Transmisor OFDM

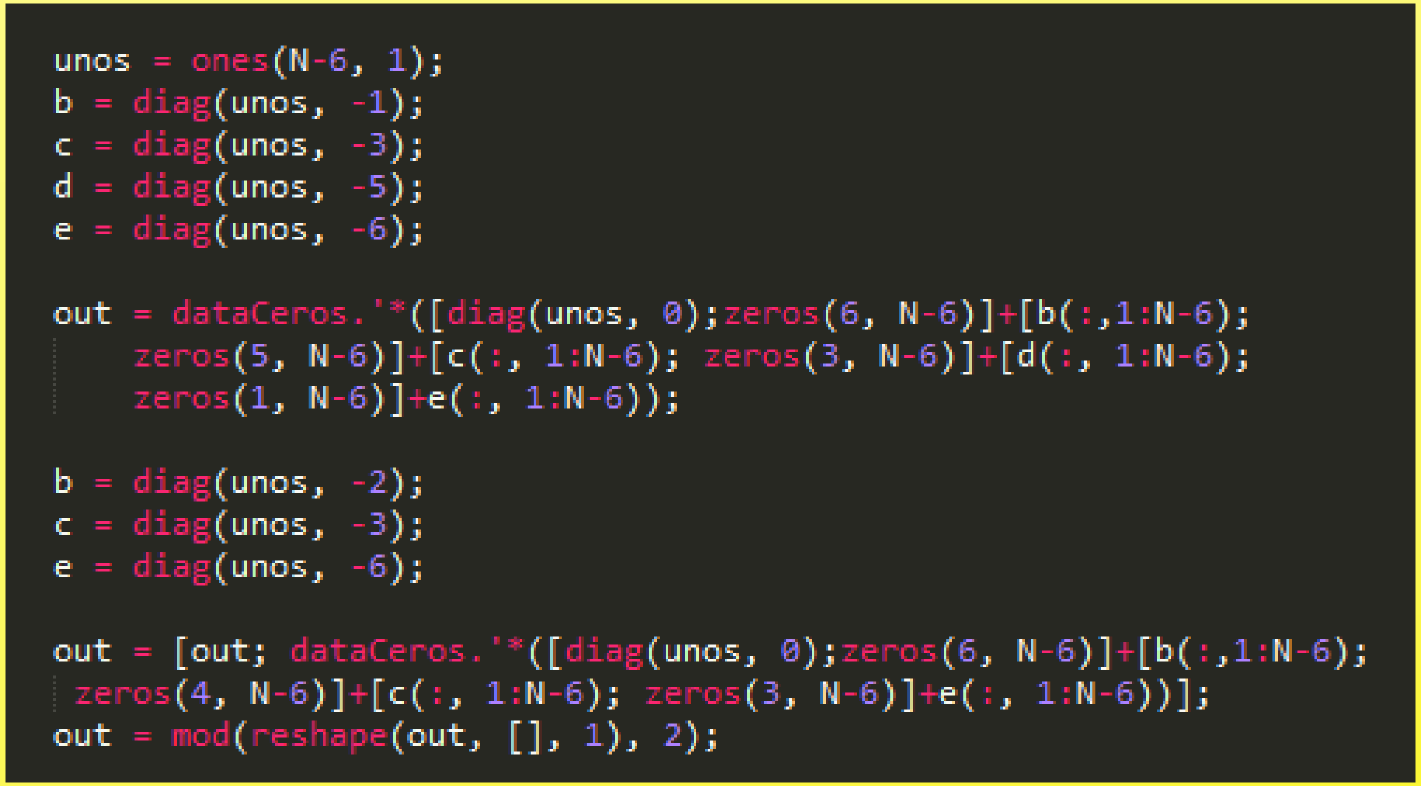
González Bonet, Santiago de Jesús

Fernández-Palacios Serrano, Jaime

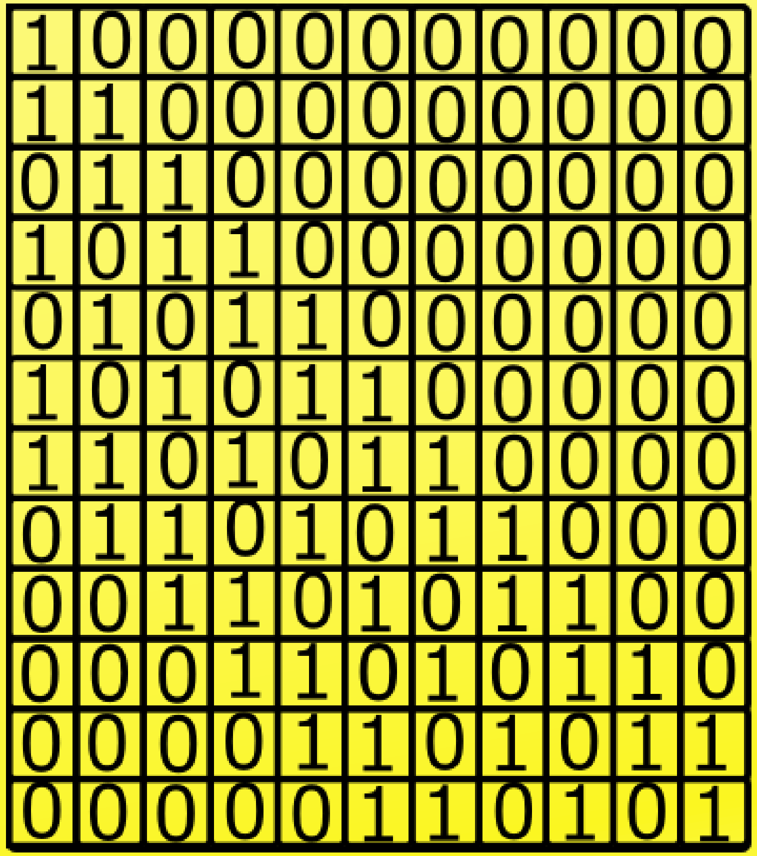
Linares Serrano, Pablo

MATLAB

Codificador Convolucional



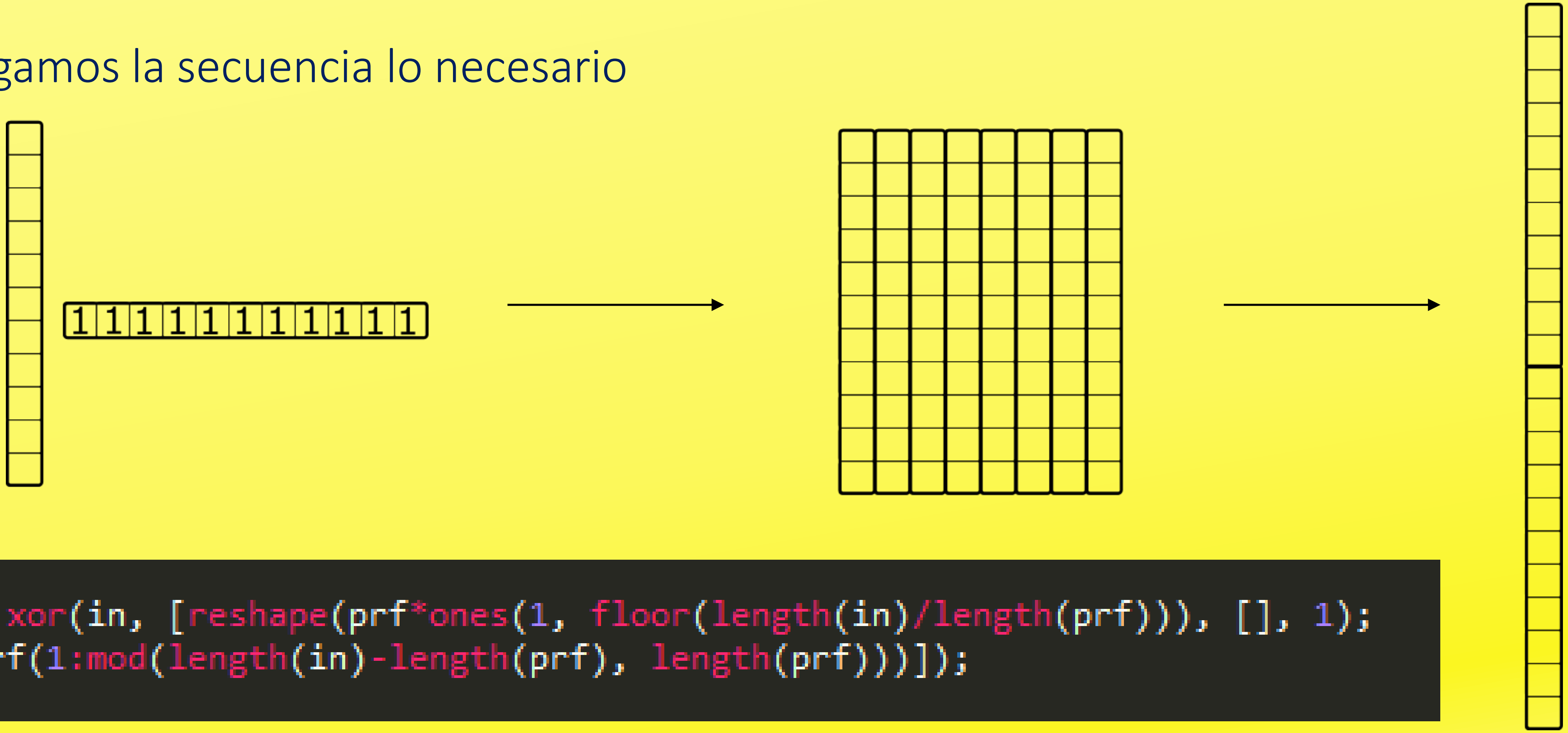
```
unos = ones(N-6, 1);
b = diag(unos, -1);
c = diag(unos, -3);
d = diag(unos, -5);
e = diag(unos, -6);

out = dataCeros.'*([diag(unos, 0);zeros(6, N-6)]+[b(:,1:N-6);
    zeros(5, N-6)]+[c(:, 1:N-6); zeros(3, N-6)]+[d(:, 1:N-6);
    zeros(1, N-6)]+e(:, 1:N-6));

b = diag(unos, -2);
c = diag(unos, -3);
e = diag(unos, -6);

out = [out; dataCeros.'*([diag(unos, 0);zeros(6, N-6)]+[b(:,1:N-6);
 zeros(4, N-6)]+[c(:, 1:N-6); zeros(3, N-6)]+e(:, 1:N-6))];
out = mod(reshape(out, [], 1), 2);
```

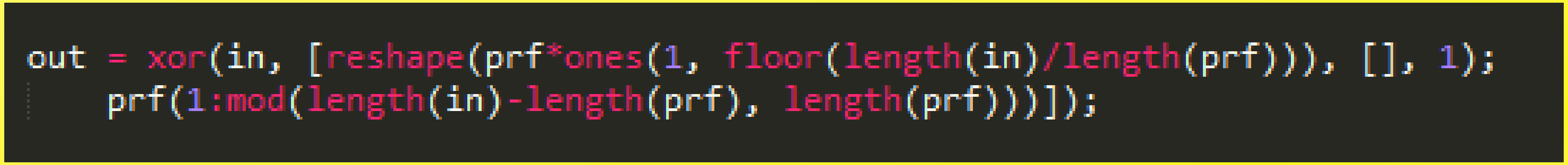


1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0
1	0	1	0	1	1	0	0	0	0	0
1	1	0	1	0	1	1	0	0	0	0
0	1	1	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	1	1	0	0
0	0	0	1	1	0	1	0	1	1	0
0	0	0	0	1	1	0	1	0	1	1
0	0	0	0	0	1	1	0	1	0	1

Scrambler

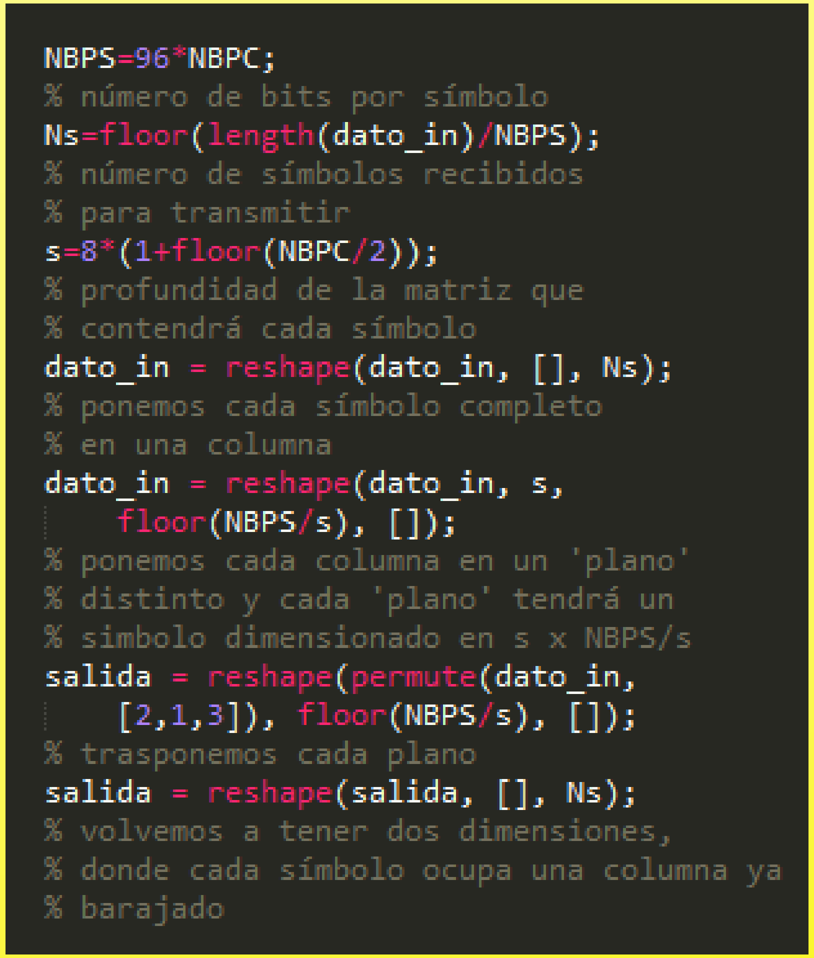
Alargamos la secuencia lo necesario



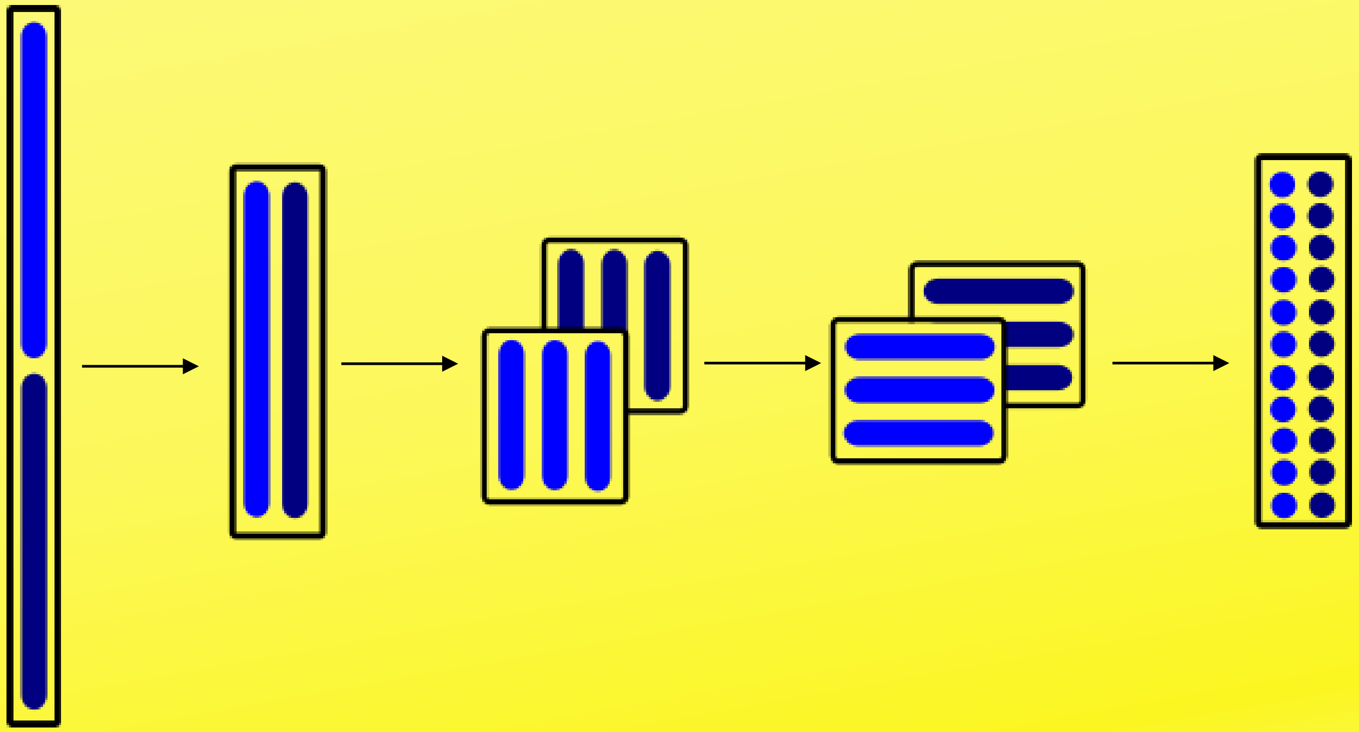


```
out = xor(in, [reshape(prf*ones(1, floor(length(in)/length(prf))), [], 1);
    prf(1:mod(length(in)-length(prf), length(prf)))]);
```

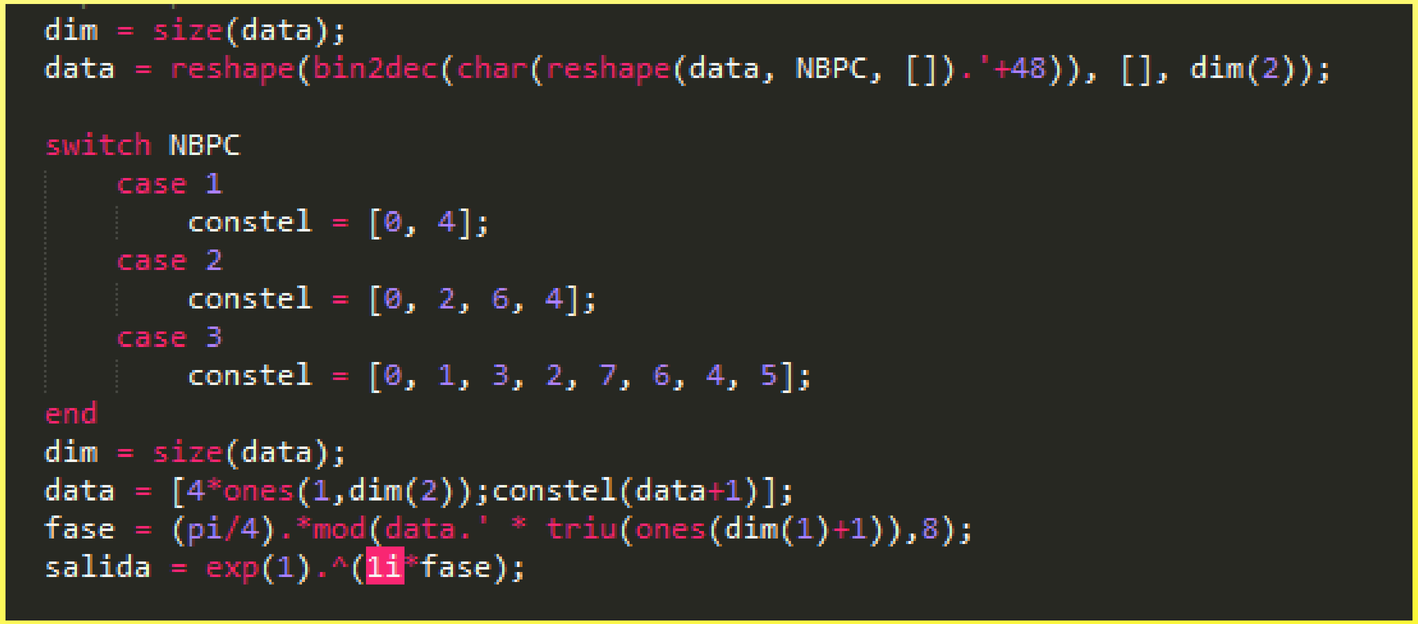
Interleaver



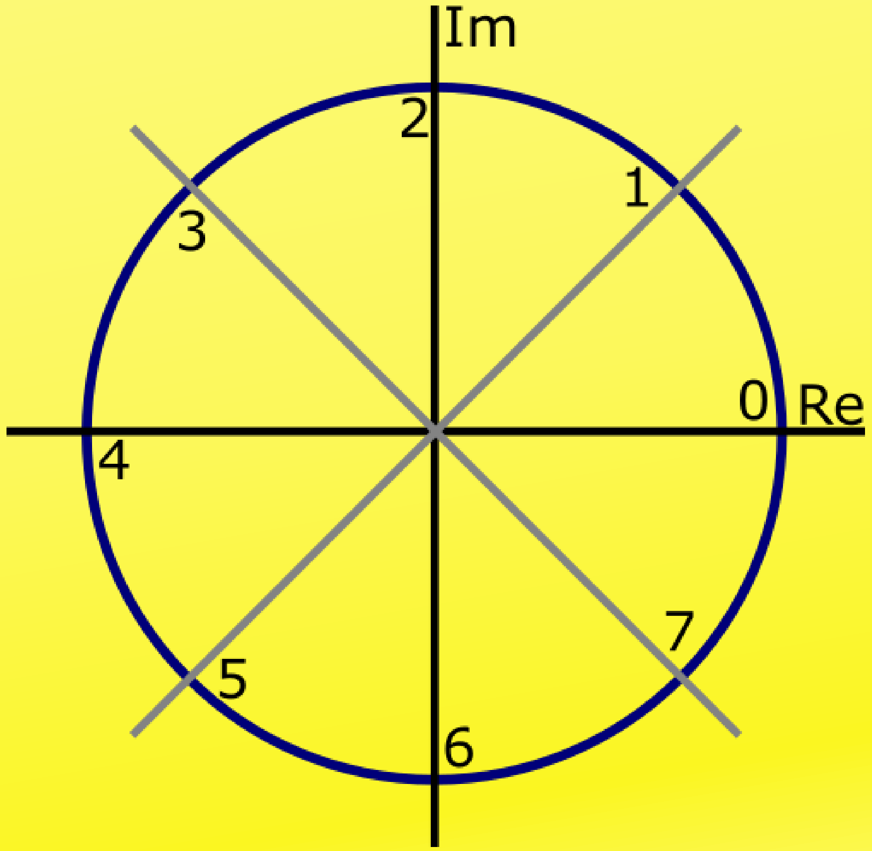
```
NBPS=96*NBPC;
% número de bits por símbolo
Ns=floor(length(dato_in)/NBPS);
% número de símbolos recibidos
% para transmitir
s=8*(1+floor(NBPC/2));
% profundidad de la matriz que
% contendrá cada símbolo
dato_in = reshape(dato_in, [], Ns);
% ponemos cada símbolo completo
% en una columna
dato_in = reshape(dato_in, s,
    floor(NBPS/s), []);
% ponemos cada columna en un 'plano'
% distinto y cada 'plano' tendrá un
% simbolo dimensionado en s x NBPS/s
salida = reshape(permute(dato_in,
    [2,1,3]), floor(NBPS/s), []);
% trasponemos cada plano
salida = reshape(salida, [], Ns);
% volvemos a tener dos dimensiones,
% donde cada símbolo ocupa una columna ya
% barajado
```



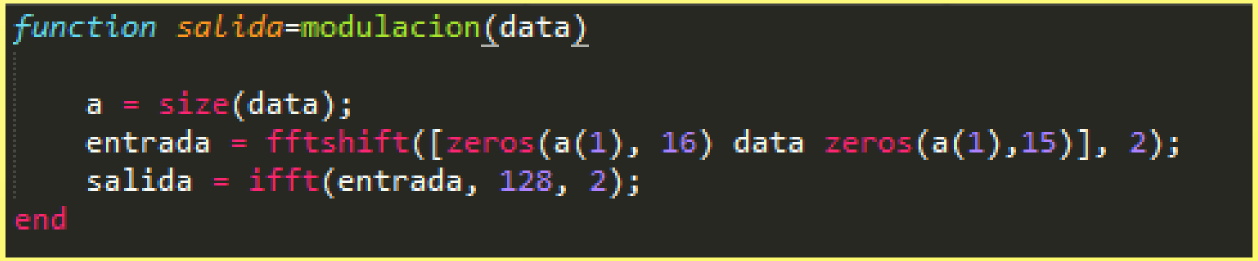
Mapper



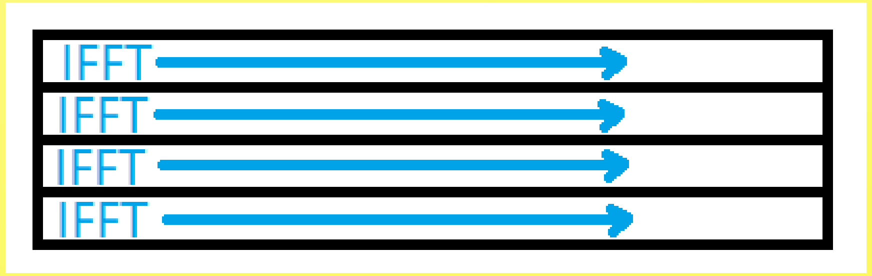
```
dim = size(data);
data = reshape(bin2dec(char(reshape(data, NBPC, []).'+48)), [], dim(2));

switch NBPC
    case 1
        constel = [0, 4];
    case 2
        constel = [0, 2, 6, 4];
    case 3
        constel = [0, 1, 3, 2, 7, 6, 4, 5];
end
dim = size(data);
data = [4*ones(1,dim(2));constel(data+1)];
fase = (pi/4).*mod(data.' * triu(ones(dim(1)+1)),8);
salida = exp(1).^(1i*fase);
```

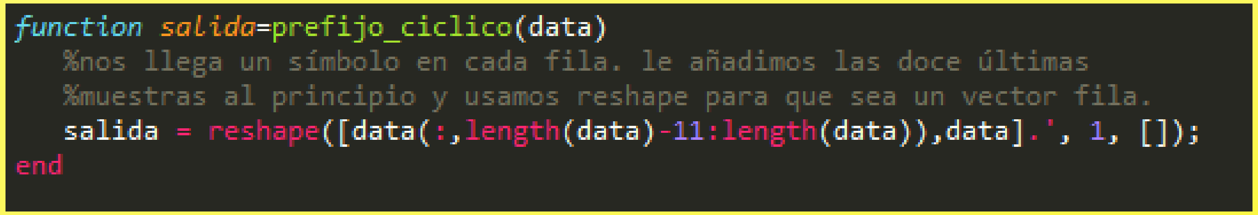


IFFT y prefijo cíclico

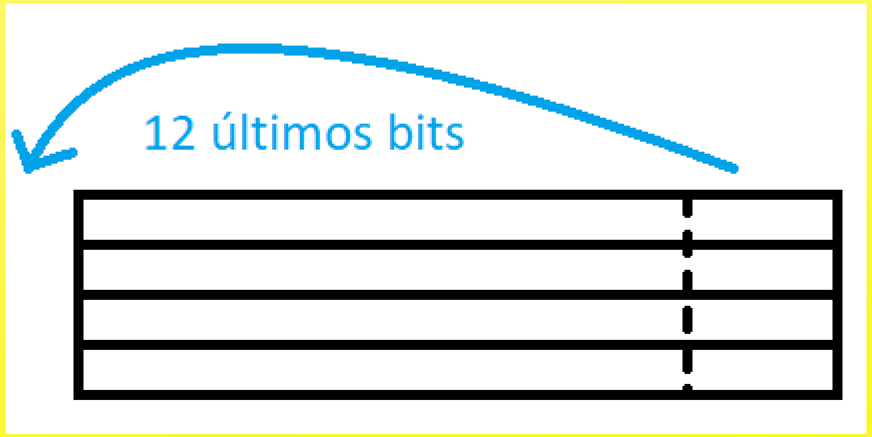


```
function salida=modulacion(data)

    a = size(data);
    entrada = fftshift([zeros(a(1), 16) data zeros(a(1),15)], 2);
    salida = ifft(entrada, 128, 2);
end
```



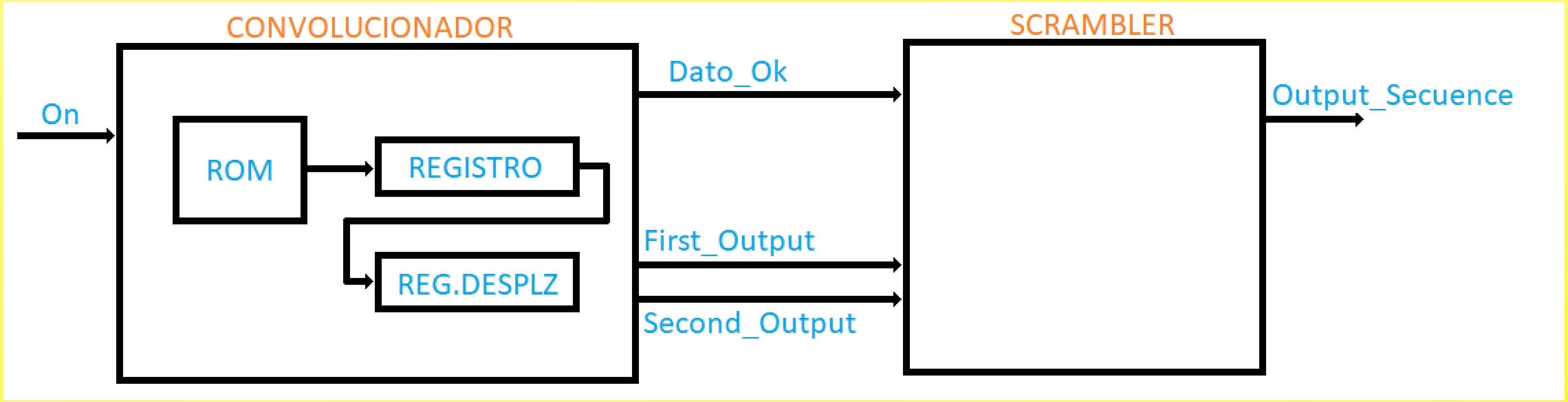


```
function salida=prefijo_ciclico(data)
   %nos llega un símbolo en cada fila. le añadimos las doce últimas
   %muestras al principio y usamos reshape para que sea un vector fila.
   salida = reshape([data(:,length(data)-11:length(data)),data].', 1, []);
end
```

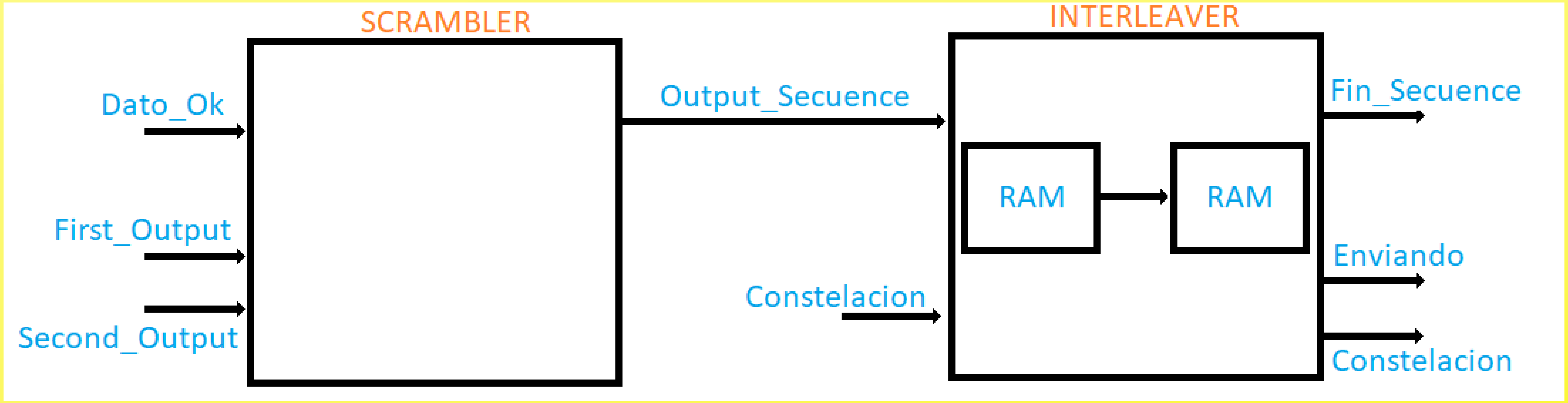


VHDL

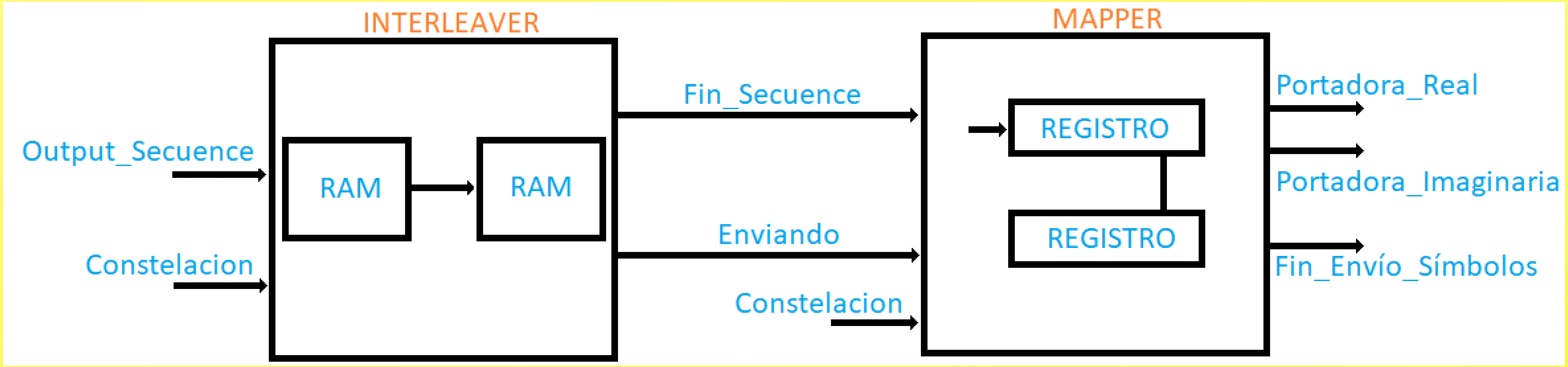
Codificador – Scrambler



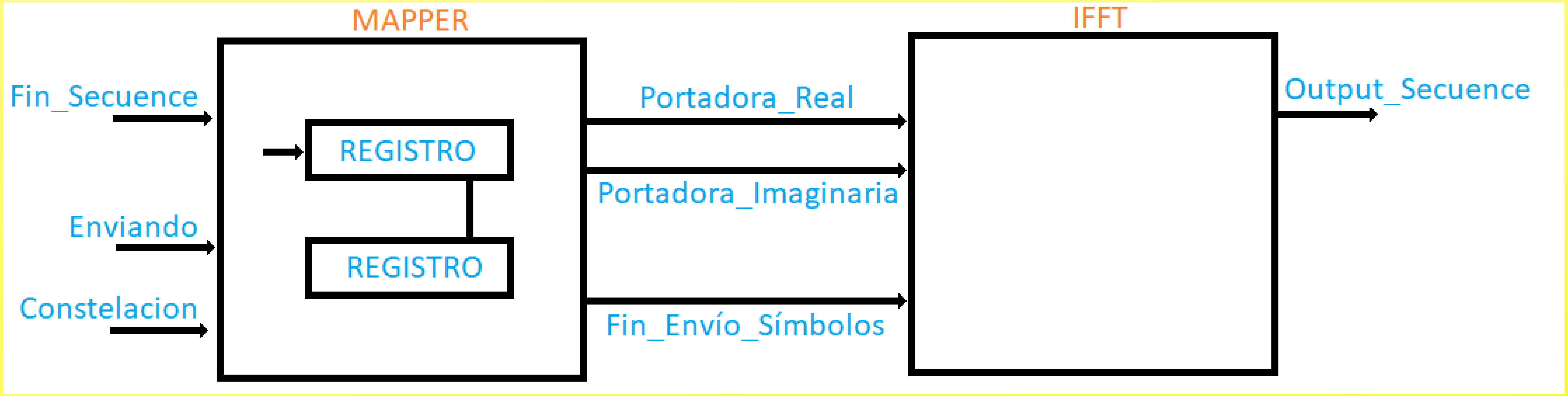
Scrambler – Interleaver



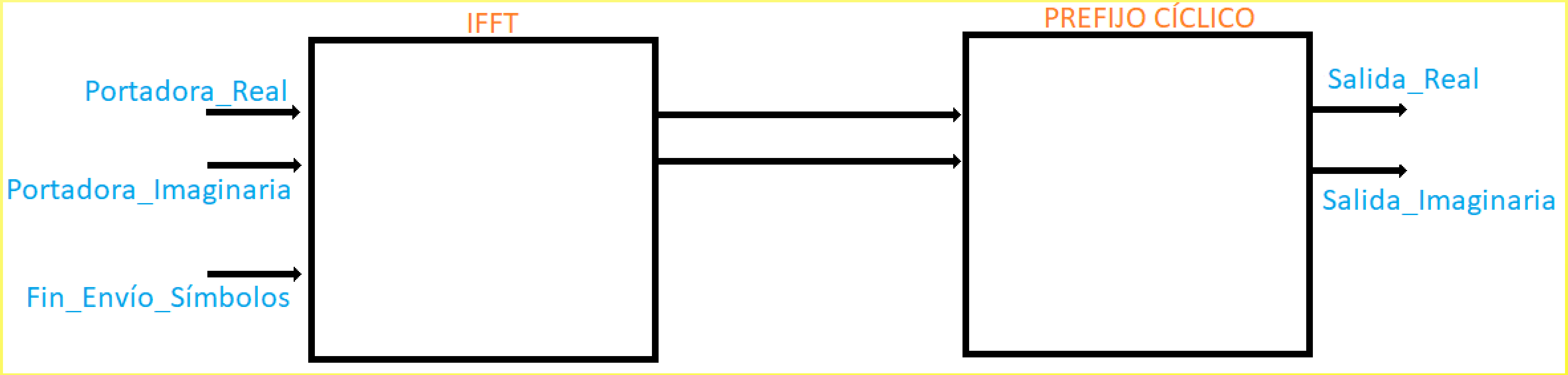
Interleaver – Mapper



Mapper – IFFT


MAPPER
IFFT
Fin_Secuence
Enviando
Constelacion
REGISTRO
REGISTRO
Portadora_Real
Portadora_Imaginaria
Fin_Envío_Símbolos
Output_Secuence

IFFT – Prefijo Cíclico



- **ALGORITMO EMPLEADO:** Radix2
- **FRECUENCIA:** 50Mhz
- **OUTPUT ORDERING:** Natural order
- **SCALING OPTIONS:** Unscaled ó Scaled
- **TAMAÑO TRANSFORMADA:** 128bits
- **ROUNDING MODES:** Convergent Rounding
- **INPUT DATA TIMING:** No offset.