIR de 400 coeficientes que deberá trabajar a una frecuencia de muestreo de 200 MHz. Para ello se dispone de los siguientes dispositivos:

- DSP ADSP SC589 que posee una capacidad de procesado de 1.8 GMACs.
- FPGA de Altera Statrix 10 Gx 500, que posee 2304 multiplicadores capaces de trabajar con una frecuencia de reloj de 500 MHz.
- FPGA de Xilinx Virtex-7, que posee 3600 multiplicadores capaces de trabajar con una frecuencia de reloj de 450 MHz.

¿Cuáles de los dispositivos anteriores se podría utilizar para diseñar el filtro? Justificar la respuesta.
$\gg$ DSP ADSP SC589

$$Carga\ Comp_{min} = coef * f_s$$

$$Carga\ Comp_{min} = 400 * (200 \times 10^6) \gg 80\ GMs$$

La capacidad del DSP es inferior a la requerida.

≫ FPGA-Statrix 10 Gx 500

$$Carga\ Comp_{disp} = \#mult * f_{clk}$$

$$Carga\ Comp_{disp} = 2304 * (500 \times 10^6) \gg 1.15\ TMACs$$

Se puede implementar el Filtro FIR en la FPGA-Stratix.

≫ FPGA- Virtex-7

$$Carga\ Comp_{disp} = \#mult * f_{clk}$$

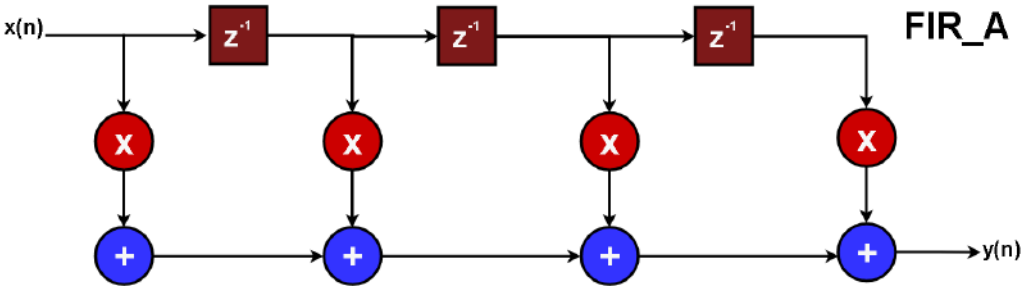
$$Carga\ Comp_{disp} = 3600 * (450 \times 10^6) \gg 1.62\ TMACs$$

Se puede implementar el Filtro FIR en la FPGA-Virtex.

6.2. EJERCICIO 2

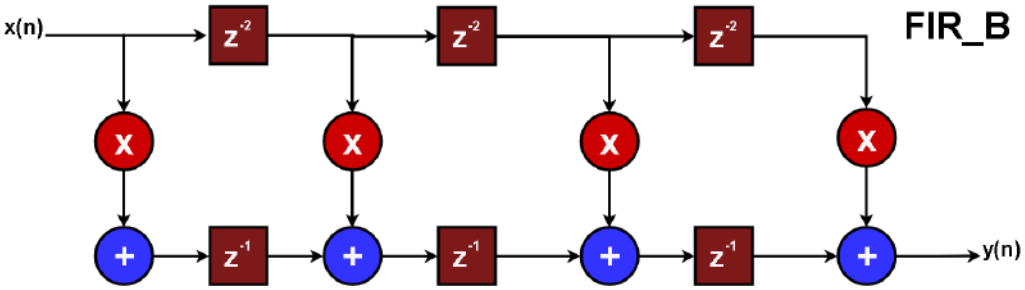
Dados los siguientes esquemas de filtros con estructuras paralelas en forma directa: **(Figura 6.1)**, **(Figura 6.2)** y **(Figura 6.3)**

Figura 6.1. Filtro Forma Directa



1. Indique si son correctas las segmentaciones realizadas en las arquitecturas FIR_B y FIR_C. Justificar la respuesta.
   Las segmentaciones realizadas en las arquitecturas FIR_B **(Figura 6.2)** y FIR_C **(Figura 6.3)** son correctas, siendo un array sistólico la arquitectura FIR_C.

Figura 6.2. Filtro Segmentado



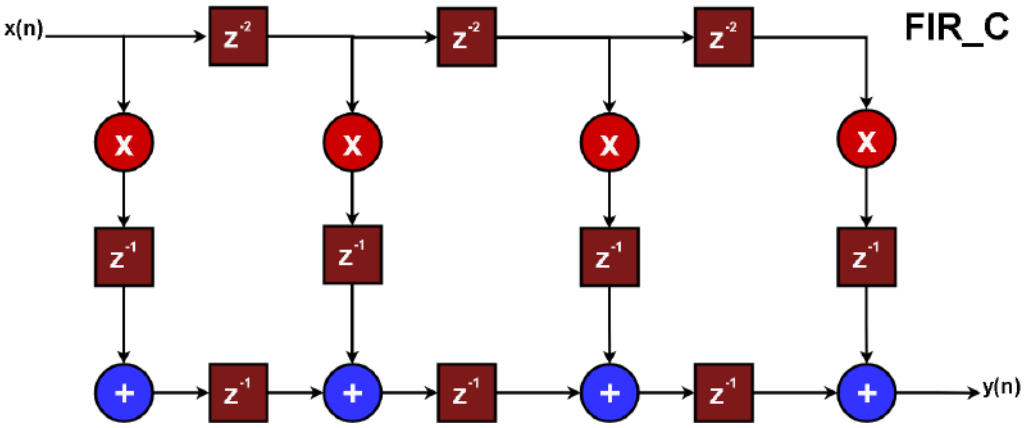
2. ¿Cuál es la latencia, dada en ciclos de reloj, de cada filtro?

   FIR_A ≫ no presenta latencia.

   FIR_B ≫ 3 ciclos de reloj.

   FIR_C ≫ 4 ciclos de reloj.

Figura 6.3. Filtro Segmentado (Array Sistólico)

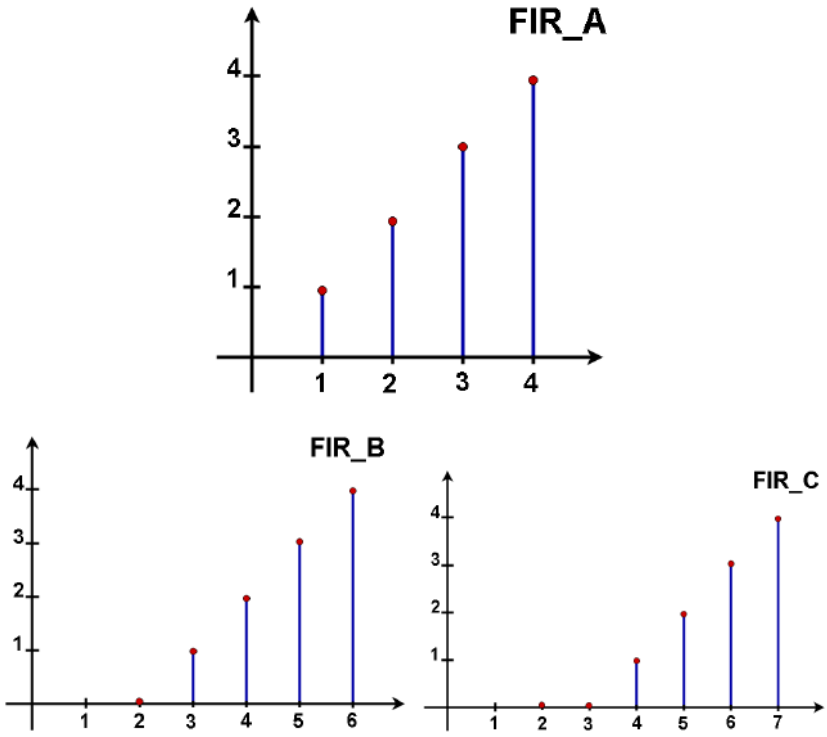


3. Dibuje la respuesta al impulso de cada filtro. Suponga: $h_1 = 1,\ h_2 = 2,\ h_3 = 3\ y\ h_4 = 4.$**(Figura 6.4)**

6.3. EJERCICIO 3

Se dispone de un dispositivo FPGA cuyos multiplicadores y sumadores tienen tiempos de propagación $t_{mul} = 3.5\ ns$ y $t_{add} = 1.5\ ns$, respectivamente.

Figura 6.4. Respuesta al Impulso FIR



¿Cuál es la frecuencia máxima de funcionamiento de cada filtro? NOTA: considere nulos los tiempos de *set-up* y propagación de los registros para resolver este ejercicio.

FIR_A $\gg$

$$f_{max} = \frac{1}{t_{mult} + 3 * t_{add}}$$

$$f_{max} = \frac{1}{3.5\ ns + 3 * (1.5\ ns)} \gg 125\ MHz$$

FIR_B $\gg$

$$f_{max} = \frac{1}{t_{mult} + t_{add}}$$

$$f_{max} = \frac{1}{3.5\ ns + 1.5\ ns} \gg 200\ MHz$$

FIR_C $\gg$

$$f_{max} = \frac{1}{t_{mult} + t_{add}}$$

$$f_{max} = \frac{1}{3.5} \gg 285.71\ MHz$$

6.4. EJERCICIO 4

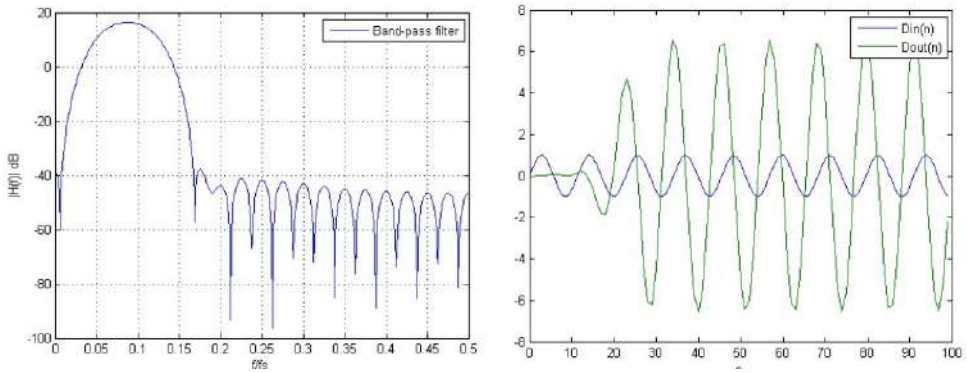
Utilizando la estructura del filtro FIR_C de la sección anterior, se va a implementar un filtro FIR paso banda de 41 coeficientes cuya respuesta en frecuencia y un ejemplo de señal filtrada se muestran en la **Figura 6.5**. Teniendo en cuenta que el filtro tiene una ganancia de 16.31 dBs y que la señal de entrada al filtro tiene un formato [10,9] y los coeficientes [16,15], ambos con signo, determine el crecimiento de datos y los formatos numéricos a la salida del filtro y en los puntos intermedios.

$D[N_d, d] \gg D[10, 9]$ Datos

$C[N_c, c] \gg C[16, 15]$ Coeficientes

$M[N_m, m]$ $\qquad G = 16.31\ dBs$

Figura 6.5. Respuesta Filtro Pasa Banda



≫ Dimensionamiento

- Multiplicador ≫ M=D*C ≫ M[26,24]
- Rango Señal Filtrada (-8,8)
- Salida Filtro ≫ [28,24]

$$2^N - 1 = v_{max} - v_{min}$$

$$N = log_2\, 15 \approx 4\ bits$$

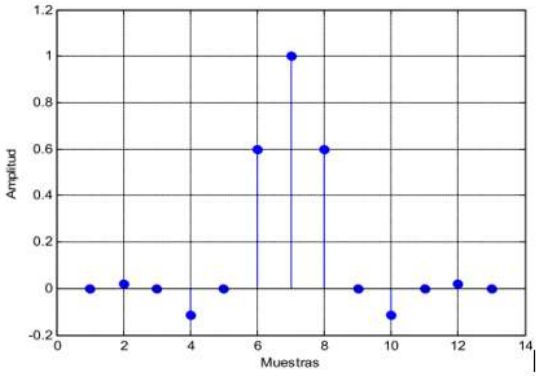
- Salida Primer Sumador ≫ [27,24]
- Salida Segundo Sumador ≫ [28,24]

6.5. EJERCICIO 5

Dado un filtro media banda de orden 12 **(Figura 6.6)** con los coeficientes indicados abajo.

1. Dibuje la arquitectura en paralelo utilizando una estructura de filtro FIR directa **(Figura 6.7)**.

Figura 6.6. Coeficientes Filtro Media Banda



Coeficientes

$h_0 = h_2 = h_4 = h_6 = h_8 = h_{10} = h_{12} = 0$

$h_1 = h_{11} = 0.018$

$h_3 = h_9 = -0.115$

$h_5 = h_7 = 0.597$

$h_6 = 1$

Figura 6.7. Esquema FIR Directo

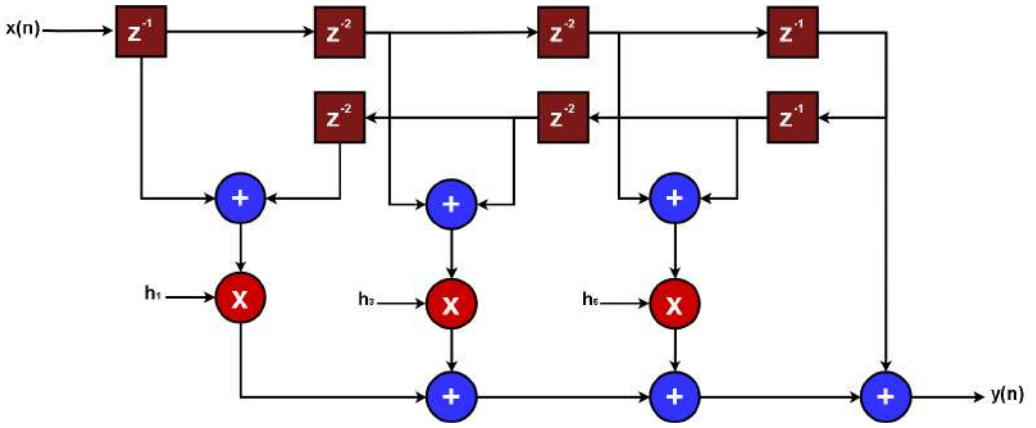
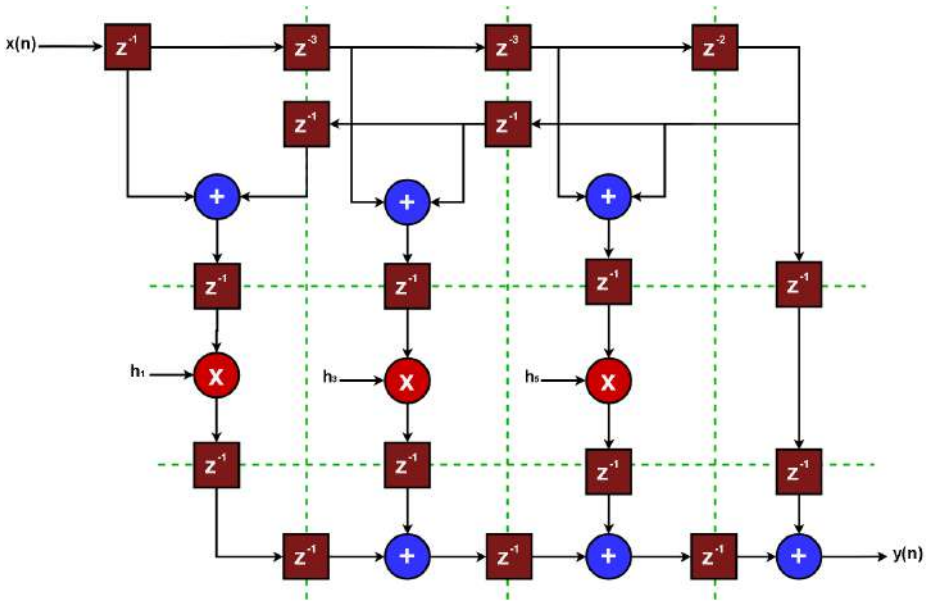


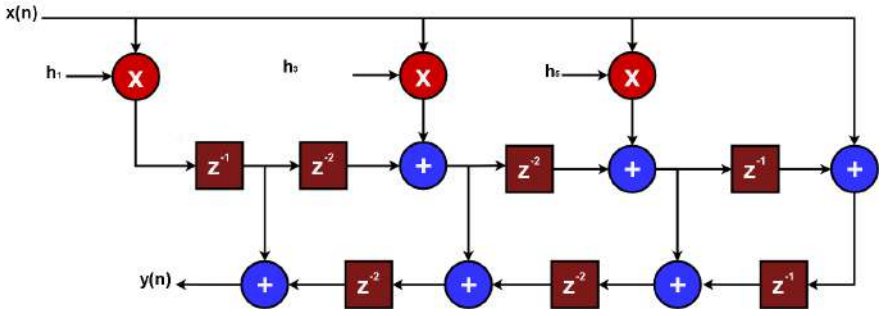
Figura 6.8. Esquema Segmentado FIR Directo



2. Dibuje la arquitectura en paralelo utilizando una estructura de filtro FIR transpuesta **(Figura 6.8)**.

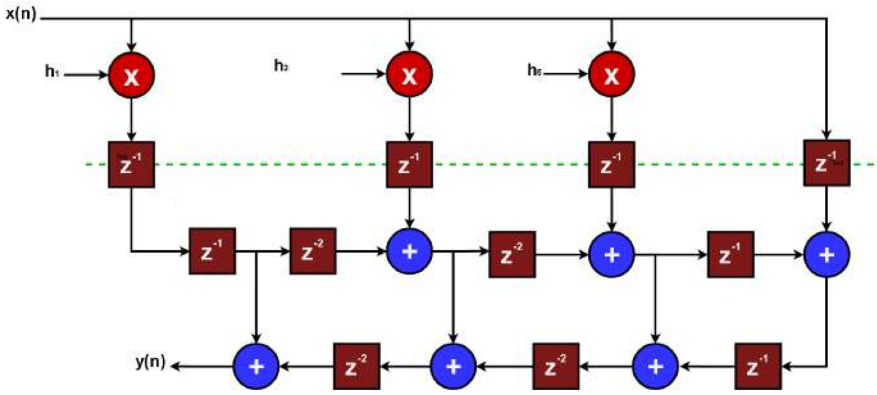
3. Segmente la arquitectura con estructura directa para que se alcance la mayor frecuencia de funcionamiento posible **(Figura 6.9).**

Figura 6.9. Esquema FIR Transpuesto



4. Segmente la arquitectura con estructura transpuesta para que se alcance la mayor frecuencia de funcionamiento posible.**(Figura 6.10)**

Figura 6.10. Esquema Segmentado FIR Transpuesto



5. Indique los recursos que requieren ambas arquitecturas segmentadas (número y tamaño de sumadores, multiplicadores y registros). Suponga que los datos y coeficientes tienen un formato [16,15] con signo y que la ganancia del filtro es 1.

≫ Estructura Directa Segmentada.

- #sumadores ≫ 3 sumadores [17,15]
  3 sumadores [33,30]
- #multiplicadores ≫ 3 mult. [33,30]
- #registros ≫ 13 registros [16,15]
  3 registros [17,15]
  6 registros [33,30]

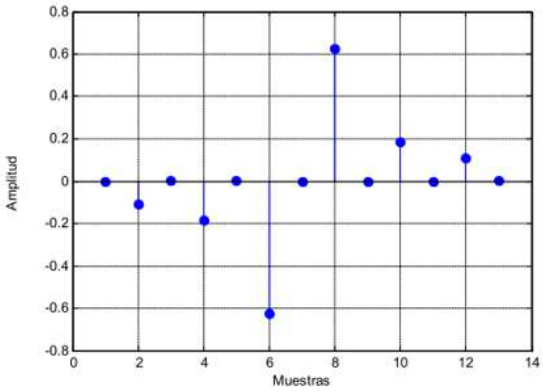
≫ Estructura Transpuesta Segmentada.

- #sumadores ≫ 6 sumadores [32,30]
- #multiplicadores ≫ 3 mult. [33,30]
- #registros ≫ 1 registro [16,15]
  14 registros [32,30]

6.6. EJERCICIO 6

Dado un filtro de Hilbert de orden 12 cuyos coeficientes se indican en la **Figura 6.11**.

Figura 6.11. Coeficientes Filtro Hilbert



Coeficientes

$h_0 = h_2 = h_4 = h_6 = h_8 = h_{10} = h_{12} = 0$

$h_1 = h_{11} = 0.018$

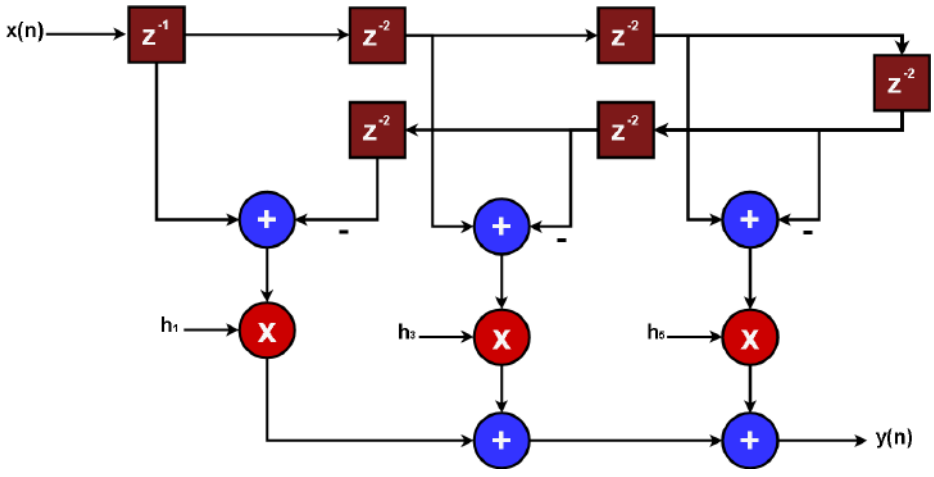
$h_3 = h_9 = -0.115$

$h_5 = h_7 = 0.597$

$h_6 = 1$

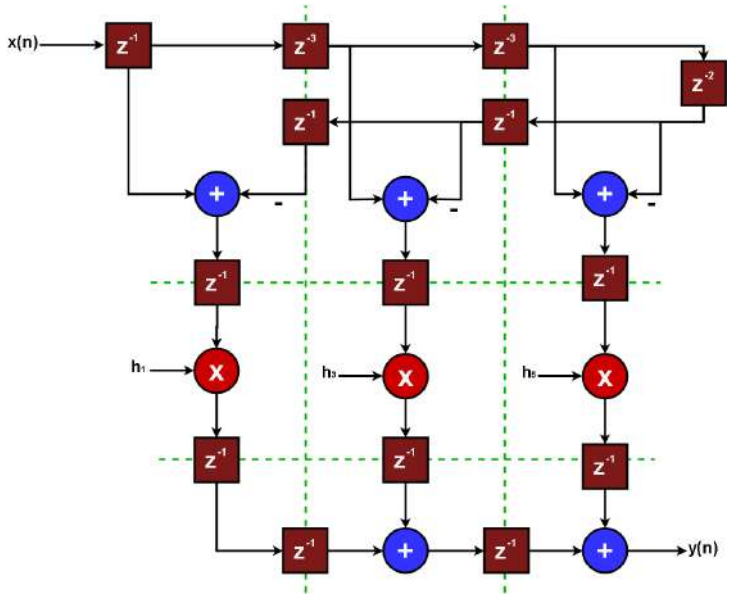
1. Dibuje la arquitectura en paralelo utilizando una estructura de filtro FIR directa **(Figura 6.12)**.

Figura 6.12. Esquema FIR Directo Hilbert



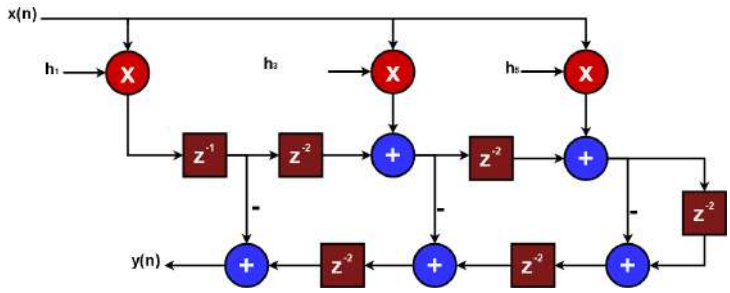
2. Segmente la arquitectura con estructura directa para que se alcance la mayor frecuencia de funcionamiento posible **(Figura 6.13)**.

Figura 6.13. Esquema Segmentado FIR Hilbert



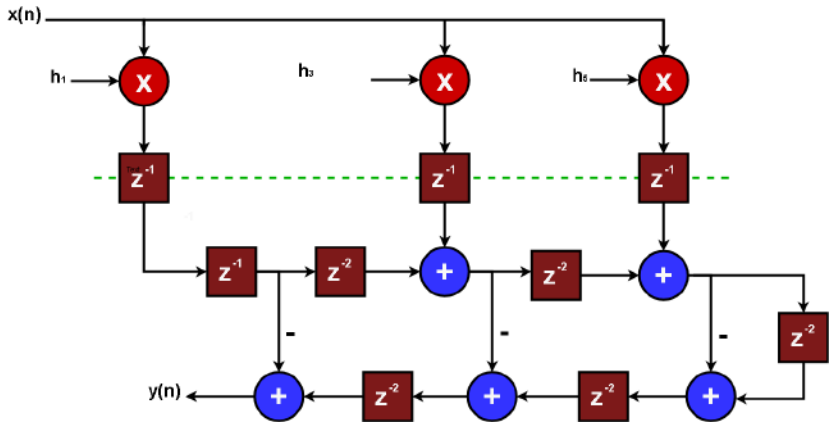
3. Dibuje la arquitectura en paralelo utilizando una estructura de filtro FIR transpuesta **(Figura 6.14)**.

Figura 6.14. Esquema FIR Transpuesto Hilbert



4. Segmente la arquitectura con estructura transpuesta para que se alcance la mayor frecuencia de funcionamiento posible **(Figura 6.15)**.

Figura 6.15. Esquema Segmentando Hilbert Transpuesto



5. Indique los recursos que requieren ambas arquitecturas segmentadas (número y tamaño de sumadores, multiplicadores y registros). Suponga que los datos y coeficientes tienen un formato [16,15] con signo y que la ganancia del filtro es 1.

   ≫ Estructura Directa Segmentada.

   - #sumadores ≫ 3 sumadores [17,15]
     2 sumadores [33,30]
   - #multiplicadores ≫ 3 mult. [33,30]
   - #registros ≫ 11 registros [16,15]
     3 registros [17,15]
     5 registros [33,30]

   ≫ Estructura Transpuesta Segmentada.

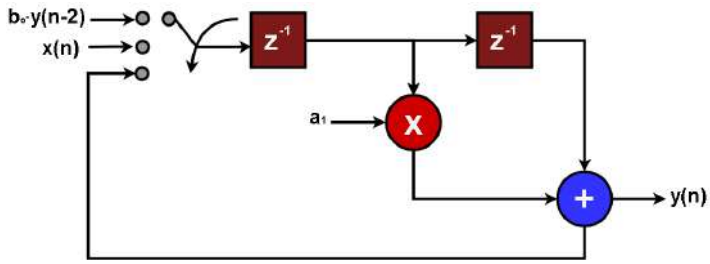
   - #sumadores ≫ 5 sumadores [32,30]
   - #multiplicadores ≫ 3 mult. [32,30]
   - #registros ≫ 1 registro [16,15]
     14 registros [32,30]

6.7. EJERCICIO 7

Una señal sinusoidal se puede generar con un filtro recursivo de segundo orden que implemente la siguiente ecuación en diferencias: $y(n) = a_1 * y(n-1) - y(n-2)$ con $y(-2) = b_o$, donde $b_o = A * \sin\left(\frac{2*\pi*f_o}{f_s}\right)$ y $a_1 = 2 * \cos\left(\frac{2*\pi*f_o}{f_s}\right)$, siendo A y f_o la amplitud y frecuencia de oscilación y f_s la frecuencia de muestreo.

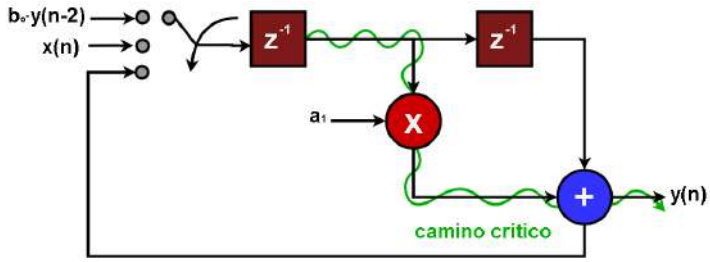
1. Dibuje el esquema de implementación en paralelo del filtro recursivo $(n) = a_1 * y(n-1) - y(n-2)$ **(Figura 6.16)**.

Figura 6.16. Esquema Implementación Ecuación Diferencial



2. Indique cuál es el camino crítico del filtro **(Figura 6.17)**.

Figura 6.17. Camino Crítico Ecuación Diferencial



3. ¿Se puede segmentar el filtro para aumentar la frecuencia de funcionamiento?

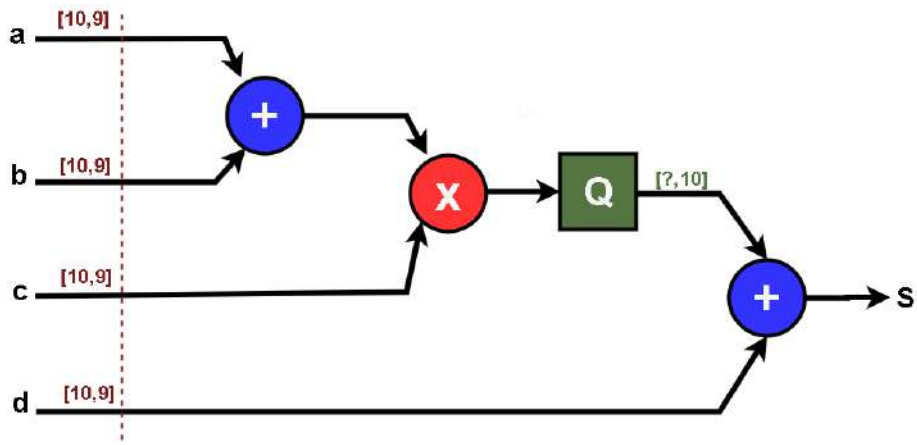
   No se puede segmentar el filtro porque la entrada depende de una retroalimentación de la salida.

CAPÍTULO 7
MISCELÁNEA EJERCICIOS

7.1. EJERCICIO 1

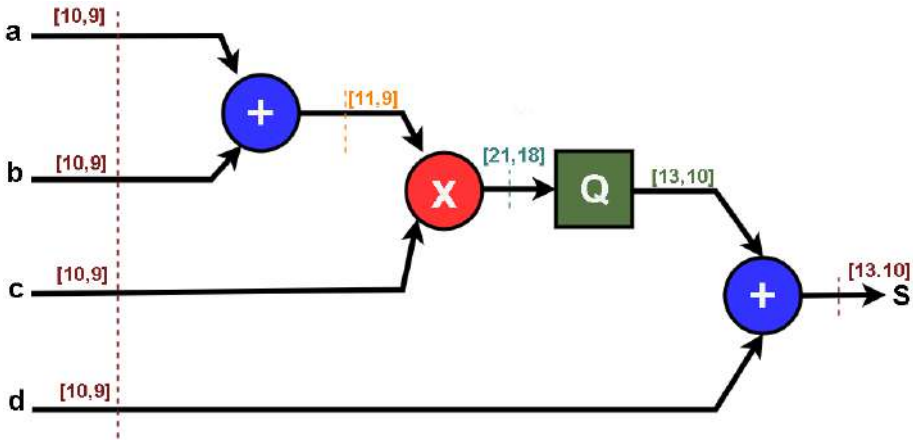
Se desea implementar en un dispositivo FPGA el operador $s = (a - b) * c + d$, siguiendo el esquema que se muestra en la **Figura 7.1**. En dicho operador, el formato de los operandos de entrada (a,b,c y d) es [10,9] en complemento a dos y la salida del multiplicador se trunca para limitar su salida a 10 bits fraccionales.

Figura 7.1. Esquema Operador s = (a-b)·c+d



1. Indique los formatos numéricos a la salida de los operadores restador, multiplicador y sumador para que no se cometa pérdida de precisión en sus cálculos y dispongan del menor número de bits posible, y a la salida del bloque Q, que simboliza el truncado del multiplicador, para que no se produzca desbordamiento **(Figura 7.2)**.

Figura 7.2. Cuantificación Operador s = (a-b)·c+d



2. Escriba el código de Matlab para modelar el operador $s = (a-b)*c+d$ incluyendo el efecto del truncado del multiplicador. Utilice la función floor de Matlab.

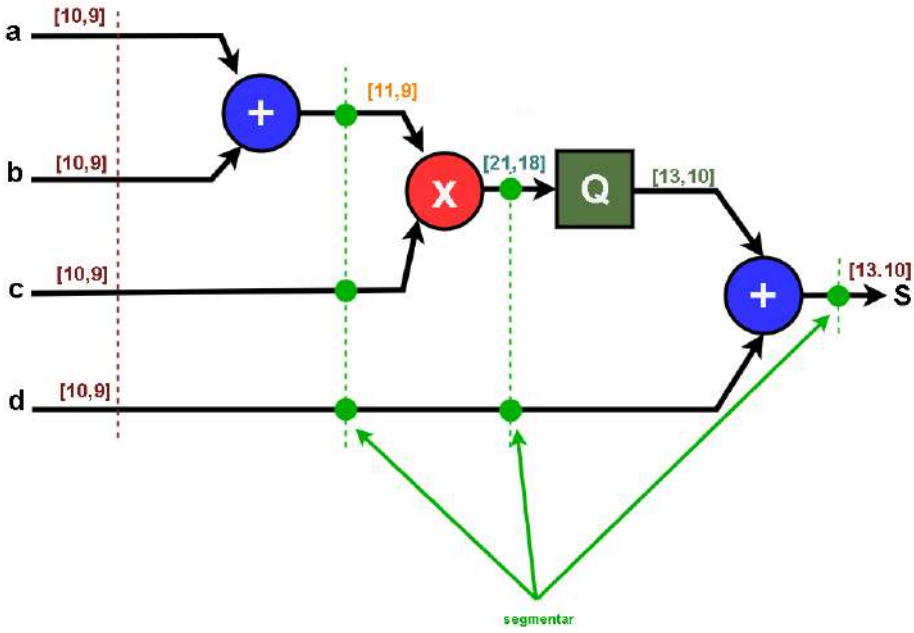
$$s = floor((a-b)*c*2^{10})*2^{-10}+d$$

3. Indique en el esquema del operador en qué puntos se debe segmentar el circuito para asegurar la máxima frecuencia de funcionamiento **(Figura 7.3)**.

4. Escriba el código Verilog del operador para que realice las operaciones siguiendo los formatos indicados en el numeral 1 y la segmentación del numeral 3. Justifique su respuesta.

```
module add_mult_add_t(
input signed [9 : 0] a,b,c,d,
input clk,
output signed [12 : 0] s
);

reg signed [10 : 0] add1_r;
```

Figura 7.3. Operador s = (a-b)·c+d Segmentado




```
wire signed [20 : 0] m;
reg signed [12 : 0] mt_r;
reg signed [9 : 0] c_r;
reg signed [9 : 0] d_r1,d_r2;
reg signed [12 : 0] add2_r;
assign m = add1_r*c_r;

always@(posedge clk)
begin
add1_r <= a - b;
mt_r <= m[20 : 8];
c_r <= c;
d_r1 <= d;
d_r2 <= d_r1;
add2_r <= mt_r + d_r2[9], d_r2[9], d_r2, 1'b0;
```

```
end

assign s = add2_r;

endmodule
```

5. Suponiendo que el tiempo de propagación de un sumador es $t_{sum} = 1.5ns$, el del multiplicador es $t_{mult} = 3.5ns$, y que se consideran nulos los tiempos de t_{su} y t_{co} de los registros, indique cuál es la frecuencia máxima de operación del operador.

   De los posibles caminos entre registros existentes en el circuito los más limitantes son los que pasan por el multiplicador. Por tanto, estos impondrán la frecuencia máxima de operación.

$$f_{max} = \frac{1}{t_{co} + t_{mult} + t_{su}}$$

$$f_{max} \approx \frac{1}{t_{mult}} = 285.7MHz$$

6. Indique el número de recursos *hardware* que se requiere para implementar el operador en un dispositivo Cyclone IV. Justifique su respuesta.

Tomando en cuenta que cada bit en los operadores (suma, multiplicación), representa un LUT en la Cyclone IV y cada bit de los registros representa un Flip-Flop, los recursos *hardware* necesarios se detallan en la **Tabla 7.1** de acuerdo al detalle de la **Figura 7.4.**

Figura 7.4. Esquema Recursos *Hardware* Cyclone IV

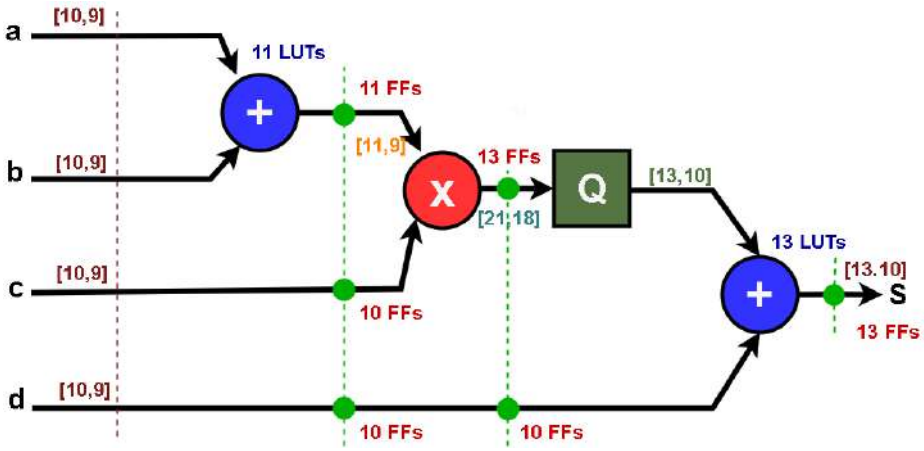


Tabla 7.1. Recursos *Hardware* Cyclone IV

RECURSO	NÚMERO
LUTs	24
FFs	67
MULTs 18X18	1

7.2. EJERCICIO 2

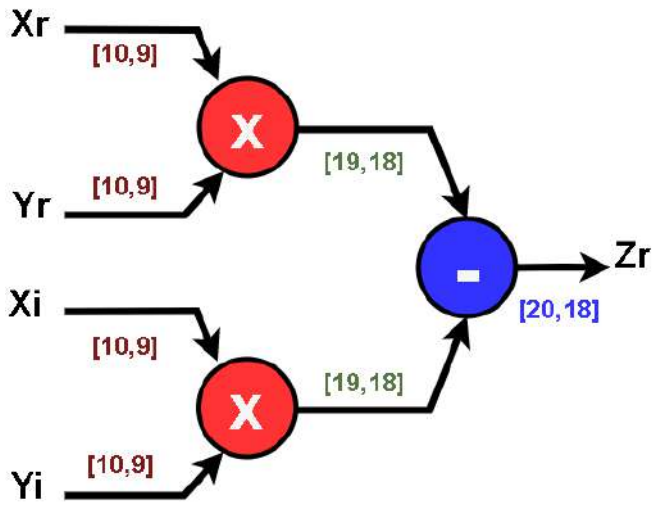
Dado los números complejos $X = X_r + jX_i$ e $Y = Y_r + jY_i$, se requiere computar el cálculo de la parte real de la multiplicación de dichos números:

$$Z_r = real(X.Y) = X_r * Y_r - X_i * Y_i$$

Los formatos numéricos de las partes reales e imaginarias de ambos números son [10,9] codificados con signo en complemento a dos, en las que se ha limitado su rango a $(-1 + 2^{(-9)}\text{-}9, 1\text{-}2^{(-9)})$, eliminando el caso en que los datos valgan -1.

1. Indique los formatos numéricos a la salida de los multiplicadores y a la salida del restador para que se compute la operación sin pérdida de precisión. Justifique su respuesta **(Figura 7.5)**.

Figura 7.5. Esquema Multiplicación Números Complejos



Las entradas con formato $S[10, 9]$ tienen un rango $(-1+2^{(-9)}\text{-}9, 1\text{-}2^{(-9)})$, por lo tanto el formato numérico de las salidas de los multiplicadores es de $S[19, 18]$. No se necesita el bit adicional para codificar el resultado 1 positivo (+1), puesto que este caso nunca se va a dar para codificar al

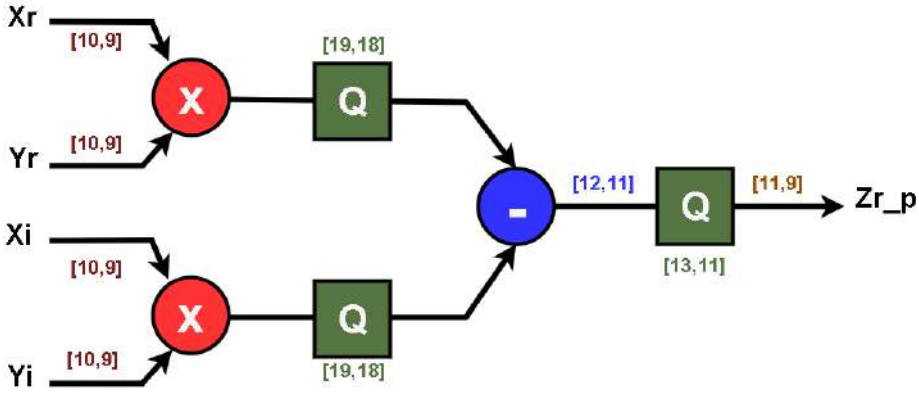
haber limitado el rango de las entradas, excluyendo el caso 1 negativo (-1). El resultado del restador puede crecer un bit, por lo tanto, su formato debe ser $S[20,18]$.

2. Suponga que el operador anterior se implementa atendiendo al siguiente modelo de precisión finita **(Figura 7.6)**:

   $Z_(r_p) = floor((floor(X_r * Y_r * 2^{11}) - (floor(X_i * Y_i * 2^{11})) * 2^{(-2)}) * 2^{(-9)}$

   Indique en que puntos del circuito se recorte la precisión y los formatos numéricos en dichos puntos.

Figura 7.6. Cuantificación Multiplicación Números Complejos



Las salidas de los multiplicadores con formato S[19,18] recortan el número de bits fraccionales para dejarlos en 11, por lo tanto, su formato es S[12,11]. El restador hace crecer los datos en un bit, por ello su formato es S[13,11] y sus bits se truncan para generar la salida con 9 bits fraccionales cuyo formato es S[11,9].

3. Modele con Verilog el operador del numeral 1, que opera con precisión completa.

```
module Mult_Zr(
input signed [9:0] Xr,Xi,Yr,Yi,
output signed [19:0] Zr);
```

```
wire signed [18:0] M1,M2; // M1

assign M1 = Xr * Yr; // M2
assign M2 = Xi * Yi; // Resta
assign Zr = M1 - M2;
endmodule
```

4. Modele con Verilog el operador del literal 2, que opera con pérdida de precisión.

```
module Mult_Zr_p(
input signed [9:0] Xr,Xi,Yr,Yi,
output signed [10:0] Zr_p);

wire signed [18:0] M1,M2;
wire signed [11:0] M1q,M2q;
wire signed [12:0] R; // M1

assign M1 = Xr * Yr;
assign M1q = M1[18:7]; // M2
assign M2 = Xi * Yi;
assign M2q = M2[18:7]; // Resta
assign R = M1q - M2q;
assign Zr_p = R[12:2];
endmodule
```

5. Suponiendo que el tiempo de propagación de un sumador es $t_{sum} = 1.5ns$, del multiplicador $t_{mult} = 3.5ns$, y que se consideran nulos los tiempos de t_{su} y t_{co} de los registros. Indique cuál es la frecuencia máxima de operación del operador 1 sin aplicarle segmentación y 2 aplicándole segmentación.

En el circuito no segmentado, el camino crítico es el de propagación a través de un multiplicador y el restador. La frecuencia máxima de operación es:

$$f_{max} = \frac{1}{t_{co} + t_{mult} + t_{sum} + t_{su}} \approx \frac{1}{t_{mult} + t_{sum}}$$

$$f_{max} = 200MHz$$

En el circuito segmentado, se registran las salidas de los multiplicadores y el camino crítico es el de propagación a través de un multiplicador, por ser este más restrictivo que el de propagación a través del restador. La frecuencia máxima de operación es:

$$f_{max} = \frac{1}{t_{co} + t_{mult} + t_{su}} \approx \frac{1}{t_{mult}} = 285.7MHz$$

7.3. EJERCICIO 3

Dada la siguiente ecuación en diferencias:

$$y(n) = [a(n) - b(n)] * [c(n) + (a(n-1) - b(n-1))]$$

1. Dibuje la arquitectura paralela con la que implementar el algoritmo con el menor número de recursos *hardware* que sea posible **(Figura 7.7)**.

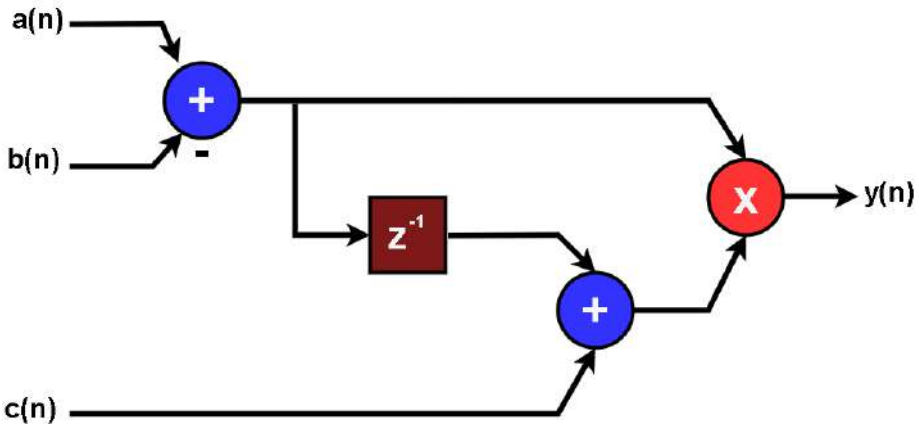
   La ecuación en diferencias se puede descomponer de la siguiente manera:

   $$s(n) = a(n) - b(n)$$

   $$y(n) = s(n) * [c(n) + s(n-1)]$$

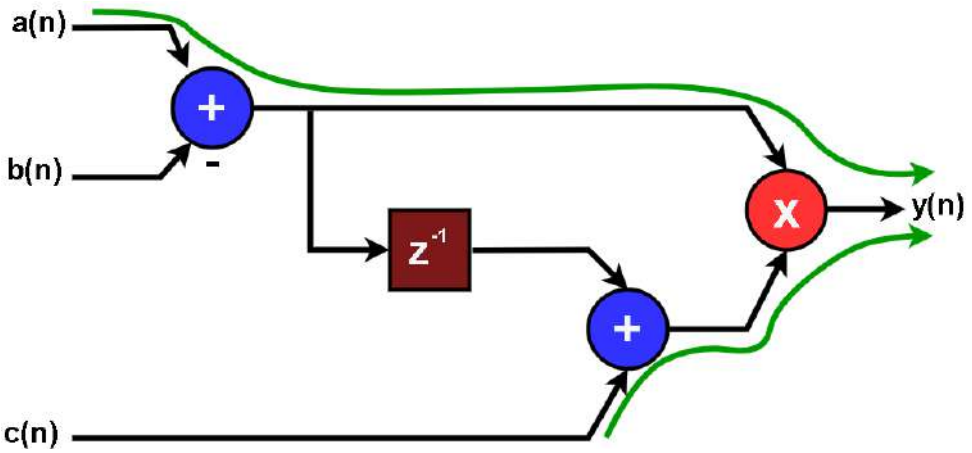
   Se puede aprovechar la salida s(n) para ahorrar un sumador.

Figura 7.7. Esquema Ecuación de Diferencias – Arquitectura Paralela



2. Dibuje el camino crítico del circuito y halle la máxima frecuencia de funcionamiento sabiendo que $t_{mult} = 3.5ns$ y $t_{add} = 1.5ns$ NOTA: considere nulos los tiempos de *set-up* y propagación de los registros para resolver este ejercicio **(Figura 7.8)**.

Figura 7.8. Camino Crítico Ecuación de Diferencias – Arquitectura Paralela

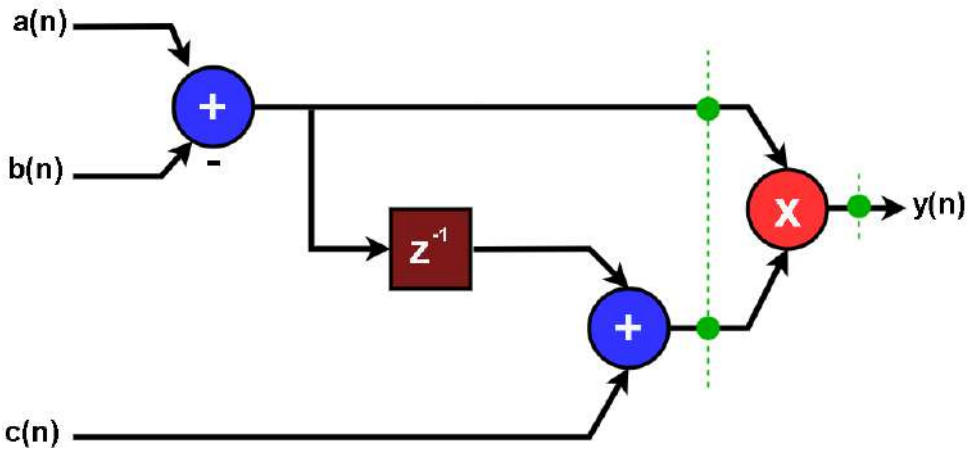


El camino crítico estará formado por un sumador y un multiplicador. Por lo tanto, la frecuencia máxima será:

$$f_{max} = \frac{1}{3.5ns + 1.5ns} = 200MHz$$

3. Aplique las técnicas de segmentación adecuadas sobre la arquitectura paralela anterior para obtener máxima velocidad. ¿Cuál será la máxima frecuencia de funcionamiento tras segmentar? **(Figura 7.9)**.

Figura 7.9. Segmentación Ecuación de Diferencias – Arquitectura Paralela



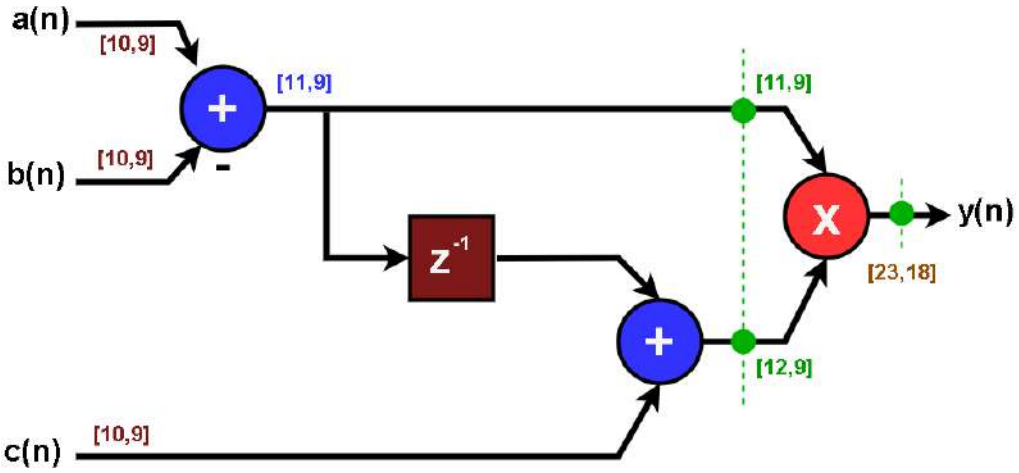
Se puede segmentar a la salida del sumador inferior y a la salida del multiplicador, tal y como se muestra en la **Figura 7.9**. De este modo, el camino crítico se reduce a un multiplicador. La máxima frecuencia de reloj será:

$$f_{max} = \frac{1}{t_{mult}} = \frac{1}{3.5ns} = 285.7MHz$$

4. Dimensione cada uno de los operadores e indique a su salida el formato numérico necesario para no perder precisión, sabiendo que el formato de a, b y c es [10,9] con signo codificado en complemento a dos **(Figura 7.10)**.

Tal y como se muestra en la figura, serán necesarios los siguientes operadores:

Figura 7.10. Cuantificación Ecuación de Diferencias – Arquitectura Paralela



- 1 sumador de 11 bits
- 1 sumador de 12 bits
- 1 multiplicador de 22 bits
- Registros: 11*3+22=55

5. Halle el número de recursos lógicos necesarios para implementar el algoritmo segmentado en un dispositivo FPGA de Altera Cyclone IV.

   Cada sumador de n bits requerirá de n LE. El sintetizador utiliza el registro de salida de cada LE para implementar el sumador con salida registrada. Por lo tanto, se obtiene:

   - 2 sumadores de 11 bits y 12 bits = 23 LE (el sumador inferior tiene las salidas registradas).
   - El operador multiplicador se implementa utilizando un multiplicador embebido incluyendo sus registros de salida.
   - Los registros que quedan se implementan utilizando 1 LE por cada registro.

Registros: 11+12=23 LE solo para registros.

6. Si se desea cuantificar la salida con 16 bits. ¿Cuál será el formato numérico de la señal de salida? Escriba cómo se modela con Matlab la aplicación de dicho formato en la salida "y" (con precisión completa) para obtener y q (salida con 16 bits).

   La señal de salida tiene formato [23,18]; por lo tanto, tiene 5 bits de parte entera. Al cuantificar con 16 bits, se deben respetar los 4 bits enteros para evitar que se produzca desbordamiento. Luego el formato de salida será [16,11].

   Esta operación se modelará en Matlab de la siguiente forma:

$$yq = floor(y * 2^{11}) * 2^{-11}$$

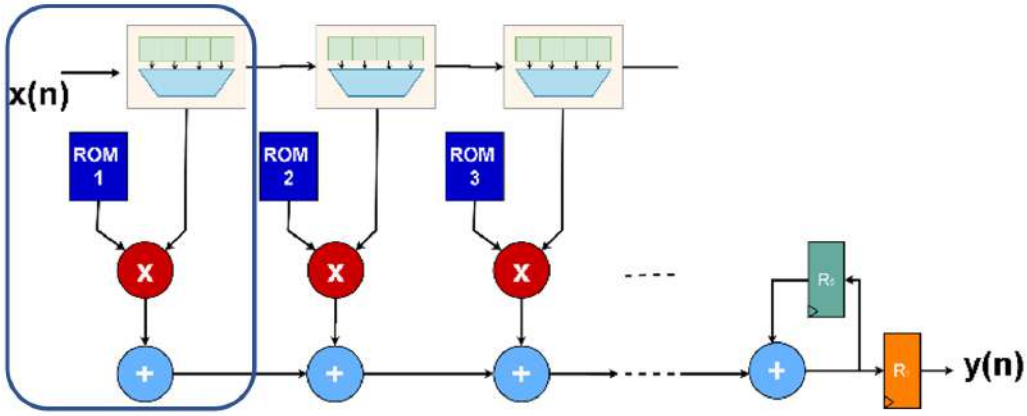
7.4. EJERCICIO 4

Se quiere implementar un filtro FIR de 300 coeficientes utilizando una arquitectura en cascada como se muestra en la **(Figura 7.11)**. Se desea trabajar a una frecuencia de muestreo de $f_S = 100\ Msps$ con una FPGA que contiene 116 multiplicadores embebidos capaces de trabajar a 350 MHz.

Los datos de entrada se cuantifican con 14 bits (formato $[14, 13]$) y los coeficientes con 10 bits (formato$[10,9]$), ambos con signo.

1. ¿Cuál debe ser la frecuencia de reloj (f_{clk}) del filtro? ¿Cuántas celdas como la marcada en la **Figura 7.11** serán necesarias para implementar el filtro utilizando el menor número de recursos?
   Teniendo en cuenta las especificaciones de velocidad, serán necesarios 3,5 ciclos para realizar el filtrado:

Figura 7.11. FIR 300 coeficientes – Arquitectura en Cascada



$$\#\ ciclos = \frac{f_{clk}}{f_s} = \frac{350MHz}{100MHz} = 3.5\ ciclos$$

$$\frac{300\ coeficientes}{3\ ciclos} = 100\ mutiplicaciones\ x\ ciclo$$

$$100\ mutiplicaciones\ < 116\ multiplicadores\ FPGA$$

Por lo tanto, se necesitan 100 celdas como la marcada en la figura y la frecuencia de reloj del filtro debe ser:

$$f_{clk} = 3 * f_s = 300\ MHz.$$

2. ¿Cuál será el tamaño de las memorias ROM que almacenan los coeficientes?
   Se debe almacenar 3 coeficientes de 10 bits en cada una de las memorias por lo que se necesita almacenar en cada memoria 3 palabras de 10 bits. El tamaño de las ROM será de 4x10 bits.

3. Suponga que los coeficientes se enumeran de la siguiente forma: $h_0, h_1, h_2, \ldots, h_{299}$. Indique cómo se rellenaría las memorias de las 2 primeras cel-

das indicando en cada memoria sus direcciones y contenidos almacenados en dichas direcciones.
En cada memoria se almacenarán 3 coeficientes consecutivos:
En la ROM_0 :

$$dir_o \gg h_0,\ dir_1 \gg h_1,\ dir_2 \gg h_2;$$

En la ROM_1 :

$$dir_o \gg h_3,\ dir_1 \gg h_4,\ dir_2 \gg h_5;$$

En todas las memorias $dir_3 \gg 0$

4. ¿Cuál será el tamaño de los registros de desplazamiento que almacenan los datos x(n)?
   Habrá que incluir un desplazamiento de 3 datos entre cada celda, por lo tanto será necesario implementar 3 registros de 14 bits entre cada celda.

5. Dimensione el tamaño necesario en los sumadores de las celdas y en el acumulador final para que el filtro pueda operar con precisión completa sin desbordamiento.
   Crecimiento Sumadores:

$$log_2\, 100 = 6,6\, bits \approx 7\, bits$$

   Crecimiento Acumulador:

$$log_2\, 3 = 1,5\, bits \approx 2\, bits$$

   Crecimiento Sumadores:

$$14 + 7 + 2 = 23\, bits$$

6. ¿Cuál será el camino crítico, frecuencia máxima de funcionamiento y la latencia del filtro sin segmentar? Suponga que $t_{mult} = 2.8ns$, $t_{add} =$

$1.5ns$, $t_{ROM} = 0.5ns$ y $t_{mux} = 0.6ns$ y considere nulos los tiempos de *set-up* t_{su} y propagación t_{co} de los registros para resolver este ejercicio.

Dado que la primera celda contiene un sumador con un operando fijo a cero, el sintetizador no lo implementará. Por lo tanto, se tendrá 99 sumadores más el acumulador, o sea 100 sumadores en total.

El camino crítico estará formado por 1 multiplexor + 1 multiplicador + 100 sumadores. La frecuencia máxima del circuito será:

$$f_{max} = \frac{1}{t_{mux} + t_{mult} + 100 * t_{add}} = \frac{1}{153.4ns}$$

$$f_{max} = 6.5MHz$$

Se necesitan 3 ciclos para obtener el resultado por lo que la frecuencia de máxima de muestreo será:

$$f_s = \frac{6.5 \times 10^6\ MHz}{3} = 2.16\ MHz$$

Para calcular la latencia, se debe considerar que se trata de una implementación secuencial, que necesitará de 3 ciclos para computar la primera muestra. Por lo tanto, la latencia será de 3 ciclos.

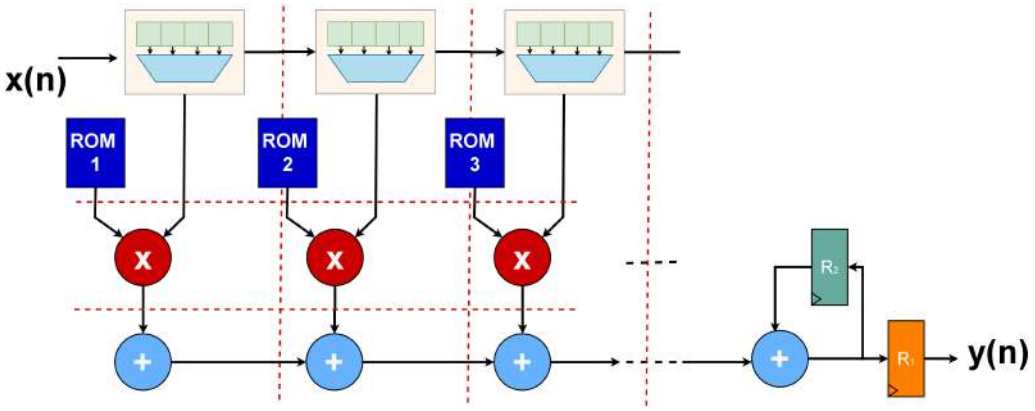
7. ¿Cuál será la máxima frecuencia de muestreo que podría alcanzar el circuito sin segmentar?
   Como se necesitan 3 ciclos para obtener el resultado, la frecuencia de máxima de muestreo será:

$$f_s = \frac{6.5 \times 10^6\ MHz}{3} = 2.16\ MHz$$

8. Indique cómo se debe segmentar el circuito anterior para que pueda alcanzar la frecuencia de muestreo requerida **(Figura 7.12)**.

   Se puede segmentar en horizontal en los siguientes puntos:

Figura 7.12. FIR Segmentado – Arquitectura en Cascada



- Salida de las memorias y del multiplexor.
- Salida de los multiplicadores.

Se puede segmentar en vertical a la salida de cada sumador, lo cual origina que se incluya un *flip-flop* más en la línea de datos.

9. ¿Qué latencia presentará el circuito segmentado?

- Dado que se trata de una implementación secuencial, son necesarios 3 ciclos de latencia para obtener el primer resultado.
- Teniendo en cuenta la segmentación dibujada en el apartado anterior, se estarán incluyendo 2 + 100 ciclos de latencia.

En total, el circuito presentará una latencia de:

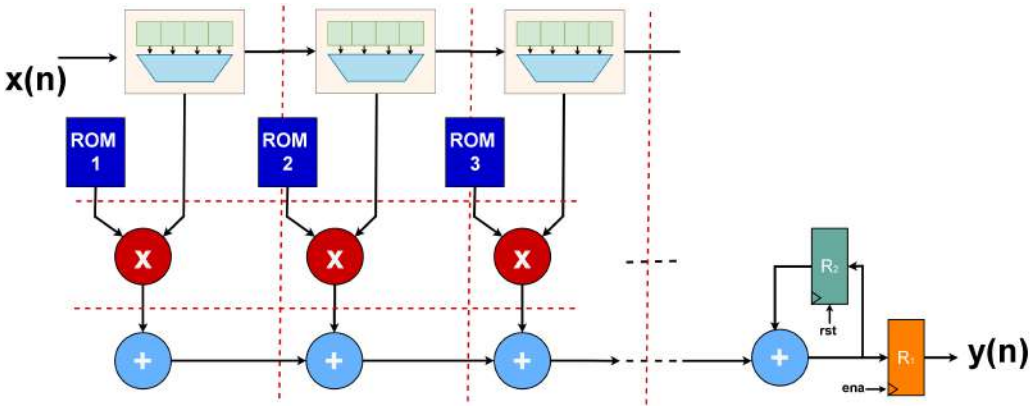
$$3 + 2 + 100 = 105 ciclos$$

10. ¿Sera necesario que algunos de los registros R1 y R2 tenga entrada de "reset"? Justifique su respuesta y explique cada cuántos ciclos de reloj se debe activar ese *reset* en caso de que sea necesario.

Se necesita una entrada de habilitación en el registro R1 para capturar el dato de salida; la señal de *enable* estará a nivel alto durante un ciclo cada tres ciclos de reloj.

El circuito segmentado, con el *reset* y *enable* se muestra en la **Figura 7.13**.

Figura 7.13. Esquema señales ENA-RST FIR

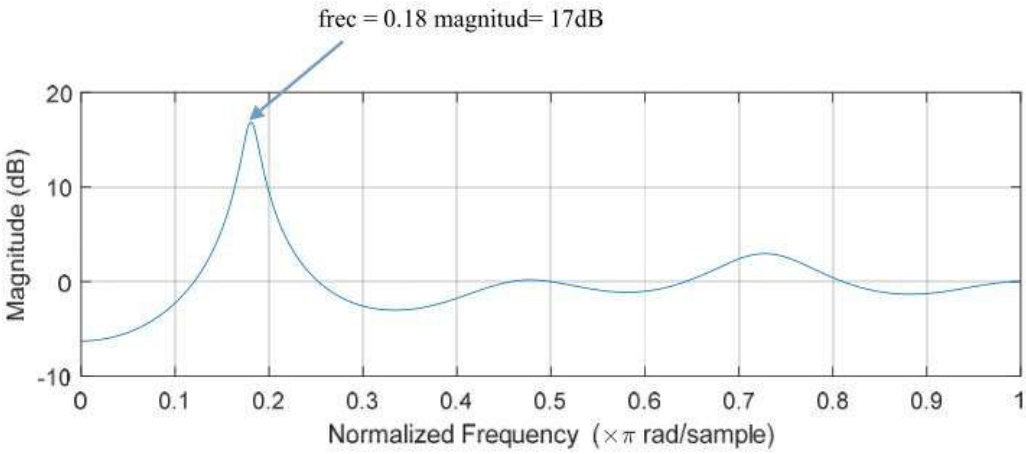


7.5. EJERCICIO 5

Se ha diseñado un filtro recursivo paso banda cuya ecuación en diferencias y respuesta en frecuencia se muestran en la **Figura 7.14**:

$$y(n) = x(n-2) - a_0 * y(n-4) - a_1 * y(n-5) - a_2 * y(n-6) - a_3 * y(n-7)$$

Figura 7.14. Respuesta en Frecuencia FIR Recursivo



1. Dibuje la arquitectura paralela con la que implementar el algoritmo anterior **(Figura 7.15)**.

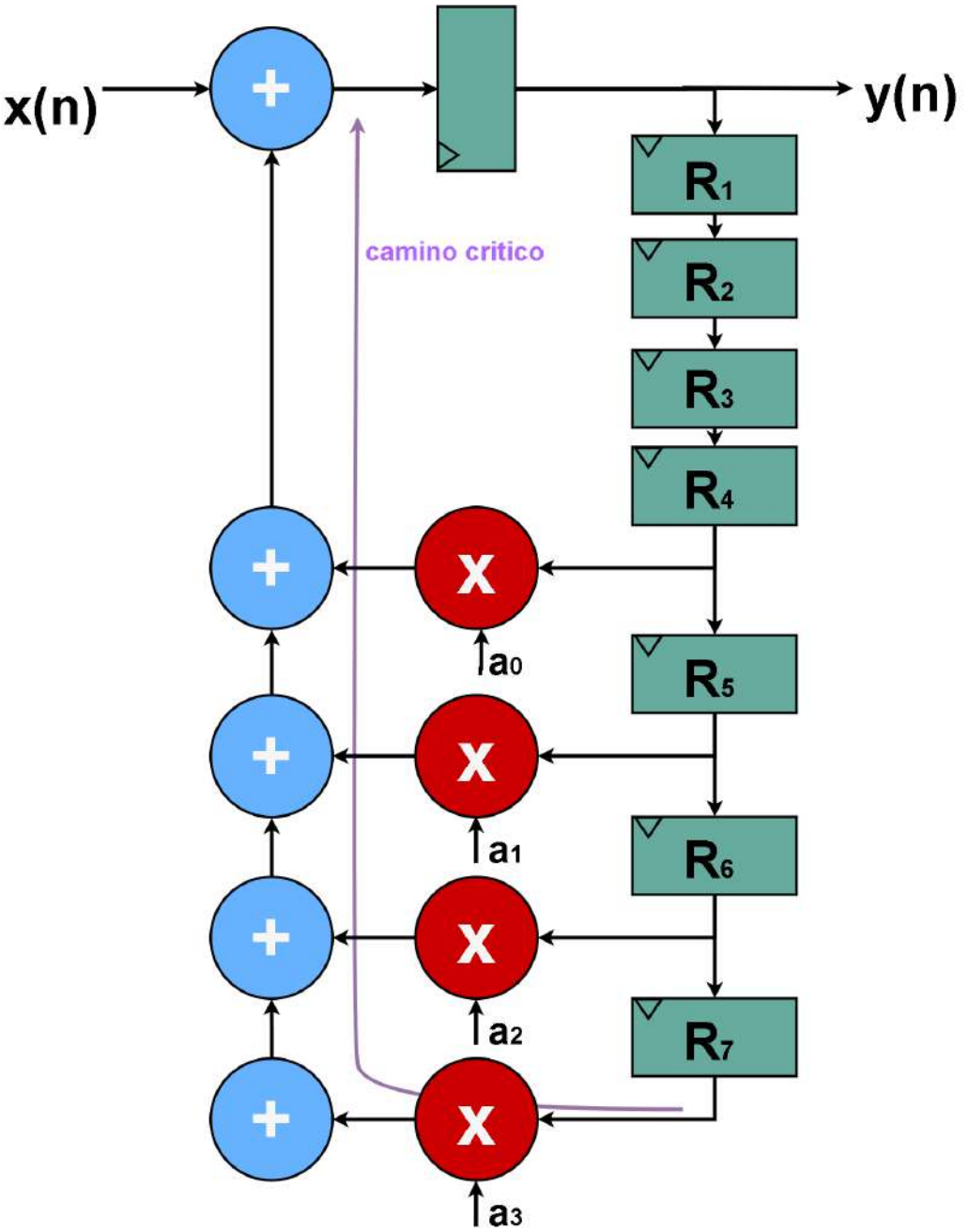
2. Sobre el esquema anterior, dibuje el camino crítico del circuito y halle la máxima frecuencia de funcionamiento sabiendo que $t_{mult} = 3.5ns$ y $t_{add} = 1.5ns$ NOTA: considere nulos los tiempos de *set-up* y propagación de los registros para resolver este ejercicio (Figura 7.15).

   El camino crítico es el que atraviesa el multiplicador cuyo coeficiente es a_3 y los cuatro sumadores en cascada.

   El tiempo de propagación de dicho camino es:

$$t_p = t_{mult} + 4 * t_{add} = 3.5 + 4 * 1.5 = 9.5ns$$

Figura 7.15. Esquema FIR Recursivo – Arq. Paralela



Por lo tanto, la máxima frecuencia de funcionamiento del circuito es:

$$f_{clk} = \frac{1}{9.5ns} = 105.3MHz$$

3. ¿Se puede aumentar la frecuencia de funcionamiento del filtro mediante la aplicación de alguna técnica arquitectural? Si su respuesta es positiva, dibuje la arquitectura del filtro resultante de la aplicación de dichas técnicas e indique cuál será la máxima frecuencia de funcionamiento **(Figura 7.16)**.

   Dado que se tiene varios registros concatenados, se puede utilizar *retimming* para reducir el camino crítico, tal y como se muestra en la Figura 7.16.

   De este modo, el camino crítico recorrerá un solo multiplicador, por lo que la frecuencia máxima de funcionamiento será:

$$t_p = \frac{1}{t_{mult}} = \frac{1}{3.5ns} = 285.7MHz$$

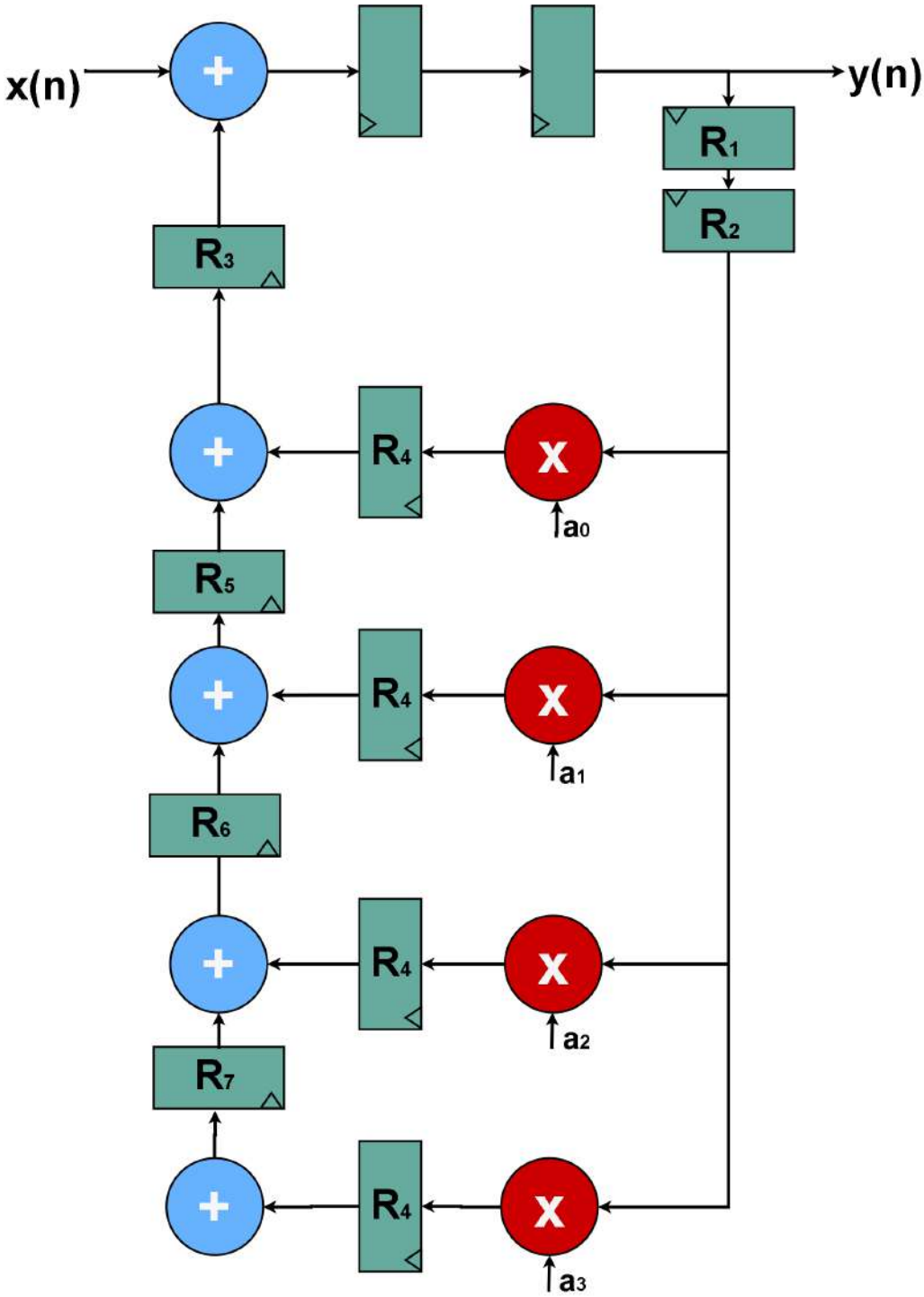
4. Se desea cuantificar la entrada de datos y los coeficientes en complemento a dos con formato [10,9]. Se debe cuantificar la salida de forma que se evite el desbordamiento y se mantengan 9 bits fraccionales de precisión a la salida, mientras que internamente el filtro deberá operar con 12 bits fracciones de precisión. Cuantifique todos los operadores internos del filtro. Si en alguna conexión entre operadores se requiere cambiar el formato indíquelo con un bloque Q y escriba el nuevo formato a su salida **(Figura 7.17)**.

   Dado que el filtro tiene una ganancia máxima de 17 dB, se puede calcular el crecimiento de la ruta de datos de la siguiente manera:

$$G = ceil(log2(10\text{^}(17/20))) = 3$$

   Por lo que serán necesarios 3 bits de crecimiento en la parte entera. Por ello, el formato de salida del filtro deberá ser sfix[13,9].

Figura 7.16. Optimización FIR Recursivo (*retimming*)



Se solicita trabajar internamente con 12 bits de precisión, por lo que:

- El formato de la ruta de datos deberá ser $sfix(16, 12)$ dado que se necesitan 4 bits para representar la parte entera.
- A los multiplicadores le llegará un dato $sfix[16, 12]$ que se multiplicará por un coeficiente $sfix[10, 9]$ por lo que su salida con precisión completa será $sfix[26, 21]$.
- Como se solicita trabajar con 12 bits de precisión, se deberá recortar 6 bits fraccionales.
- La cuantificación a la salida de los multiplicadores será $sfix[16, 12]$ (se cogen los 16 bits [24:9]). Se representa este cuantificador con el bloque Q2.
- Los sumadores serán todos de 16 bits.
- El cuantificador Q1 deberá elegir los bits [15:3] para quedarse con el formato deseado $(sfix[13, 9])$.

7.6. EJERCICIO 6

Partiendo de la ecuación en diferencias del ejercicio anterior, se desea realizar una implementación serie para conseguir el mínimo uso de recursos. En la **Figura 7.18**, se muestra el esquema de la implementación serie. Cada uno de los registros presentes en el circuito poseen una entrada de habilitación ce y una de *reset* rst.

$$y(n) = x(n-2) - a_0 * y(n-4) - a_1 * y(n-5) - a_2 * y(n-6) - a_3 * y(n-7)$$

1. Explicar el funcionamiento del circuito secuencial propuesto indicando claramente cuántos ciclos se necesitan para realizar el cómputo completo y qué operación se realiza en cada ciclo.

Figura 7.17. Cuantificación Filtro Recursivo

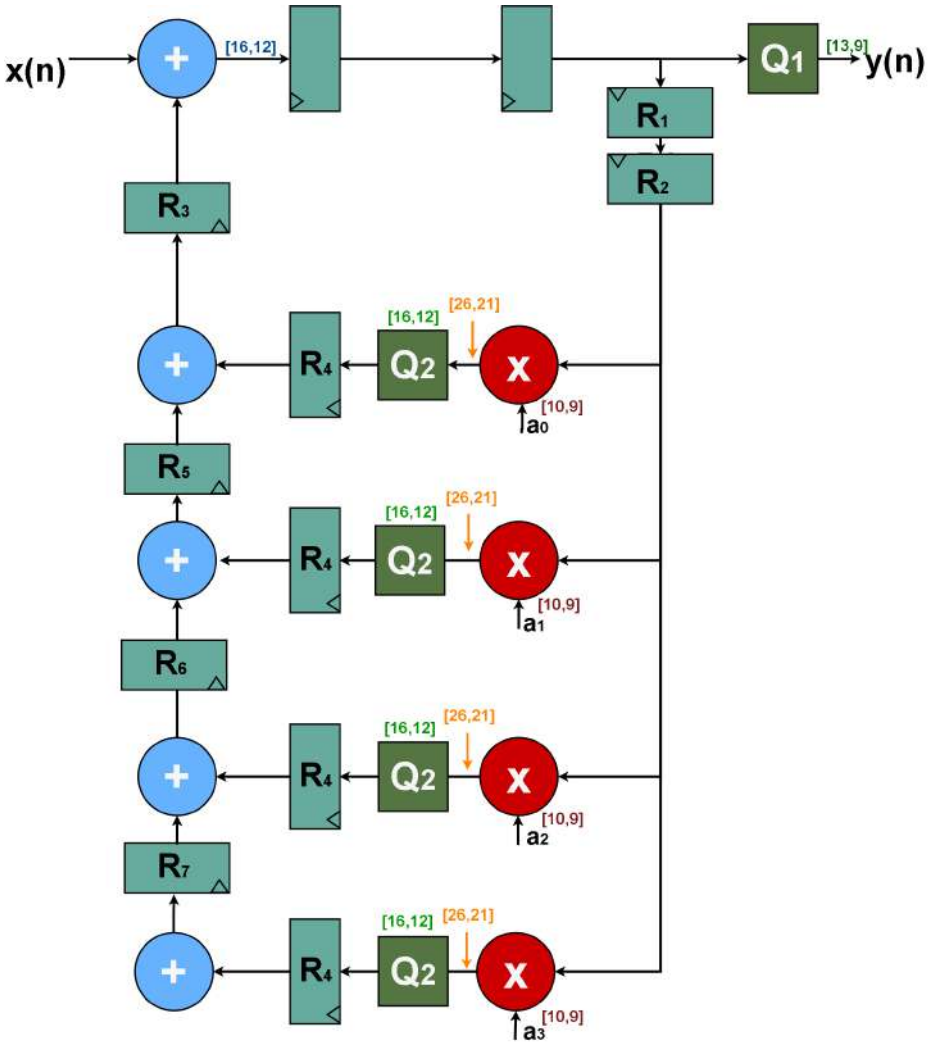
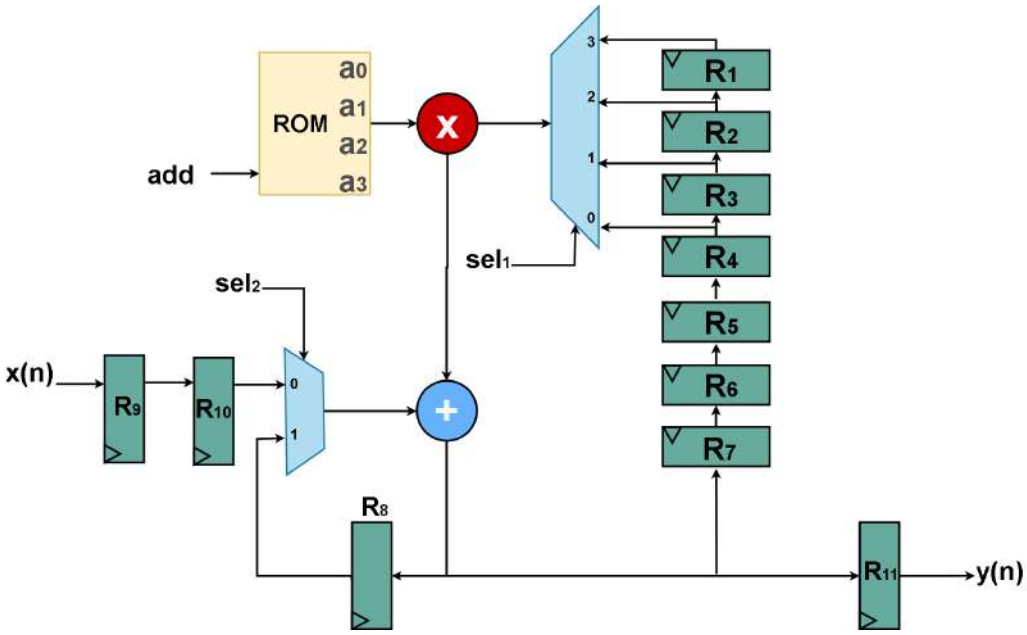


Figura 7.18. Implementación Serie Filtro Recursivo



El circuito propuesto utilizará cuatro ciclos para computar la muestra $y(n)$. En el primer ciclo, se realizará la suma de $x(n-2)+a_0*y(n-4)$, de manera que se selecciona la entrada 0 del multiplexor controlado por $sel2$.

En los siguientes ciclos se realiza la acumulación del resultado del ciclo anterior más a $x*y(n-(4+x))$ siendo $x=[1,2,3]$. Para ello se debe seleccionar la entrada 1 del multiplexor.

2. Cada uno de los registros presentes en el circuito poseen una entrada de habilitación ce y una de *reset* rst. Justifique, para cada registro, cómo tienen que ser estas líneas de control.

   Los registros $R1$ a $R7$ se utilizan para desplazar los datos de la realimentación. Deberán trabajar a la frecuencia de muestreo por lo que necesitarán una señal de habilitación $ce1$ que controle el desplazamiento

de los datos cada $\frac{f_{clk}}{4}$. Estos registros llevarán un *reset* $rst1$ que solo se activará al inicializar el circuito.

El registro $R8$ es el que se encarga de acumular los productos parciales. Este registro necesita realizar un *reset* $rst2$ cuando comience el cómputo de una nueva muestra. La entrada de habilitación de este registro $ce2$ estará siempre a nivel alto, permitiéndole funcionar a $fclk$.

Los registros R9 y R10 se utilizan para cargar los datos de entrada x(n). Estos registros deberán funcionar a $f_s = \frac{f_{clk}}{4}$ por lo que necesitan una señal de habilitación que lo permita ce1. El *reset* de estos registros se utilizará para inicializar el circuito (igual que rst1).

El registro R11 se utiliza para obtener la salida y se controla de la misma manera que R1-R7; de hecho, podría evitarse este registro obteniendo la salida directamente del registro R7.

Resumiendo:

- $R1 - R7$ y $R9 - R11$ se controlan con $ce1$ y $rst1$
- $R8$ se controla con $ce2$ y $rst2$

3. ¿Cuál será el tamaño de la memoria ROM que almacenará los coeficientes si se cuantifican con 10 bits? ¿Cuántas LUT se requieren si se implementase como memoria distribuida de un dispositivo FPGA Cyclone V?

   Se necesita almacenar 4 coeficientes de $10\ bits$ cada uno, por lo que el tamaño de la memoria será $2^2 x 10\ bits$.

   Dado que la memoria es pequeña, se podría implementar de manera distribuida, usando $10\ LUT$ de $2\ entradas$.

4. ¿Cuál es la frecuencia máxima de funcionamiento del circuito anterior?

Considere los siguientes tiempos: $t_{add} = 1.5ns$,$t_{mux} = 2ns$,$t_{mult} = 3.5ns$ y $t_{ROM} = 2ns$. ¿Cuál será la frecuencia máxima de muestreo del circuito? Justifique la respuesta.

El camino crítico recorre 1 multiplexor, 1 multiplicador y 1 sumador, por lo que la frecuencia máxima será:

$$fclk_{max} = \frac{1}{2ns + 3.5ns + 1.5ns} = \frac{1}{7ns} = 142.8MHz$$

El circuito necesita cuatro ciclos para completar el cálculo de y(n) por lo que la frecuencia máxima de muestreo será:

$$fs_{max} = \frac{fclk_{max}}{4} = 35.7MHz$$

5. ¿Es posible segmentar el circuito para alcanzar mayor frecuencia de muestreo?

Si se segmenta a la salida del multiplexor, la salida de la ROM y la salida del multiplicador, el camino crítico vendrá dado por el operador más lento, que en este caso es el multiplicador. De este modo se obtendrá:

$$fclk_{max} = \frac{1}{3.5ns} = 285.7MHz.$$

Con esta segmentación, la latencia del circuito aumenta en dos ciclos por lo que la frecuencia de muestreo máxima será:

$$fs_{max} = \frac{fclk_{max}}{4 + 2} = \frac{285.7}{6} = 47.6MHz$$

Por lo que sí se consigue mejorar la frecuencia de muestreo al segmentar.

6. Explique claramente el funcionamiento de todas las señales de control que se utilizarán para controlar el circuito sin segmentar. Realice una tabla en la que indique el valor de todas las líneas de control en cada ciclo **(Tabla 7.2)**.

La señal de habilitación de los registros $ce1$ se activará cada cuatro ciclos para permitir el desplazamiento de los datos de la realimentación $y(n-x)$, la entrada de un nuevo dato $x(n-2)$ y la salida del resultado $y(n)$. Esta señal se activará durante el último ciclo, en el cual se han calculado las cuatro acumulaciones.

Por otro lado, el registro del acumulador $R8$ necesitará limpiar la acumulación cada cuatro ciclos para evitar que se afecte al cálculo de la siguiente muestra. Esta operación se realizará utilizando el $reset$ del registro rst_2. Se realiza este *reset* en el primer ciclo.

La memoria y el multiplexor de los datos se direccionarán de la misma manera, seleccionando consecutivamente los coeficientes a_0, a_1, a_2 y a_3, a la vez que se seleccionan los datos $y(n-4), y(n-5), y(n-6)$ y $y(n-7)$. Por lo tanto, las líneas de control add y $sel1$ son idénticas.

Tabla 7.2. Señales de Control

ciclo	add	sel_1	sel_2	cel_1	rst_1	ce_2	rst_2
0	00	00	0	0	0*	1	1
1	01	01	1	0	0	1	0
2	10	10	1	0	0	1	0
3	11	11	1	1	0	1	0

* sólo se activa para inicializar el circuito completo

7. Partiendo de la **Tabla 7.2** del numeral anterior, haga las simplificaciones que considere oportunas para incluir el control en una memoria del menor tamaño posible **(Tabla 7.3)**.

Se puede reducir el número de señales de control ya que:

- add y sel_1 son iguales.
- ce_2 siempre está a 1 (no hace falta ponerla).
- rst_2es la inversa de sel_2.

Tabla 7.3. Memoria Simplificada

Posición	Valor	add	sel_2	cel_1	rst_2
0	00001	00	0	0	1
1	01100	01	1	0	0
2	10100	10	1	0	0
3	11110	11	1	1	0

Por lo tanto se puede reducir a 5 bits de control add(2 bits), sel_2, ce_1, rst_2 , aunque se podría reducir a 4 ya que $rst_2 = not(sel_2)$.

El contenido de la memoria será el siguiente:

8. Trabajando con el circuito sin segmentar, se desea aplicar la misma cuantificación que en el literal 4 del ejercicio anterior. Cuantificar la salida de todos los operadores internos del filtro e indique claramente qué bits se seleccionan como salida del filtro **(Figura 7.19)**.

   La ganancia máxima del filtro es $17\,dB$ por lo que necesita $ceil(log_2 10^{\frac{17}{20}}) = 3\,bits$ para evitar el desbordamiento a la salida del filtro.

El formato de la ruta de datos será $[16, 9]$, los $9\,bits$ de precisión más los $4\,bits$ enteros que se necesita para evitar el desbordamiento).

Al multiplicador le llega un dato $[16, 12]$ que se multiplicará por un coeficiente $[10, 9]$.

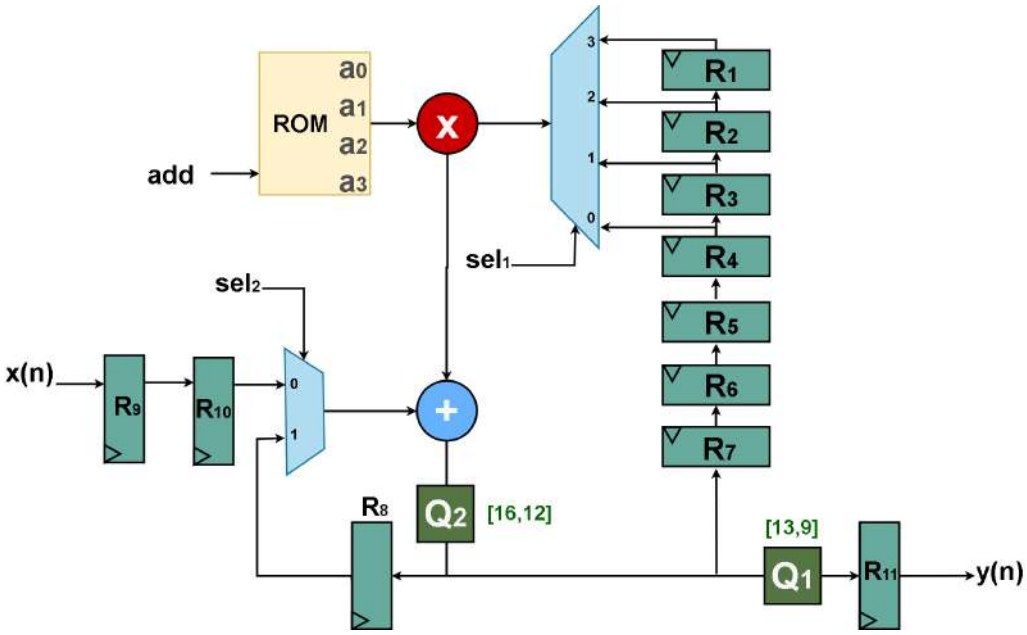
Esto entrega una cuantificación de salida del multiplicador $[26, 21]$. Como se va a trabajar con $12\,bits$ de precisión y $4\,enteros$ se quedarán con los bits $[24, 9]$ obteniendo un formato $[16, 12]$.

El acumulador será de $16\,bits$.

7.7. EJERCICIO 7

Se pretende modelar con el lenguaje Verilog un filtro FIR secuencial tal como se muestra en el esquema de implentacion del filtro (**Figura 7.20**), que compense la respuesta en frecuencia de un Filtro CIC. La frecuencia de muestreo de

Figura 7.19. Cuantificación FIR – Serie

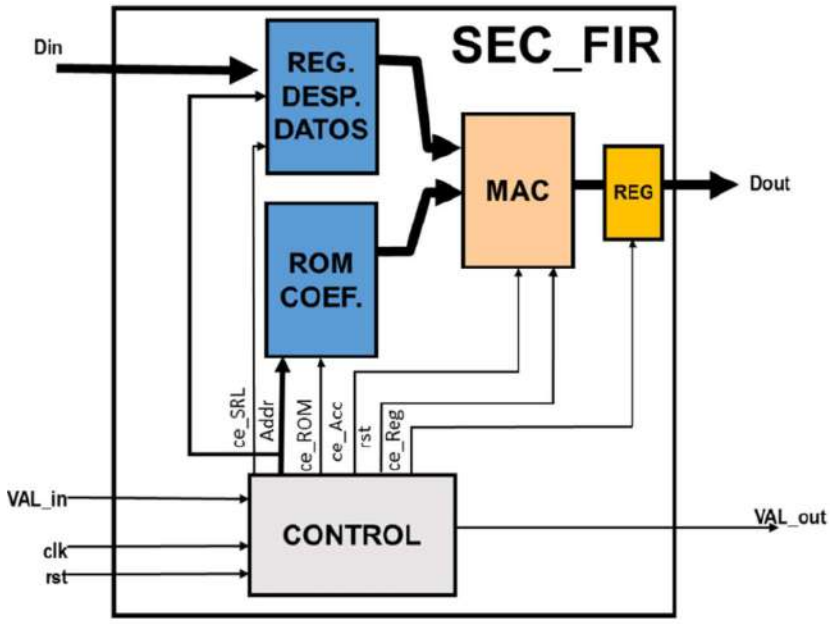


los datos de entrada será de **50 kHz**, mientras que la frecuencia de reloj del filtro será de **100 MHz**.

El módulo **SEC_FILTER** dispone de los siguientes puertos:

- Din: entrada del dato cuantificado con formato *signed* [**Win, Win-1**], siendo **Win=16 bits**.
- VAL_in: entrada binaria que informa de que existe una muestra válida en **Din**.
- rst: *reset* del sistema.
- clk: entrada de reloj.
- Dout: salida del dato filtrado, con formato *signed* [**Wout, Wout-3**] siendo **Wout=19 bits**.
- VAL_out: salida binaria que informa de que existe una muestra válida en **Dout**.

Figura 7.20. Esquema Filtro FIR Compensador CIC

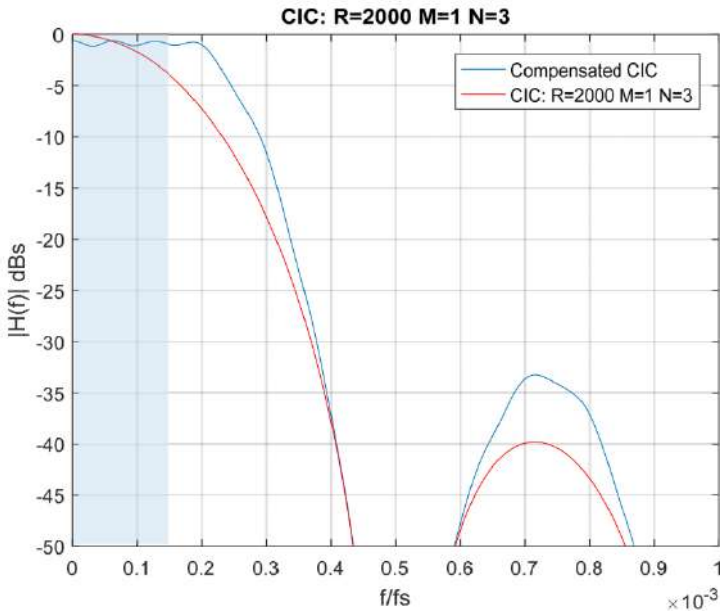


Las especificaciones del Filtro son:

- Arquitectura secuencial
- Número de etapas: **N=17**
- Coeficientes simétricos
- Tamaño de la entrada: **Win = 16 bits**
- Tamaño de la salida **Wout = 19 bits**
- El modelo del filtro deberá ser sintetizable y podrá ser implementado en un dispositivo FPGA Cyclone IV EP4CE115F29C7 funcionando a una frecuencia de reloj de **125 MHz.**
- El ancho de banda de la señal de entrada será de **15 kHz.**

La respuesta en frecuencia de los filtros CIC es muy mala a altas frecuencias. Para mejorarla, se implementan filtros compensadores previos a los filtros CIC que aplican una ganancia más alta conforme crece la frecuencia. El resultado de aplicar filtros compensadores es bastante notable, como muestra la **Figura 7.21**:

Figura 7.21. Respuesta en Frecuencia CIC Compensado y Sin Compensar



La implementación de este filtro es posible realizarla mediante arquitectura secuencial por:

$$f_s = 50kHz$$

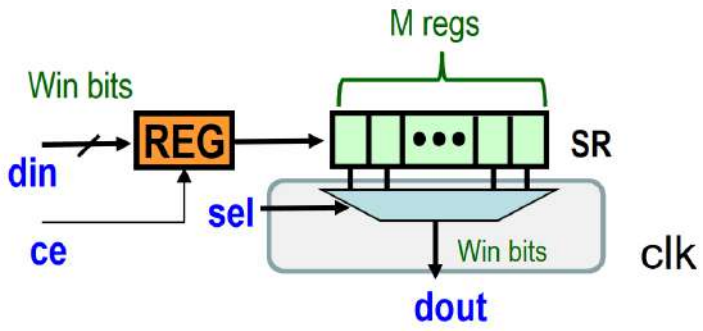
$$f_{clk} = 100MHz$$

$$\#Max\ iter = \frac{f_{clk}}{f_s} = \frac{100\ MHz}{50\ kHz} = 2000$$

Se tiene en cuenta que el filtro presenta **17 etapas**, más que suficiente con respecto a las **2000** posibles, por lo que no es necesaria otro tipo de arquitectura, como paralela o semisecuencial.

REG_MUX: incluye el registro de desplazamiento y el multiplexor de salida **Figura 7.22**. Se parametriza el número de bits del dato de entrada **Win**. Posee dos entradas de control, una para seleccionar el dato **Addr** y otra para habilitar el registro de desplazamiento (**ce_SRL**, activo a nivel alto).

Figura 7.22. Diagrama REG_MUX



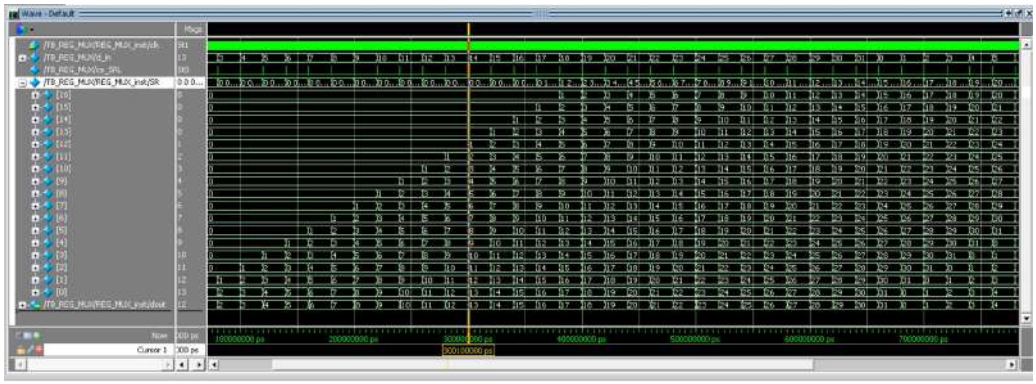
El código del módulo **REG_MUX.v** programado es:

```
module REG_MUX
#(parameter Win=16,
parameter Num = 17)
(input signed [Win-1:0] din ,
input [4:0] sel,
input clk,
input ce,
output reg signed [Win-1:0] dout);

reg signed [Win-1:0] SR [Num-1:0];
integer n;

//Inicializacion
initial begin
for ( n = Num-1; n>-1; n = n-1)
```

```
SR[n] = 16'b0;
end

//Registro de Desplazamiento
always @ (posedge clk)
begin
if (ce == 1'b1)
begin
for (n =Num-1; n>0; n = n-1)
begin
SR[n] <= SR[n-1];
end
SR[0] = din;
end
end

//Multiplexor
always @(posedge clk)
dout = SR[sel];
endmodule
```

Como se muestra en la **Figura 7.23**, se comprueba el correcto funcionamiento del módulo:

Figura 7.23. Diagrama MULT_ACC



MULT_ACC: el multiplicador-acumulador(**Figura 7.24**) parametriza el tamaño de las dos entradas (**Win:** cuantificación de los datos, **Wc:** cuantificación de los coeficientes). Posee dos señales de control, una para hacer el *reset* del acumulador (síncrono y activo a nivel alto) y otra para habilitar el inicio de la acumulación cada vez que entra una nueva muestra (**ce_Acc**, activo a nivel alto).

Para el dimensionamiento de las operaciones, se tiene en cuenta el crecimiento de los bits: en la multiplicación, la suma de los bits de los **2** operandos **(Win+Wc)**; y en el acumulador, se toma en cuenta la ganancia del filtro calculada como el sumatorio del absoluto de los coeficientes cuantificados **[18,15]**, quedando:

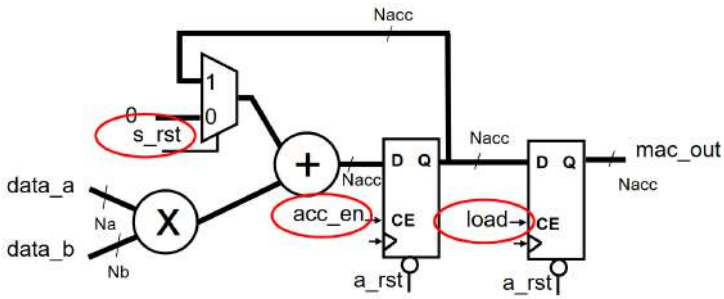
$$log2(sum(abs(h\_q))) = log2(2,4379) = 1,2856 \approx 2\ bits\ de\ crecimiento$$

Cuantificando el acumulador con (Win+Wc+2) bits.

El código del módulo **MULT_ACC.v** programado es:

```
module MULT_ACC
#(parameter Win=16,
parameter Wc = 18)
(input signed [Win-1:0] din ,
input signed [Wc-1:0] coef ,
input clk,
```

Figura 7.24. Test Bench Módulo REG_MUX

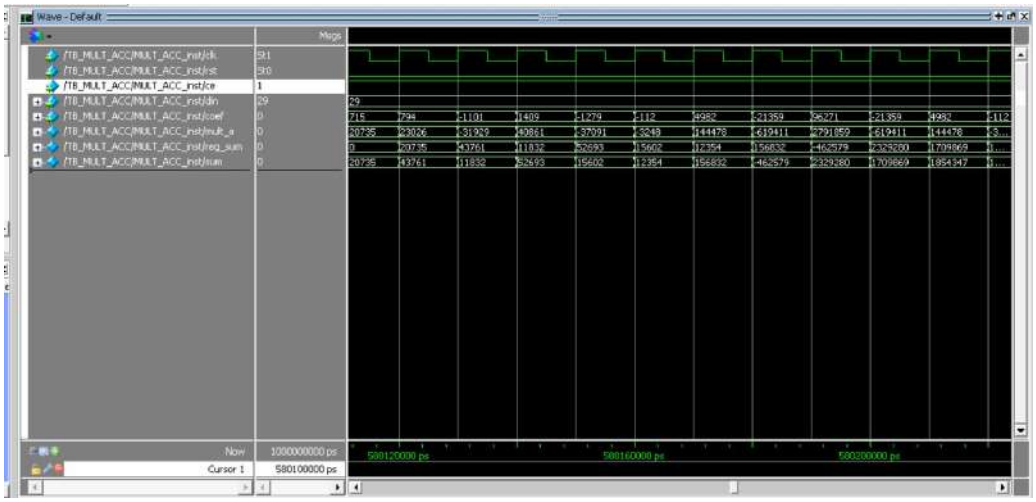



```
input ce,
input rst,
output signed [Wc-1:0] dout);

wire signed [Win+Wc-1:0] mult_a;
reg signed [Win+Wc+1:0] reg_sum=0;
wire signed [Win+Wc+1:0] sum;

assign mult_a = din * coef;
assign sum = !ce? reg_sum : mult_a + reg_sum;

always @ (posedge clk)
begin
if (rst)
reg_sum = 0;
else
reg_sum = sum;
end
assign dout = reg_sum[33:16];
endmodule
```

Como se muestra en la **Figura 7.25**, se comprueba el correcto funcionamiento del

módulo:

Figura 7.25. Test Bench Módulo MULT_ACC



ROM: se parametriza la cuantificación de los coeficientes **Wc**. Posee dos entradas de control, una para el direccionamiento de los coeficientes **Addr** y la otra para habilitar la salida del coeficiente (**ce_ROM**, activo a nivel alto).

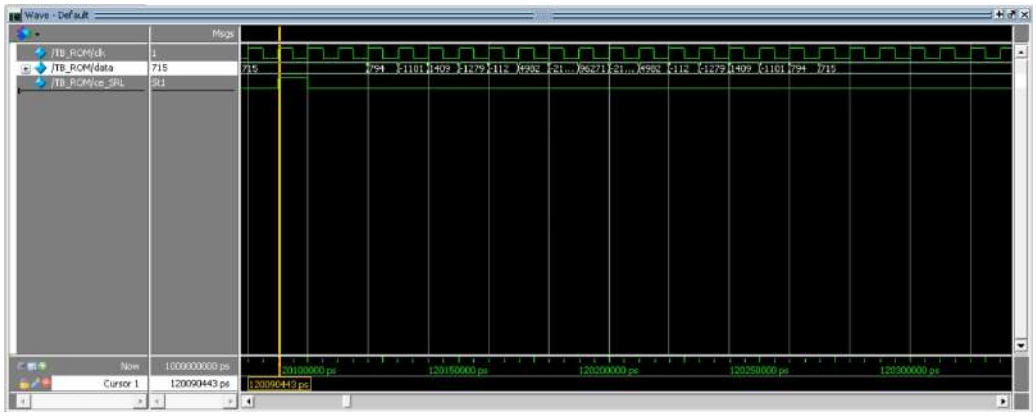
El código del módulo **ROM.v** programado es:

```
module ROM
#(parameter Wc = 18,
parameter Num = 17)
(input [4:0] addr,
input clk,
input ce,
output reg signed [Wc-1:0] data);

reg [Wc-1:0] rom [Num-1:0];
initial
begin
$readmemb(çoef.txt",rom);
end
```

```
always @(posedge clk)
if (ce)
data = rom[addr];
else
data = 0;

endmodule
```

La lectura de los coeficientes de la Memoria ROM se la hace directamente de un archivo plano .txt.

Figura 7.26. Test Bench Módulo ROM



ROM: es la Máquina de Estados que controla la salida de la memoria ROM, el funcionamiento de la celda multiplicador acumulador y el registro de salida del filtro.

El código del módulo **CONTROL.v** programado es:

```
module CONTROL
#(parameter Win=16, // Cuantificacion de la entrada y salida
parameter Num = 17) // Numero de datos a procesar
( input val_in, // dato valido
input clk,
```

```
input rst,
output reg [4:0] Addr, // direccion
output reg ce_SRL, // habilitacion desplazamiento
output reg ce_ROM, // habilitacion lectura ROM
output reg ce_Acc, // habilitacion acumulador
output reg rst_Acc, // reset acumulador
output reg ce_Reg // habilitacion salida
);
parameter S0=0,S1=1,S2=2,S3=3;
reg [1:0] Estado;
reg [4:0]count;
integer i;

// Actualizar estado con flanco de reloj o reset
always @ (posedge clk)
if (rst) Estado=S0;
else
begin
case(Estado)
S0: if(!val_in) // Espera dato valido
Estado = S0;
else
Estado = S1;
S1: if(count = Num-1) // Filtrado de Datos
begin
Estado = S1;
count = count+1;
end
else
begin
```

```
Estado = S2;
count = 0;
end
S2: Estado = S3; // Registro del dato
S3: Estado = S0; // Reset Acumulador
default: Estado = S0;
endcase
end

// Determinar salidas en funcion del estado actual
always @(Estado,count,val_in)
begin
case(Estado)
S0: begin Addr=count; ce_SRL=val_in; ce_ROM=0; ce_Acc=0; rst_Acc=0; ce_Reg=0;
end // Espera de Val_in
S1: begin Addr=count; ce_SRL=val_in; ce_ROM=1; ce_Acc=1; rst_Acc=0; ce_Reg=0;
end // MULT_ACC enable
S2: begin Addr=count; ce_SRL=val_in; ce_ROM=0; ce_Acc=0; rst_Acc=0; ce_Reg=1;
end // Registro Datos
S3: begin Addr=count; ce_SRL=val_in; ce_ROM=0; ce_Acc=0; rst_Acc=1; ce_Reg=0;
end // Reset Acumulador
default: begin Addr=count; ce_SRL=val_in; ce_ROM=0; ce_Acc=0; rst_Acc=0;
ce_Reg=0; end //So por defecto
endcase
end
endmodule
```

En la **Figura 7.27** se identifican los 4 estados definidos en la máquina de estados:

Como se muestra en la **Figura 7.28**, se comprueba el correcto funcionamiento del módulo:

Figura 7.27. Máquina de Estados CONTROL.v

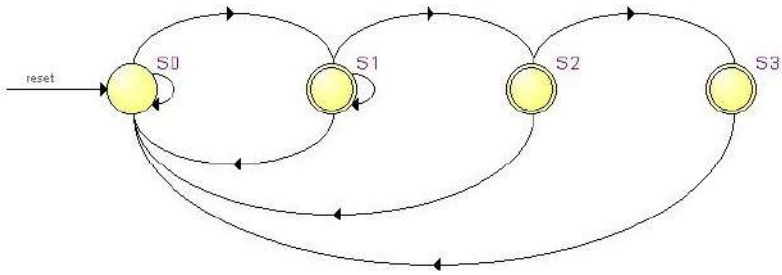
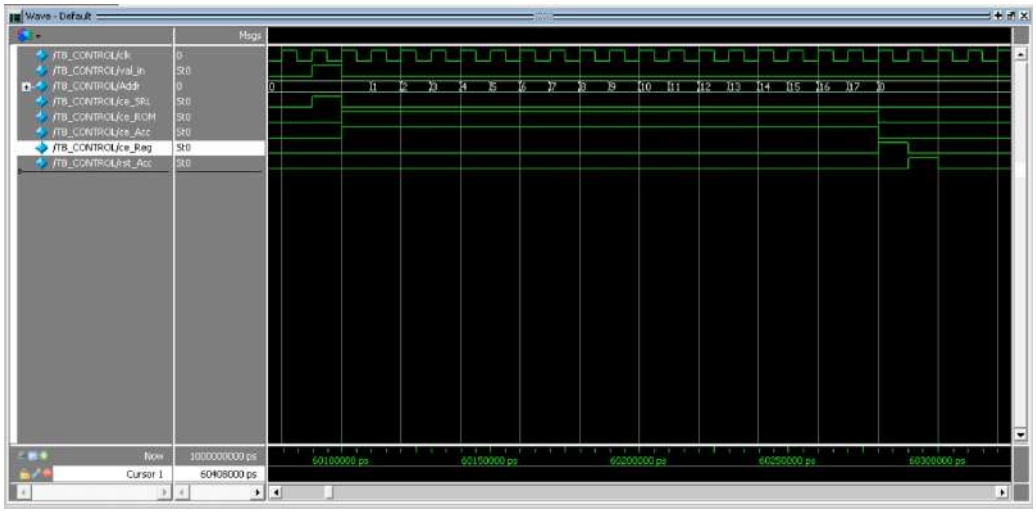


Figura 7.28. Test Bench Módulo CONTROL‘



SEC_FILTER: modelo completo del filtro secuencial. En primer lugar, se construye el filtro a partir de los módulos previamente diseñados con precisión completa para que no haya pérdida de información. Como se pide una salida definida **18 bits**, se trunca a la salida del **MULT_ACC** con ese formato, registrando los **18 bits** únicamente.

El código del módulo **SEC_FILTER.v** programado es:

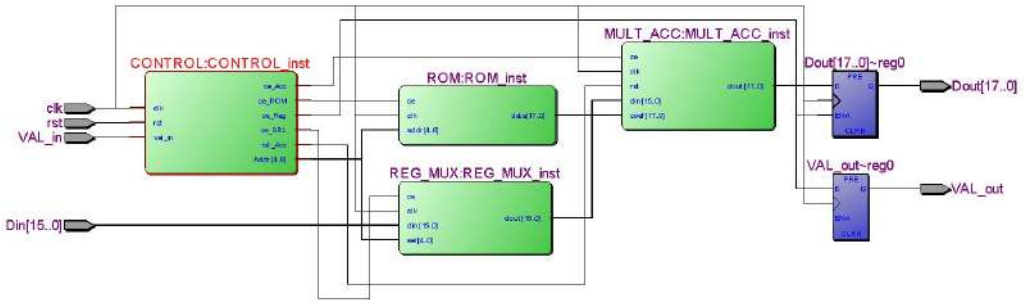
```
module SEC_FILTER
#(parameter Win=16,
parameter Wc = 18,
parameter Num=17)
(input signed [Win-1:0] Din,
input clk,
input VAL_in,
input rst,
output reg VAL_out,f
//output reg signed [Win+Wc+1:0] Dout); // Salida Precision Completa
output reg signed [Wc-1:0] Dout); // Salida

wire [4:0] sel,Addr;
wire signed[Win-1:0] dreg;
wire signed[Wc-1:0] dout;
wire signed[Wc-1:0] coef;
wire ce_SRL, ce_ROM, ce_Acc, rst_Acc,ce_Reg;

CONTROL #(.Win(Win), .Num(Num)) CONTROL_inst
(
.val_in(VAL_in),
.clk(clk),
.rst(rst),
.Addr(Addr),
```

```
.ce_SRL(ce_SRL),
.ce_ROM(ce_ROM),
.ce_Acc(ce_Acc),
.rst_Acc(rst_Acc),
.ce_Reg(ce_Reg)
);
REG_MUX #(.Win(Win), .Num(Num)) REG_MUX_inst
(
.din(Din),
.sel(Addr),
.clk(clk),
.ce(ce_SRL),
.dout(dreg)
);
ROM #(.Wc(Wc),.Num(Num)) ROM_inst
(
.addr(Addr),
.clk(clk),
.ce(ce_ROM),
.data(coef)
);
MULT_ACC #(.Win(Win), .Wc(Wc)) MULT_ACC_inst
(
.din(dreg),
.coef(coef),
.clk(clk),
.ce(ce_Acc),
.rst(rst_Acc),
.dout(dout)
);
```

```
always @(posedge clk)
if (ce_Reg)
begin
Dout = dout;
VAL_out = 1'b1;
end
else
VAL_out = 1'bo;
endmodule
```

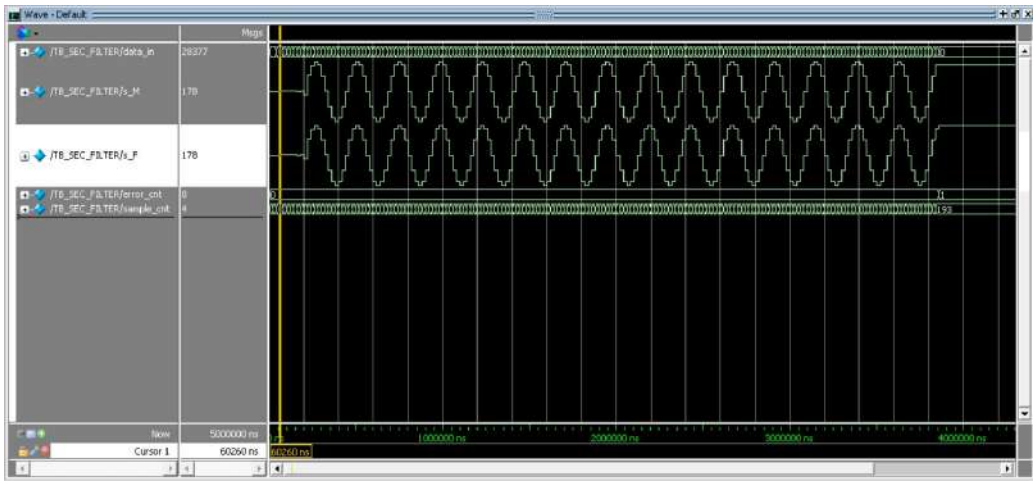
La **Figura 7.29** muestra el RTL de la implementación del módulo **SEC_FILTER.v** con salida truncada:

Figura 7.29. Esquema Módulo SEC_FILTER‘



Como se muestra en la **Figura 7.30**, se comprueba el correcto funcionamiento del módulo completo con la salida truncada a **18 bits**:

Figura 7.30. Test Bench Módulo SEC_FILTER‘



Finalmente, se comprueba que la frecuencia máxima obtenida es **137.8 MHz**, como se ve en la **Figura 7.31**:

Figura 7.31. Frecuencia Máxima de Funcionamiento - SEC_FILTER

Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	137.8 MHz	137.8 MHz	clk	

El haber alcanzado la frecuencia máxima de funcionamiento deseada, es un indicador claro de que se realizó un correcto diseño del filtro FIR secuencial.

BIBLIOGRAFÍA

[1] Keshab K. Parhi. *VLSI digital signal processing systems: Design and Implementation*. Wiley-Interscience 2014.

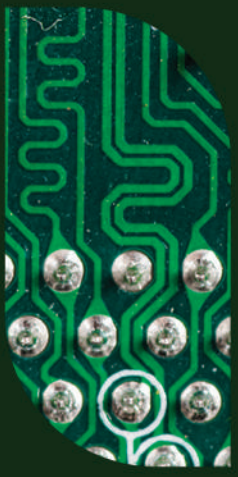
[2] S. A. Khan. *Digital Design of Signal Processing Systems: A practical Approach*. Wiley-Interscience 2011.

[3] R. Woods. *FPGA-based Implementation of Signal Processing Systems*. Wiley-Interscience 2008.

La presente obra, *Ejercicios resueltos de procesado de señal en FPGA*, está destinada a estudiantes, docentes, investigadores y profesionales en general que tengan conocimiento de procesado de señal en sistemas electrónicos, manejo de FPGA, programación en lenguaje HDL, modelado de precisión finita, circuitos aritméticos y los diferentes tipos de de arquitectura *software* que se pueden implementar sobre las FPGA.
El presente libro es resultado de la investigación desarrollada en el Máster Universitario en Sistemas Electrónicos, especialidad Sistemas Electrónicos Digitales, cursado en la Universitat Politécnica de Valencia.

Hugo Moreno Avilés nació en Riobamba, Ecuador; ingeniero en Electrónica, Automatización y Control, especialista en Comunicación Radio, Microondas y Fibra Óptica, doctor en Investigación en Sistemas e Informática y docente de la Facultad de Informática y Electrónica de la Escuela Superior Politécnica de Chimborazo. Tiene experiencia en Dirección y Gestión de Proyectos de Investigación. Autor de artículos científicos en congresos y revistas internacionales.

Jhon Jairo Cevallos nació en Riobamba, Ecuador; ingeniero en Electrónica, Control y Redes Sociales, máster en Ingeniería de Sistemas Electrónicos en la especialidad de Sistemas Electrónicos Digitales y especialista de Investigación, Proyectos y Transferencia de Tecnologías del Instituto de Investigación de la Escuela Superior Politécnica de Chimborazo.



ISBN: 978-9942-40-450-3
9 789942 404503