Transmisión OFDM basada en el Estándar IEEE 802.11

Matías Roqueta

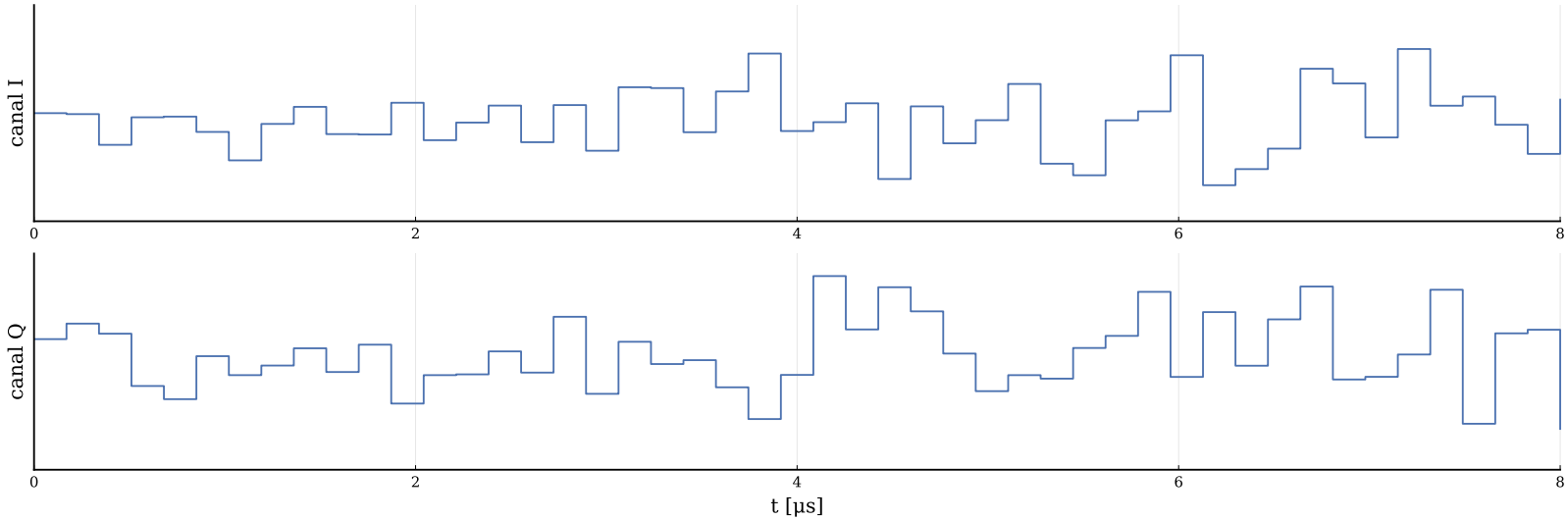
ICNPG, Instituto Balseiro

Problema a Resolver

Transformar una trama de bits en una señal temporal compleja:

```
0101 1101 1011 1101 0110 0110 0011 1001 1000 1001 0011
1011 1101 1011 0111 0001 1010 0000 1101 1001 1010 1010
1000 1011 1100 0010 1010 0111 1100 1101 0101 1100 1011
```

↓ Modulación OFDM ↓



Descripción de OFDM

- *Orthogonal Frequency Division Multiplexing* consiste en descripción en frecuencia de las señales.
- La unidad fundamental transmitida es el *Símbolo OFDM*.
- Cada símbolo corresponde a 48 números complejos.
- Cada número complejo en un símbolo corresponde a n bits, dependiendo de la modulación.
- Entonces, un símbolo corresponde a $N = 48\,n$ bits.
- En el ejemplo se usó modulación 16-QAM con $n = 4$, por lo que un símbolo OFDM codificará $N = 192$ bits.
- Se implementaron 3 etapas
  - Entrelazado $\rightarrow$ Modulación $\rightarrow$ IFFT

Etapa Entrelazado

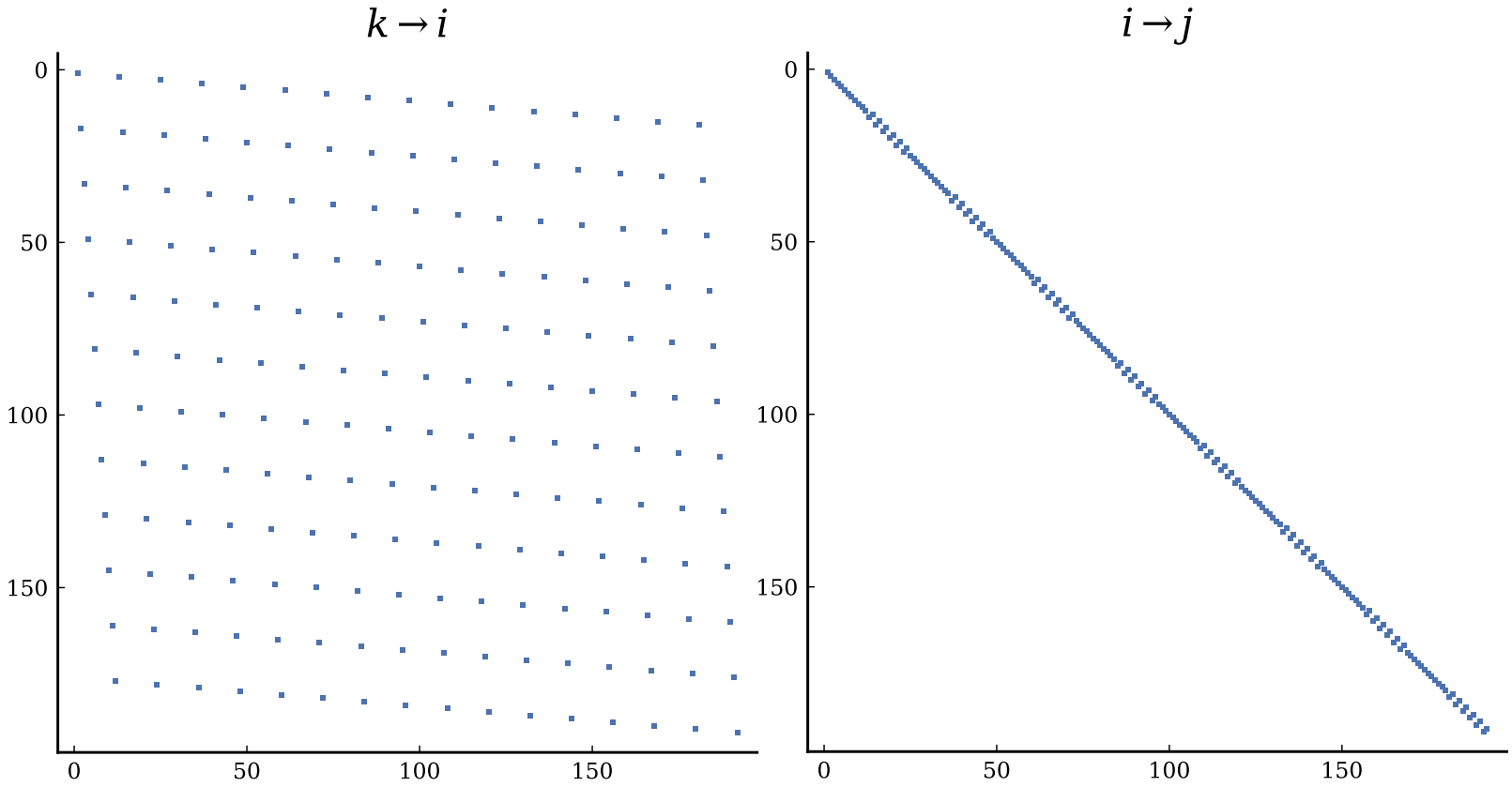
- La trana de bits se subdivide en bloques de $N = 192$ bits y cada bloque se entrelaza
- Consiste en una permutación de índices de la trama de bits. Se realiza en dos etapas: $k \rightarrow i \rightarrow j$
- Las reglas de cambio de índice son las siguientes

$$i = \frac{N}{16} \times (k \bmod 16) + \left\lfloor \frac{k}{16} \right\rfloor$$

$$j = \frac{N}{2} \times \left\lfloor \frac{i}{\frac{N}{2}} \right\rfloor + \left[i + N - \left\lfloor \frac{16 \times i}{N} \right\rfloor \right] \bmod \frac{N}{2}$$

Etapa Entrelazado

- Las reglas de entrelazado se pueden interpretar como productos por matrices ralas en donde los **1**s indican cuales elementos se permutan.



Etapa Modulación

Consiste en asignar a cada grupo de n bits consecutivos un número complejo según alguna constelación

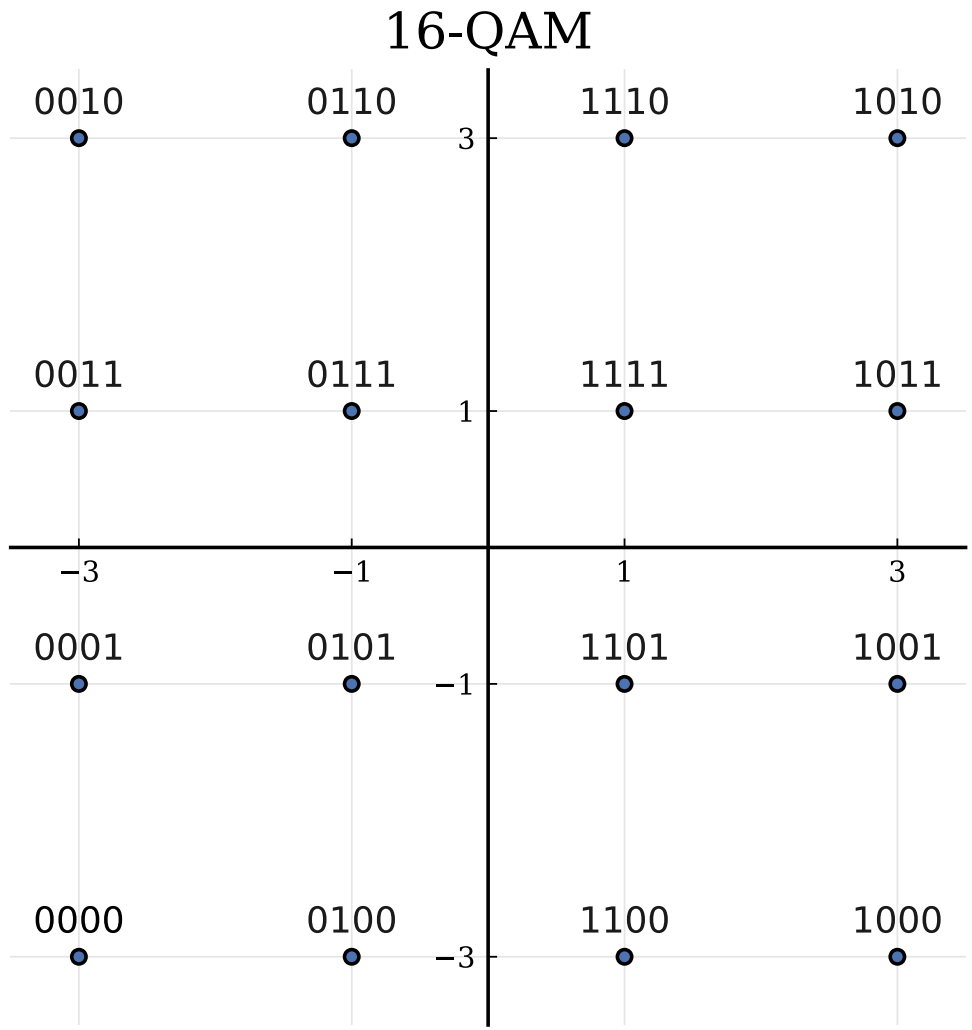
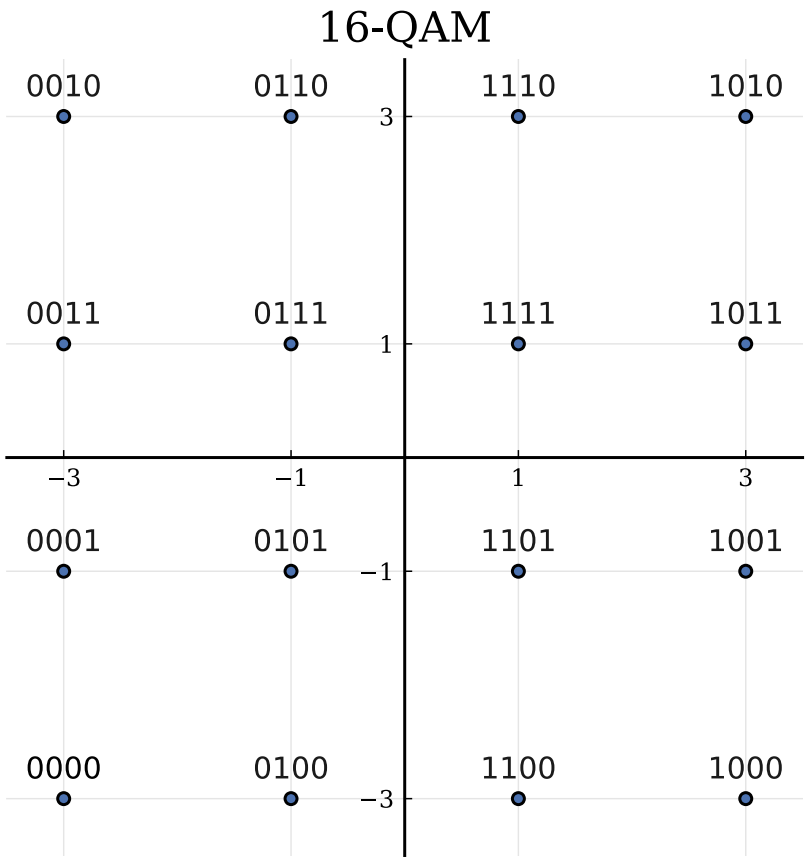


Figure 1: Ejemplo: Constelación 16-QAM, en donde $n = 4$.

Etapa Modulación

La constelación 16-QAM tiene algunas propiedades



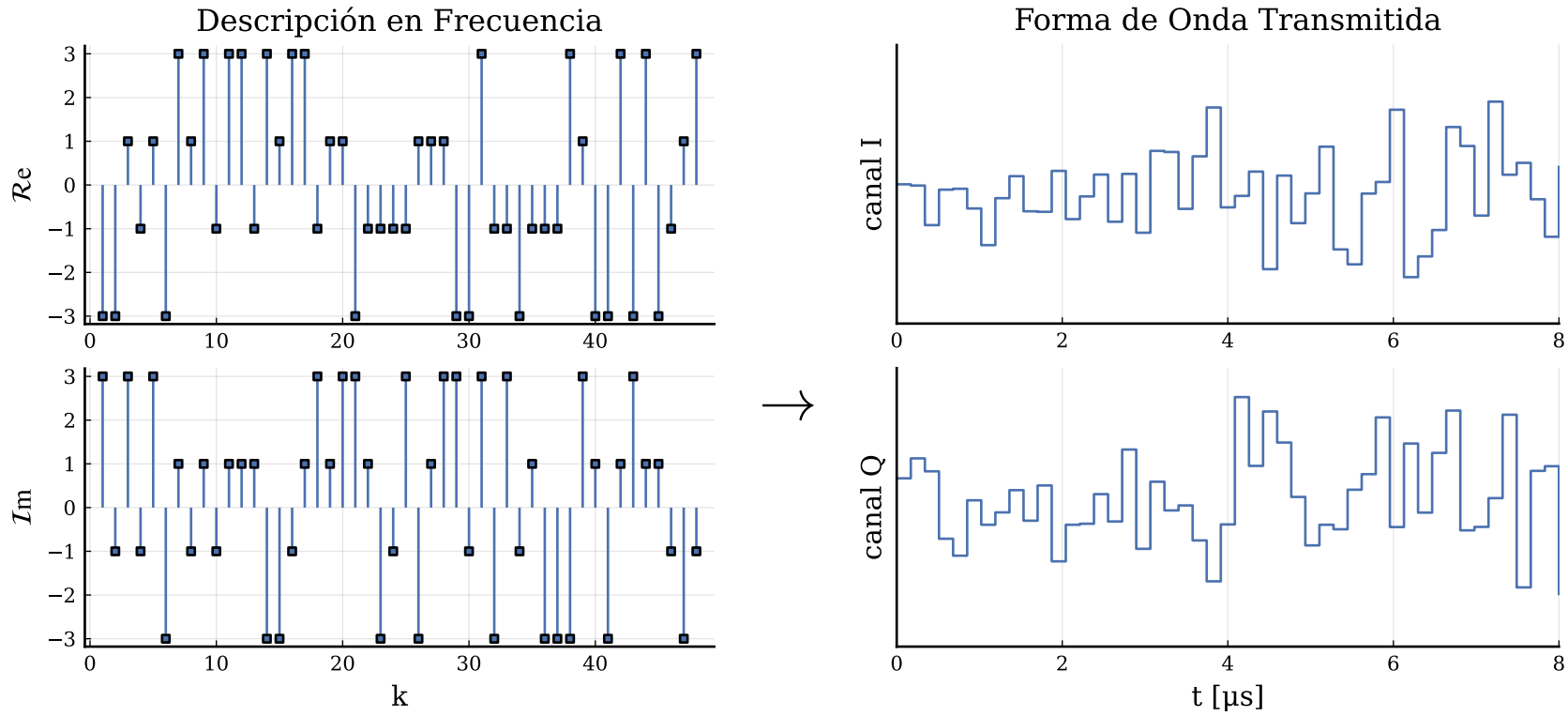
Primeros bits $\longrightarrow$ parte real
Últimos bits $\longrightarrow$ parte imag

Bits		Valor
00	$\longrightarrow$	-3
01	$\longrightarrow$	-1
11	$\longrightarrow$	1
10	$\longrightarrow$	3

Ejemplo: 1001$\longrightarrow 3 - j$

Etapa IFFT

- El bloque de 192 bits se transformó en 48 números $\mathbb{C}$
- Se interpretan como descripción en frecuencia de la señal, y se transforma al domino temporal usando una IFFT



Entrelazado Paralelo

Se aplica simultáneamente la transformación $k \rightarrow j \rightarrow i$

Para eso se instancian todos los índices de la trama y se interpretan de la siguiente forma

$$k' = \text{offset} + k$$

$$\underbrace{[k = 0 \quad \cdots \quad k = N-1}_{\textit{offset} = 0} \quad \underbrace{k = 0 \quad \cdots \quad k = N-1}_{\textit{offset} = 1} \quad \underbrace{k = 0 \quad \cdots \ ]}_{\cdots}$$

Las transformaciones se aplican sobre los índices k obteniendo los índices i, y se recupera el vector de nuevos índices

$$i' = \text{offset} + i$$

Función Entrelazado Paralelo

```
function interleave(stream::CuArray)
  N = 192
  s = 2
  all_idxs = CuArray{Int}(0:length(stream)-1)
  ks = all_idxs .% 192
  is = N÷16 .* (ks.%16) .+ floor.(ks.÷16)
  js = s.*floor.(is.÷s) .+ (is.+N.-floor.(16.*is./N)).%s
  offs = Int.(floor.(all_idxs./N).*N)
  return stream[js.+ offs .+ 1]
end
```

Modulación Paralelo

Estrategia para implementar la modulación de una trama de N bits usando QPSK en dos pasos:

$$\begin{bmatrix} 0\,1\,0\,1 & 1\,1\,0\,1 & 1\,0\,1\,1 & 1\,1\,0\,1 & 0\,1\,1\,0 & 1\,1\,1\,0 & \cdots \end{bmatrix}_N$$

1- Se reorganiza en una matriz de 4 filas y $N/4$ columnas

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & \cdots \\ 1 & 1 & 0 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 1 & 0 & 1 & 1 & \cdots \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots \end{bmatrix}_{4\times\frac{N}{4}}$$

2- Cada columna se reduce a su respectivo número complejo

$$\begin{bmatrix} -1-j & 1-j & -1+j & 1-j & -1+3j & 1+3j & \cdots \end{bmatrix}_{\frac{N}{4}}$$

Función Modulación

```
function modulate(stream::CuArray)
  N = length(stream)/4 |> Int
  result = CuVector{ComplexF64}(undef, N)
  block = reshape(stream, 4, N)
  @cuda threads=192 blocks=N modulate_kernel(result, block)
  return result
end

function modulate_kernel(result::CuVector, bits::CuMatrix)
  bits_map(x, y) = x ? (y ? 1 : 3) : (y ? -1 : -3)
  for idx in eachindex(result)
    @inbounds result[i] =
        bits_map(bits[1,idx], bits[2,idx]) + im*
        bits_map(bits[3,idx], bits[4,idx])
  end
  return nothing
end
```

IFFT Paralelo

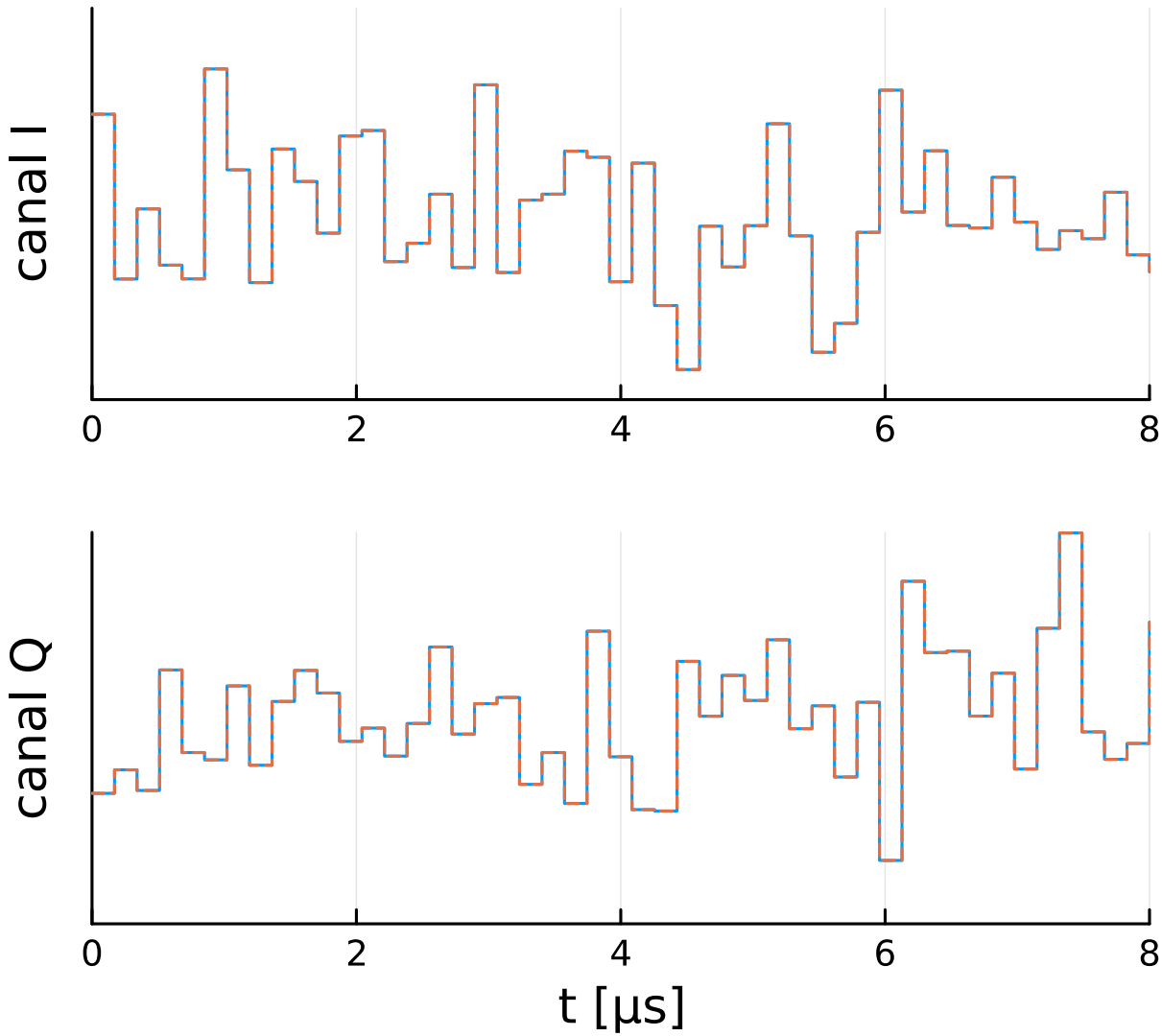
Se reorganiza el vector de N elementos en una matriz de $48 \times N/48$ y se aplica la IFFT por columnas.

$$\begin{bmatrix} x_0 & x_{48} & x_{96} \\ \vdots & \vdots & \vdots \\ x_{47} & x_{95} & x_{143} \end{bmatrix}_{48\times\frac{N}{48}} \xrightarrow{\text{IFFT}} \begin{bmatrix} \check{x}_0 & \check{x}_{48} & \check{x}_{96} \\ \vdots & \vdots & \vdots \\ \check{x}_{47} & \check{x}_{95} & \check{x}_{143} \end{bmatrix}_{48\times\frac{N}{48}}$$

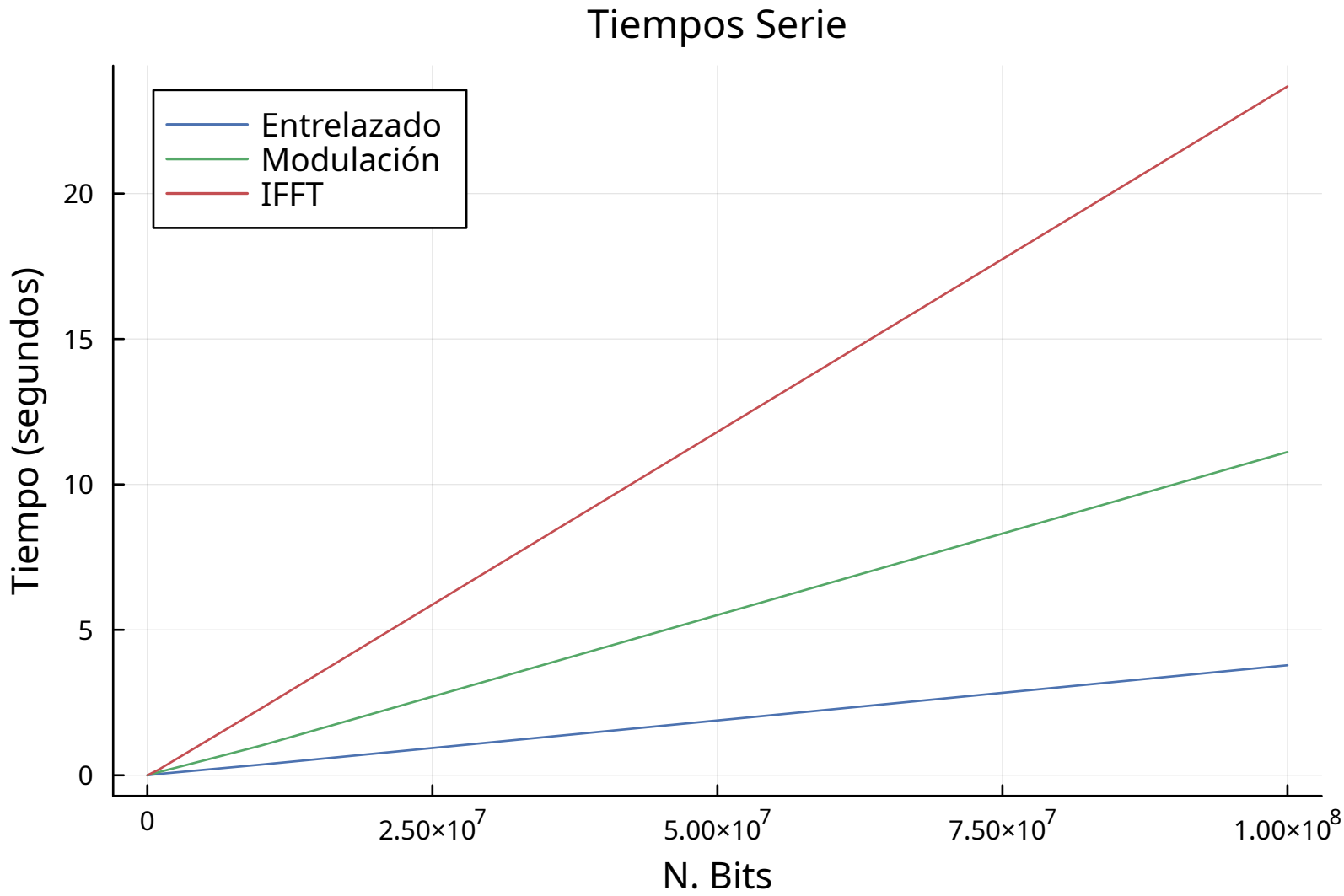
Luego se vuelve a organizar el resultado en un vector de N elementos y se retorna.

```
function to_waveform(stream::CuArray)
    block = reshape(stream, 48, :)
    t_block = CUFFT.ifft(block, 1)
    return reshape(t_block, size(stream))
end
```

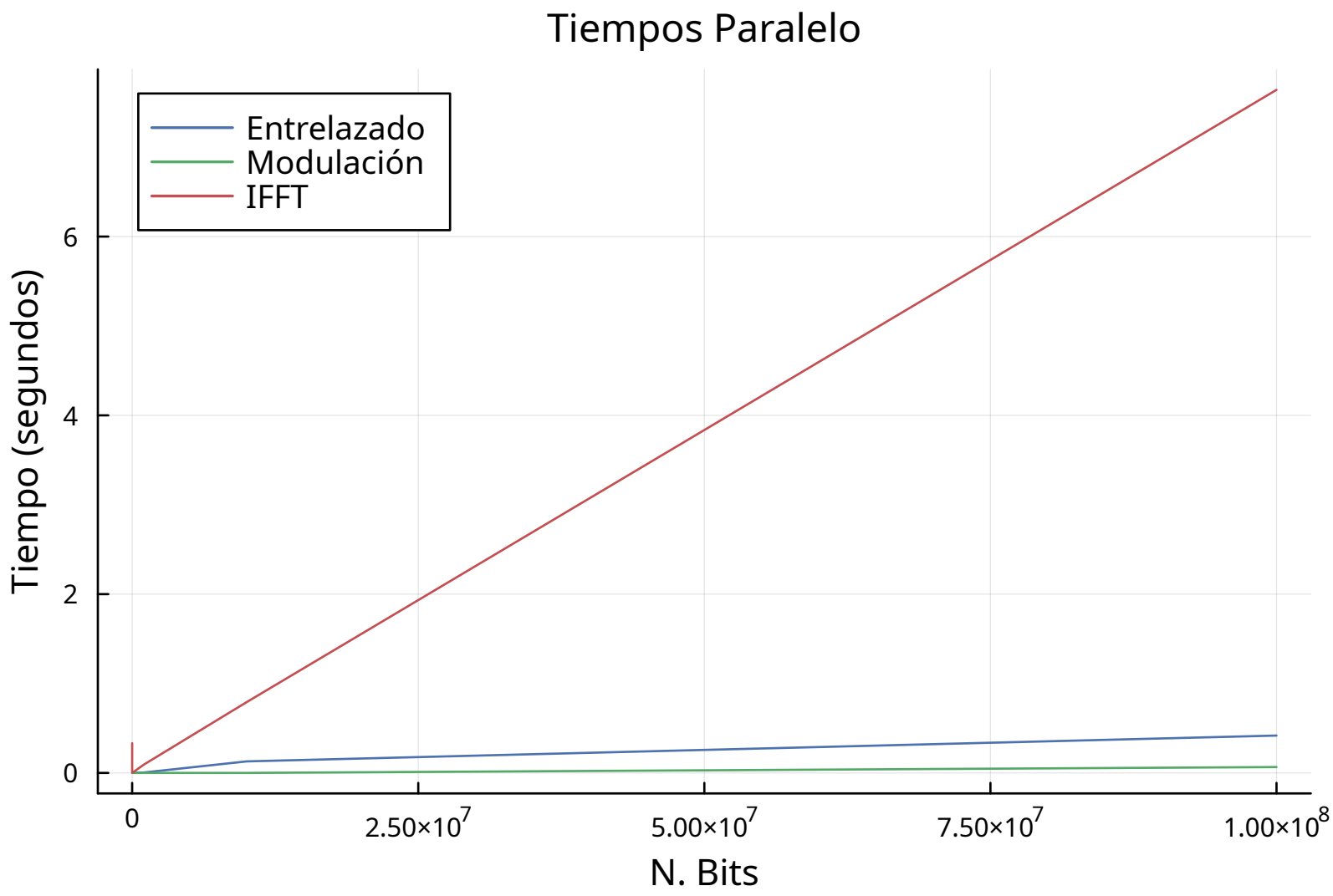
Validación Igual Resultado Serie - Paralelo



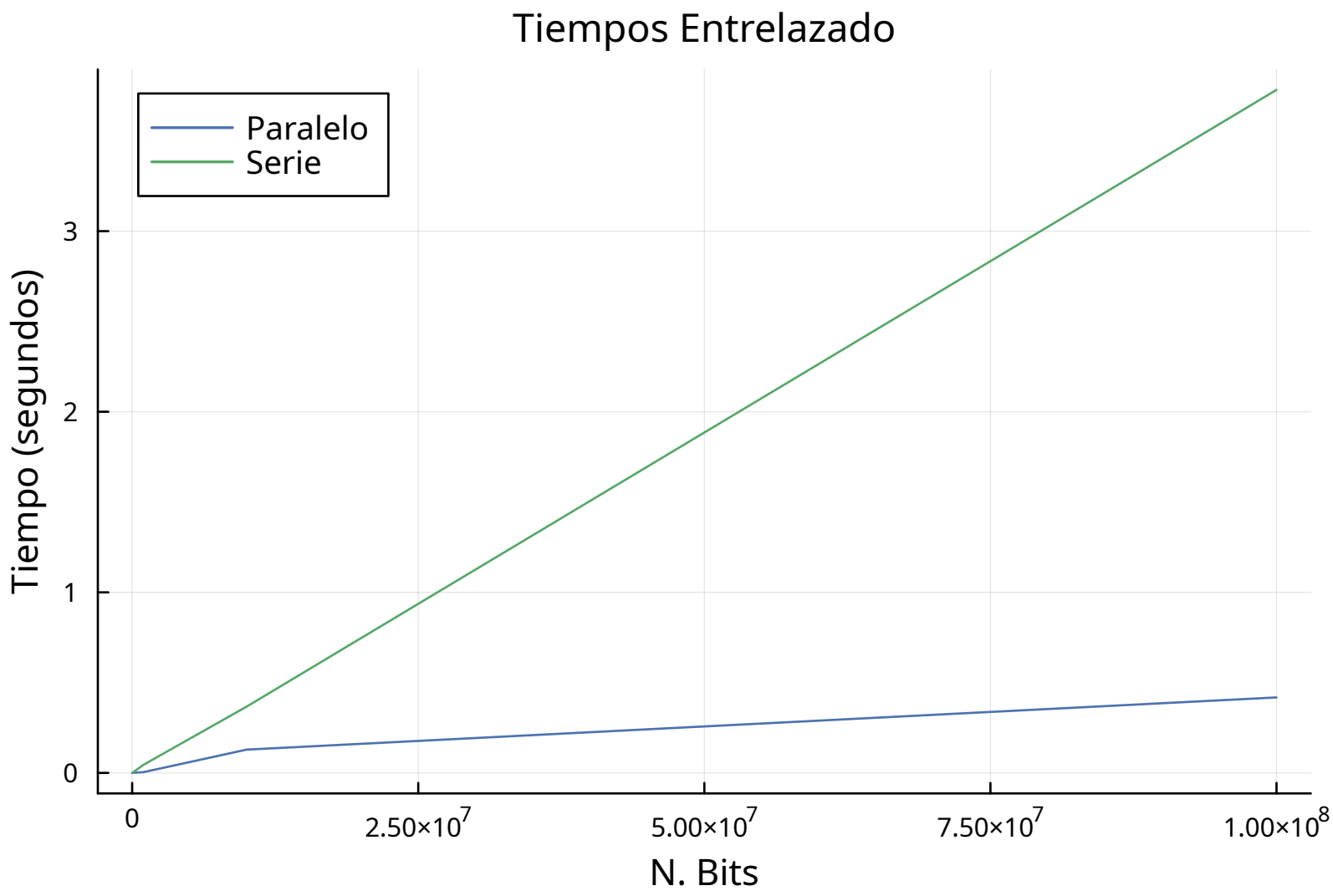
Tiempo de Etapas en Serie



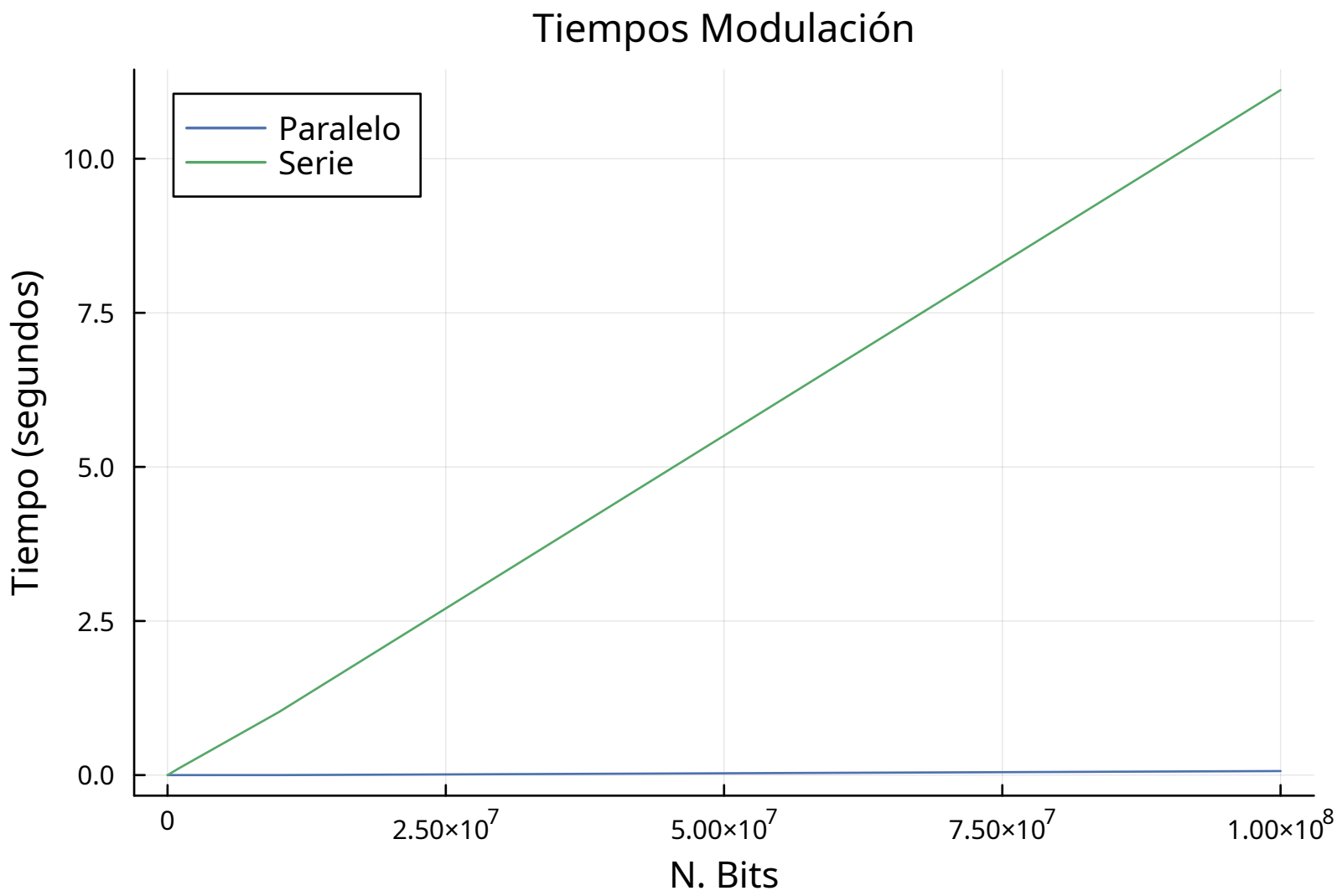
Tiempo Etapas en Paralelo



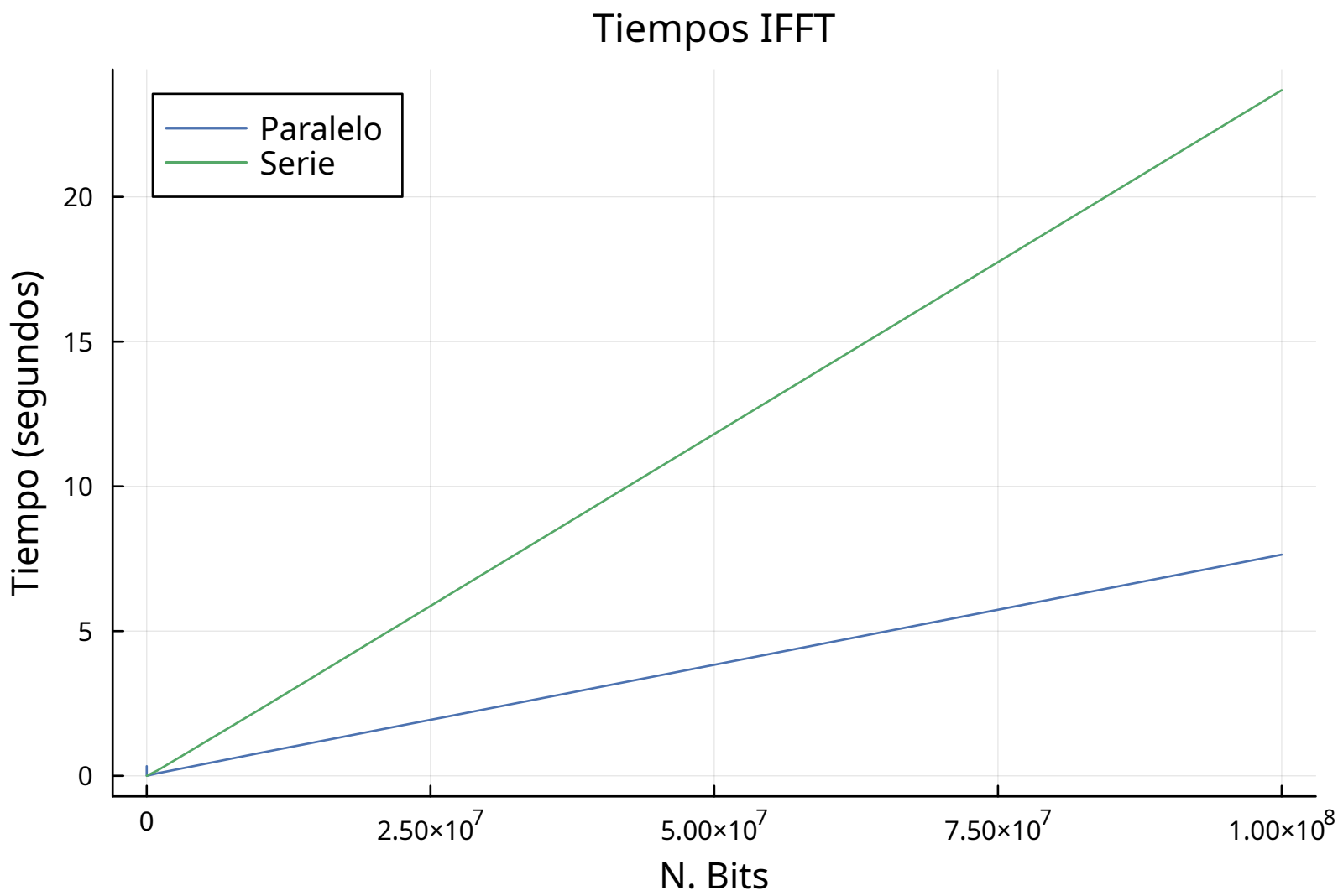
Aceleración Entrelazado



Aceleración Modulación



Aceleración IFFT



Conclusiones

- En todas las etapas el tiempo es lineal con el número de bits, lo cual era esperado.
- En todos los casos hay aceleración cuando se paraleliza el código. En el peor de los casos es una aceleración de $5\times$
- La máxima aceleración se obtuvo cuando se utilizó un kernel y se eligió el número de hilos y de bloques.
- Para optimizar el entrelazado paralelo se puede implementar un kernel en lugar de usar broadcast.
- Para optimizar la IFFT se considera configurar el parámetro `batch` de `CUFFT.plan_ifft`.