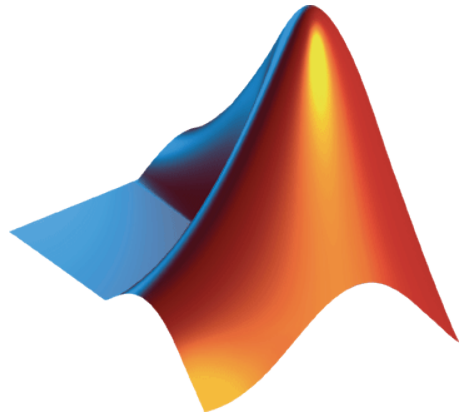
Modeling a 4G-LTE System in MATLAB



Houman Zarrinkoub PhD.
Product Manager
Signal Processing & Communications
MathWorks

Agenda

- 4G LTE and LTE Advanced standards
- MATLAB and communications system design
- Case study: A 4G LTE system model in MATLAB
  - Modeling & simulation
  - Simulation acceleration
  - Path to implementation
- Summary

4G LTE and LTE Advanced

Distinguishing Features

- Motivation
  - Very high capacity & throughput
  - Support for video streaming, web browsing, VoIP, mobile apps
- A true global standard
  - Contributions from all across globe
  - Deployed in AMER, EMEA, APLA
- A true broadband mobile standard
  - From 2 Mbps (UMTS)
  - To 100 Mbps (LTE)
  - To 1 Gbps (LTE Advanced)

Standard	Low Mobility	High Mobility
EDGE	~250 kbps	
WCDMA / UMTS	2 Mbps	384 kbps
HSDPA	14 Mbps	
HSPA+	42 Mbps	
LTE (R8 or R9)	100 Mbps	
4G Requirement	1 Gbps	100 Mbps
LTE Advanced (*)	> 1 Gbps	> 100 Mbps

How is this remarkable advance possible?

- Integration of enabling technologies with sophisticated mathematical algorithms
  - OFDM (Multi-carrier transmission, large bandwidth requires attention to multipath fading)
  - MIMO (multiple antennas)
  - Turbo Coding (near Shannon limit, e.g., near capacity channel coding)

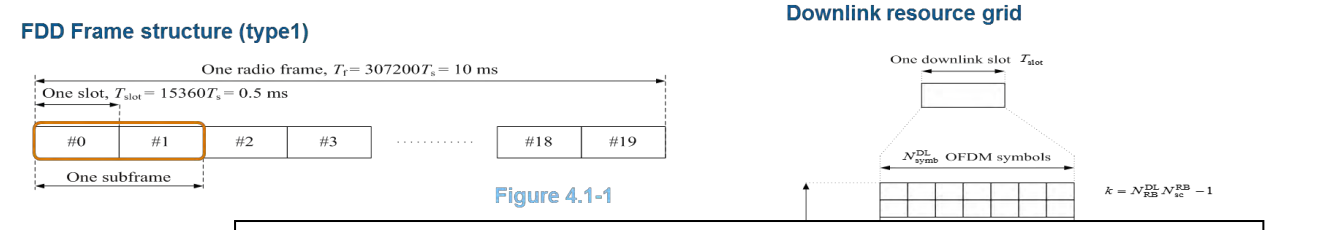
- Smart usage of resources and bandwidth (smart engineering techniques)
  - Adaptive modulation
  - Adaptive coding
  - Adaptive MIMO
  - Adaptive bandwidth

What MATLAB users care about LTE standard?

- Academics
  - Researchers contributing to future standards
  - Professors
  - Students
- Practitioners
  - System Engineers
  - Software designers
  - Implementers of wireless systems
- Challenge in interaction and cooperation between these two groups
- MATLAB is their common language

Challenges:
From specification to implementation

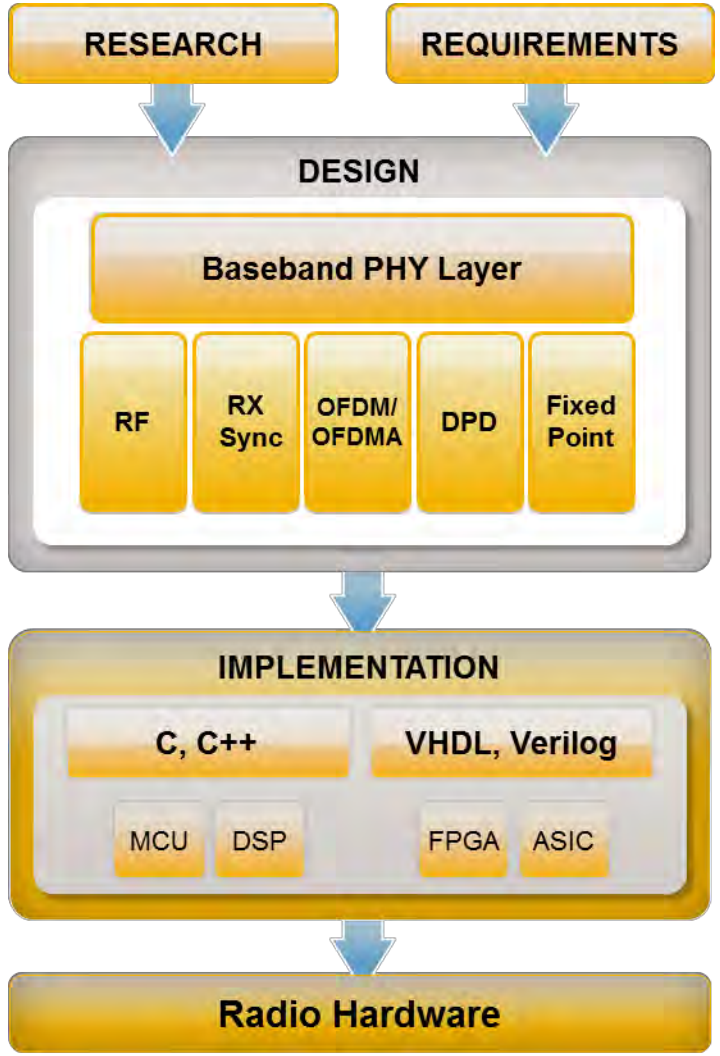
- Simplify translation from specification to a model as blue-print for implementation
- Introduce innovative proprietary algorithms
- Dynamic system-level performance evaluation
- Accelerate simulation of large data sets
- Address gaps in the implementation workflow



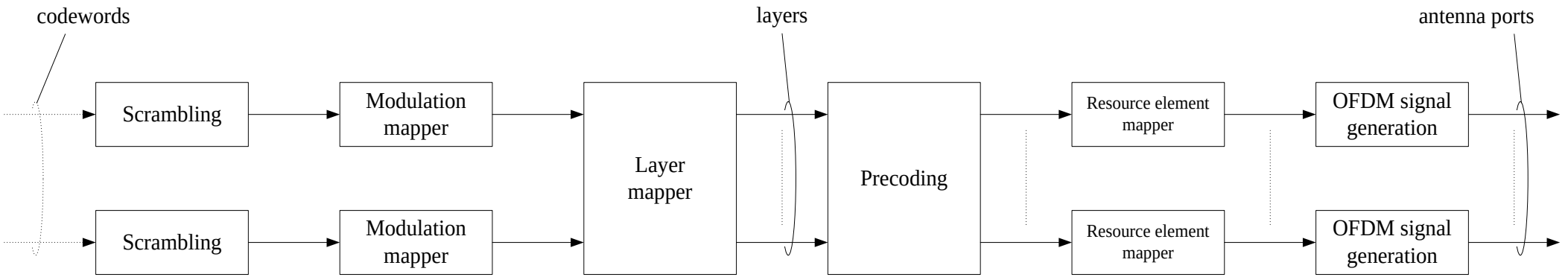

```
51    -- Component Configuration Statements
52    FOR ALL : filter_bank_left
53      USE ENTITY work.filter_bank_left(rtl);
54
55    -- Signals
56    SIGNAL filter_bank_left_out1           : std_logic_vector(15 DOWNTO 0);  -- ufix16
57    SIGNAL filter_bank_right_out1          : std_logic_vector(15 DOWNTO 0);  -- ufix16
58
59  BEGIN
60    -- <S6>/filter_bank_left
61    u_filter_bank_left : filter_bank_left
62      PORT MAP( clk => clk,
63                reset => reset,
64                enb => enb,
65                input => LeftIn,   -- sfix16_En15
66                parameters => Parameters,   -- uint8 [10]
67                gain => Gain,   -- sfix16_En11
68                data_out => filter_bank_left_out1   -- sfix16_En8
69                );
70
71    -- <S6>/filter_bank_right
72    u_filter_bank_right : filter_bank_left
73      PORT MAP( clk => clk,
74                reset => reset,
75                enb => enb,
76                input => RightIn,   -- sfix16_En15
77                parameters => Parameters,   -- uint8 [10]
78                gain => Gain,   -- sfix16_En11
79                data_out => filter_bank_right_out1   -- sfix16_En8
80                );
81
82    LeftOut <= filter_bank_left_out1;
83
84    RightOut <= filter_bank_right_out1;
```

Where does MATLAB fit in addressing these challenges?

- MATLAB and Communications System Toolbox are ideal for LTE algorithm and system design
- MATLAB and Simulink provide an environment for dynamic & large scale simulations
- Accelerate simulation with a variety of options in MATLAB
- Connect system design to implementation with
  - C and HDL code generation
  - Hardware-in-the-loop verification



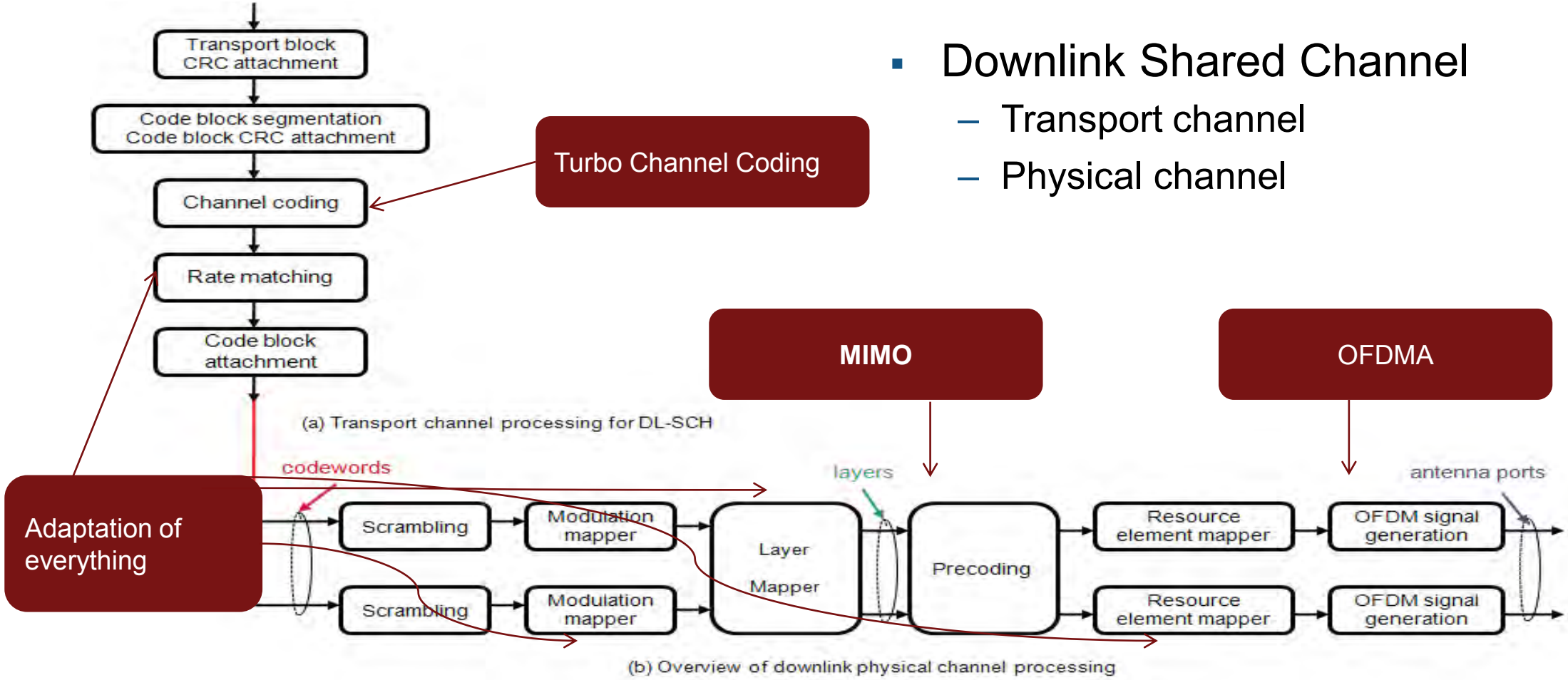
Case Study:
Downlink physical layer of LTE (Release 10)



Modeling and simulation

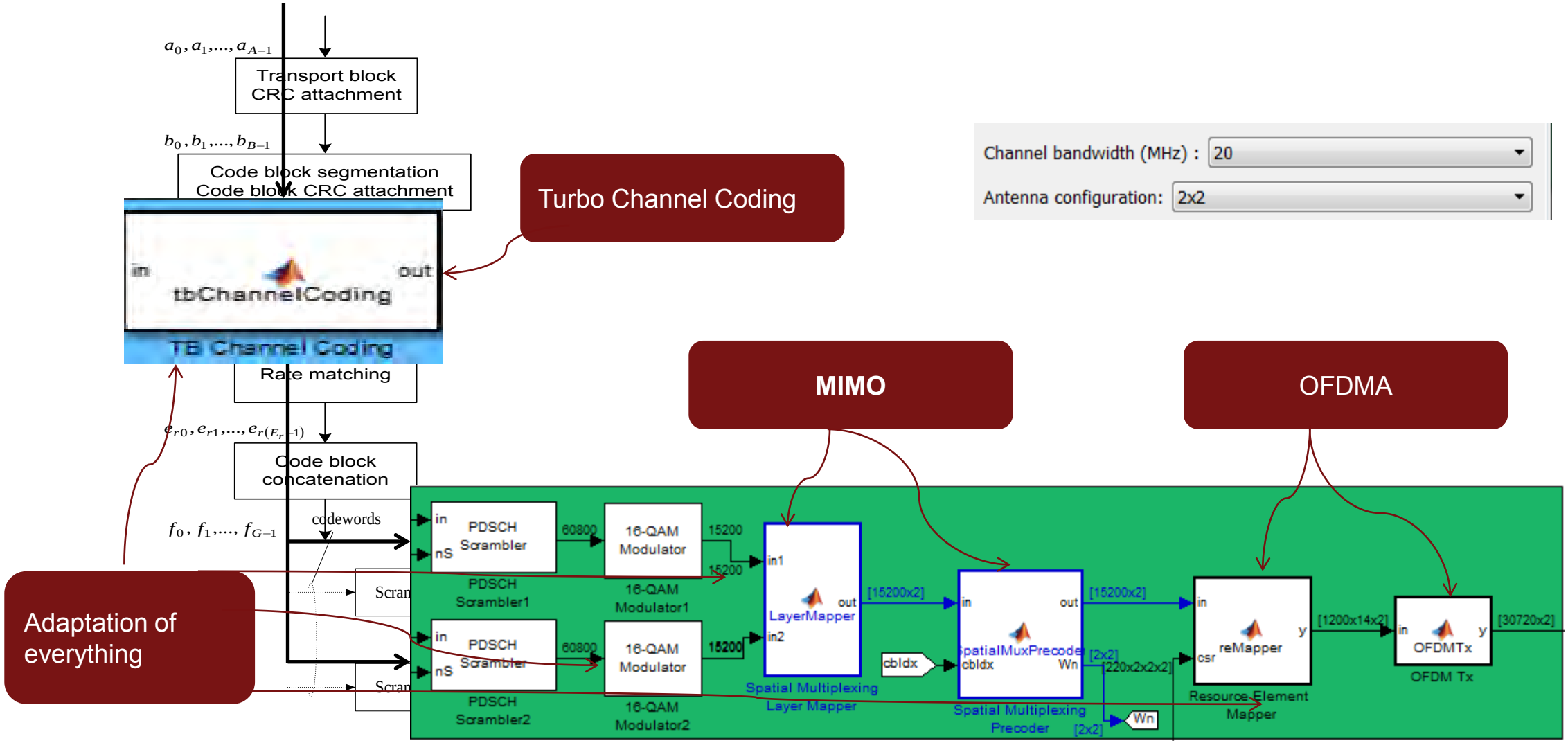
LTE Physical layer model in standard

- Downlink Shared Channel
  - Transport channel
  - Physical channel



Reference: 3GPP TS 36 211 v10 (2010-12)

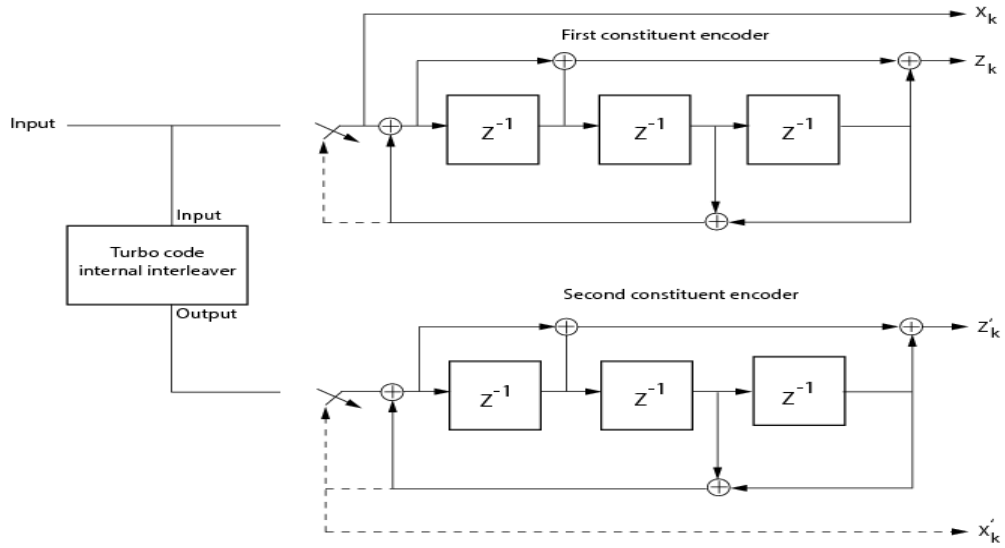
LTE Physical layer model in MATLAB

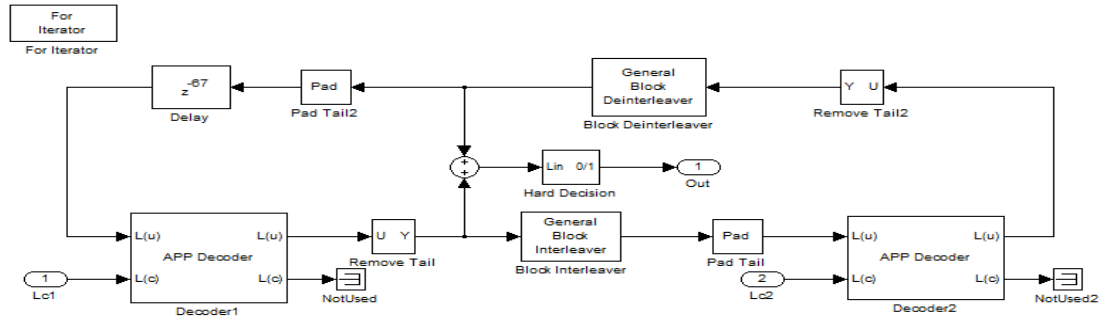


>> Open LTE system model

Overview of Turbo Coding

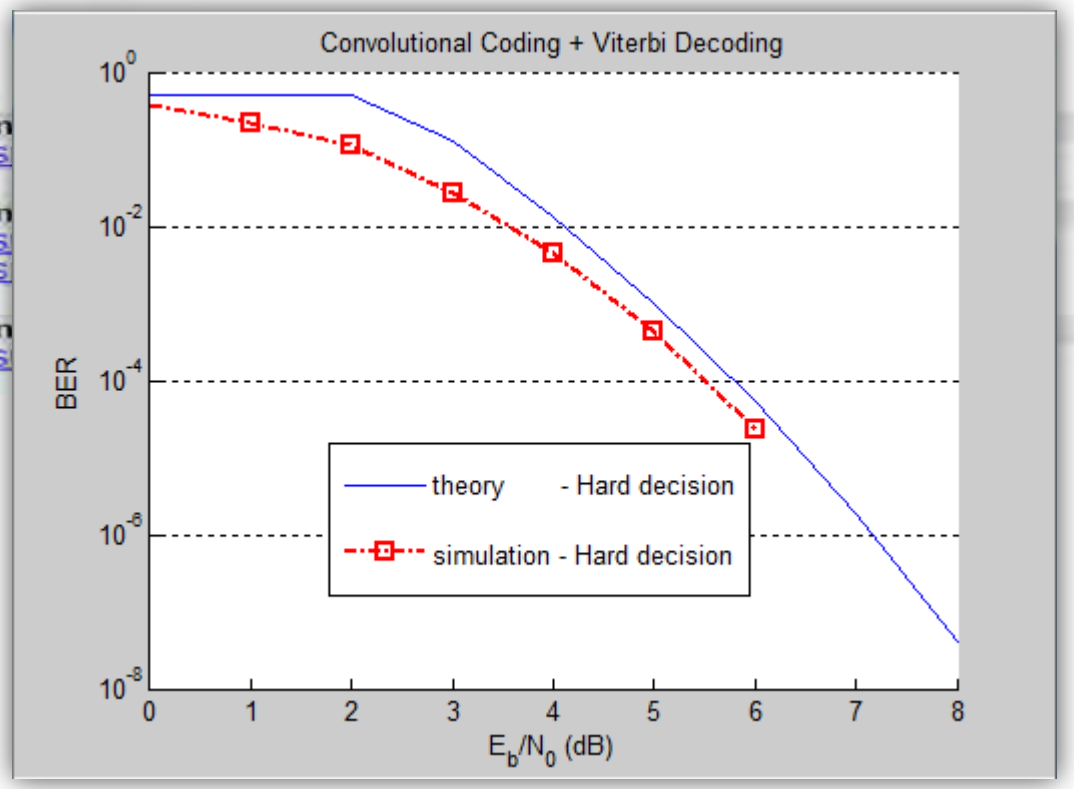
- Error correction & coding technology of LTE standard
- Performance: Approach the channel capacity (Shannon bound)
- Represents an evolution of convolutional coding
- Based on an iterative decoding scheme





MATLAB Demo

Modeling and Simulation of a transceiver system, showcasing:
Coding Gain:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding
- Turbo Encoding & Decoding



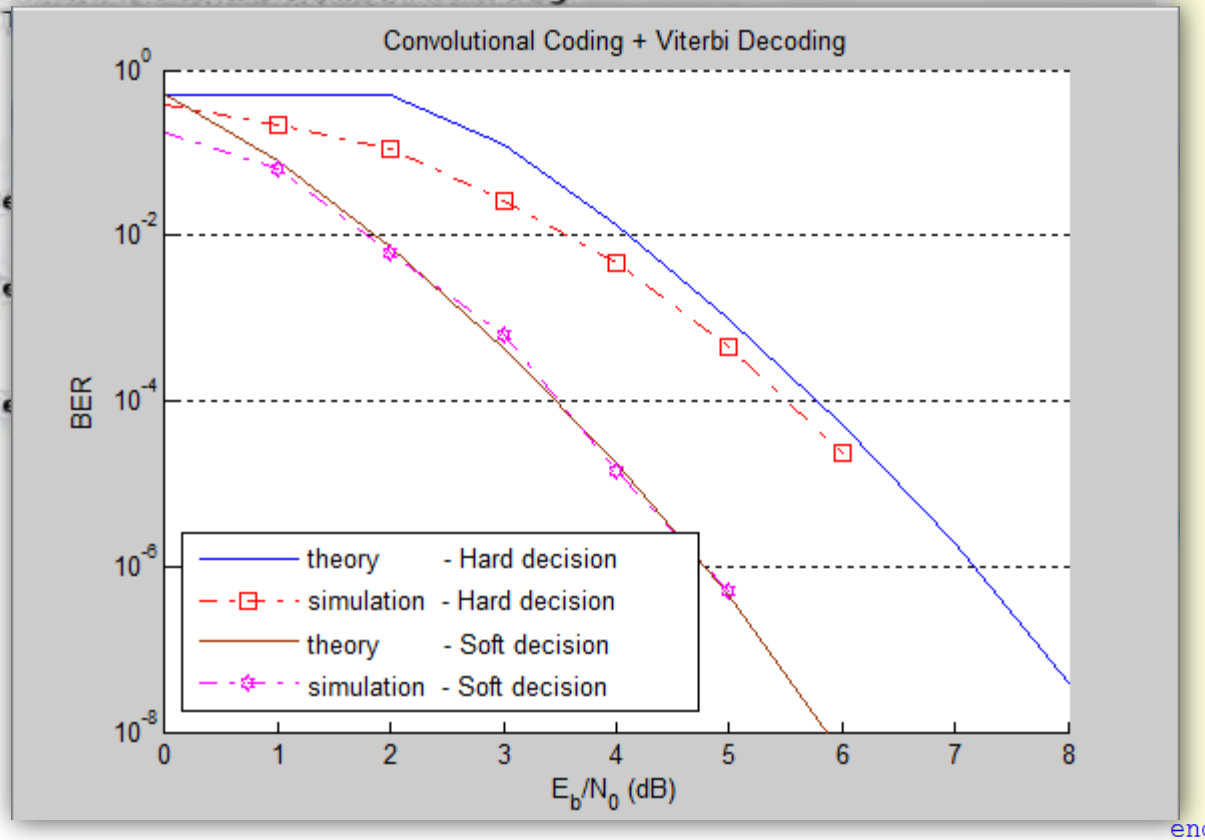

```
1   function [ber bits]=fcn_04PSK_Viterbi(EbNo,MaxNumErrs,MaxNumBits)
2   %% Initialization
3   % Components
4   persistent hModulator hAWGN hDeModulator hBitError
5   persistent hConvEncoder hViterbi
6   if isempty(hModulator)
7       hModulator = comm.PSKModulator(...
8           'ModulationOrder',4, ...
9           'PhaseOffset',0, ...
10          'BitInput',true);
11      hAWGN = comm.AWGNChannel;
12      hDeModulator = comm.PSKDemodulator(...
13
14
15
16          'DecisionMethod','Hard decision');
17      hBitError = comm.ErrorRate;
18
19
20
21
22          'OutputDataType', 'double',...
23          'TerminationMethod','Terminated');
24  end% Parameters
```

= comm.ConvolutionalEncoder(

= comm.ViterbiDecoder(…
'InputFormat', 'Hard',…

MATLAB Demo

Modeling and Simulation of a transceiver system, showc
Coding Gain:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding



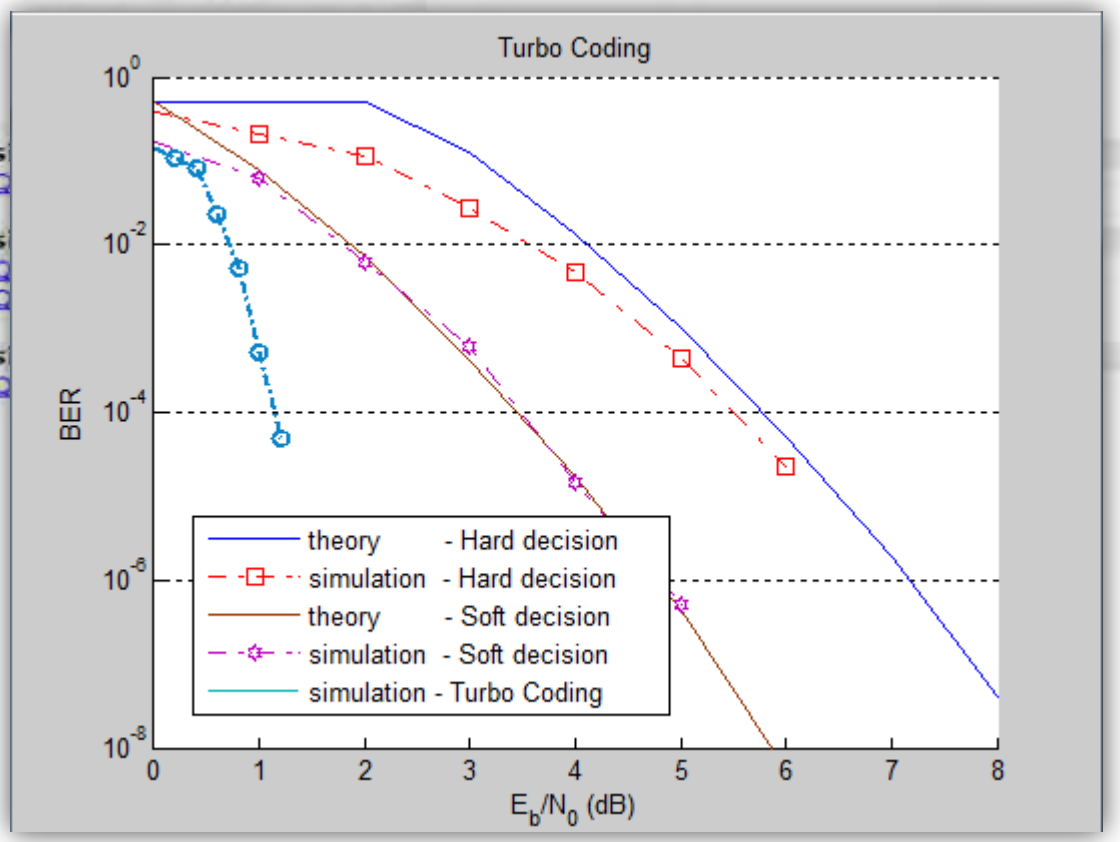

```
function [ber bits]=fcn_04PSK_Viterbi_Soft(EbNo,MaxNumErrs,MaxNumBits)
%% Initialization
persistent hModulator hAWGN hDeModulator hBitError
persistent hConvEncoder hViterbi hQuantizer
if isempty(hModulator)
    hModulator = comm.PSKModulator(...
        'ModulationOrder',4, ...
        'PhaseOffset',0, ...
        'BitInput',true);
    hAWGN = comm.AWGNChannel(...
    hDe
        'VarianceSource', 'Input port');
    hBitError = comm.ErrorRate;
    hQuantizer=dsp.ScalarQuantizerEncoder(...
        'Partitioning',  'Unbounded',...
        'BoundaryPoints', QuantizerBoundaries,...
    hConv
    hVite
        'SoftInputWordLength', 4,...
        'OutputDataType', 'double',...
        'TerminationMethod','Terminated');
end
```

= comm.QPSKDemodulator(
'DecisionMethod','Log-Likelihood ratio'

= comm.ViterbiDecoder(…
'InputFormat', 'Soft',…

MATLAB Demo

Modeling and Simulation of a transceiver system, showcasi
Coding Gain:
- Convolutional Encoding & Viterbi decoding
- Hard and Soft Decision decoding
- Turbo Encoding & Decoding




```
function [ber bits]=fcn_04PSK_zTurbo_Soft(EbNo,MaxNumErrs,MaxNumBits)
%% Initialization
persistent hModulator hAWGN hDeModulator hBitError
persistent hTurboEncoder hTurboDecoder
if isempty(hModulator)
    hModulator = comm.QPSKModulator('PhaseOffset',0,'BitInput',true);
    hAWGN = comm.AWGNChannel(...
        'NoiseMethod','Variance',
    hTurboEncoder=comm.TurboEncoder(...
        'TrellisStructure',Trellis,
    h
        'NumIterations',6);
end
```

= comm.TurboEncoder

= comm.QPSKDemodulator(
'DecisionMethod','Log-Likelihood ratio'

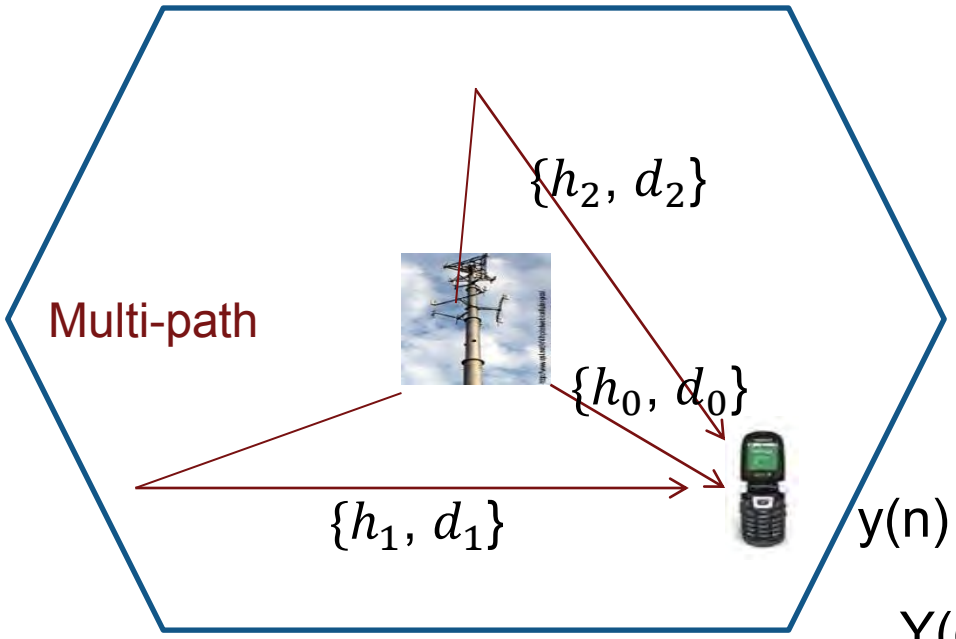
= comm.TurboDecoder(…
'NumIterations', 6,…

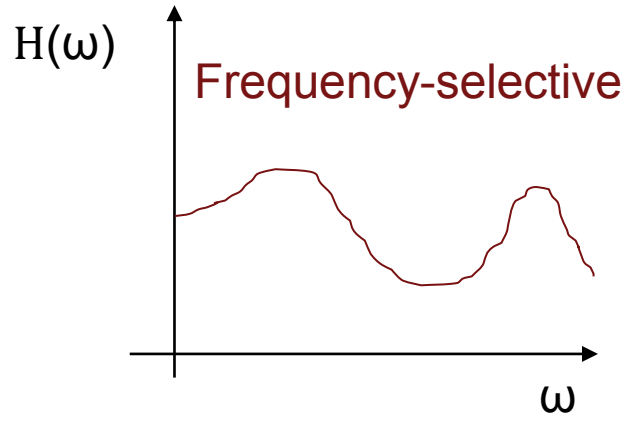
OFDM Overview

- Orthogonal Frequency Division Multiplexing
  - Multicarrier modulation scheme (FFT-based)
- Sample the spectrum at uniform intervals called sub-carriers
  - Transmit data independently at each sub-carrier
- Most important feature
  - Robust against multi-path fading
  - Using low-complexity frequency-domain equalizers

OFDM & Multi-path Fading

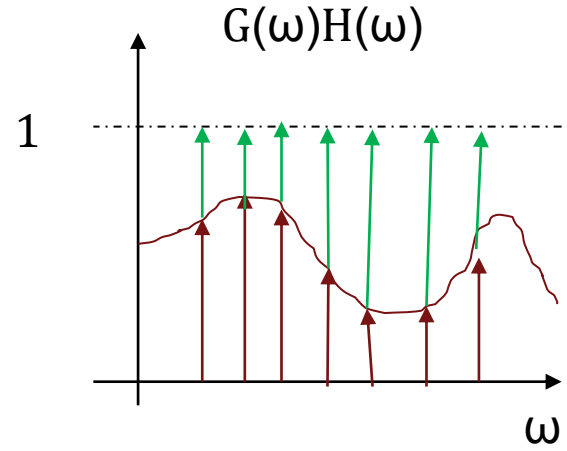
- Multi-path propagation leads to frequency selective fading
- Frequency-domain equalization is less complex and perfectly matches OFDM
- We need to know channel response at each sub-carrier – We need pilots





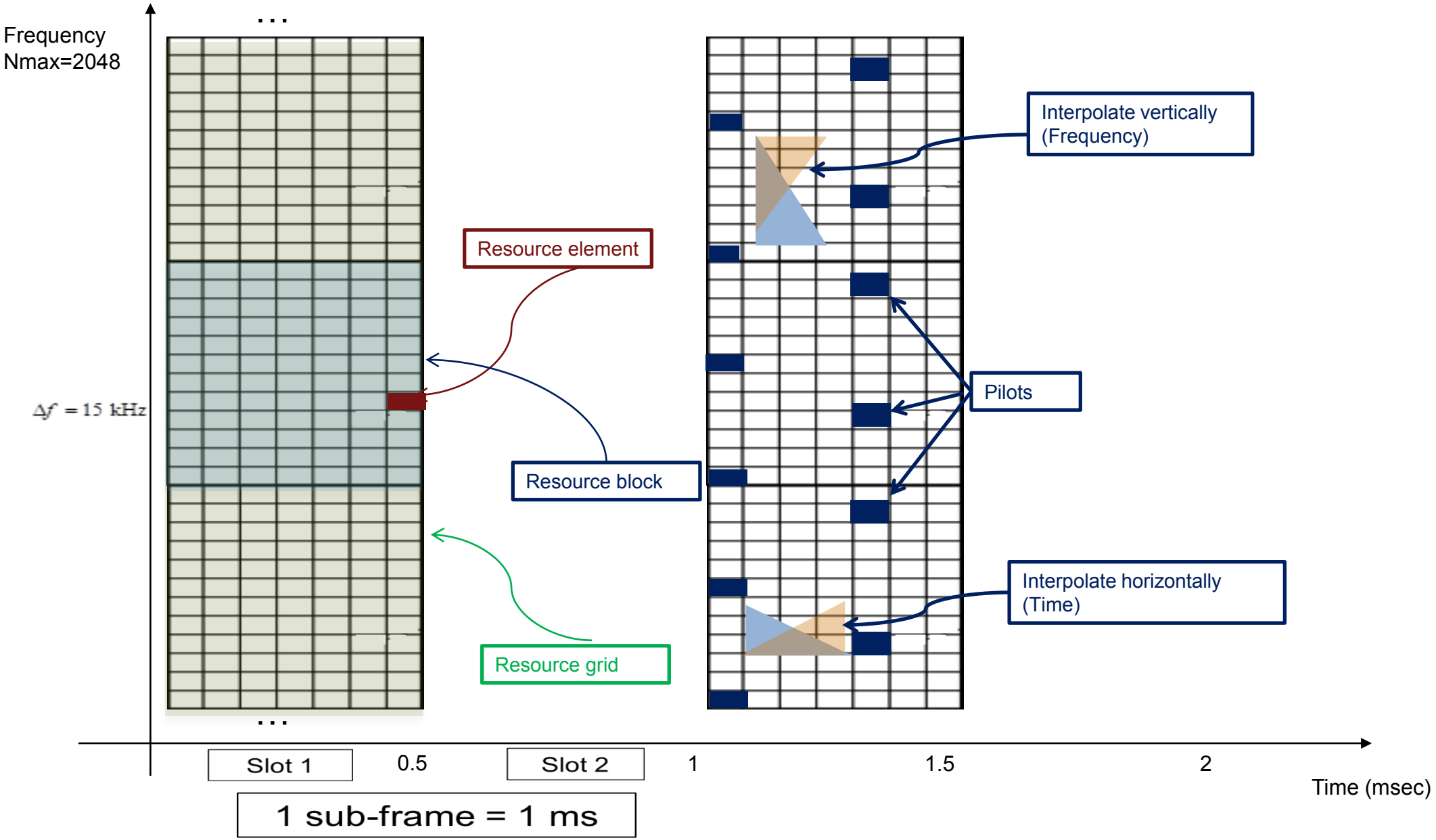
$$y(n) = \sum_{n=0}^{N} h_n \, x(n - d_n)$$

$$Y(\omega) = H(\omega)X(\omega)$$



Frequency-domain equalization $\quad \text{If } G(\omega_k) \approx H^{-1}(\omega_k) \quad G(\omega_k)\, Y(\omega_k) \approx X(\omega_k)$

How Does LTE Implement OFDM?



How to Implement LTE OFDM in MATLAB

```
function y = OFDMTx(in, prmLTEPDSCH)
% OFDM Transmitter includes:
%   Pack data with guards, add DC, reorder
%   IFFT and gain scale - per symbol per antenna port
%   Add cylic prefix - different length across symbols in a slot
%   Serialize subframe per antenna port
[len, numSymb, numLayers] = size(in);

switch prmLTEPDSCH.Nrb
        N = 128;
        cpLen0 = 10; cpLenR = 9;

Case 25, N=512;
    case 25
        N = 512;
        cpLen0 = 40; cpLenR = 36;
    case 50
        N = 1024;

Case 100, N=2048;
    case 100
        N = 2048;
        cpLen0 = 160; cpLenR = 144;
end
```

Depending on Channel Bandwidth

Set Frequency-domain FFT size

```
% IFFT processing
X=ifft(tmp,N,1);
```

Create OFDM signal

```
%% Determine Pilot indices in subframe - per antenna port
% Assign Pilot Signals signals to the grid
csrIdx = zeros(Nrb*2, 2, 2, numLayers);
%   Get the common first antenna port sequence indexes
vsh = mod(NcellID, 6);
numRb = (0:2*Nrb-1).';

= 6*numRb + mod(v+vsh, 6) + 1;
lexes
end
```

Transmitter:
Place pilots in regular intervals

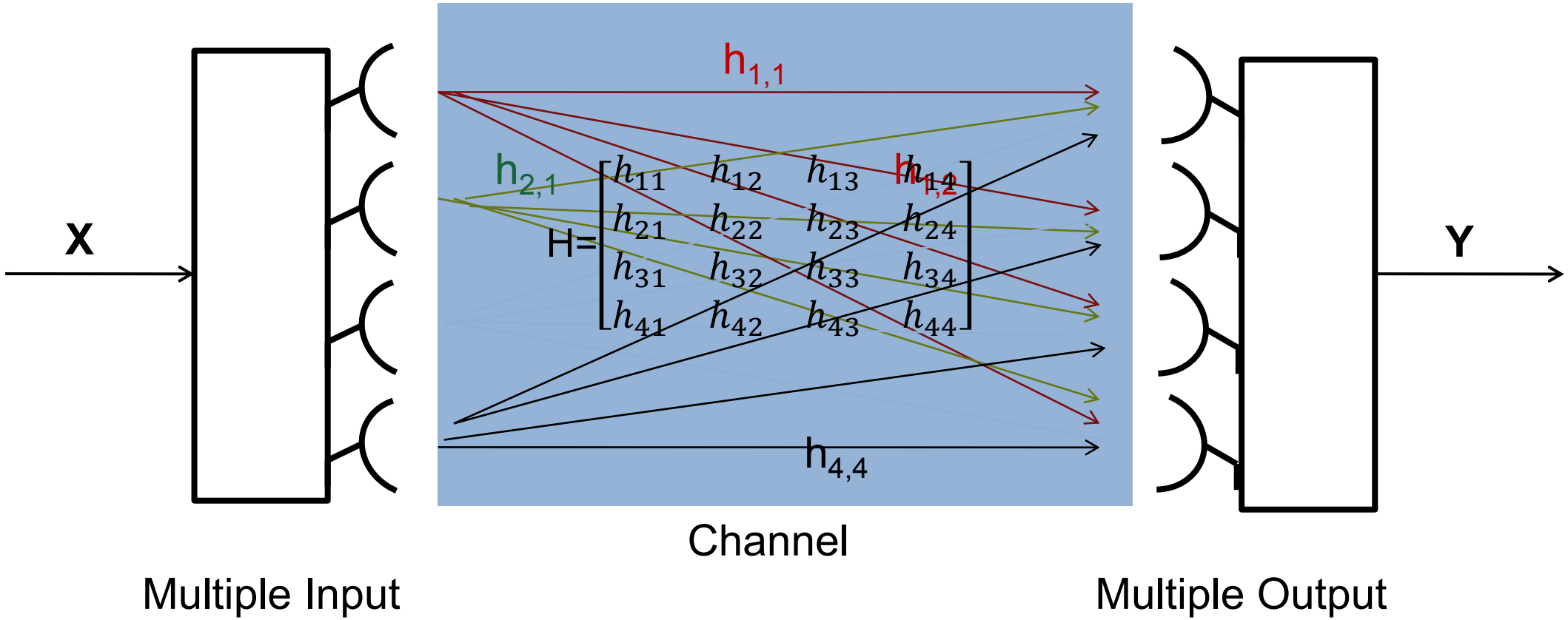
```
% Time average two pilots over the slots, then interpolate (F)
% between the 4 averaged values, repeat for all symbols in sframe

= mean([hp(:,1,1,n) hp(:,3,1,n)])
h1_a1 = mean([hp(:, 1, 1, j) hp(:, 3, 1, j)],2);
h1_a2 = mean([hp(:, 2, 1, j) hp(:, 4, 1, j)],2);

= mean([hp(k,2,1,:) hp(k,4,1,:)])
```

Receiver:
Estimate channel by interpolating in time & frequency

MIMO Overview

- Multiple Input Multiple Output
- Using multiple transmit and receive antennas

$$Y = H*X + n$$



Where is MIMO being used?

- Several wireless standards
  - 802.11n: MIMO extensions of WiFi as of 2008
  - 802.16e: As of 2005 in WiMax Standard
  - 3G Cellular: 3GPP Release 6 specifies transmit diversity mode
  - 4G LTE
- Two main types of MIMO
  - Spatial multiplexing
  - Space-Time Block Coding (STBC)

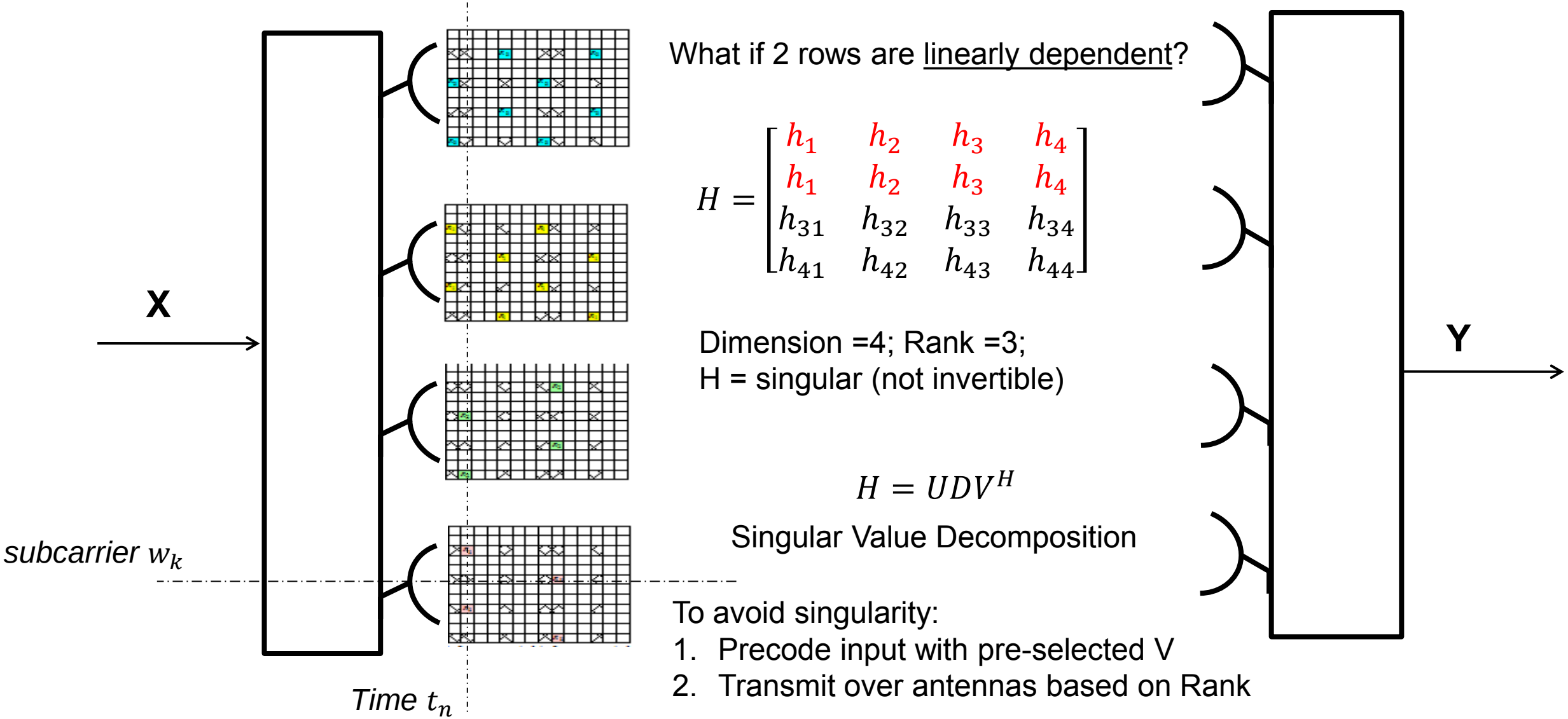
Spatial Multiplexing

- MIMO technique used in LTE standard
- Divide the data stream into independent sub-streams and use multiple transmit antennas
- MIMO is one of the main reasons for boost in data rates
  - More transmit antennas leads to higher capacity
- MIMO Receiver essentially solves this system of linear equations

$$Y = HX + n$$

MIMO-OFDM overview

$$Y = H * X + n$$



What if 2 rows are linearly dependent?

$$H = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \\ h_1 & h_2 & h_3 & h_4 \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

Dimension =4; Rank =3;
H = singular (not invertible)

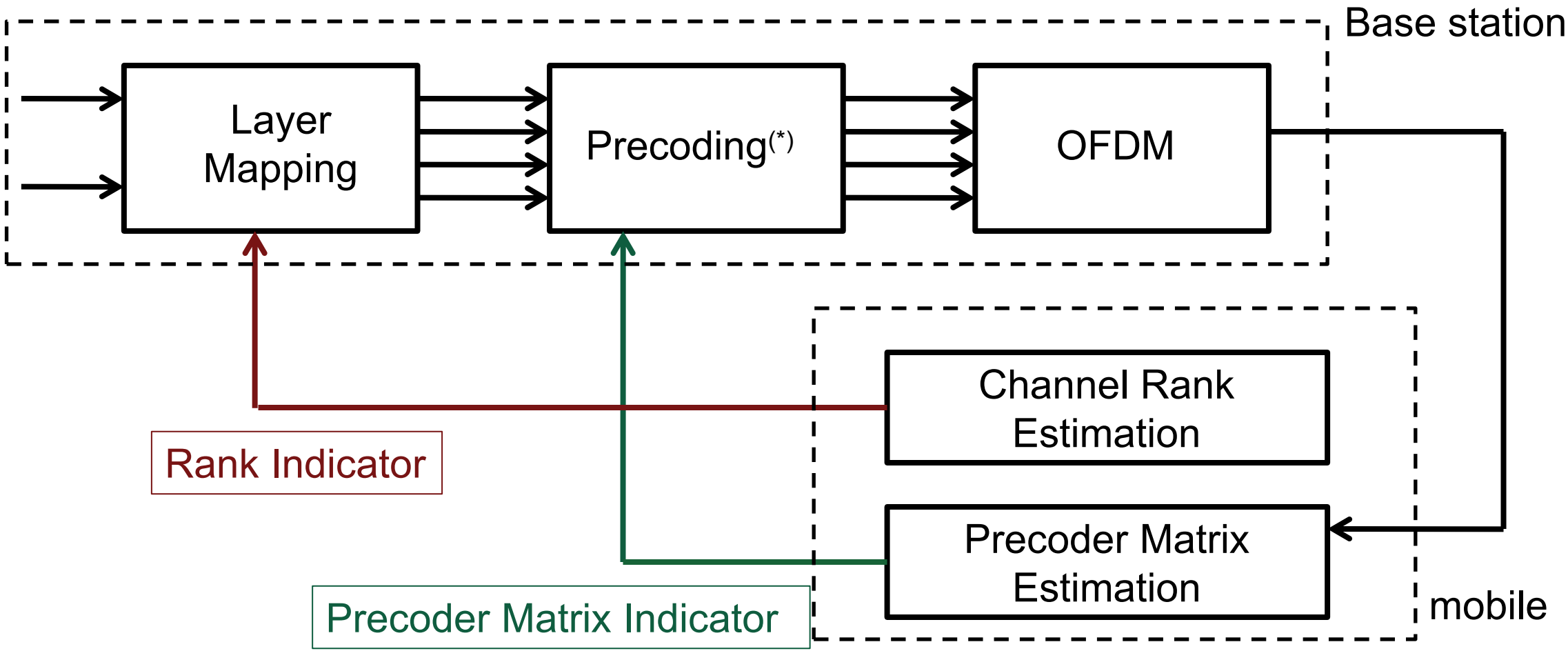
$$H = UDV^H$$

Singular Value Decomposition

To avoid singularity:

1. Precode input with pre-selected V
2. Transmit over antennas based on Rank

Adaptive MIMO: Closed-loop Pre-coding and Layer Mapping



Adaptive MIMO in MATLAB

- In Receiver:
- Detect V = Rank of the H Matrix
- = Number of layers

- In Transmitter: (next frame)
- Based on number of layers
- Fill up transmit antennas with available rank

```
1   function out = LayerMapper(in1, in2, prmLTEPDSCH)
2   % LTE Layer mapper for spatial multiplexing.
3   %   Assumes two codeword input for spatial multiplexing.
4   %   As per TS 36.211 v10.0.0, Section 6.3.3.2.
5
6   %#codegen
7
8   % Assumes the incoming codewords are of the same length for now
```

V= prmLTEPDSCH.numLayers;

```
13  inLen2 = size(in2, 1);
14  switch q
15     case 1  % Single codeword
16        % for numLayers = 1,2,3,4
```

Switch V

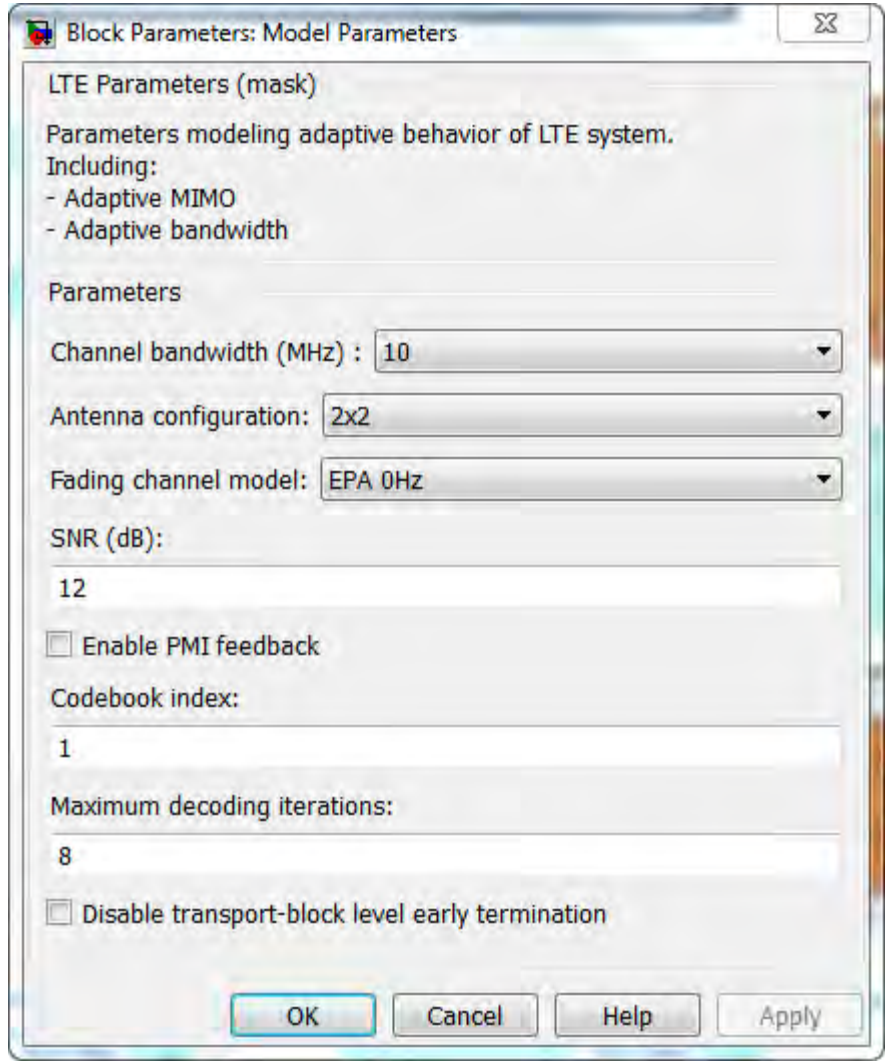
```
20          switch v
21             case 2
22                out = complex(zeros(inLen1, v));
23                out(:,1) = in1;
24                out(:,2) = in2;
25             case 3 % => different length input codewords
26                assert(false, '3 layers for 2 codewords is not implemented yet');
27             case 4
```

```
case 4
    out = complex(zeros(inLen1/2, v));
    out(:,1:2) = reshape(in1, 2, inLen1/2).';
    out(:,3:4) = reshape(in2, 2, inLen2/2).';
```

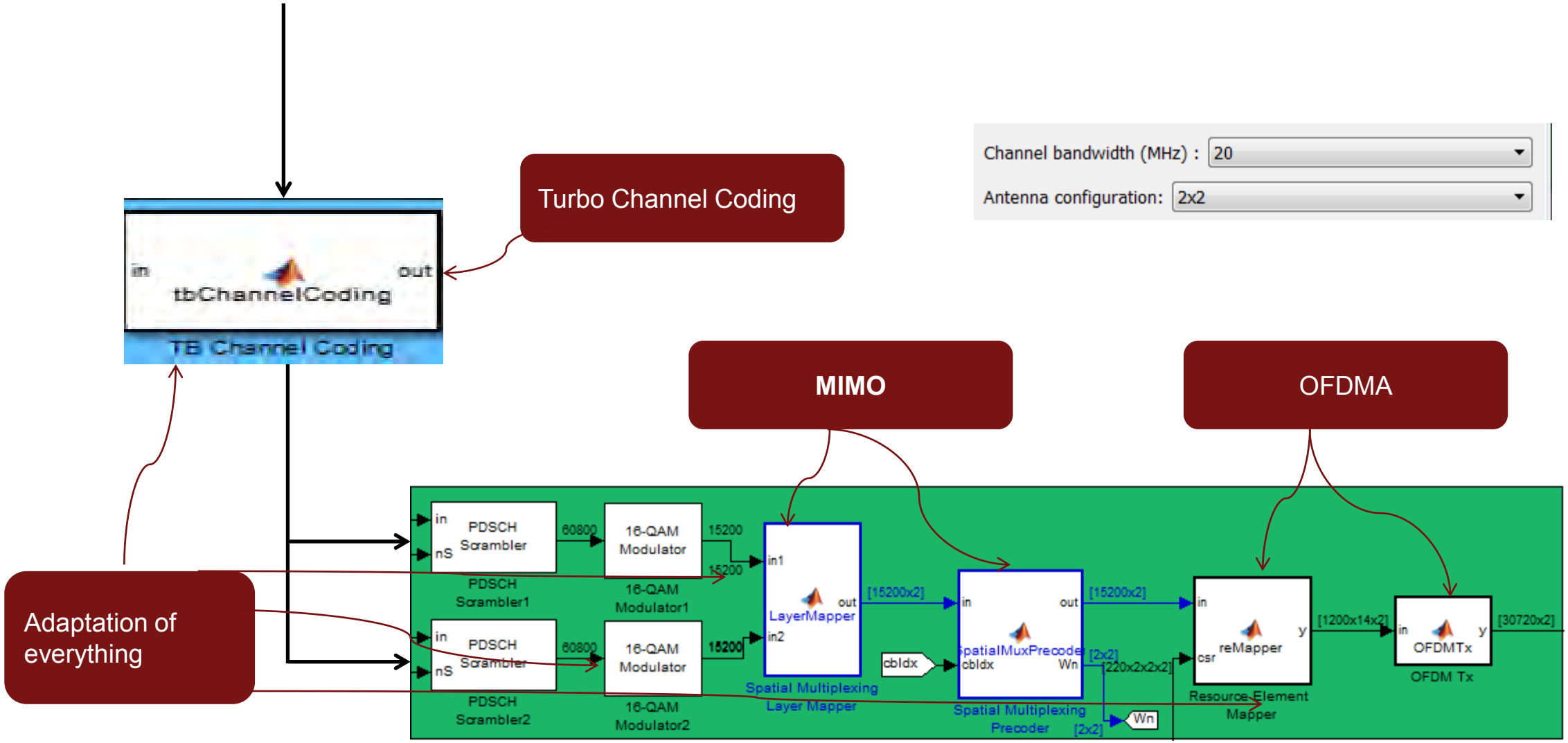
```
37             case 7 % => different length input codewords
38                assert(false, '7 layers for 2 codewords is not implemented yet');
39             case 8
40                out = complex(zeros(inLen1/4, v));
41                out(:,1:4) = reshape(in1, 4, inLen1/4).';
42                out(:,5:8) = reshape(in2, 4, inLen2/4).';
43          end
44  end
```

Link Adaptation Overview

- Examples of link adaptations
  - Adaptive modulation
    - QPSK, 16QAM, 64QAM
  - Adaptive coding
    - Coding rates from (1/13) to (12/13)
  - Adaptive MIMO
    - 2x1, 2x2, …,4x2,…, 4x4, 8x8
  - Adaptive bandwidth
    - Up to 100 MHz (LTE-A)

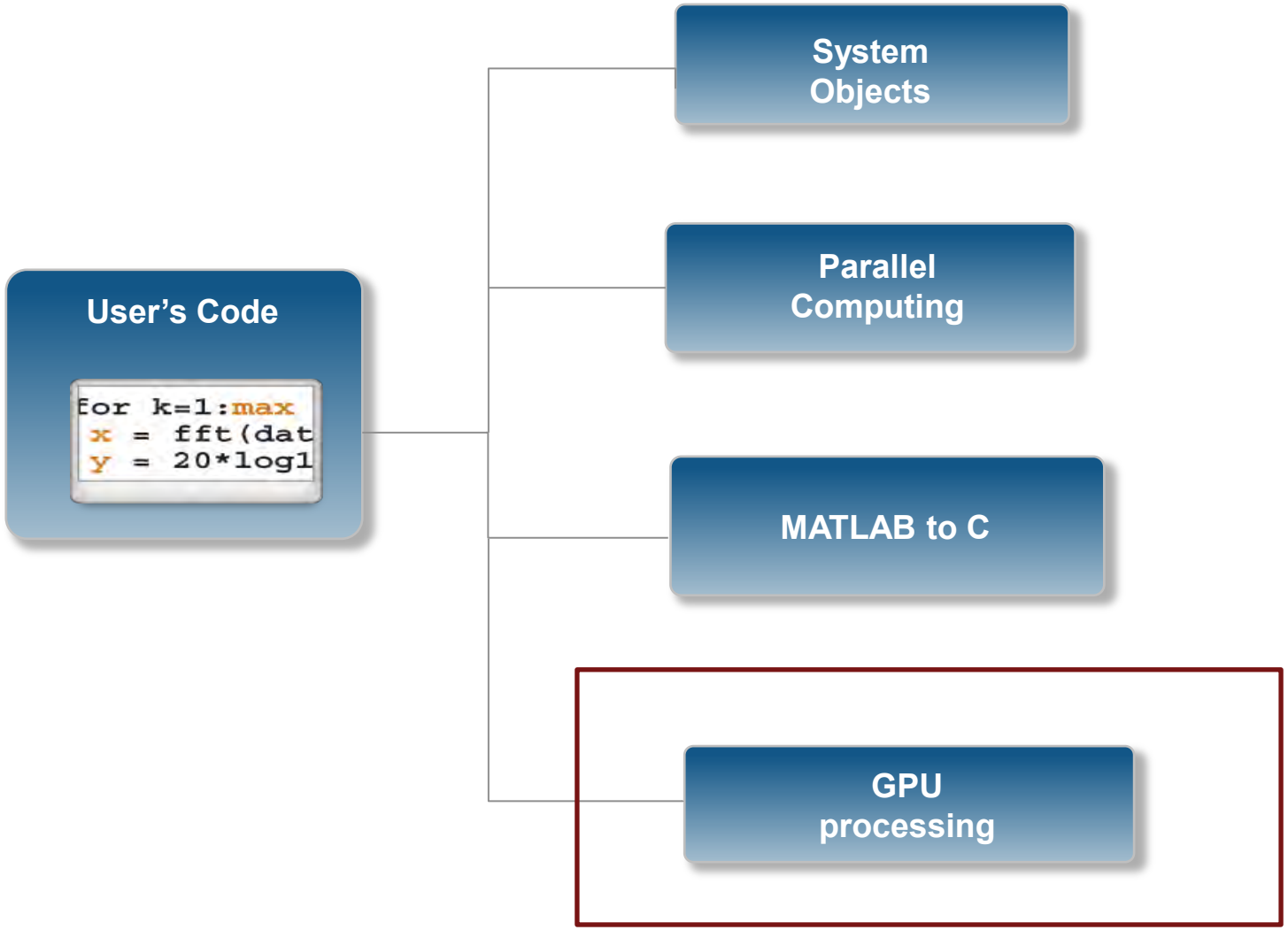


LTE Physical layer model in MATLAB



Simulation Acceleration

Simulation acceleration options in MATLAB



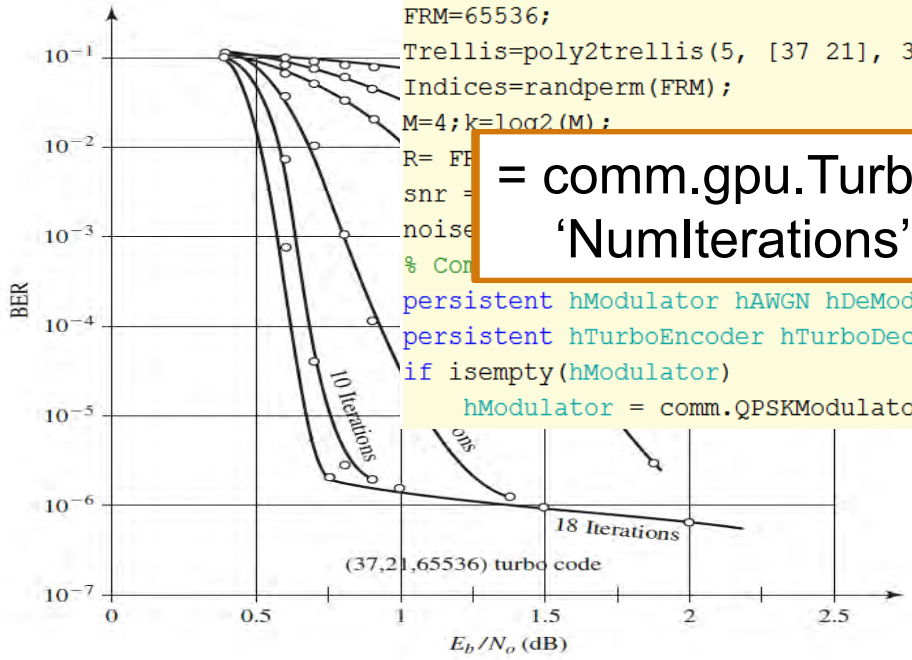
GPU Processing with Communications System Toolbox

- Alternative implementation for many System objects take advantage of GPU processing
- Use Parallel Computing Toolbox to execute many communications algorithms directly on the GPU
- Easy-to-use syntax
- Dramatically accelerate simulations

GPU System objects
comm.gpu.TurboDecoder
comm.gpu.ViterbiDecoder
comm.gpu.LDPCDecoder
comm.gpu.ViterbiDecoder
comm.gpu.PSKDemodulator
comm.gpu.AWGNChannel

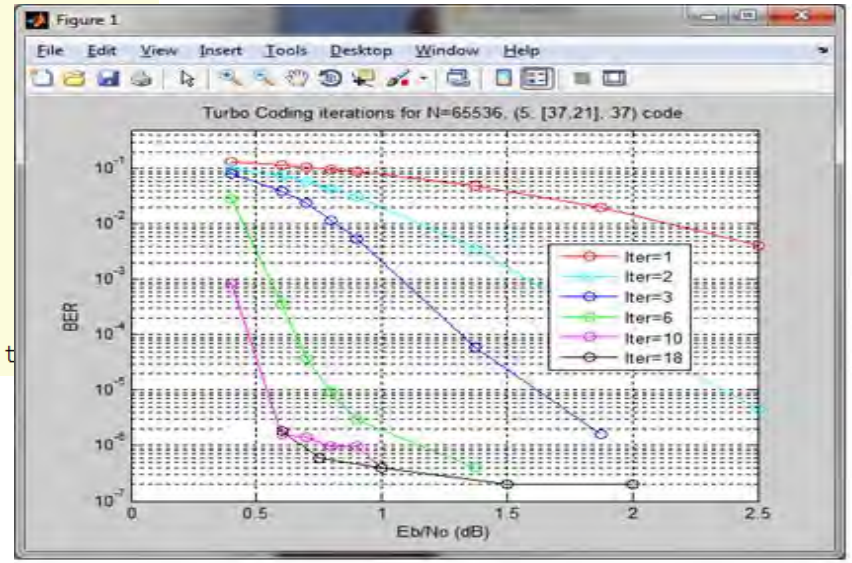
Example: Turbo Coding

- Impressive coding gain
- High computational complexity
- Bit-error rate performance as a function of number of iterations

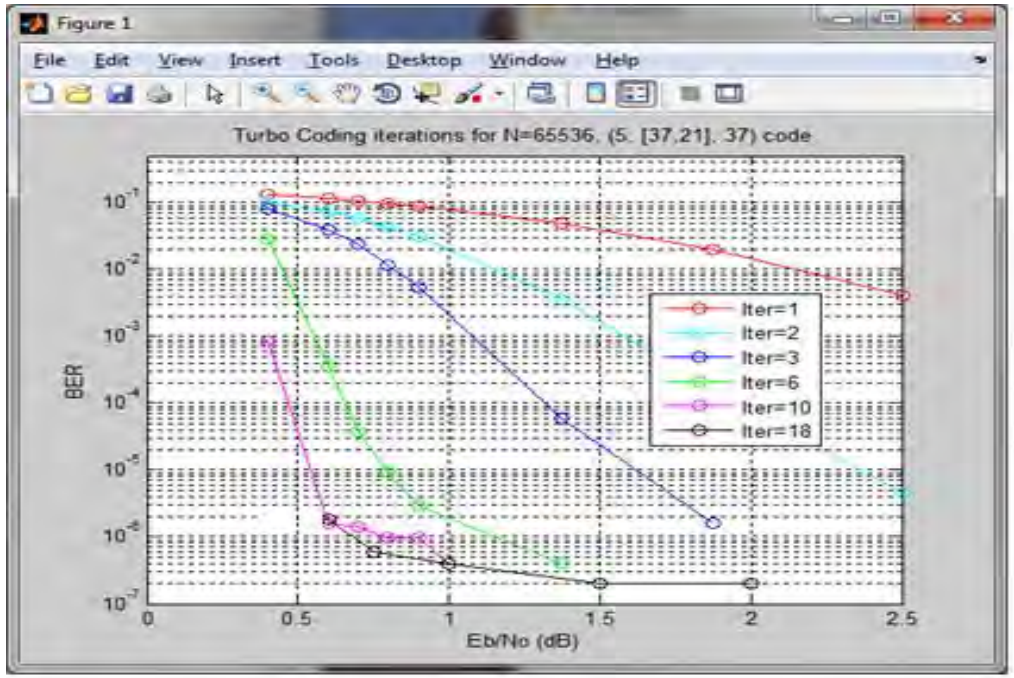



```
function [ber bits]=Fcn4_04PSK_Turbo_Soft(EbNo,MaxNumErrs,MaxNumBits)
%#codegen
% Initialization
FRM=65536;
Trellis=poly2trellis(5, [37 21], 37);
Indices=randperm(FRM);
M=4;k=log2(M);
R= FR
snr =
noise
% Con
persistent hModulator hAWGN hDeModulator hBitError
persistent hTurboEncoder hTurboDecoder
if isempty(hModulator)
    hModulator = comm.QPSKModulator('PhaseOffset',0,'BitInput',t
```

= comm.gpu.TurboDecoder(…
'NumIterations', N,…



Simulation acceleration with GPU System objects



- Same numerical results

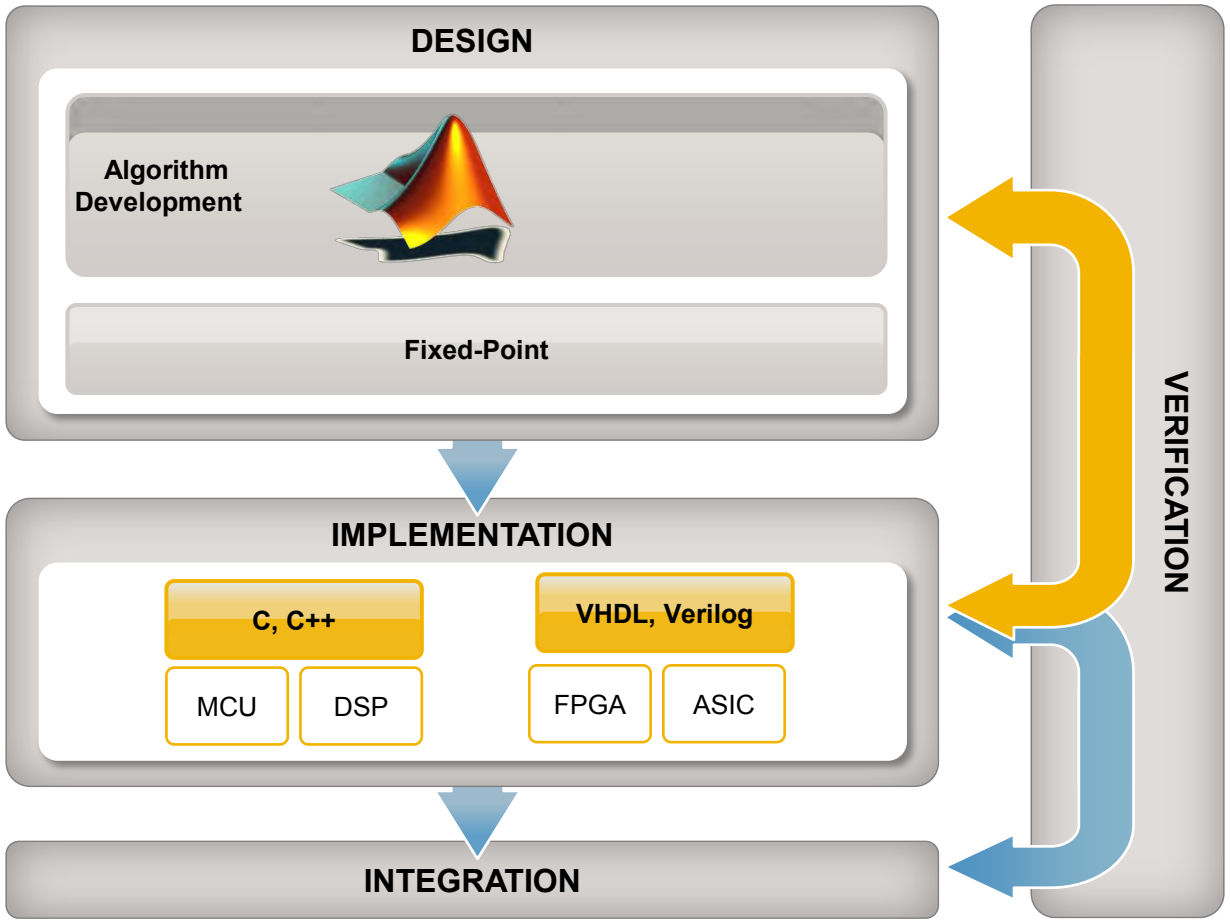
```
= comm.TurboDecoder(…
    'NumIterations', N,…
```

```
= comm.gpu.TurboDecoder(…
    'NumIterations', N,…
```

Version	Elapsed time	Acceleration
CPU	8 hours	1.0
1 GPU	40 minutes	12.0
Cluster of 4 GPUs	11 minutes	43.0

Path to implementation
C/C++ and HDL Code Generation

Path to implementation



Mathematical modeling and algorithm development

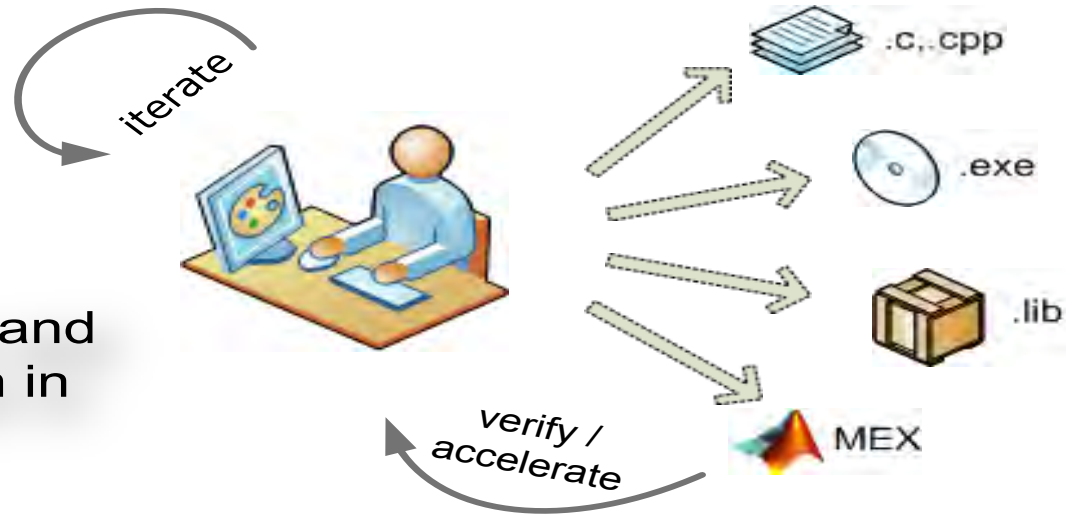
Model elaboration including fixed-point numerical representation

Automatic code generation for rapid on-target prototyping of Hardware/Software

Co-simulation for verification in Hardware / Software

Automatic Translation of MATLAB to C

Algorithm Design and Code Generation in MATLAB

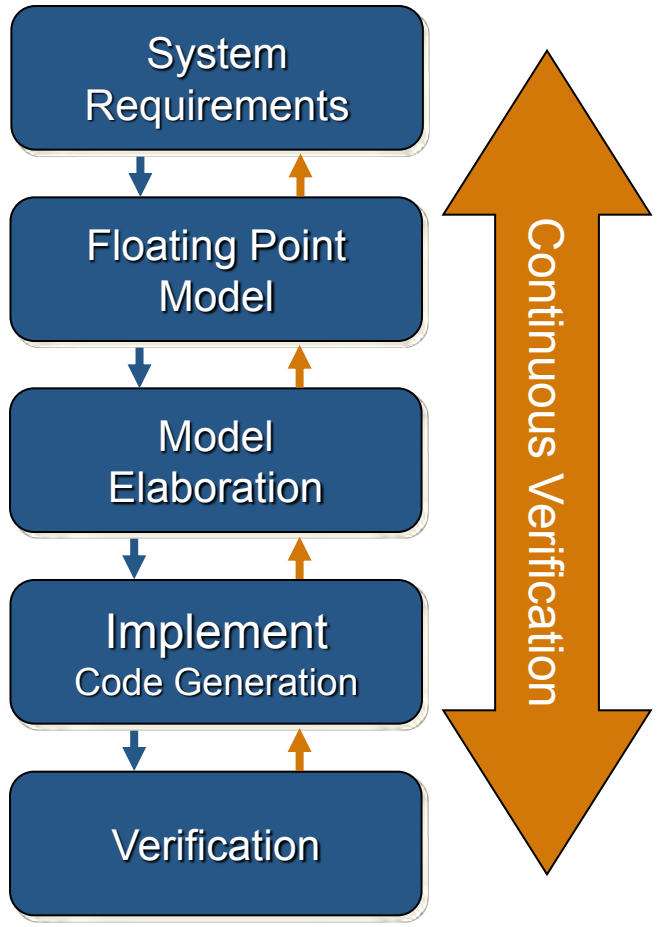


With MATLAB Coder, design engineers can

- Maintain one design in MATLAB
- Design faster and get to C/C++ quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB

HDL Workflow

- Floating Point Model
  - Satisfies System Requirements
    - Executable Specification
  - MATLAB and/or Simulink Model
- Model Elaboration
  - Develop Hardware Friendly Architecture
  - Convert to Fixed-Point
    - Determine Word Length
    - Determine Binary Point Location
- Implement Design
  - Generate HDL code using HDL Coder
  - Import Custom and Vendor IP
- Verification
  - Software co-simulation with HDL simulator
  - Hardware co-simulation



Key Points

- MATLAB is an ideal language for LTE modeling and simulation
- Communications System Toolbox extends MATLAB capabilities with algorithms for communications system design
- You can accelerate simulation with a variety of options in MATLAB
  - Parallel computing, GPU processing, MATLAB to C
- Address implementation workflow gaps with
  - Automatic MATLAB to C/C++ and HDL code generation
  - Hardware-in-the-loop verification