OFDM and FFT

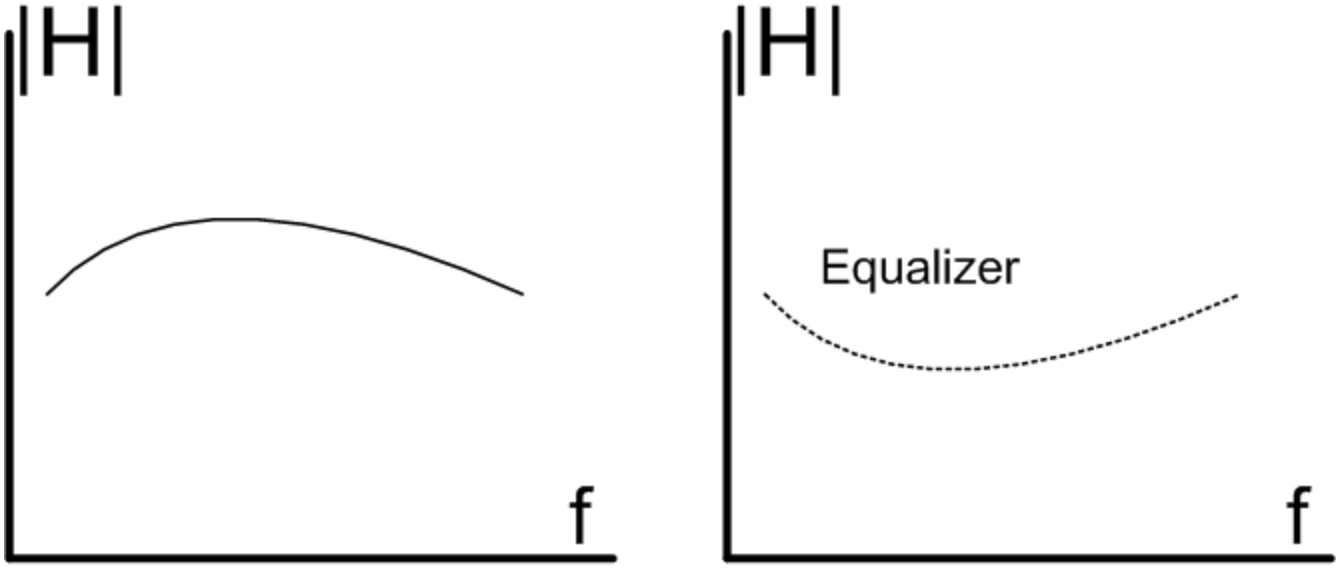
Cairo University Faculty of Engineering
Department of Electronics and Electrical Communications
Dr. Karim Ossama Abbas
Fall 2010

Contents

- OFDM and wideband communication in time and frequency domain
- Implementation using FFT

Frequency selective wireless channel

- A wireless channel with multi-path is frequency selective
- This frequency selectivity must be corrected at the receiver
- Correction is through equalization, or inversion of the channel

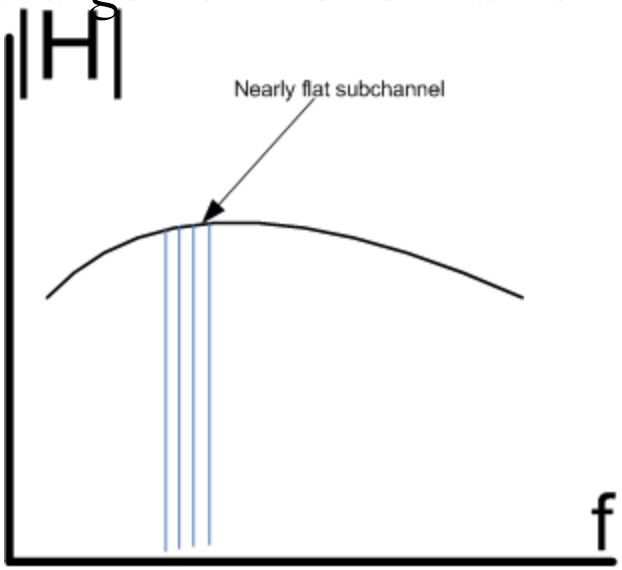


Naïve equalization

- The simplest and most direct form of equalization is to use an FIR filter
- The number of taps in the filter grows quadratically with the bandwidth of the channel
- The FIR equalizer is thus too complex to implement if the channel is too selective or too wideband

OFDM and dividing the channel

- In OFDM "Orthogonal Frequency Division Multiplexing" the channel is divided into multiple narrowband subchannels
- If the bandwidth of the subchannels is narrow enough, they will each be effectively flat
- Thus the task of equalizing one wideband channel is changed into equalizing a large number of narrowband channels

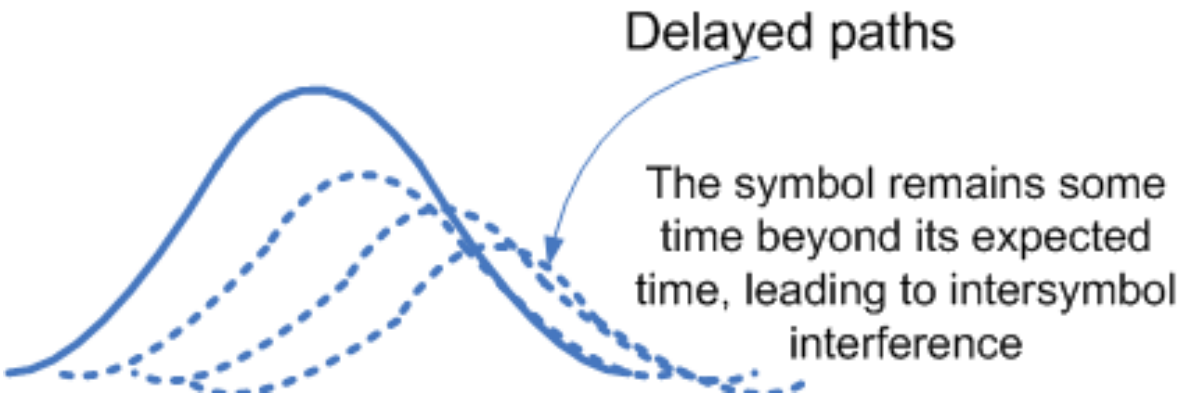


Questions about OFDM

- With so many narrow channels next to each other, how can you be sure they are not interfering with each other?
- How is this implemented? How do you combine so many subcarriers, do you have to use as many RF chains as there are subcarriers?

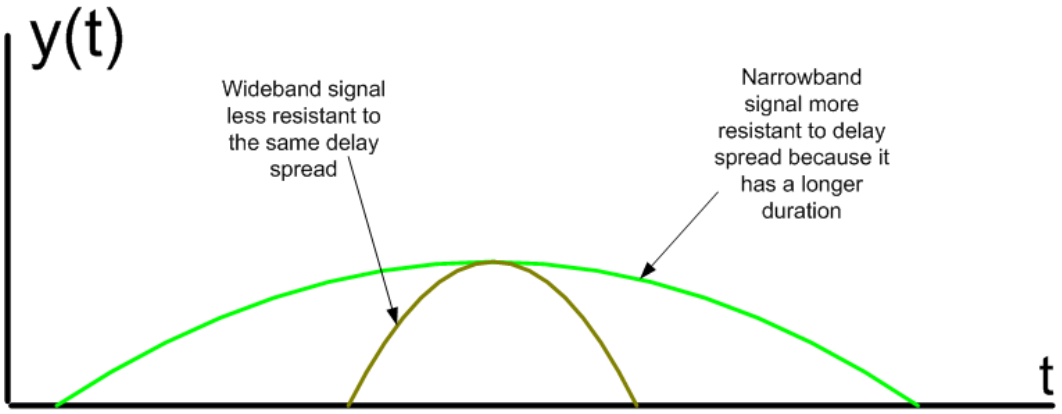
The time domain equivalent problem

- The problem of a frequency selective channel has a time-spreading analog in time-domain
- Thus frequency selectivity is equal to spreading of the time-domain signal waveforms
- This can lead to ISI



The time-domain solution

- OFDM divides a wideband channel into parallel narrowband ones
- What this means is that the symbol waveform will be wider than the original waveform (by a factor of the total bandwidth to subchannel bandwidth)
- Thus the effect of the same time-shift on the signal is much smaller (and can be effectively removed)

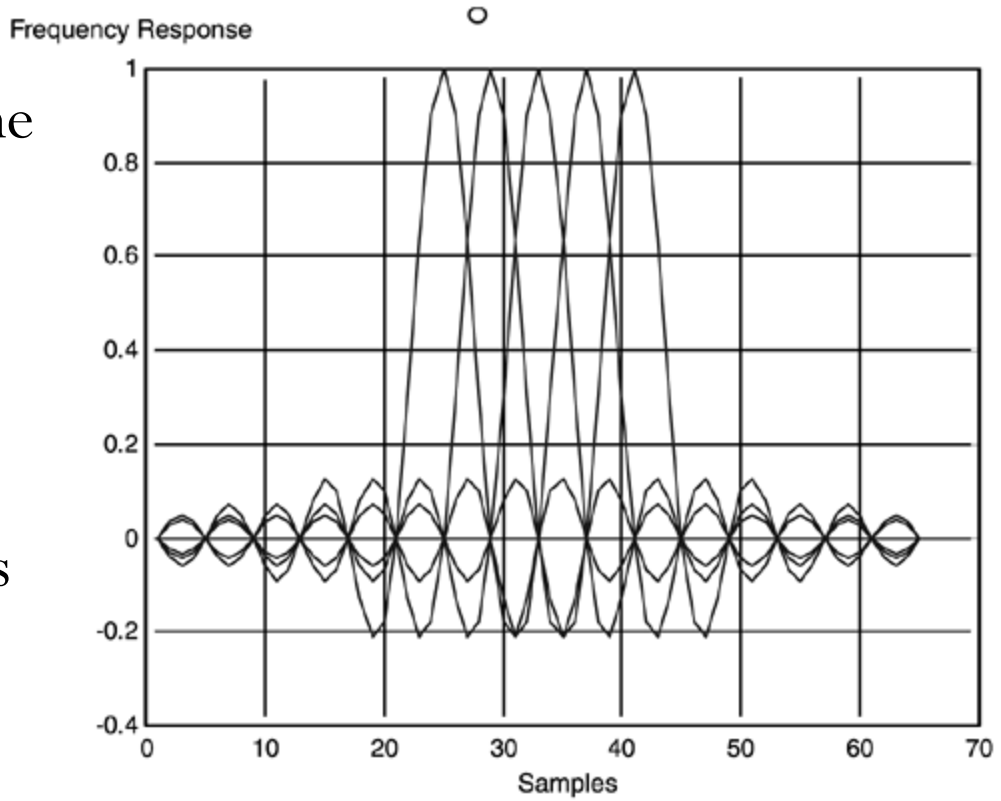


Subcarrier orthogonality

- The subcarriers of an OFDM system are chosen to be orthogonal
- What this means is that the waveforms have zero crossings at each other's maxima
- This leads to zero intercarrier interference!

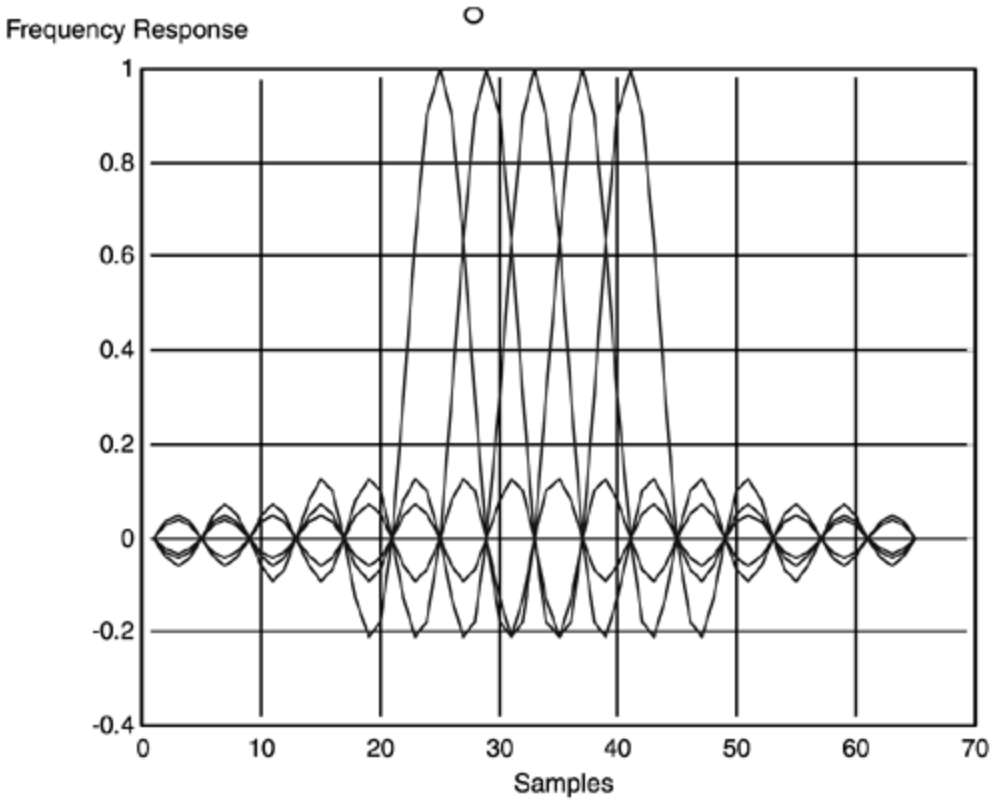
Condition for subcarrier orthogonality

- Each subcarrier with information is a sinc with the first zero crossing at 1/T (symbol duration)
- Thus the minimum spacing between subcarriers is 1/T
- Thus the carrier frequencies are:
- $f_i = f_c + i/T \quad i = 0, 1, \ldots, N-1$



Orthogonal subcarriers

- On each subcarrier, a baseband modulated data stream is sent
- The streams are sent in parallel
- Another way to look at orthogonality is that any two subcarriers form an orthogonal sinusoid pair over a symbol period



Choosing subchannel bandwidth

- How many subcarriers should we have? Depends on:
  - Channel bandwidth
  - Channel selectivity
- Now assume that Bc (coherence bandwidth)=200kHz, how many subcarriers will you use in 20MHz?
  - 5*20e6/(200e3)=500 -> 512
- Thus we assume the channel is flat only over Bc/5, this is a rule of thumb

Fighting ISI

- Now assume we have a delay spread of 10μsec, how long as a ratio of the symbol does a guard interval need to be to remove ISI in:
  - Single carrier 20MHz
  - OFDM 20MHz with 512 subcarriers
- In single carrier -> Symbol time = 1/20e6=50nsec, guard interval is 2000% the symbol duration
- In OFDM the ratio is 2000/512=3.9%
- Thus using a guard interval to combat ISI is reasonable in OFDM

What is in the guard interval?

- The guard interval contains a repetition of the beginning of the OFDM symbol, called a cyclic prefix
- This allows a linear convolution to be equivalent to a circular convolution, the latter is much simpler in DFT

Implementation

- One question remaining is if we need as many RF chains as there are subcarriers
- The answer is of course not, examine what we do a the transmitter:
  - We encode and baseband modulate on each subcarrier
  - Then we have all the subcarriers in frequency domain
  - How do you take that and transform it into one time-domain signal?

Taking the subcarriers from frequency domain to time domain

- What takes a sequence from f-domain to t-domain?
  - Inverse Fourier Transform
  - Inverse Discrete Time Fourier Transform (IDTFT) for discrete time sampled systems
  - Inverse Discrete Fourier Transform (IDFT) deals with finite samples both in time and frequency)
  - Inverse Fast Fourier Transform (IFFT) is an efficient implementation of IDFT
- Thus OFDM and FFT/IFFT always come together

Complexity of implementation

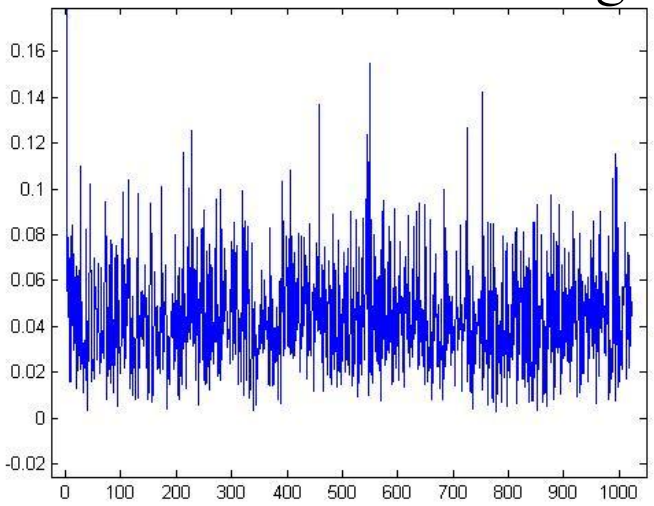
- At the receiver, each subcarrier will need to be inverted, this is equivalent to one complex multiplication
- We also need to perform the FFT, this grows in a semilog fashion
- The overall complexity thus grows nearly linearly
- For N-subcarriers the complexity is:
- N+Nlog2(N) complex multiplications
  - N from equalization
  - NlogN from the FFT

Advantages of OFDM

- Simple implementation using DSP
  - FFT allows much easier equalization than a straight filter
- Ability to deal with bad wideband channels
  - Allows better spectral efficiency and better overall throughput
- Combats delay spread
  - The time-domain counterpart of frequency selectivity
- Insensitive to symbol timing errors
  - Due to the guard interval it doesn't matter if the receiver sampling is a little off

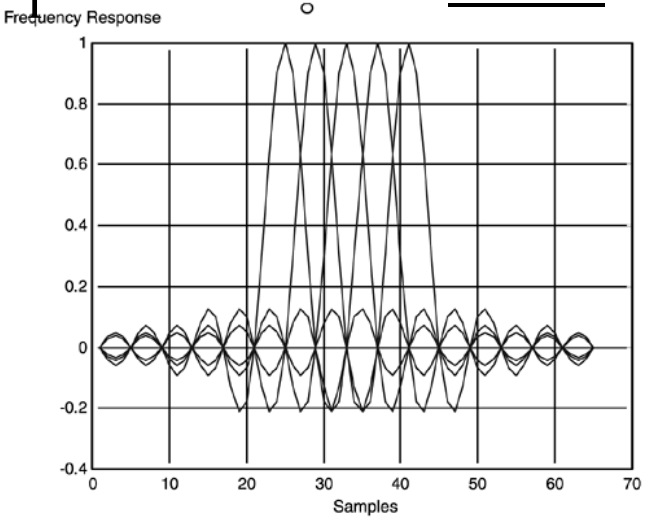
Weak points of OFDM - PAPR

- OFDM has very bad Peak to average power ratio (PAPR)
- PAPR is defined as: Peak power in the time domain symbol / Average power of the symbol
- OFDM looks rather white in frequency domain due to presence of closely packed subcarriers
- In time-domain this leads to noise-like waveforms with high peaks relative to the mean

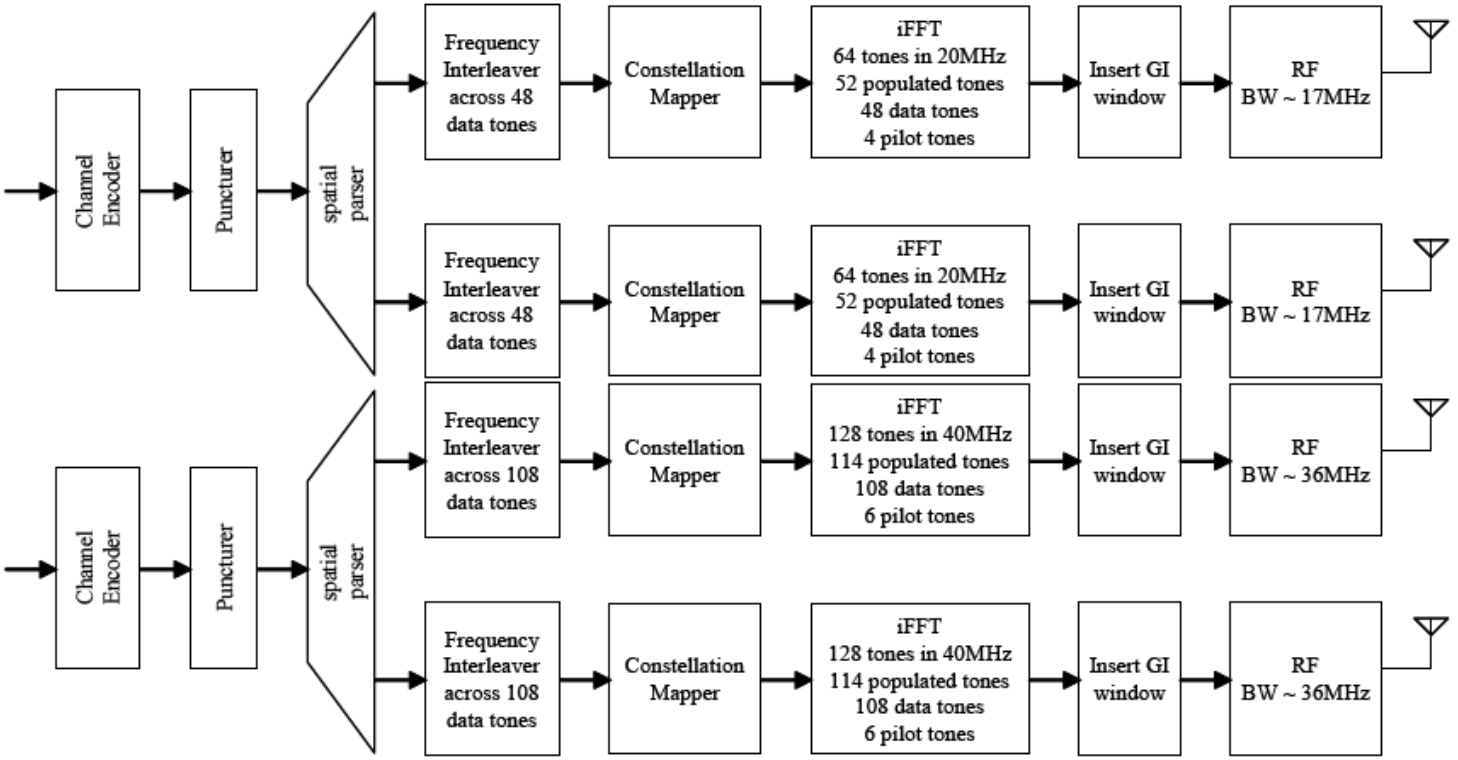


Sensitivity to frequency offsets

- OFDM's greatest strength is its immunity to timing problems
- However, it is extremely sensitive to offsets in frequency and phase between the transmitter and the receiver
- If the receiver makes an error about the center frequency, even if small, then the FFT sees all subcarrier frequencies offset
- Any small offset in the subcarrier frequencies leads to loss of orthogonality
- This is a major issue leading to severe loss in performance



Review of the 802.11n transceiver



- At the transmitter: Encode, puncture, parse, and interleave bits -> Move to symbol domain through constellation mapper -> Move to time domain by IFFT -> insert GI and send
- At the receiver: FFT to go to f-domain -> invert channel -> demap -> depuncture and deinterleave -> channel decode (recovery and synch functions additional)

DFT

The Fourier Transform takes a signal from time domain to frequency domain

Both signals are continuous and infinite in width

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi jft} dt$$

The discrete-time Fourier transform converts a discrete time signal into a continuous frequency domain.

Note that both signals are still infinitely wide

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$

The Discrete Fourier Transform (DFT) converts a finite discrete time signal into a finite discrete frequency domain. This is done by sampling frequency on the unit circle. This is very suitable for DSP since the signals are discretised and finite length

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} k = 0,1,\ldots,N-1$$

DFT complexity

- In DFT we compute N frequency domain points
- Each frequency point is calculated by performing N complex multiplications then N complex additions
- Thus we need N^2 complex multiplies and N^2 complex adds

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} \quad k = 0,1,\ldots,N-1$$

FFT

- FFT is mathematically identical to DFT in floating point
- The FFT, however, is faster
- The complexity of FFT is O(NlogN) instead of O(N^2)
- This saving can vary between FFT algorithms and a solid bound is very hard to define, but the log limit is good

Cooley-Tukey

- The Cooley-Tukey algorithm is the most commonly used FFT algorithm
- Cooley-Tukey uses a divide and conquer approach towards the FFT, breaking it down into successively smaller FFT's
- We are going to consider the radix-2 algorithm, but higher radix algorithms are also commonly used in practice (particularly radix 4)

FFT decimation in time – Step 1

Begin with normal FFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} k = 0,1,\ldots,N-1$$

Now divide the N-point FFT into two N/2 point FFTs

One for the even and one for the odd indices

(summation is linear, so there is no effect)

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-2\pi jk(2m)/N} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-2\pi jk(2m+1)/N} k = 0,1,\ldots,N-1$$

FFT decimation in time – Step 1

Define the following:

$$W_N^{lk} = e^{-2\pi jkl/N}$$

The equations then become:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_N^{2mk} + \sum_{m=0}^{N/2-1} x_{2m+1} W_N^{(2m+1)k} k = 0,1,\ldots,N-1$$

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_N^{2mk} + W_N^k \sum_{m=0}^{N/2-1} x_{2m+1} W_N^{2mk} k = 0,1,\ldots,N-1$$

$$X_k = E_k + W_N^k O_k k = 0,1,\ldots,N-1$$

FFT decimation in time – Step 1

Each of E and O is length N/2, DFT is cyclic, thus we only calculate N/2 points each and calculate the remaining N/2 points from the first N/2

$$E_k = E_{k+N/2}$$

$$O_k = O_{k+N/2}$$

$$W_N^{k+N/2} = e^{-2\pi j(k+N/2)/N} = e^{-2\pi jk/N} e^{-\pi jN/N} = -W_N^k$$

$$X_k = E_k + W_N^k O_k, k = 0,1,\ldots, N/2-1$$

$$= E_{k-N/2} - W_N^k O_{k-N/2}, k = N/2,\ldots, N$$

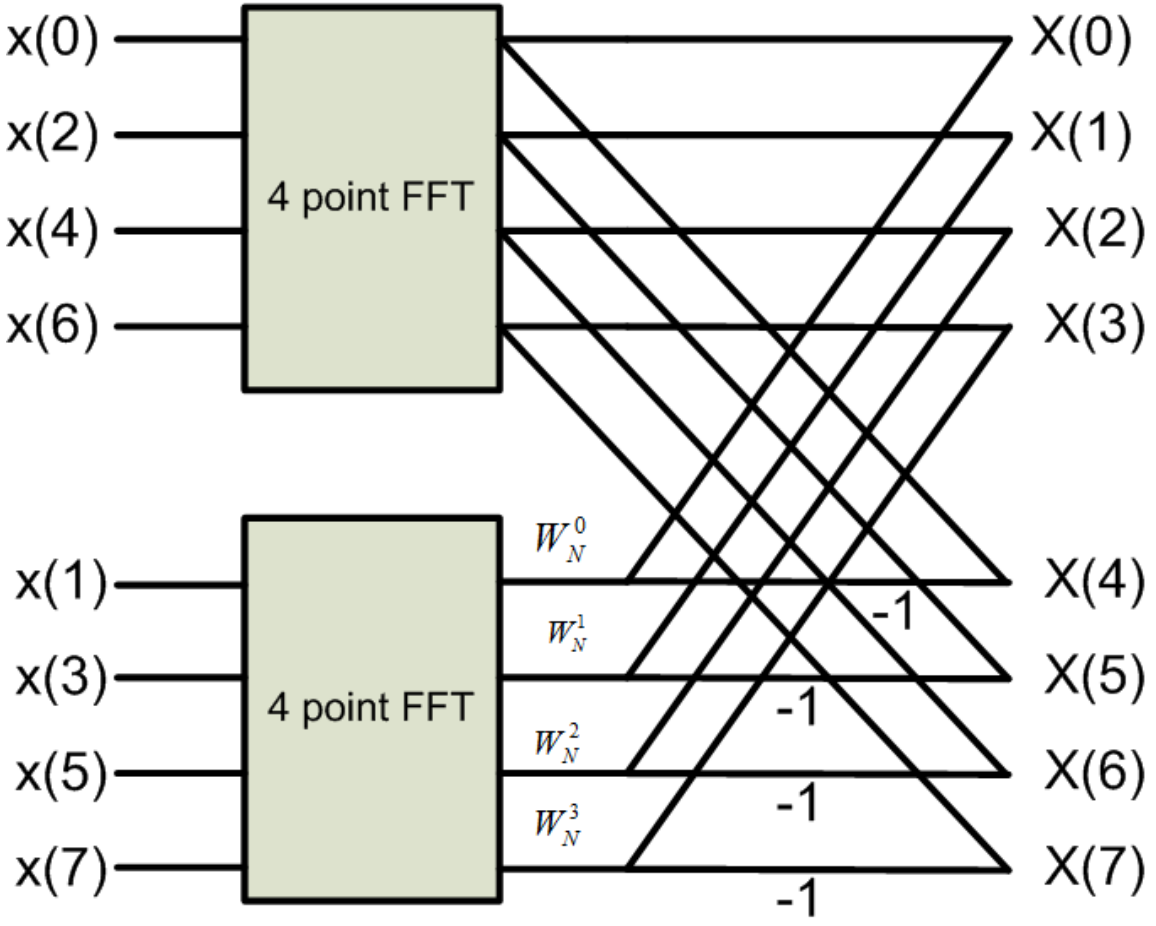
What happened to complexity so far

$$X_k = E_k + W_N^k O_k, k = 0,1,\ldots, N/2-1$$
$$= E_{k-N/2} - W_N^k O_{k-N/2}, k = N/2,\ldots, N$$

- A straight DFT requires N^2 operations
- Here we will calculate 2 sets of N/2 point DFTs then add an additional N multiplications for the twiddle factor
- Complexity becomes 2(N/2)^2+N=(N^2)/2+N
- Projecting further, the two N/2 FFTs will be further broken into N/4 FFTs and so on leading to the logarithmic complexity

Step 1 example

- Consider application to an 8-point FFT



FFT decimation in time – step 2

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_N^{2mk} + W_N^k \sum_{m=0}^{N/2-1} x_{2m+1} W_N^{2mk} \quad k = 0,1,\ldots,N-1$$

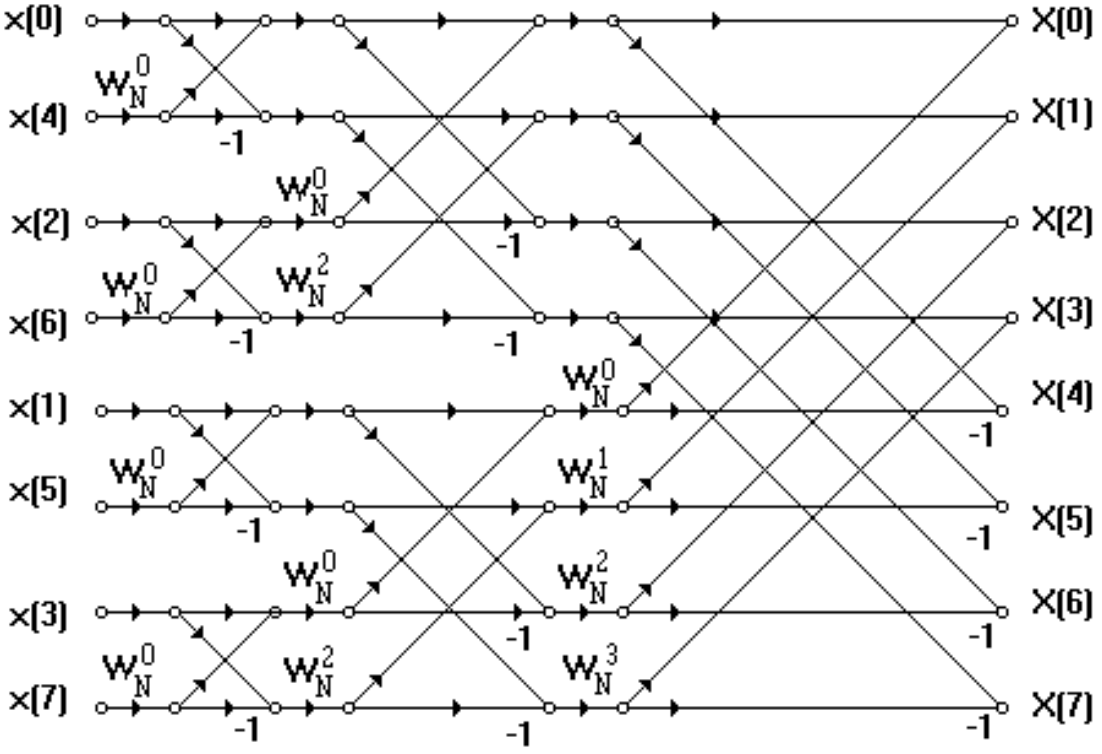
$$X_k = \sum_{m=0}^{N/4-1} x_{2(2m)} W_N^{2(2m)k} + W_N^{2k} \sum_{m=0}^{N/4-1} x_{2m+1} W_N^{2(2m)k}$$

$$+W_N^k \{ \sum_{m=0}^{N/4-1} x_{2(2m+1)} W_N^{2(2m)k} + W_N^{2k} \sum_{m=0}^{N/4-1} x_{2(2m+1)+1} W_N^{2(2m)k} \}, k = 0,1,\ldots,N-1$$

- If the two FFT sequences are further divided into even and odd parts, then we end up with four smaller FFTs combined together in a certain order

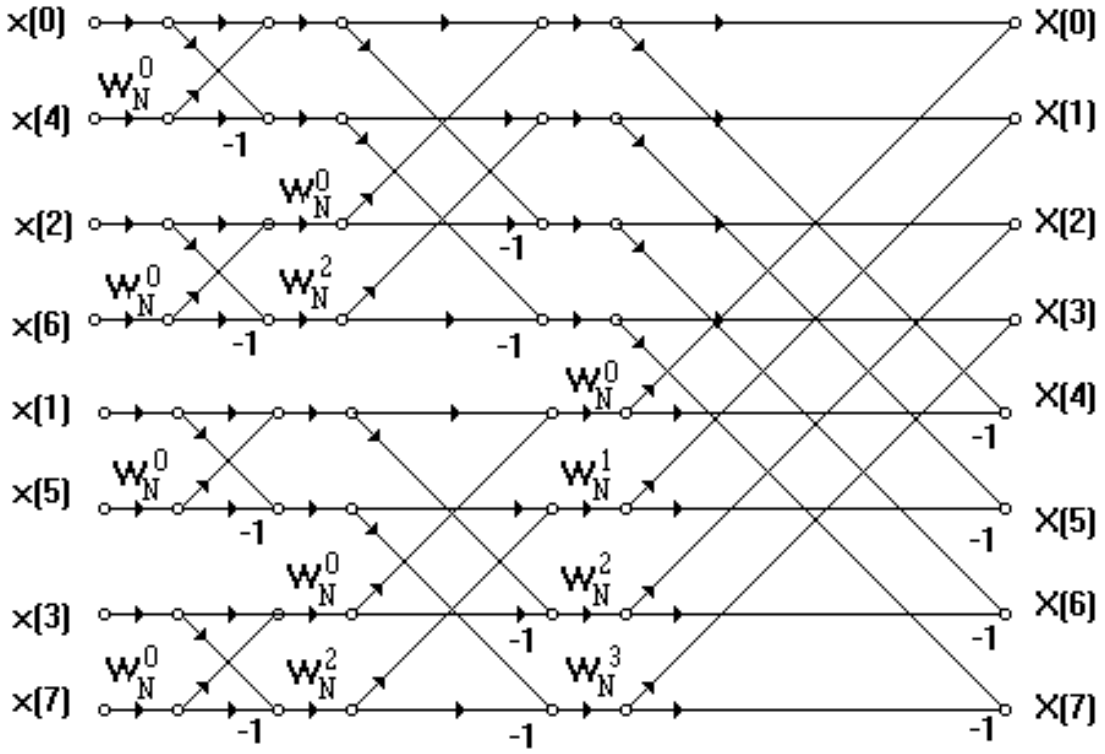
FFT decimation in time – final

- In its final form, the FFT is calculated from 2-point FFTs (radix 2)
- Further 2 point combinations yield the final 8-points



Bit-reversal

- Note in the full butterfly that the inputs are not in order
- This is necessary since the successive division into odd and even requires this



Bit reversal

- If the inputs are in order, the outputs will not be
- Choose one or the other depending on:
  - Application
  - Memory arrangement
  - Connection with following and previous stages
- The order of the inputs is obtained by reversing the Endianess of the index
  - 001->100 (1->4)
  - 010->010 (2->2)
  - 011->110 (3->6)
  - 100->001 (4->1)
  - 101->101 (5->5)
  - 110->011 (6->3)
  - 111->111 (7->7)

Bit reversal

- Bit reversal is useful in software implementations where sorting output vectors could be an issue
- In principle, hardware applications incur no additional cost to rearrange either inputs or outputs (hardwiring)
- However, most modern FFT circuits must support multiple subcarrier lengths, thus requiring complicated sorting circuits, this is where bit reversal could be useful

FFT and latency

- In an N-point FFT N samples are termed an OFDM symbol
- To calculate any output of the FFT we need to have all inputs

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} \quad k = 0,1,\ldots,N-1$$

- This introduces an inevitable latency to FFT = symbol duration = 4μsec in 802.11n
- This latency is often used by later stages to buffer their own startup times
- If a later stage can absorb the latency by operating faster than necessary for a short period, the FFT latency effect can be removed for packets of reasonable length

FFT or IFFT

- From the computational point of view the two are basically the same
- It is thus not hard and very common (basic requirement) to find them implemented using the same circuit FFT/IFFT

$$FFT$$

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} k = 0,1,\ldots,N-1$$

$$IFFT$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{2\pi jkn/N} n = 0,1,\ldots,N-1$$

FFT/IFFT

- There are two differences between the two:
  - IFFT is scaled by N (extremely minor, constant scaling)
  - There is a phase difference in the exponentials (major)

FFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} k = 0,1,\ldots,N-1$$

IFFT

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{2\pi jkn/N} n = 0,1,\ldots,N-1$$

FFT/IFFT

- The (scaled) IFFT can be calculated by first conjugating inputs to FFT and then conjugating the outputs again
- Conjugate is a relatively cheap operation (sign inversion)

$$IFFT$$

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{2\pi jkn/N} n = 0,1,\ldots,N-1$$

$$IFFT = (FFT(x^*))^*$$

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi jkn/N} k = 0,1,\ldots,N-1$$

$$Nx_n = (\sum_{k=0}^{N-1} X_k^* e^{-2\pi jkn/N})^*, n = 0,1,\ldots,N-1$$

FFT/IFFT

$$Nx_n = (\sum_{k=0}^{N-1} X_k^* e^{-2\pi jkn/N})^*, n = 0,1,...,N-1$$

$$= (\sum_{k=0}^{N-1} (X_{kr} - jX_{ki})(\cos(2\pi jkn/N) - j\sin(2\pi jkn/N)))^*$$

$$\Omega = 2\pi jkn/N$$

$$Nx_n = (\sum_{k=0}^{N-1} (X_{kr} - jX_{ki})(\cos\Omega - j\sin\Omega))^*$$

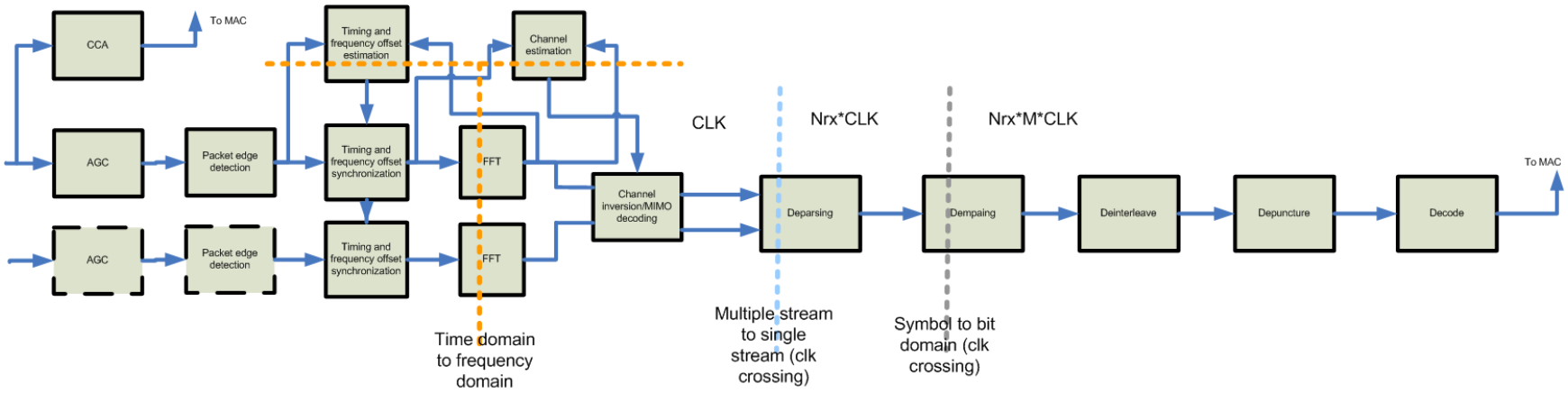
$$= (\sum_{k=0}^{N-1} X_{kr}\cos\Omega - X_{ki}\sin\Omega - j(X_{ki}\cos\Omega + X_{kr}\sin\Omega))*$$

$$= \sum_{k=0}^{N-1} X_{kr}\cos\Omega - X_{ki}\sin\Omega + j(X_{ki}\cos\Omega + X_{kr}\sin\Omega) = \sum_{k=0}^{N-1} X_k e^{\Omega}$$

Linking FFT to OFDM again

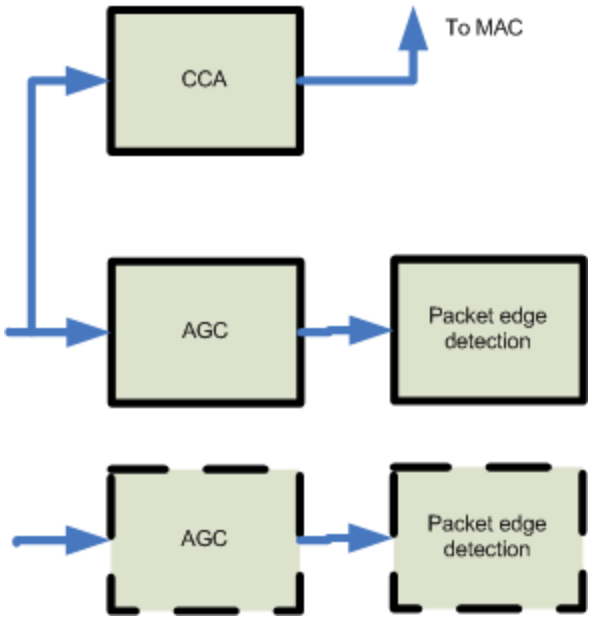
- In OFDM, there are a number of subcarriers between which the channel is divided
- This is then equivalent to the symbol length which is N in FFT
- Standards do not specify the kind of algorithm through which DFT is implemented, only the number of subcarriers and the bandwidth

The 802.11n transceiver revisited



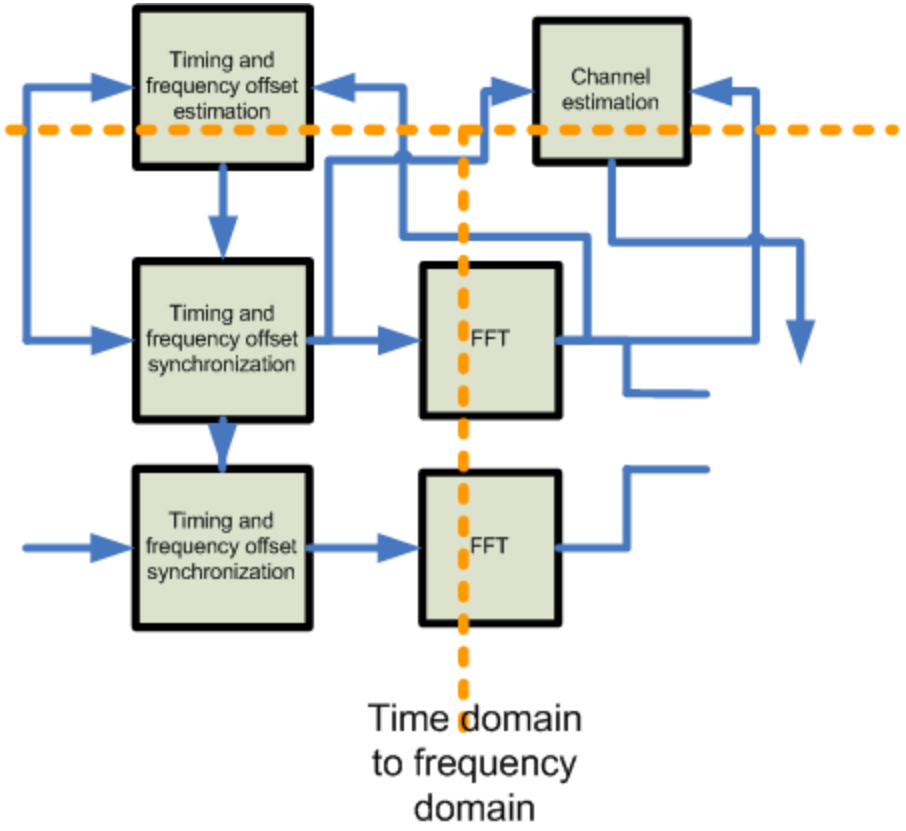
Receiver I

- Two data streams from RF (assume 2x2 MIMO)
- CCA: Clear channel assessment: Figure if the channel is busy and inform MAC (if not busy maybe we can send), can be on one or multiple streams
- AGC: Automatic gain control according to nearness of transmitter to prevent amplifier saturation (maybe only on one stream)
- Packet edge detection: A coarse estimate of when a packet is detected (maybe on one stream)



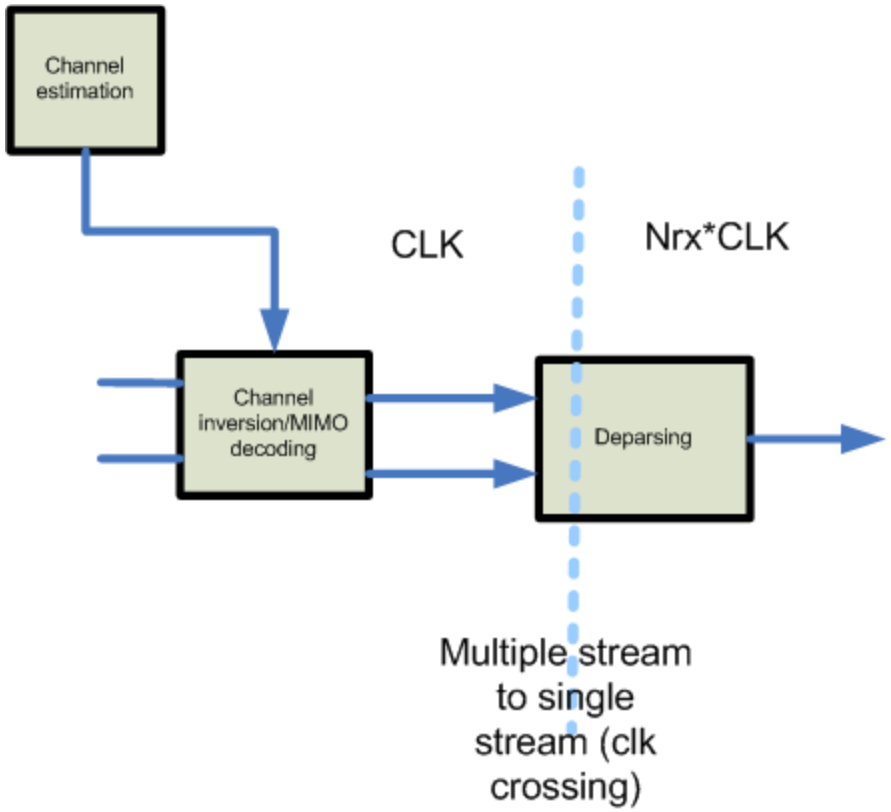
Receiver 2

- Next the start of an OFDM packet is detected, the frequency offset of the carrier is also corrected. This is critical in OFDM (can be in time or frequency or both)
- To do this, the offsets are estimated from header AND pilots in frequency, time or both
- FFT converts to frequency domain
- Channel estimtation guesses H and the noise in time or frequency domain

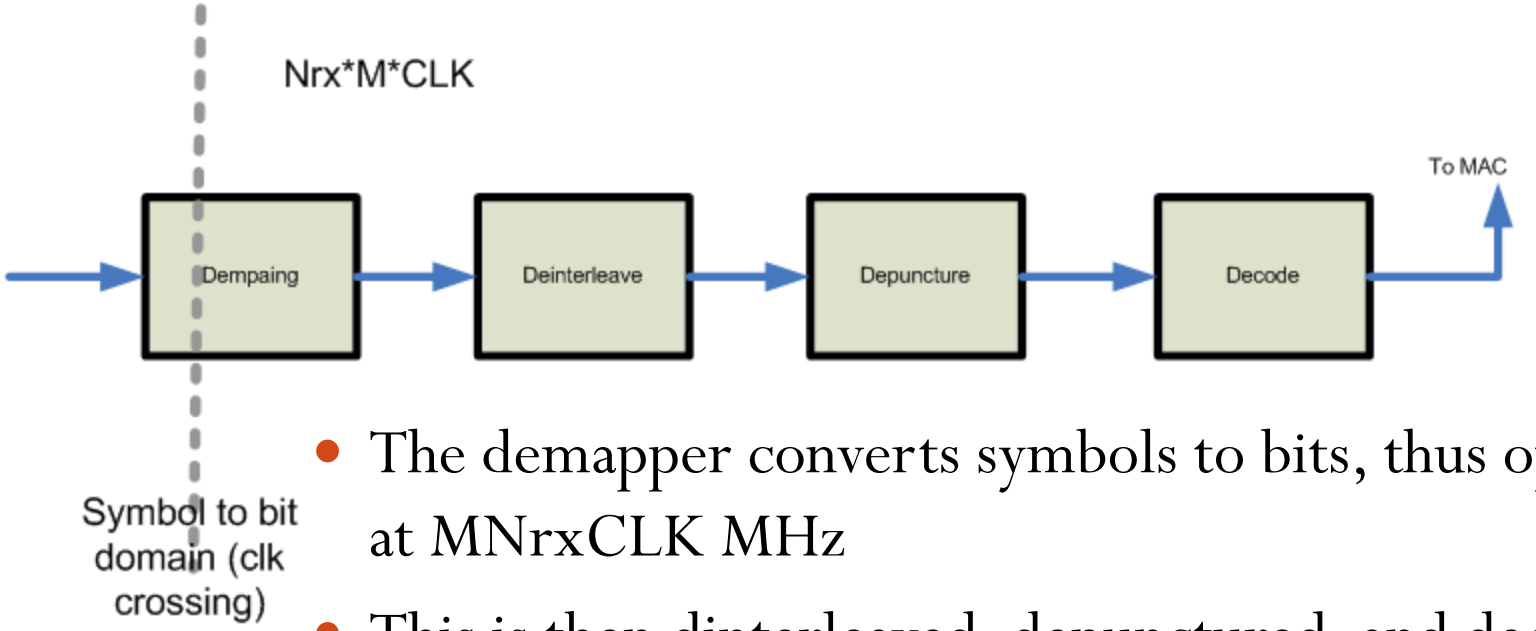


Receiver 3

- The channel estimates are used to invert the channel
- At this point we still have two spatial streams (Nrx=2)
- After channel inversion, you should be able to observe a proper constellation and see only white noise (no rotation), we have proper modulated symbols
- Deparsing combines the streams into one symbol stream. We are in symbol frequency domain operating at Nrx*CLK



Receiver 4



- The demapper converts symbols to bits, thus operating at MNrxCLK MHz
- This is then dinterleaved, depunctured, and decoded into an information bitstream sent to higher layers

Assignment

- Assume you are working with an FPGA with complex multipliers that can be operated at 200MHz
- You are designing an FFT/IFFT circuit to support the 802.11n basic mode (64 subcarriers in 20MHz)
- Count the number of complex multipliers necessary in the circuit
- Draw a block diagram of the circuit implementation
- Briefly describe the operation of the circuit's state machine
- Bonus: Write a VHDL description of the circuit