FFT Core de tamaño reducido para comunicaciones OFDM

Autores: Ing. Andrés D. Cassagnes, Ing. Federico G. Zacchigna, Ing. Octavio Alapago, Dr. Ing. Ariel Lutenberg

Laboratorio de Sistemas Embebidos (LSE)
Facultad de Ingeniría
Universidad de Buenos Aires
FIUBA

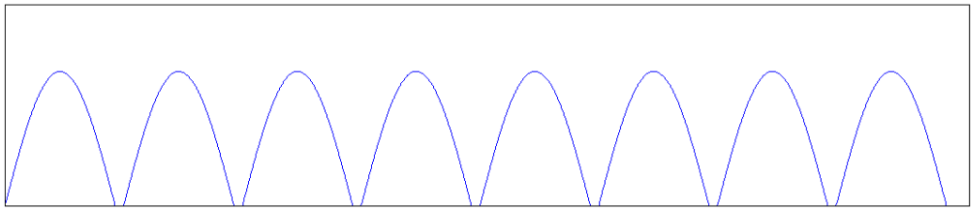
16/08/2018

OFDM

- Orthogonal Frequency Divider Multiplexing

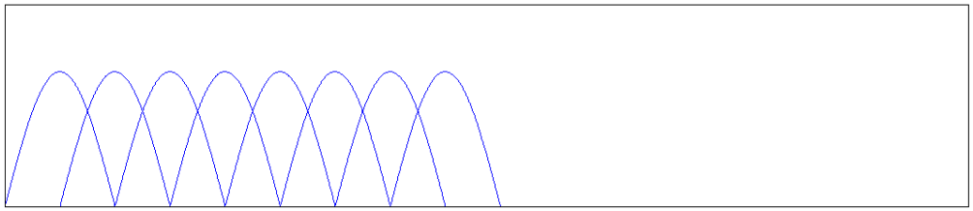
OFDM

- Orthogonal Frequency Divider Multiplexing
- Divide la información en múltiples frecuencias



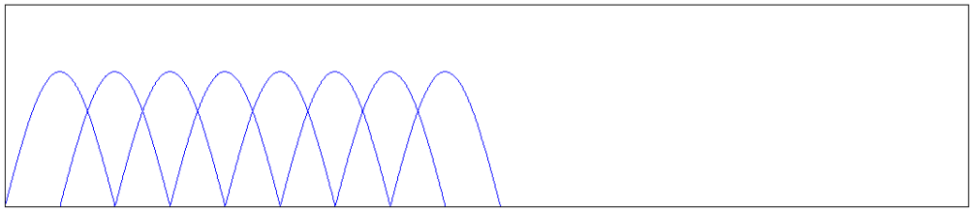
OFDM

- Orthogonal Frequency Divider Multiplexing
- Divide la información en múltiples frecuencias
- Bandas de frecuencia solapadas



OFDM

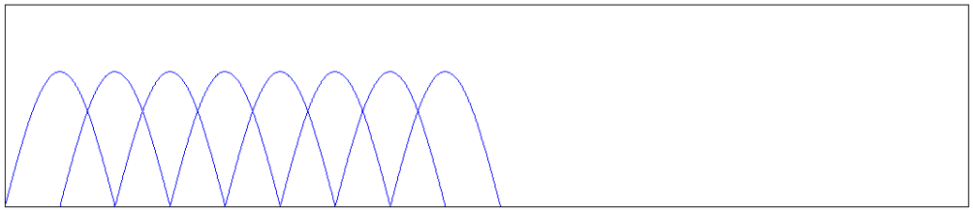
- Orthogonal Frequency Divider Multiplexing
- Divide la información en múltiples frecuencias
- Bandas de frecuencia solapadas
  - Representación matemática



$$s_k(t - kT) = \sum_{i=-N/2}^{N/2-1} x_{i,k} e^{j2\pi\left(\frac{i}{T}\right)(t-kT)} \quad (1)$$

OFDM

- Orthogonal Frequency Divider Multiplexing
- Divide la información en múltiples frecuencias
- Bandas de frecuencia solapadas
  - Representación matemática



$$s_k(t - kT) = \sum_{i=-N/2}^{N/2-1} x_{i,k} e^{j2\pi\left(\frac{i}{T}\right)(t-kT)} \quad (1)$$

  - Asumiendo que $x_{i,k}$ es constante a lo largo del período de símbolo T, se puede utilizar una IDFT/DFT para modular.

FFT

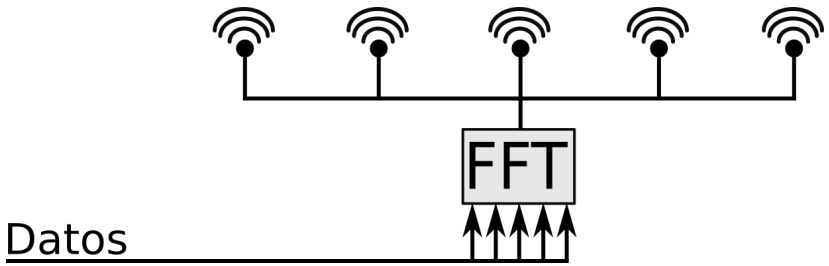
- Transformada rápida de Fourier

FFT

- Transformada rápida de Fourier
- Sumas, restas y multiplicaciones

FFT

- Transformada rápida de Fourier
- Sumas, restas y multiplicaciones
- Cada salida = suma y resta de todas las entradas



Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).
  - Frecuencia de muestreo mínima de 8126984 sps (según estándar ISDB-T)

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).
  - Frecuencia de muestreo mínima de 8126984 sps (según estándar ISDB-T)
  - Entrada y salida continua

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).
  - Frecuencia de muestreo mínima de 8126984 sps (según estándar ISDB-T)
  - Entrada y salida continua
  - Aritmética de punto fijo

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).
  - Frecuencia de muestreo mínima de 8126984 sps (según estándar ISDB-T)
  - Entrada y salida continua
  - Aritmética de punto fijo
  - Unidad de escalamiento configurable en ejecución con opción de seleccionar la etapa a escalar y el método (redondeo/truncamiento)

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Diseñar un modulador/demodulador OFDM para un sistema de telecomunicaciones definido por software que cumpla con el estándar ISDB-T
- Que sirva también como unidad de cómputo FFT/IFFT para procesamiento de señales.
- Requerimientos
  - Longitud configurable, incluyendo al menos 2K, 4K y 8K muestras (ISDB-T).
  - Frecuencia de muestreo mínima de 8126984 sps (según estándar ISDB-T)
  - Entrada y salida continua
  - Aritmética de punto fijo
  - Unidad de escalamiento configurable en ejecución con opción de seleccionar la etapa a escalar y el método (redondeo/truncamiento)
  - Bajo consumo de recursos comparados con otras implementaciones[1][2]

[1] Implementación abierta orientada a ISDB-T: Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”

[2] Implementación propietaria Xilinx LogiCORE FFT 7.1

Objetivos

- Realizar una evaluación de desempeño

Objetivos

- Realizar una evaluación de desempeño
  - Funcional

Objetivos

- Realizar una evaluación de desempeño
  - Funcional
  - Ruido / error

Objetivos

- Realizar una evaluación de desempeño
  - Funcional
  - Ruido / error
  - Distorsión armónica

Objetivos

- Realizar una evaluación de desempeño
  - Funcional
  - Ruido / error
  - Distorsión armónica
  - Recursos de HW

Objetivos

- Realizar una evaluación de desempeño
  - Funcional
  - Ruido / error
  - Distorsión armónica
  - Recursos de HW
- Realizar una comparativa con desarrollos de terceros para evaluar el diseño realizado

Objetivos

- Realizar una evaluación de desempeño
  - Funcional
  - Ruido / error
  - Distorsión armónica
  - Recursos de HW
- Realizar una comparativa con desarrollos de terceros para evaluar el diseño realizado
- Proponer trabajos futuros para continuar y mejorar el diseño.

SELECCIÓN DE LAS ARQUITECTURAS

Arquitectura

- Algoritmo: Radix-r

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)
- Implementación: arquitectura iterativa

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)
- Implementación: arquitectura iterativa
  - Un solo bloque para todos los cálculos

Arquitectura

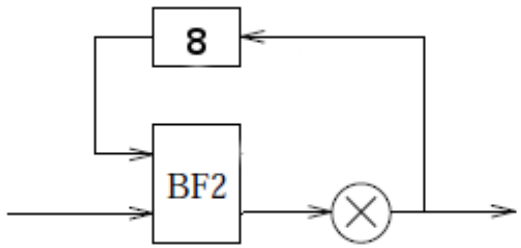
- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)
- Implementación: arquitectura iterativa
  - Un solo bloque para todos los cálculos
  - Es la implementación más pequeña (requerimiento)

Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)
- Implementación: arquitectura iterativa
  - Un solo bloque para todos los cálculos
  - Es la implementación más pequeña (requerimiento)
  - Se computa una etapa a la vez

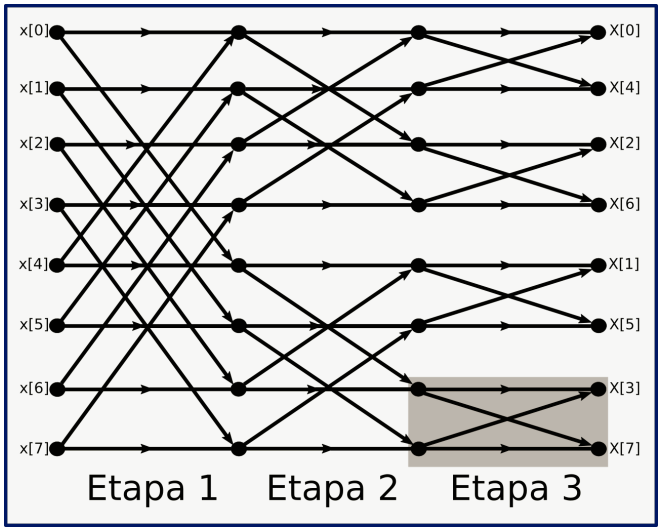
Arquitectura

- Algoritmo: Radix-r
  - Flexibilidad en la longitud
  - Simplicidad en la implementación
  - Posibilidad de reutilizar módulos
- Longitud del bloque: 2 y 4
  - No implican multiplicaciones no triviales (solo por twiddle factors)
- Implementación: arquitectura iterativa
  - Un solo bloque para todos los cálculos
  - Es la implementación más pequeña (requerimiento)
  - Se computa una etapa a la vez

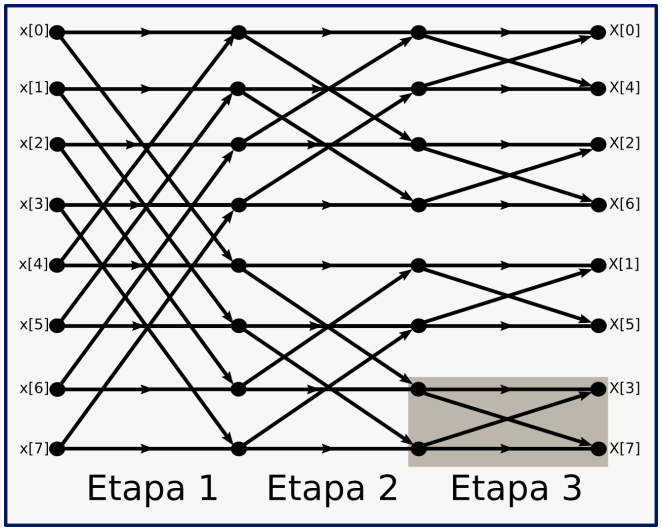


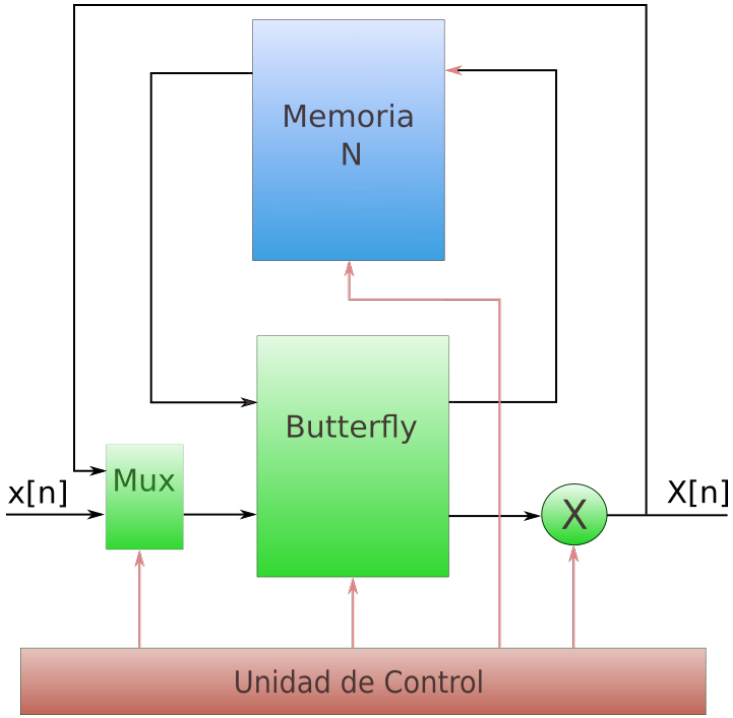
IMPLEMENTACIÓN DE LAS ARQUITECTURAS

Radix-2 Iterativa

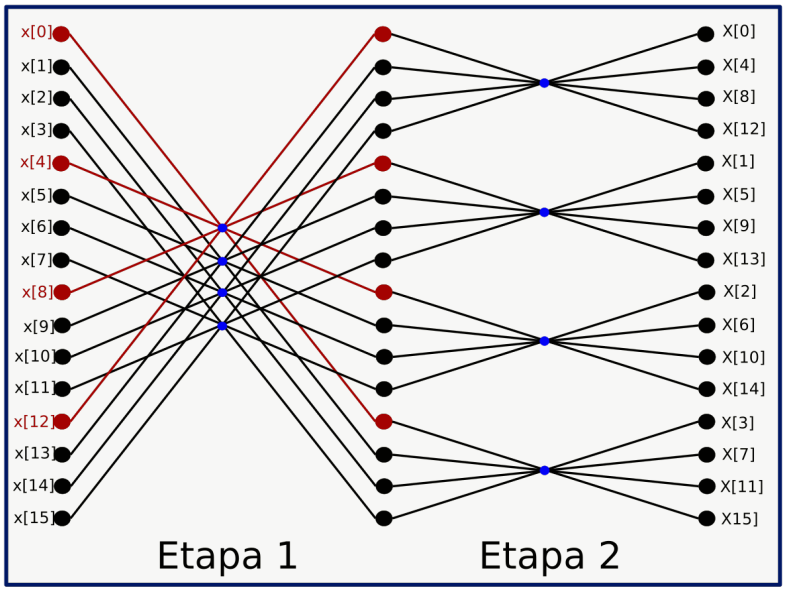


Radix-2 Iterativa

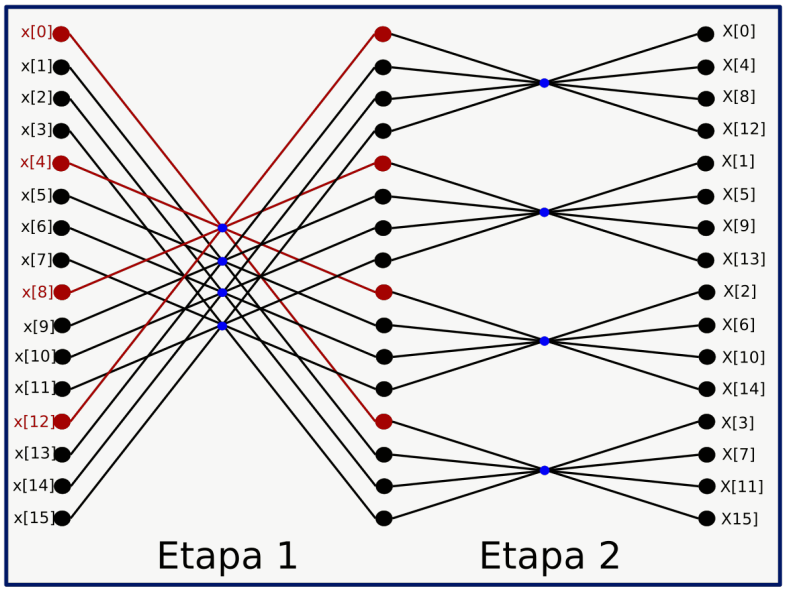


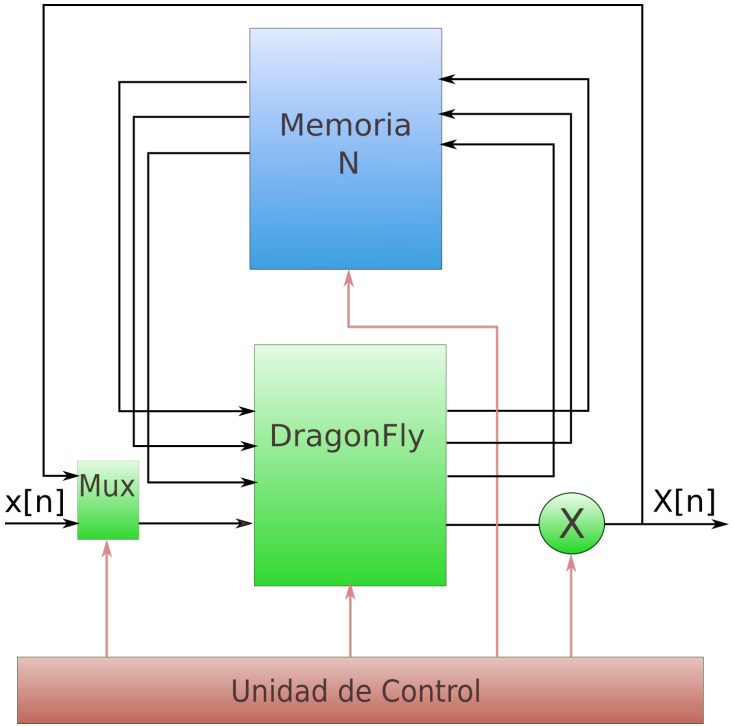


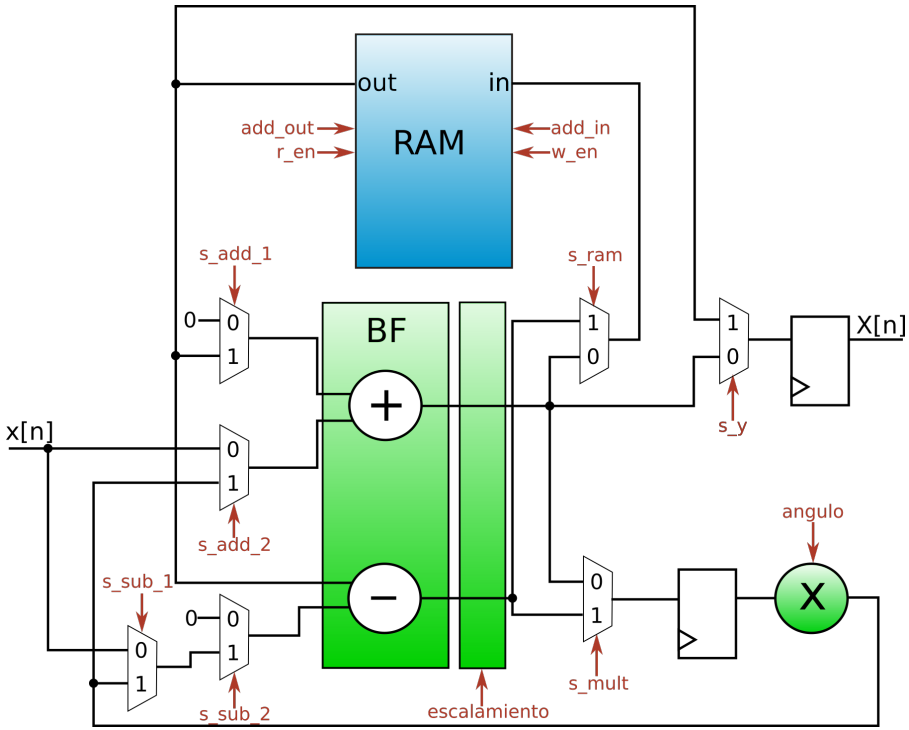
Radix-4 Iterativa

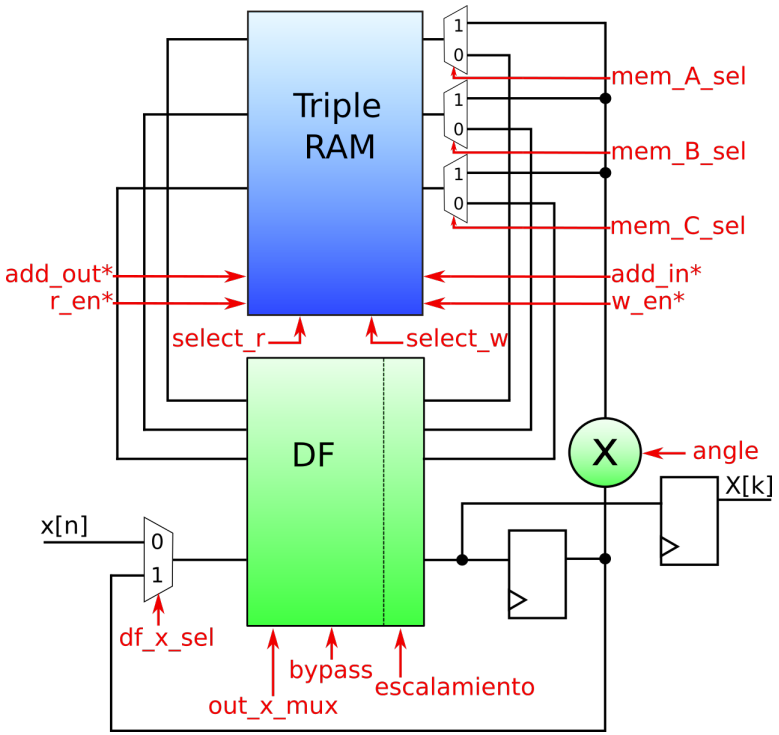


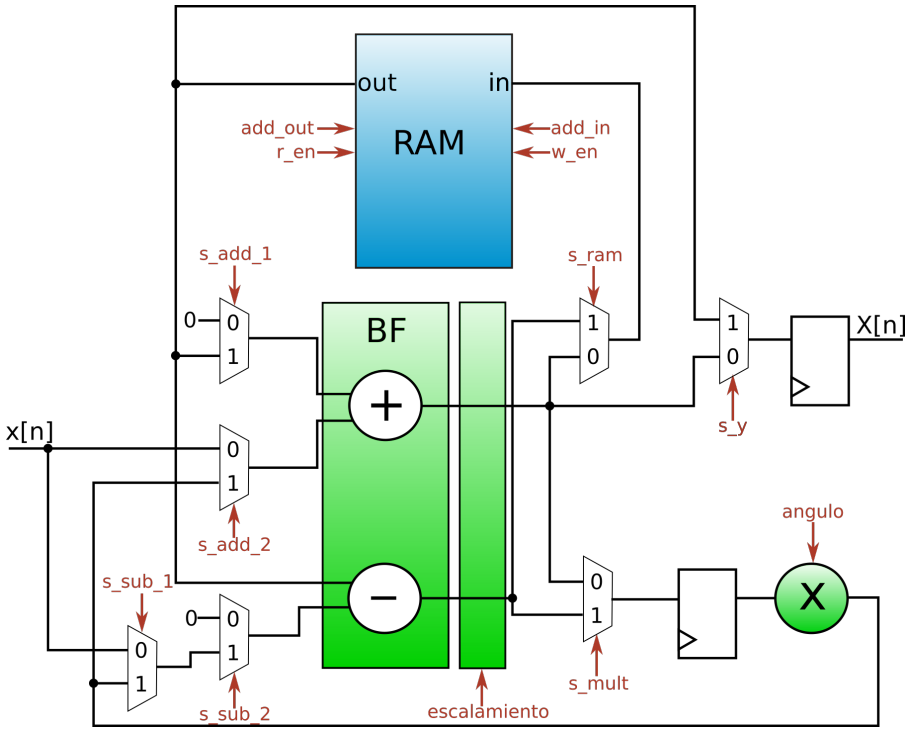
Radix-4 Iterativa

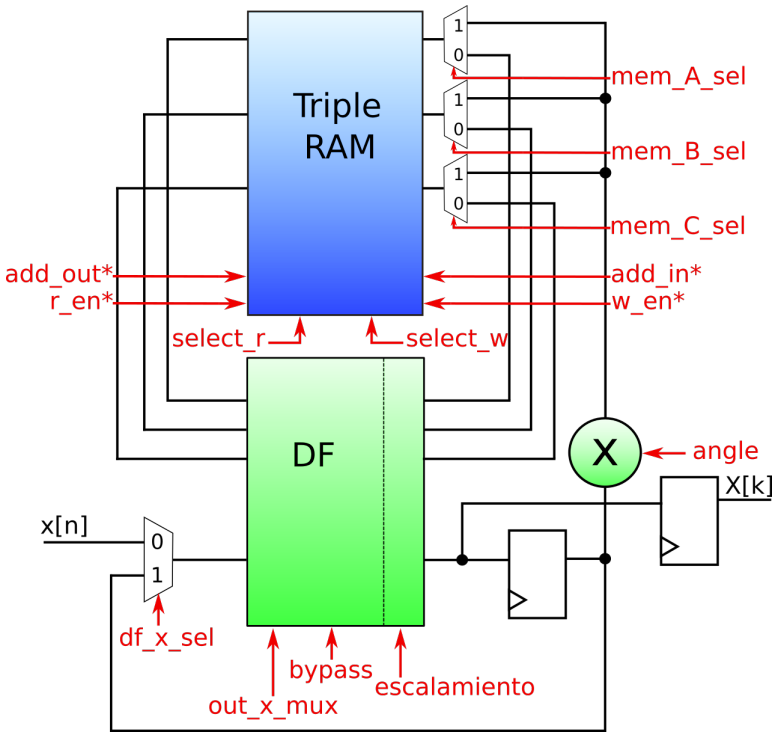




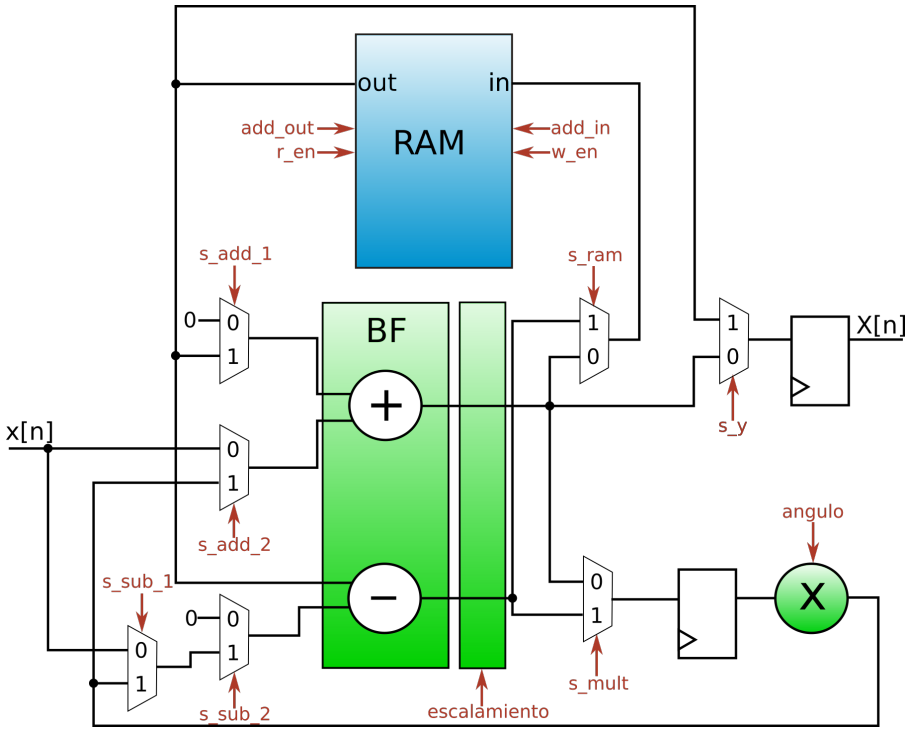

out
in
RAM
add_out
r_en
add_in
w_en
s_add_1
s_ram
BF
X[n]
x[n]
s_y
s_add_2
angulo
s_sub_1
s_sub_2
escalamiento
s_mult
X

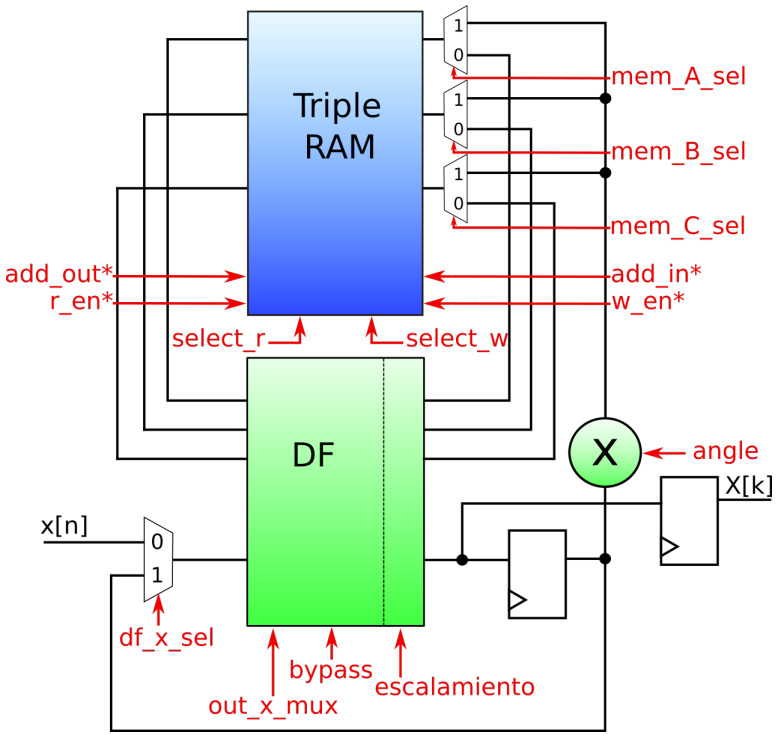

Triple
RAM
mem_A_sel
mem_B_sel
mem_C_sel
add_out*
r_en*
add_in*
w_en*
select_r
select_w
DF
X
angle
X[k]
x[n]
df_x_sel
bypass
escalamiento
out_x_mux



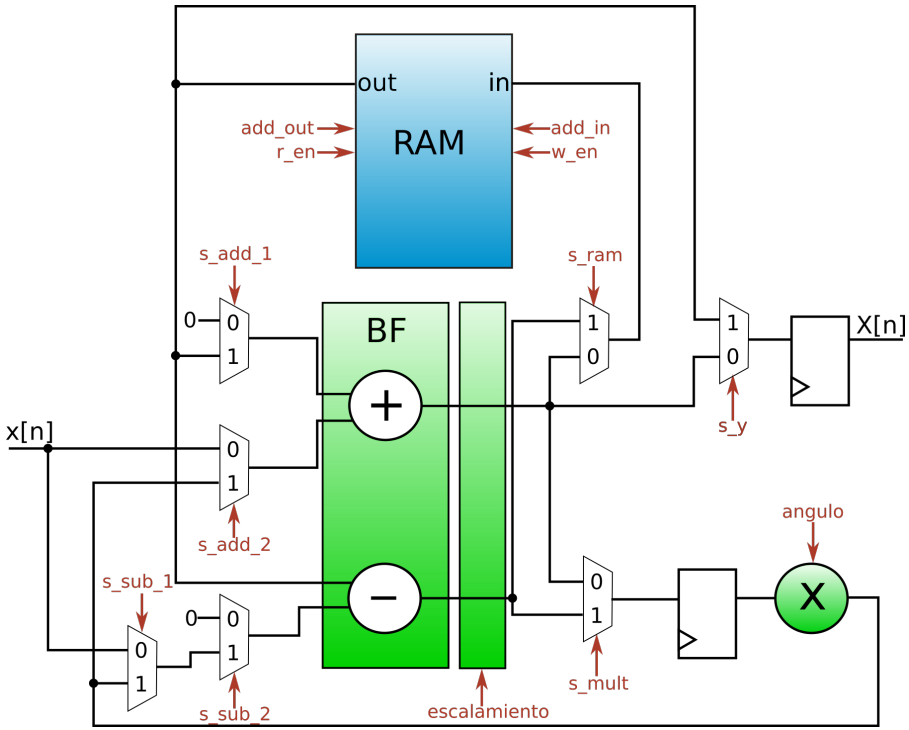


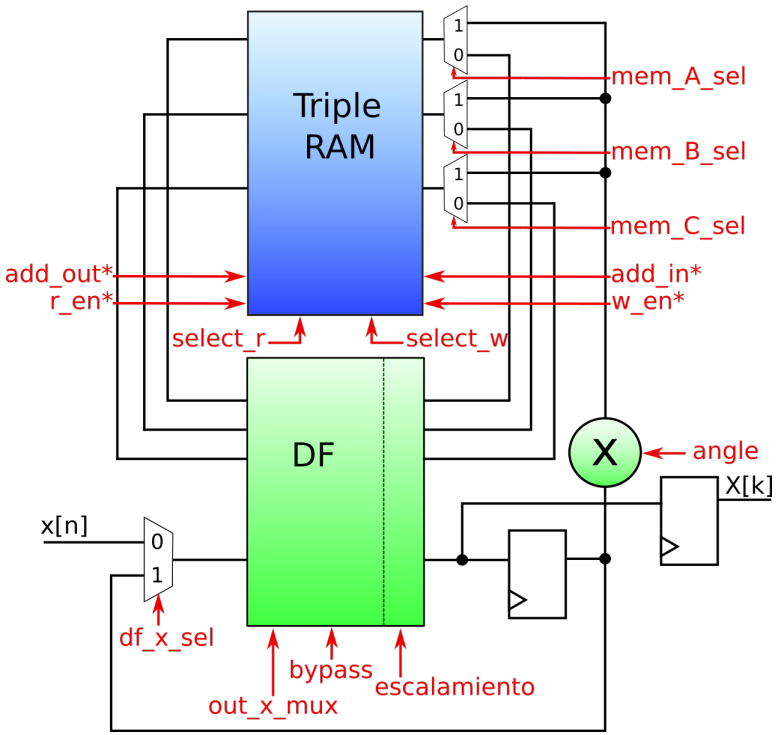
- Unidad aritmética (incluyendo unidad de escalamiento)



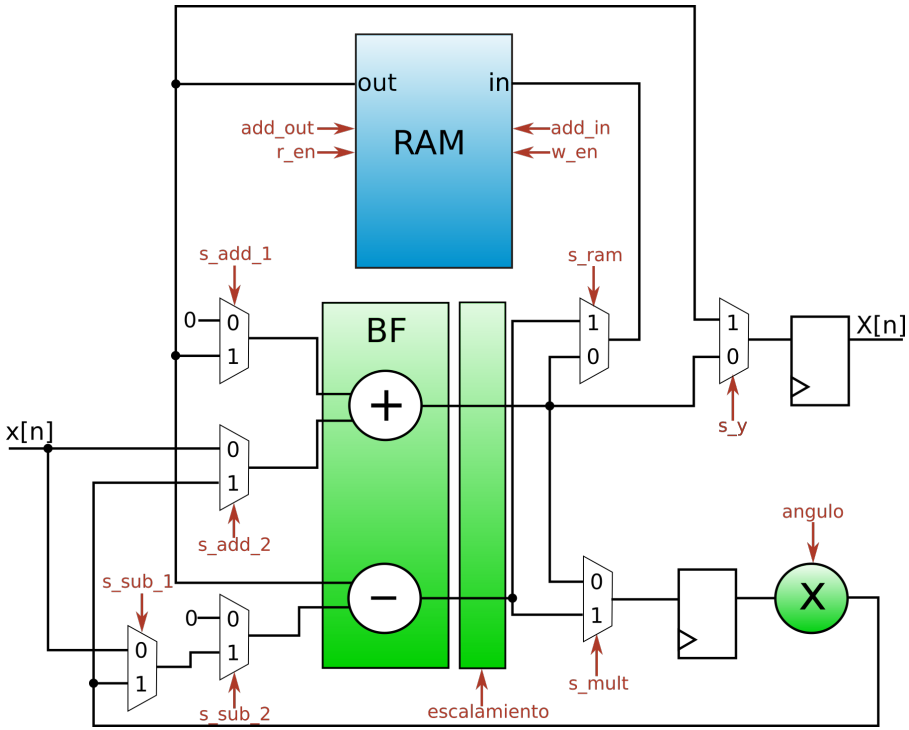


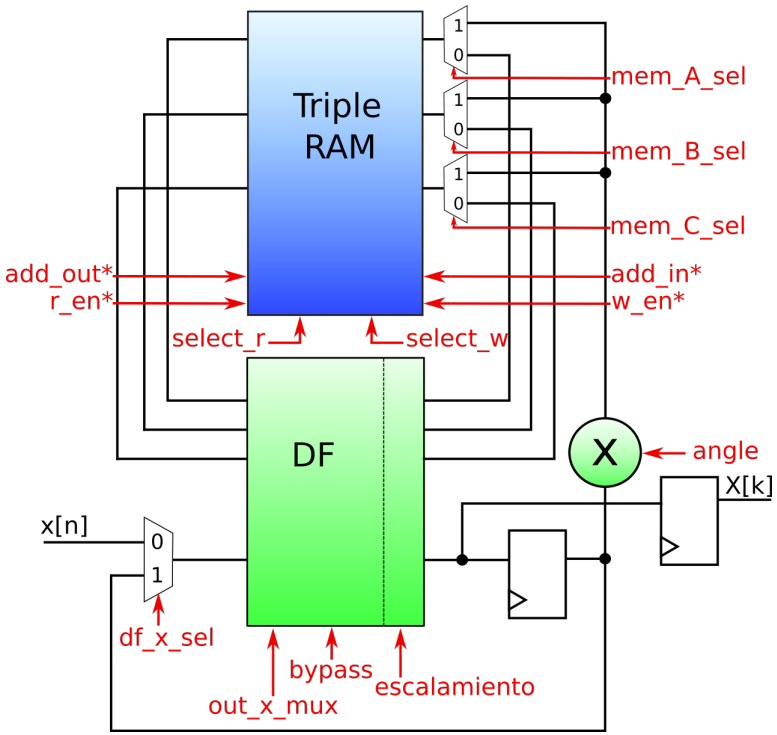
- Unidad aritmética (incluyendo unidad de escalamiento)
- Multiplicador



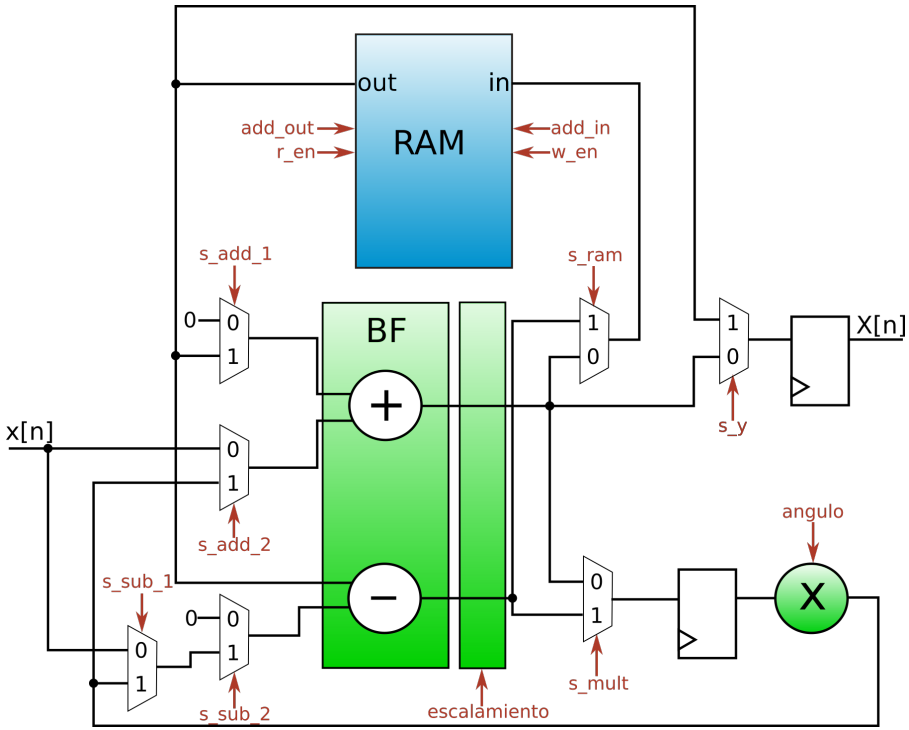


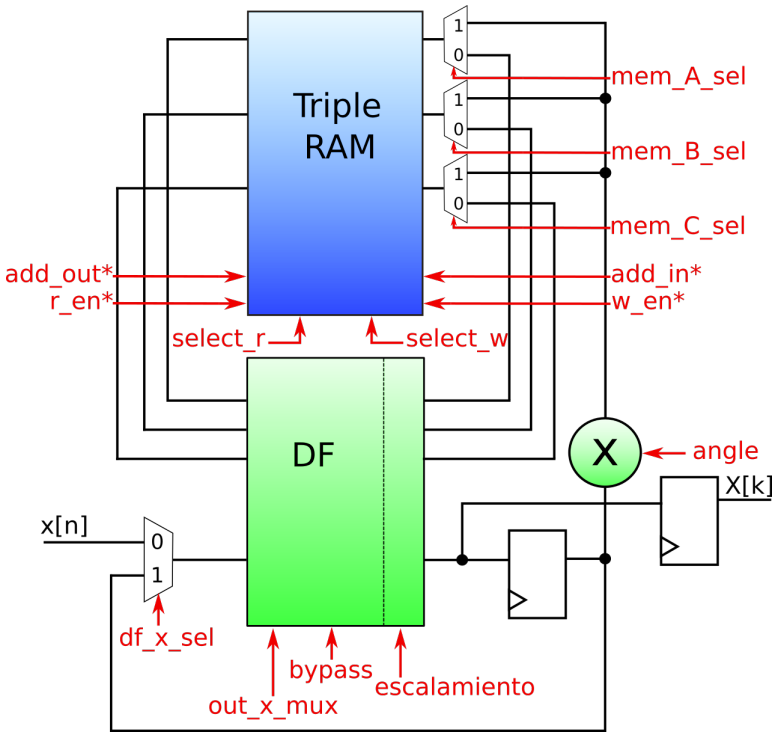
- Unidad aritmética (incluyendo unidad de escalamiento)
- Multiplicador
- Memoria





- Unidad aritmética (incluyendo unidad de escalamiento)
- Multiplicador
- Memoria
- Datapath





- Unidad aritmética (incluyendo unidad de escalamiento)
- Multiplicador
- Memoria
- Datapath
- Unidad de control

CARACTERIZACIÓN Y PRUEBAS

Listado de pruebas

- Transformación de señales patrón

Listado de pruebas

- Transformación de señales patrón
- Medición del error

Listado de pruebas

- Transformación de señales patrón
- Medición del error
- Medición de la THD

Listado de pruebas

- Transformación de señales patrón
- Medición del error
- Medición de la THD
- Efecto del escalamiento

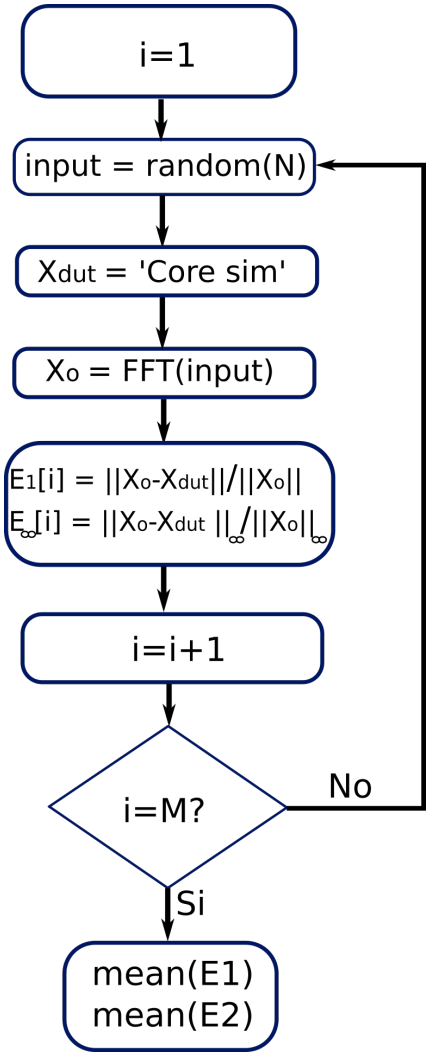
Listado de pruebas

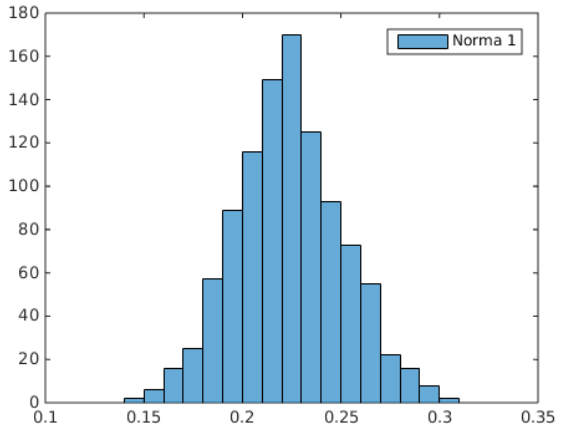
- Transformación de señales patrón
- Medición del error
- Medición de la THD
- Efecto del escalamiento
- Medición de los recursos necesarios

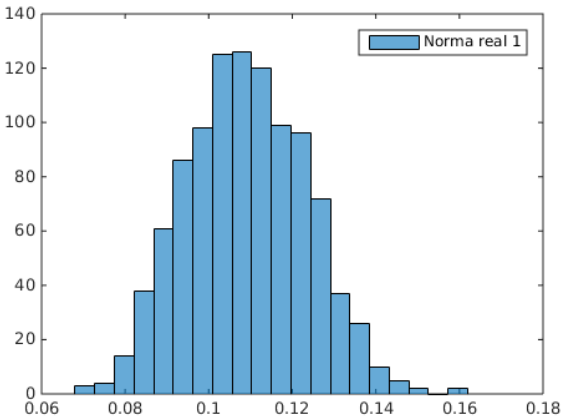
Listado de pruebas

- Transformación de señales patrón
- Medición del error
- Medición de la THD
- Efecto del escalamiento
- Medición de los recursos necesarios
- Comparación con core FFT abierto para modulación OFDM/ISDB-T

Medición del error



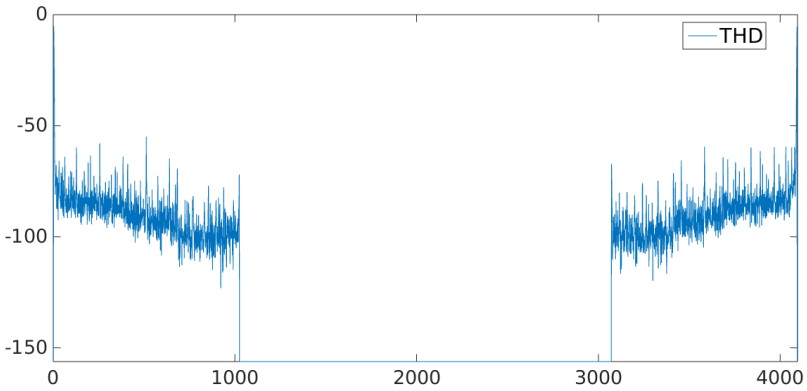




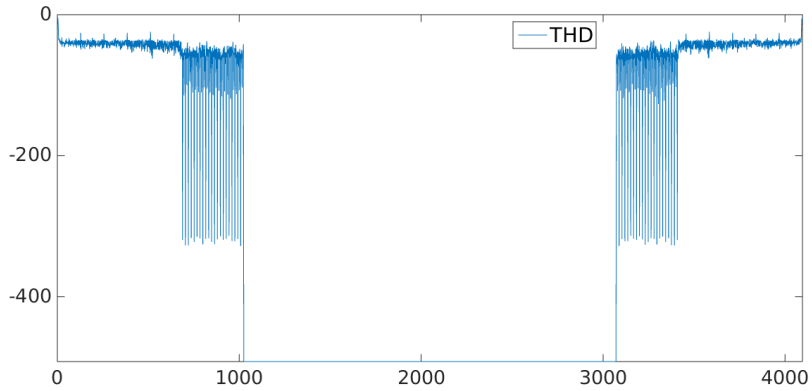
Resultados de la medición de error

	1024, 12	**1024, 16**	**4096, 12**	**4096, 16**
R-2, cordic	0,092	0,006	0,099	0,008
R-2, Mult.	0,232	0,003	0,340	0,108
R-4, cordic	0,077	0,003	0,074	0,007
R-4, Mult.	0,224	0,002	0,334	0,105
Kiss FFT		0,017		0,035
Xilinx FFT v7.1	0,0007	0,0001	0,0008	0,0004

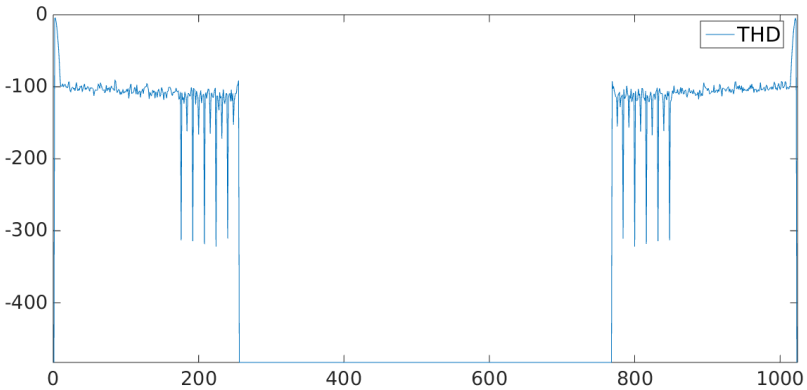
Medición de la THD



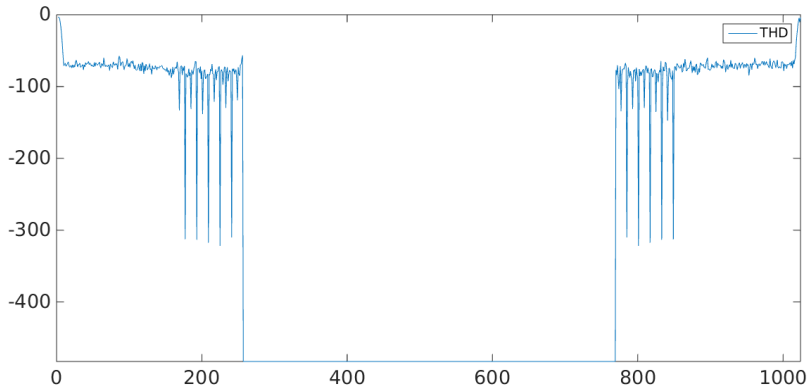
Radix-2, Cordic, 16 bits



Kiss FFT. C++. 16 bits

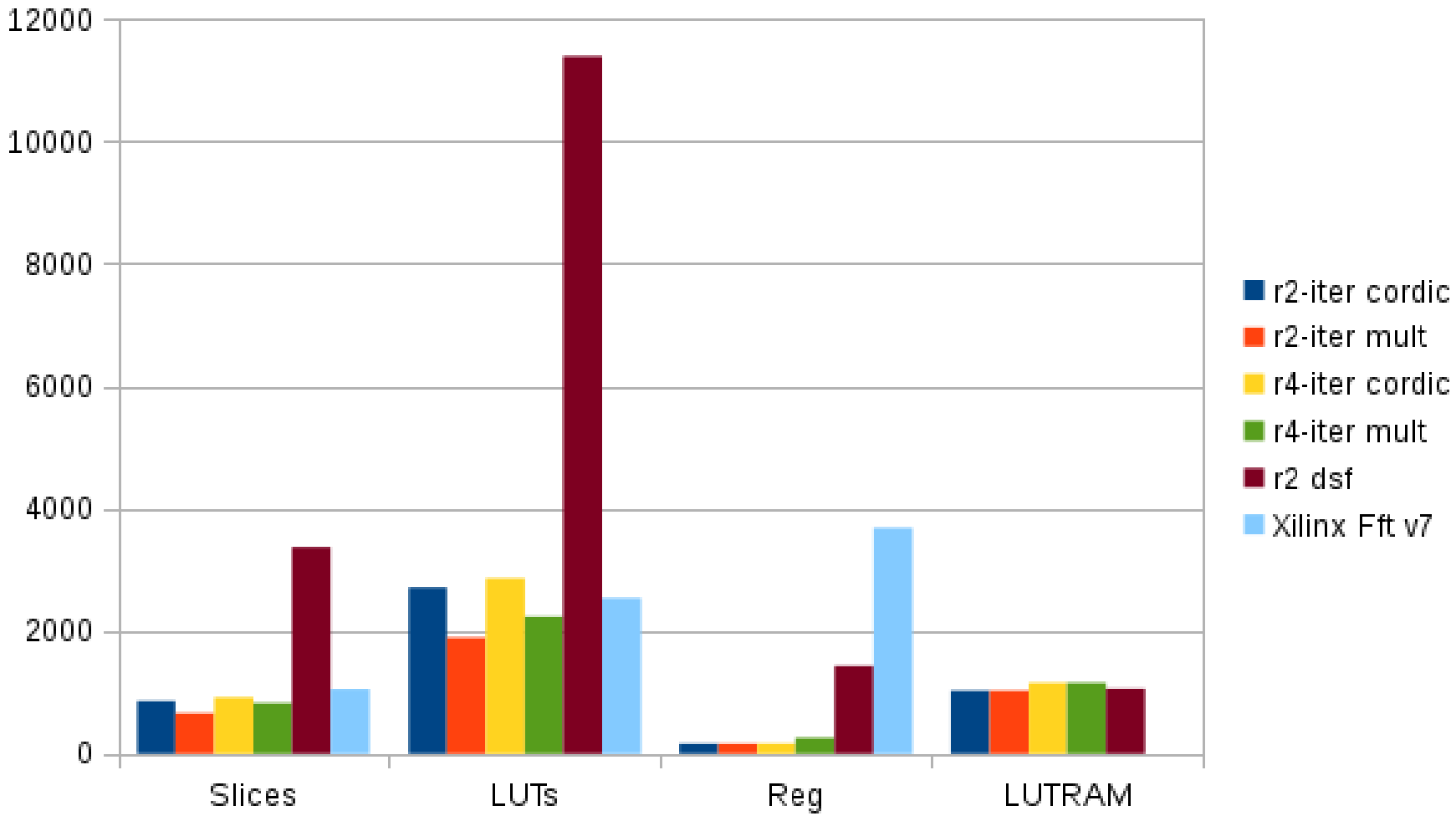


Radix-4, Mult., 16 bits

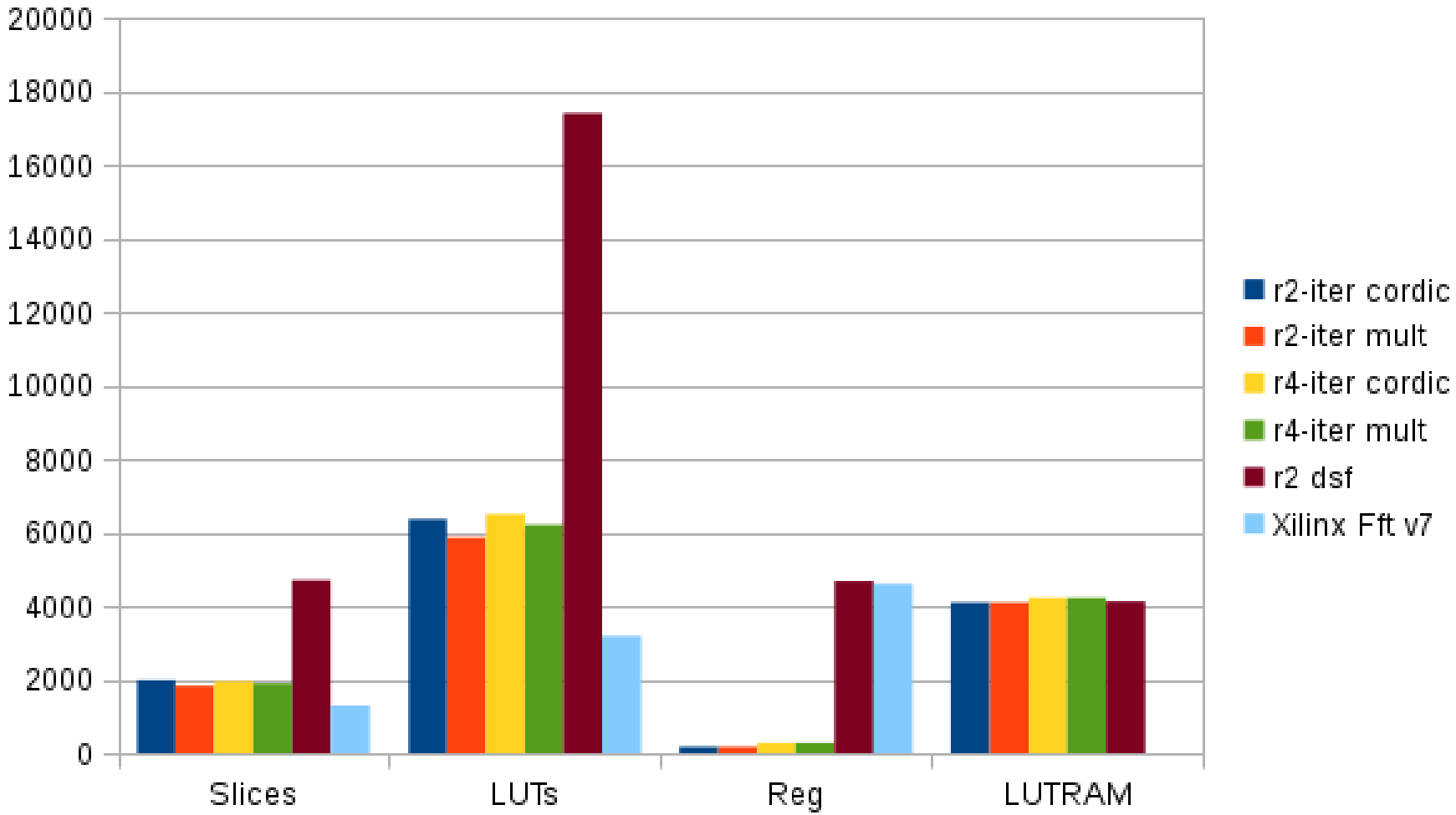


Xilinx LogiCORE FFT 7.1

Tamaño de implementación para 1024 puntos



Tamaño de implementación para 4096 puntos



Comparación con IP Core FFT abierto

- IP Core abierto para modulación/demodulación OFDM para ISDB-T (Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”)

Comparación con IP Core FFT abierto

- IP Core abierto para modulación/demodulación OFDM para ISDB-T (Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”)
- Comparación con la versión básica del IP Core, 8K muestras, 16 bits

Comparación con IP Core FFT abierto

- IP Core abierto para modulación/demodulación OFDM para ISDB-T (Melo, R., Salomón, F., Valinoti, B., (2016) “IP core FFT configurable en Runtime”)
- Comparación con la versión básica del IP Core, 8K muestras, 16 bits

	Iterative radix-2	**Reference core**
FF	533	1334
LUT	3046	4133
BRAM	62	62
MUL		48
MHz	107	61

CONCLUSIONES Y TRABAJOS FUTUROS

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima
  - Radix-2: clock: 107 MHz -> 8.23 Msamples/sec.

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima
  - Radix-2: clock: 107 MHz -> 8.23 Msamples/sec.
  - Radix-4: clock: 81 MHz -> 11.57 Msamples/sec.

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima
  - Radix-2: clock: 107 MHz -> 8.23 Msamples/sec.
  - Radix-4: clock: 81 MHz -> 11.57 Msamples/sec.
- Se cumplió el requerimiento de baja demanda de recursos/espacio.

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima
  - Radix-2: clock: 107 MHz -> 8.23 Msamples/sec.
  - Radix-4: clock: 81 MHz -> 11.57 Msamples/sec.
- Se cumplió el requerimiento de baja demanda de recursos/espacio.
- Las arqiutecturas fueron implementadas en Verilog HDL.

Conclusiones

- Se presentaron dos arquitecturas radix-r iterativas para cómputo de FFT/IFFT
  - Radix-2 iterativa
  - Radix-4 iterativa
  - Unidad de multiplicación por Twiddle Factors basada en algoritmo Cordic
  - Unidad de multiplicación compleja eficiente para producto por Twiddle Factors
  - Unidad de escalamiento (redondeo/truncamiento) configurable en ejecución
- Frecuencia de muestreo por encima de la frecuencia mínima
  - Radix-2: clock: 107 MHz -> 8.23 Msamples/sec.
  - Radix-4: clock: 81 MHz -> 11.57 Msamples/sec.
- Se cumplió el requerimiento de baja demanda de recursos/espacio.
- Las arqiutecturas fueron implementadas en Verilog HDL.
- Se generaron además herramientas de testing en Verilog, C++ y Matlab scriptng.

Trabajos Futuros

- Estudiar posibles implementaciones de algoritmos de *dithering* para reducir el ruido generado en las arquitecturas.

Trabajos Futuros

- Estudiar posibles implementaciones de algoritmos de *dithering* para reducir el ruido generado en las arquitecturas.
- Modificar el módulo de rotación Cordic agregando un pipeline que permita aumentar la velocidad de *clock* de las arquitecturas, sin agregar ciclos de *clock* extra al cómputo total de la FFT.

PREGUNTAS?

MUCHAS GRACIAS!

FIN!