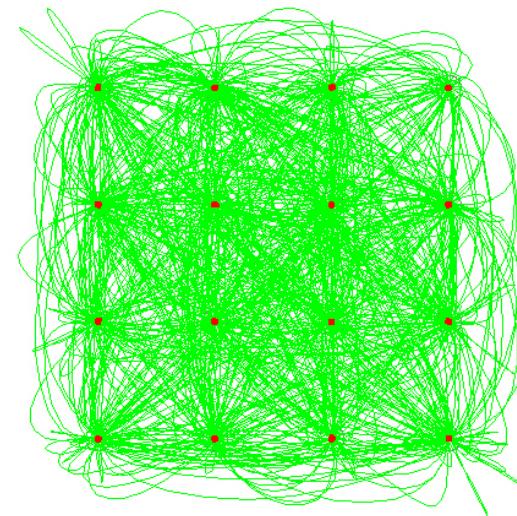


# Implementing Spectrally Efficient Digital Transmitters Using DSPs

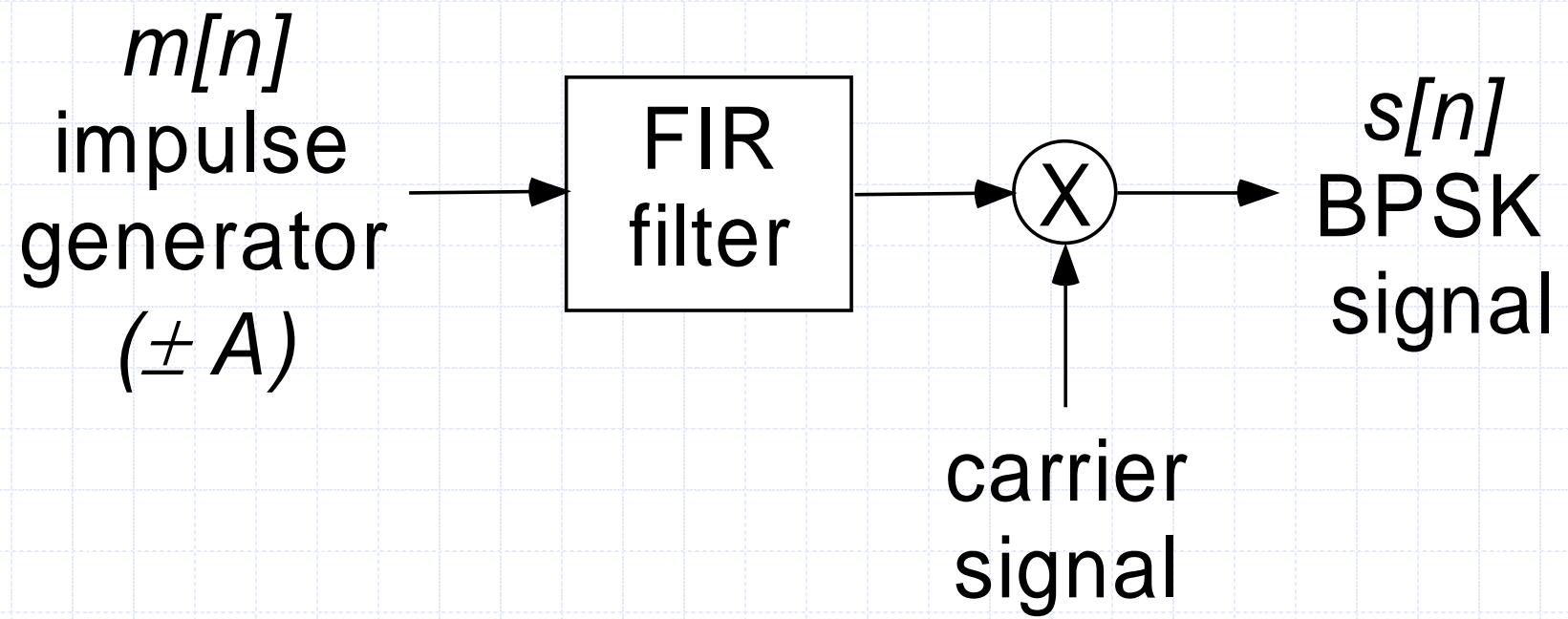
Dr. Thad B. Welch, PE



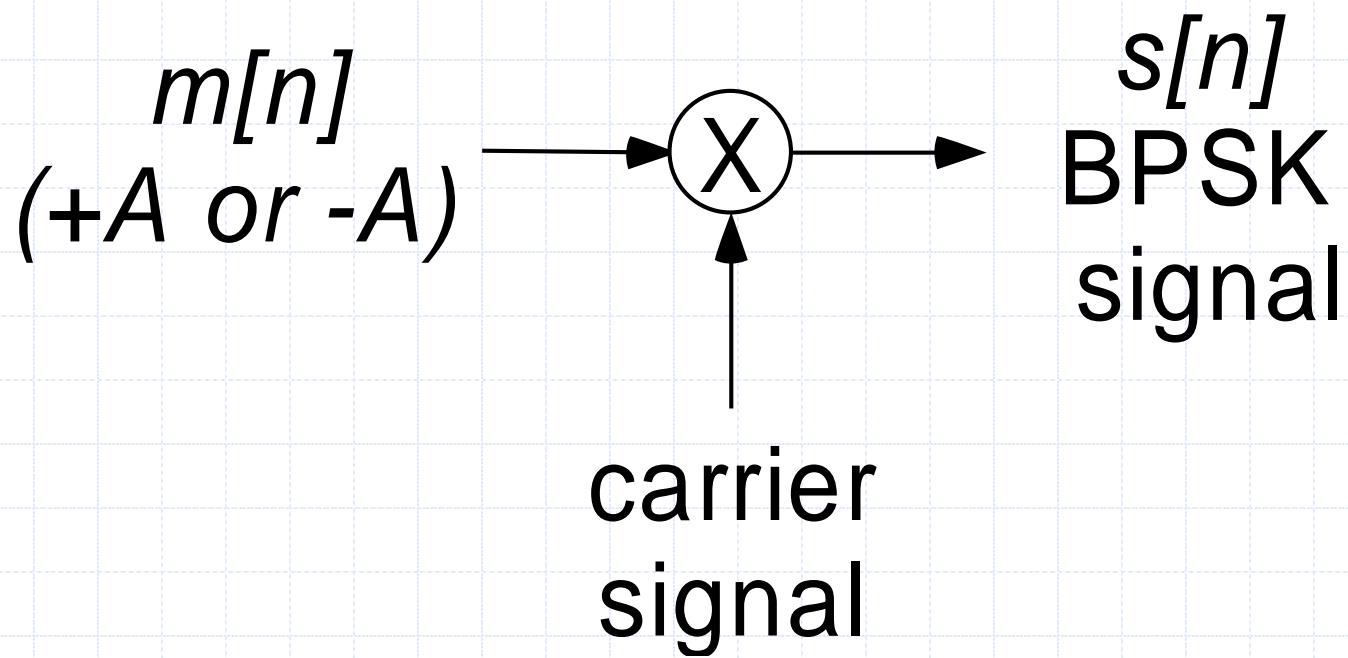
# Overview

- ◆ Review of a few basic block diagrams
  - BPSK, QPSK, 16-QAM ... transmitters!
- ◆ ISI minimization
- ◆ Spectral efficiency
- ◆ Implementation issues
  - Table lookup-based symbol mapping
  - Impulse modulation
  - Reduced complexity NCOs
- ◆ Demonstrations!

# BPSK Transmitter

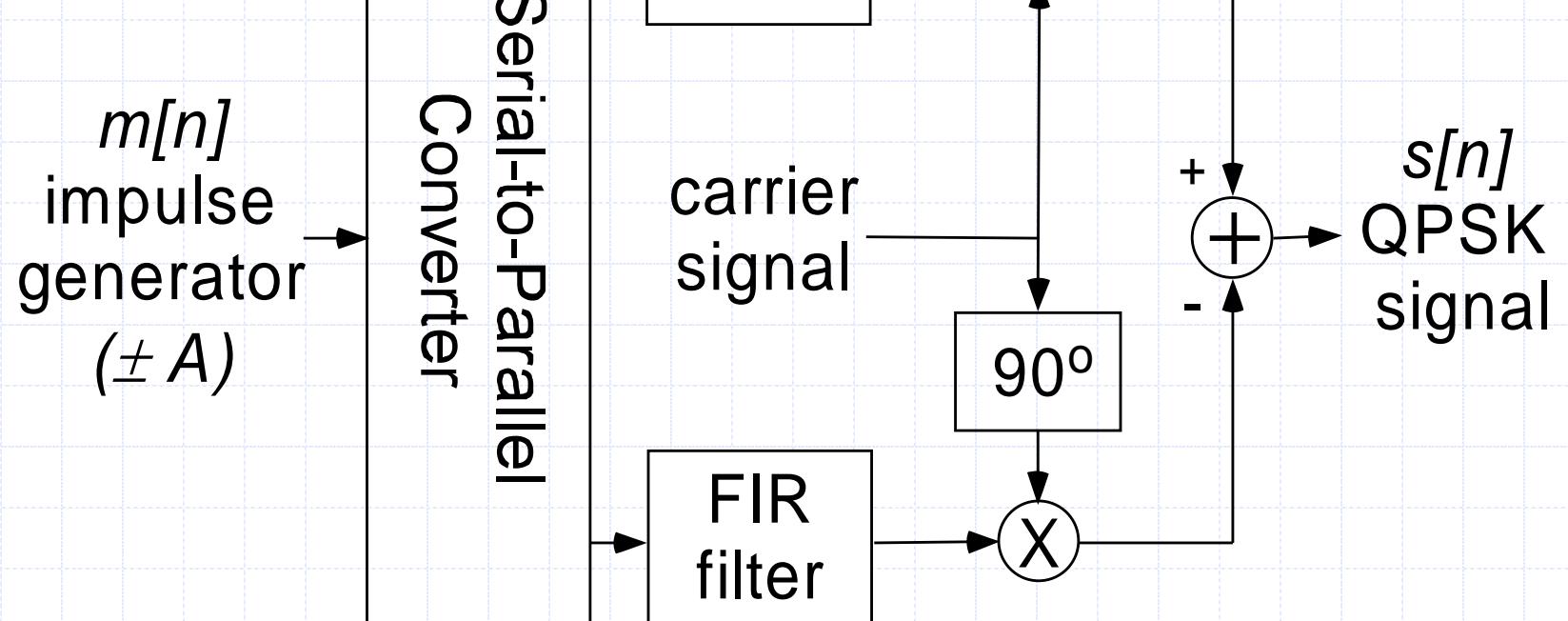


# But Why Bother?

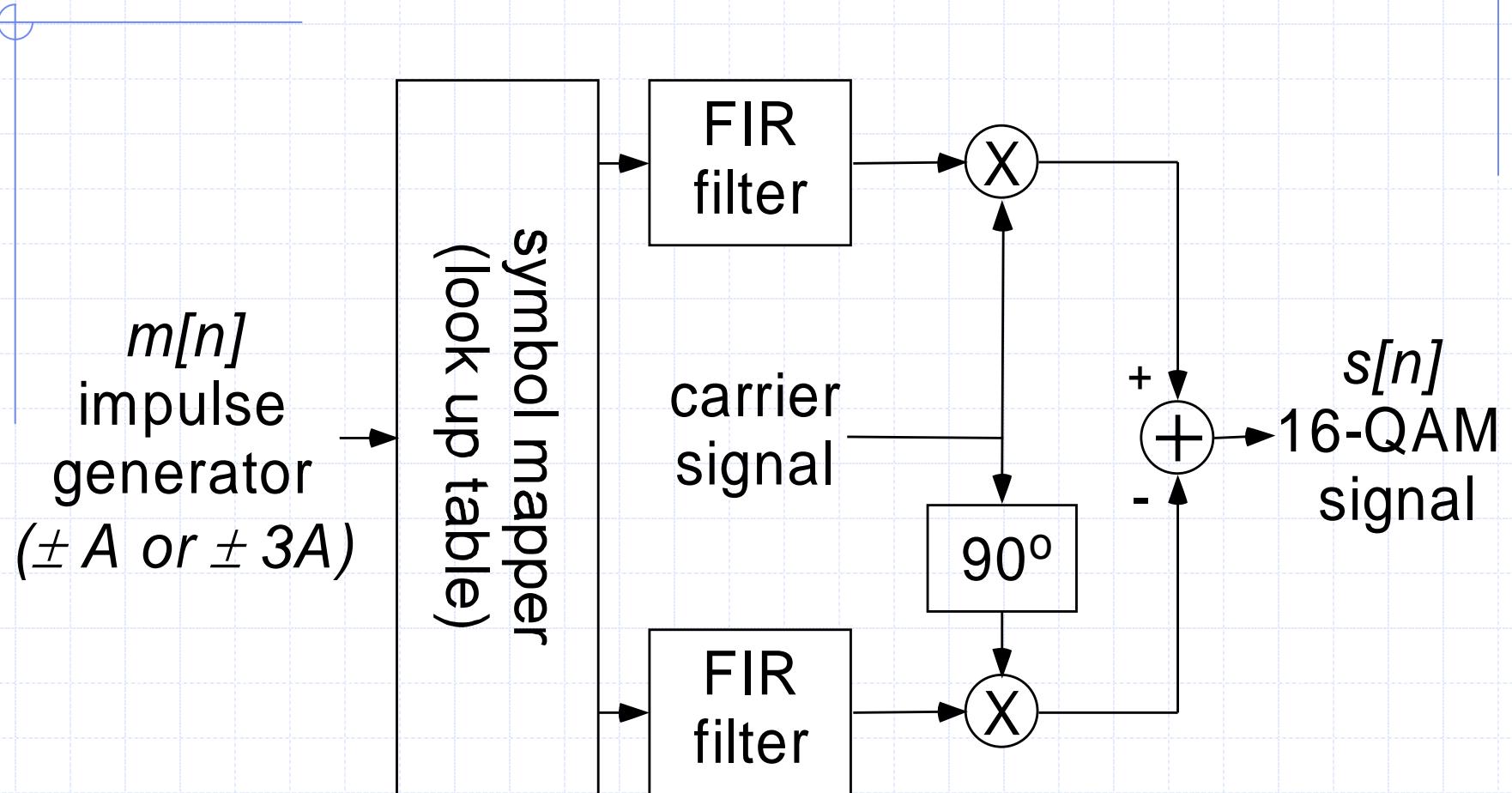


# It's Time For A Real-time Spectral Containment Demo

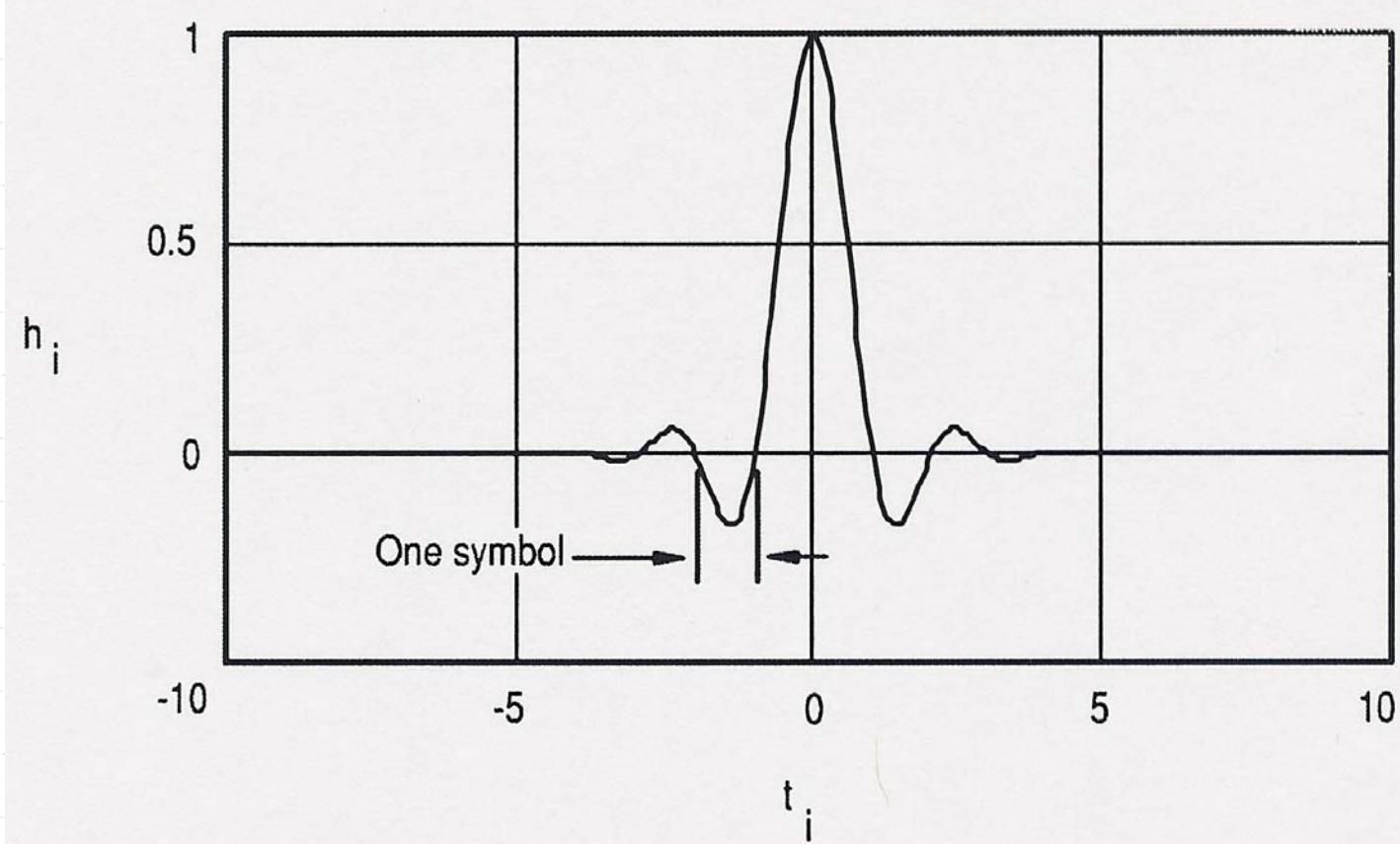
# QPSK Transmitter



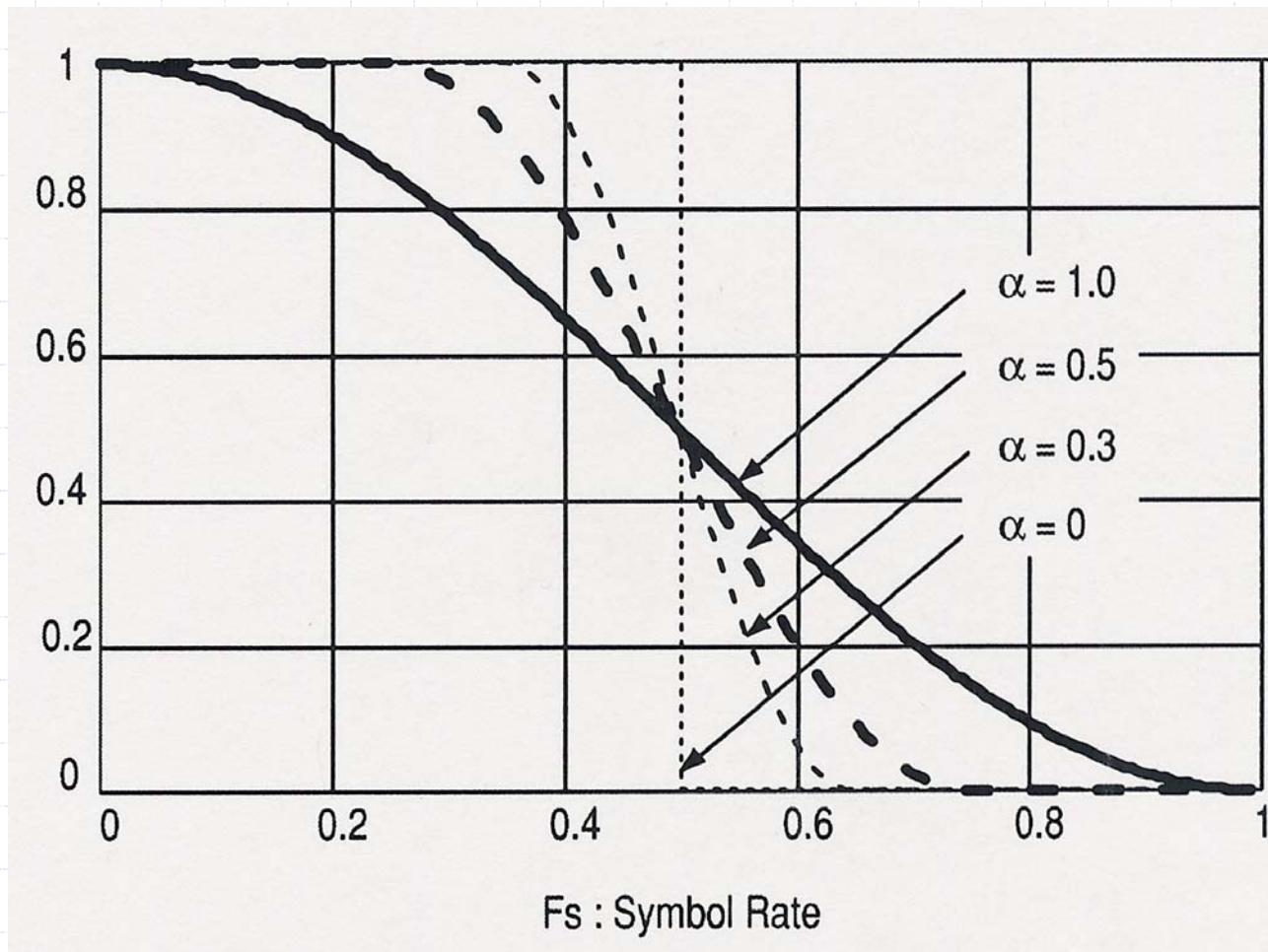
# 16 QAM Transmitter



# Pulse Shaping – ISI Reduction



# Spectral Containment



# Spectral Efficiency

$$\eta = \frac{\text{bits per second}}{\text{bandwidth}} = \frac{\text{bps}}{\text{Hz}}$$

# Theoretical Bandwidth is Proportional to the Symbol Rate

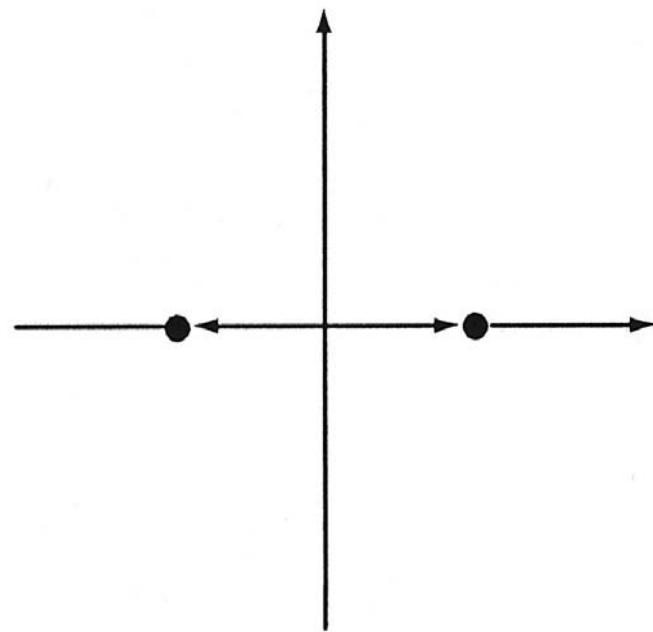
$$BW_{baseband} = \frac{D}{2}(1 + \alpha)$$

$$BW_{RF} = D(1 + \alpha)$$

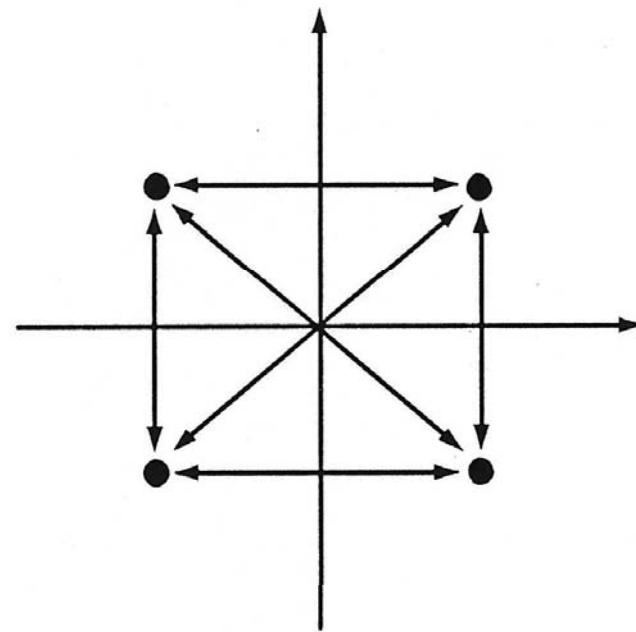
$D$  is the symbol rate

$\alpha$  is the roll-off factor

# Constellation Diagrams

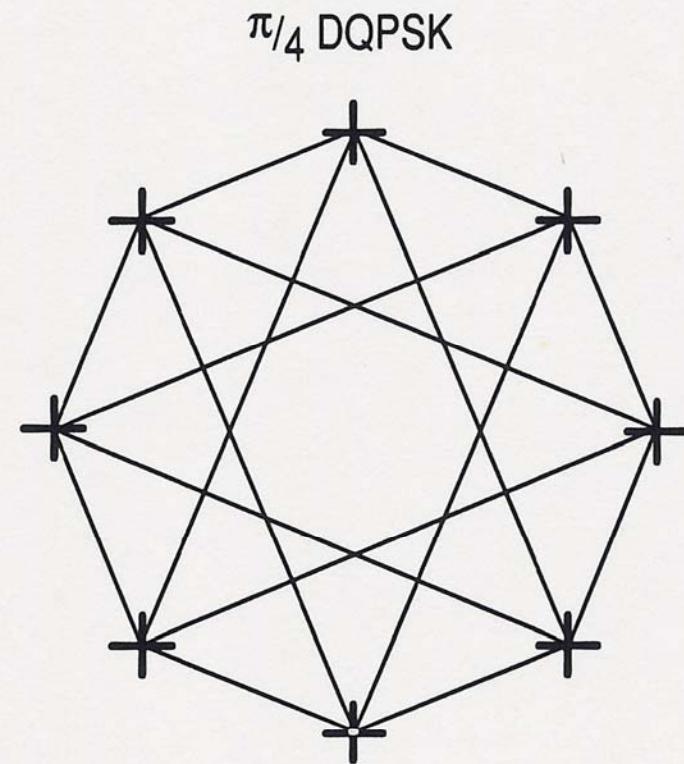
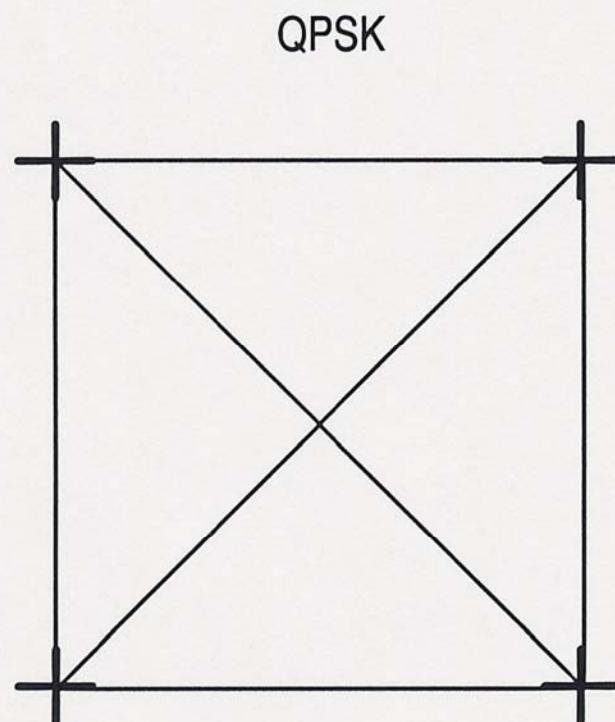


BPSK  
One Bit Per Symbol



QPSK  
Two Bits Per Symbol

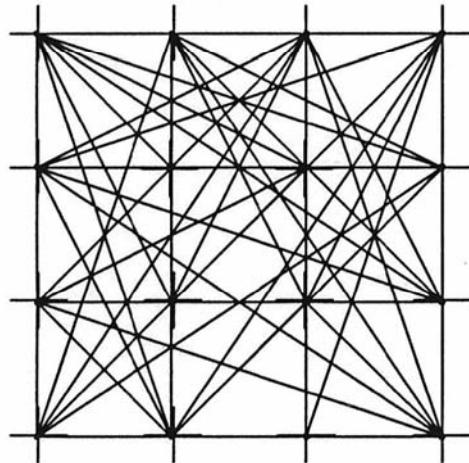
# Constellations Aren't Unique



Both formats are 2 bits/symbol

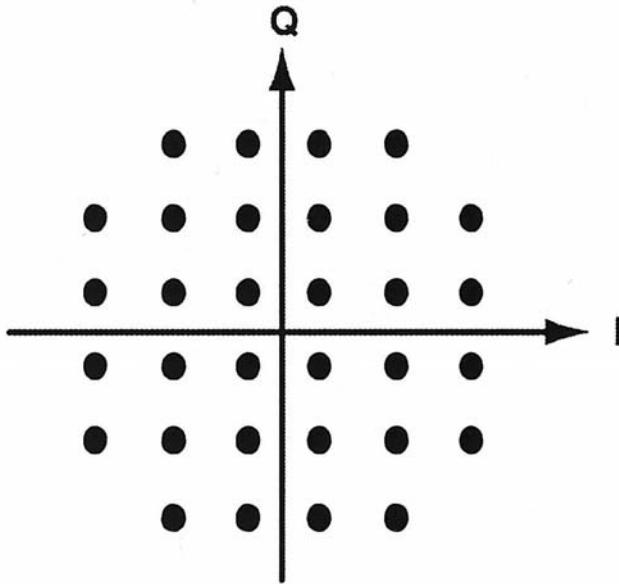
# Some Higher Order Modulation Schemes

Vector Diagram



16QAM  
Four Bits Per Symbol  
Symbol Rate = 1/4 Bit Rate

Constellation Diagram



32QAM  
Five Bits Per Symbol  
Symbol Rate = 1/5 Bit Rate

# Example - BPSK

$$BW_{RF} = D(1 + \alpha)$$

$D$  is the symbol rate = 2400 bps = 2400 sps

$\alpha$  is the roll-off factor = 0.5

$$BW_{RF} = D(1 + \alpha) = 2400(1.5) = 3600 \text{ Hz}$$

# Example – QPSK

$$BW_{RF} = D(1 + \alpha)$$

$D$  is the symbol rate = 2400 bps = 1200 sps

$\alpha$  is the roll-off factor = 0.5

$$BW_{RF} = D(1 + \alpha) = 1200(1.5) = 1800 \text{ Hz}$$

# Example – 16-QAM

$$BW_{RF} = D(1 + \alpha)$$

$D$  is the symbol rate = 2400 bps = 600 sps

$\alpha$  is the roll-off factor = 0.5

$$BW_{RF} = D(1 + \alpha) = 600(1.5) = 900 \text{ Hz}$$

# Bandwidth Summary

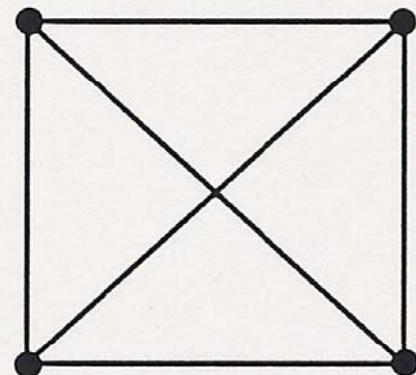
- ◆ BPSK – 3600 Hz (2/3 bps/Hz)
- ◆ QPSK – 1800 Hz (4/3 bps/Hz)
- ◆ 16-QAM – 900 Hz (8/3 bps/Hz)
  
- ◆ BLAST system (lab) . . . > 20 bps/Hz
  - BLAST . . . Bell Labs layered space time
- ◆ So why not keep going?



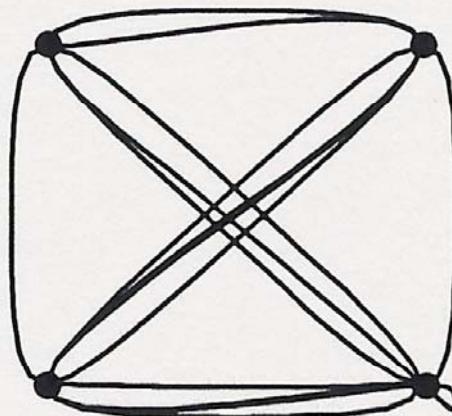
# It's Time For A Real-time Spectral Containment Demo Comparing BPSK, QPSK, and 16-QAM

# Effects of Pulse Shaping

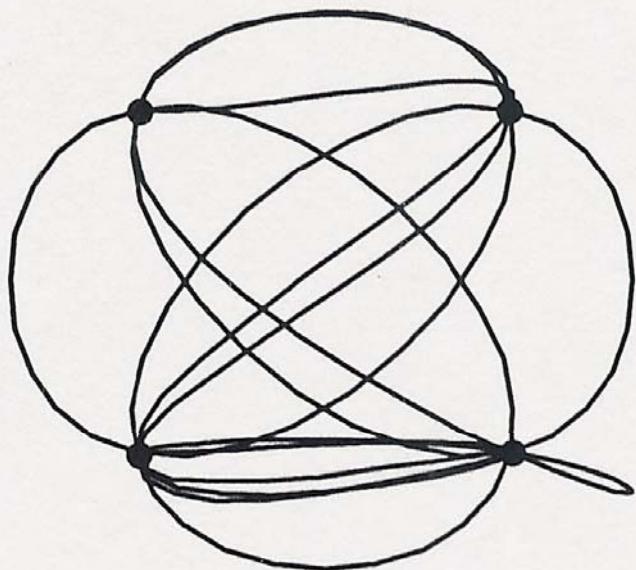
QPSK Vector Diagrams



No Filtering



$\alpha = 0.75$



$\alpha = 0.375$

# Implementation Issues

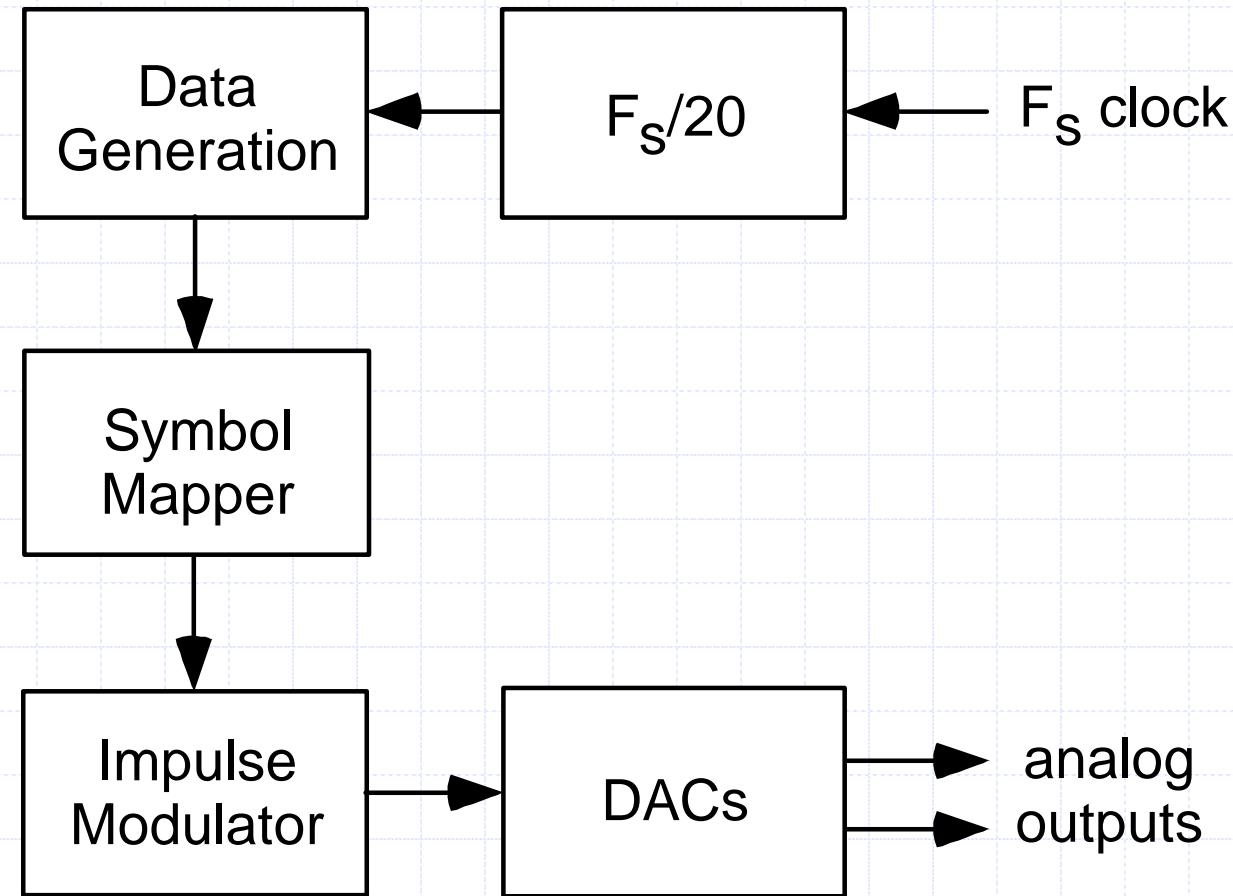
# 16-QAM System Specifications

- ◆ System must operate at 9,600 bits/second
- ◆ Must create/generate the random data
- ◆ Signal constellation points associated with the output of the system must be as shown in the textbook
- ◆ Scaling of the constellation is allowed

# A Few Design Details

- ◆ Audio codec
  - 48,000 samples per second
  - 16 bits per sample
  - Stereo (2 channels)
- ◆ 9,600 bps is 2,400 symbols per second
- ◆  $48,000 / 2,400 = 20$  samples per symbol

# System Block Diagram

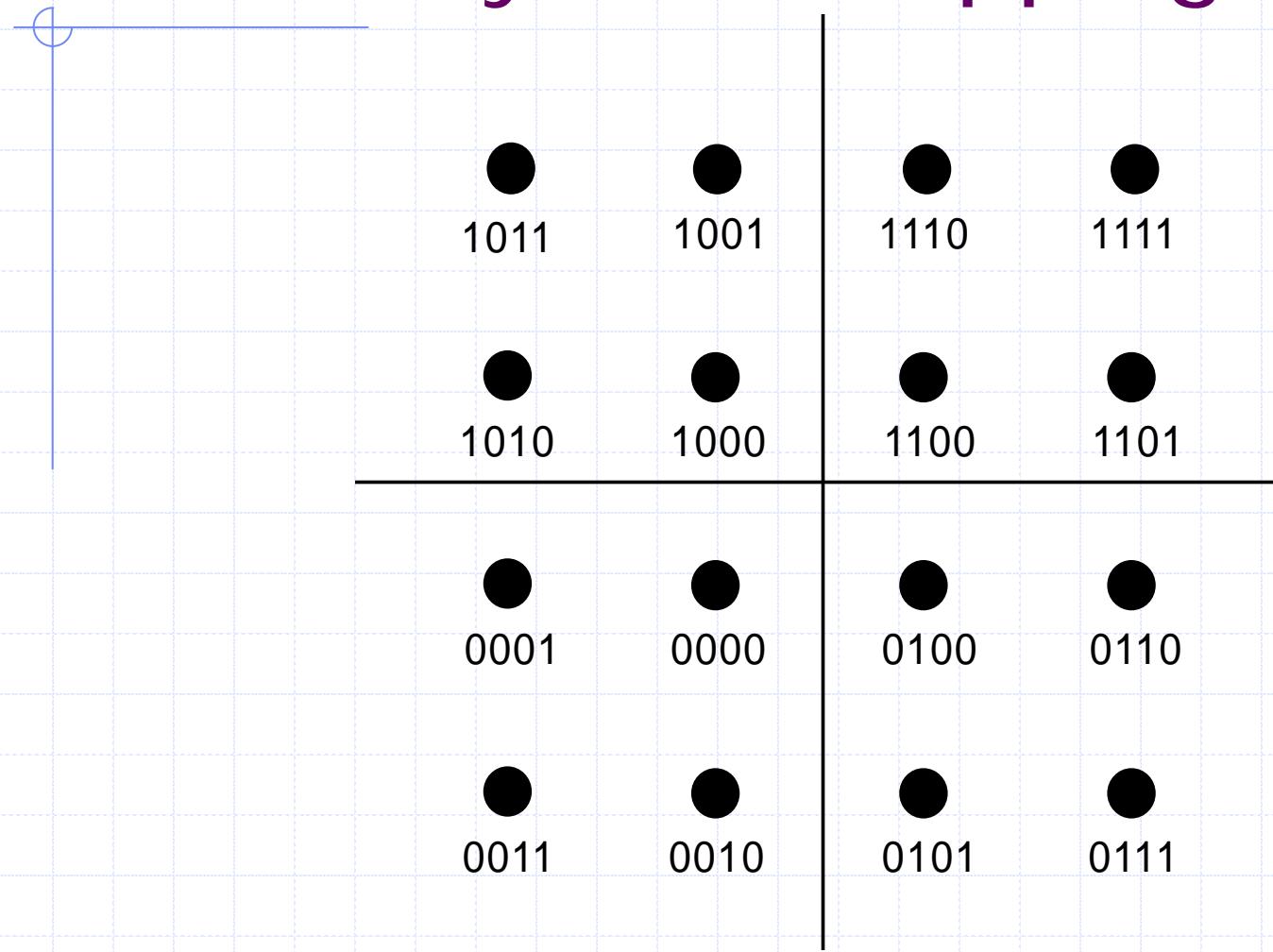


# Implementation – part 1

## ◆ Data Generation

- I added ... **#include <stdlib.h>**
  - ◆ allows for a call of **rand()**
- **symbol = rand() % 16;**
  - ◆ modulo operations involve division
  - ◆ division is computationally expensive
- **symbol = rand() & 15;**
  - ◆ AND with 0000 0000 0000 1111
  - ◆ results in 0, 1, 2, ..., 15
  - ◆ this is less computationally expensive

# Symbol Mapping



# Implementation – part 2

## ◆ Symbol Mapper

- `xLeft[0] = scaleFactor * QAM16_table[symbol];`
- `xRight[0] = scaleFactor * QAM16_table[symbol+16];`

## ◆ QAM16\_table.c (partial listing)

```
#include "QAM16_table.h"

float QAM16_table[32] = {

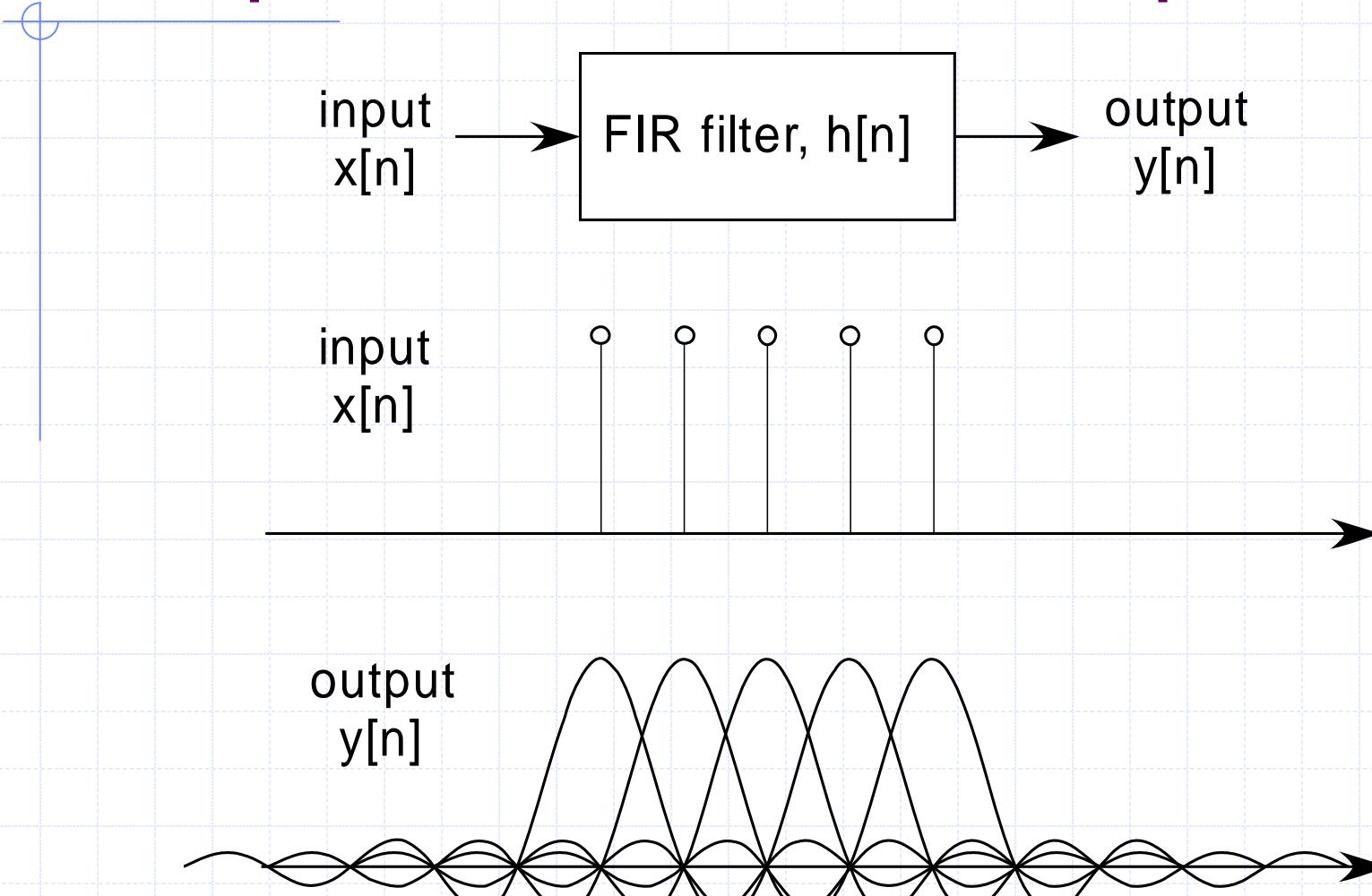
    -5000, /* QAM16_table[0] */
    -15000, /* QAM16_table[1] */
    -5000, /* QAM16_table[2] */
    ...}
```

# Implementation – part 3

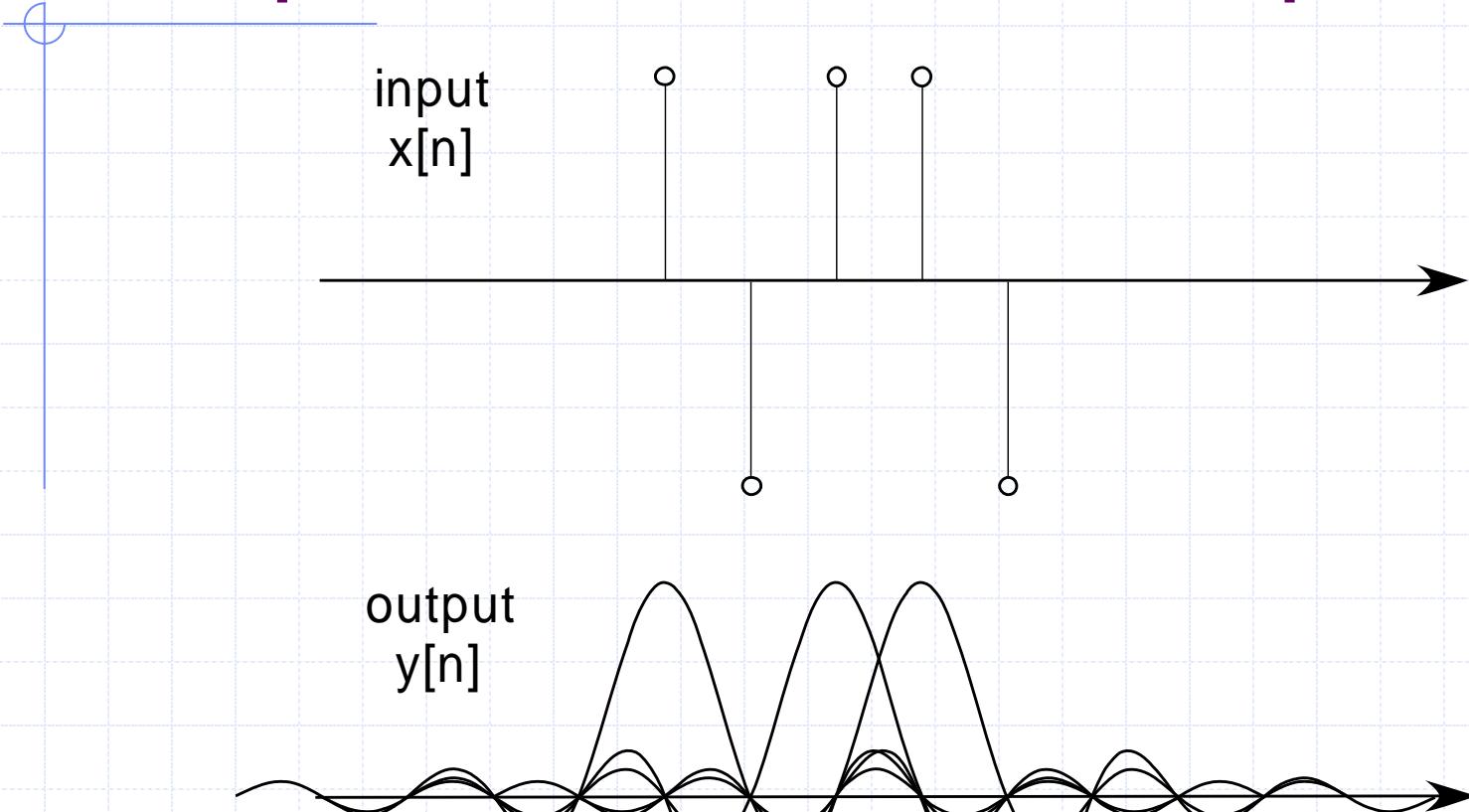
## ◆ Impulse Modulation

- Generate one impulse per symbol
- The impulse's value comes from the symbol mapper
- Use this impulse to excite an FIR filter
- The FIR filter can have whatever shape you desire,  $h[n]$
- The output of the filter is the superposition of the individual impulse responses

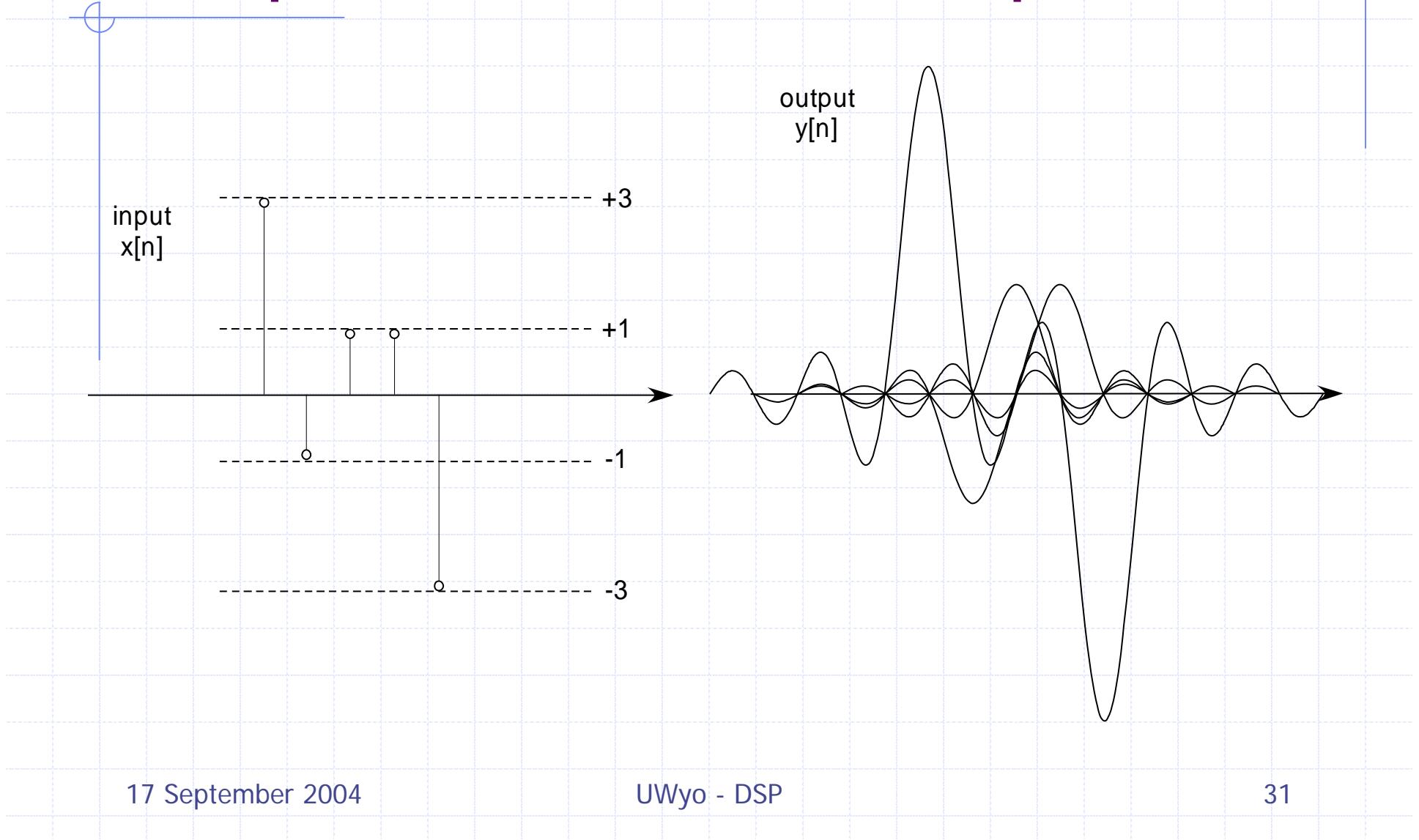
# Impulse Modulation – part 1



# Impulse Modulation – part 2



# Impulse Modulation – part 3



# Implementation – part 4

## ◆ Impulse modulator . . . FIR filter

```
yLeft = 0;
```

```
yRight = 0;
```

```
for (i = 0; i < 10; i++) {  
    yLeft += xLeft[i]*B[counter + 20*i];  
    yRight += xRight[i]*B[counter + 20*i];  
}
```

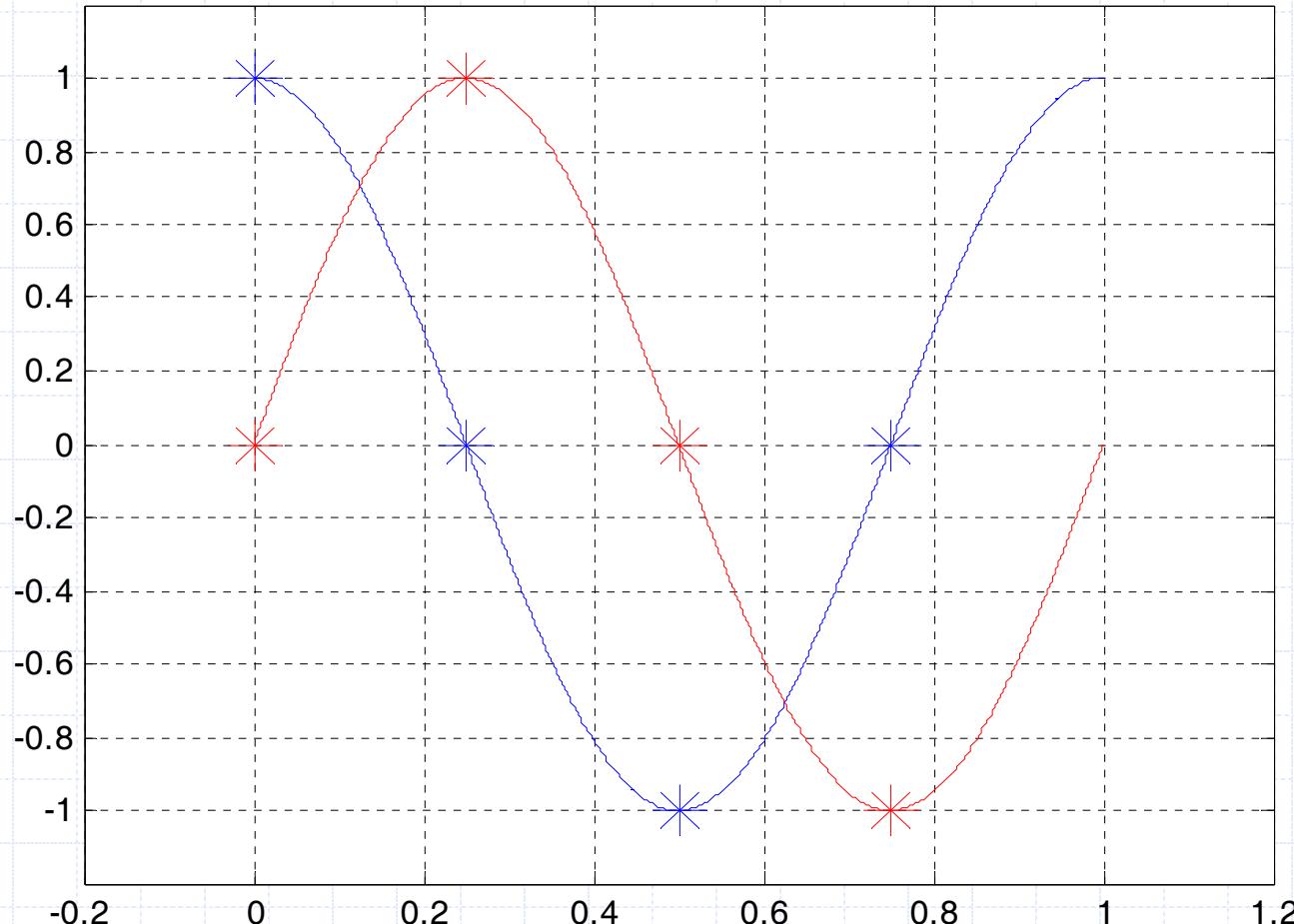
# Implementation – part 5



**Output =**

**yRight \* cosTable[counter] –  
yLeft \* sinTable[counter];**

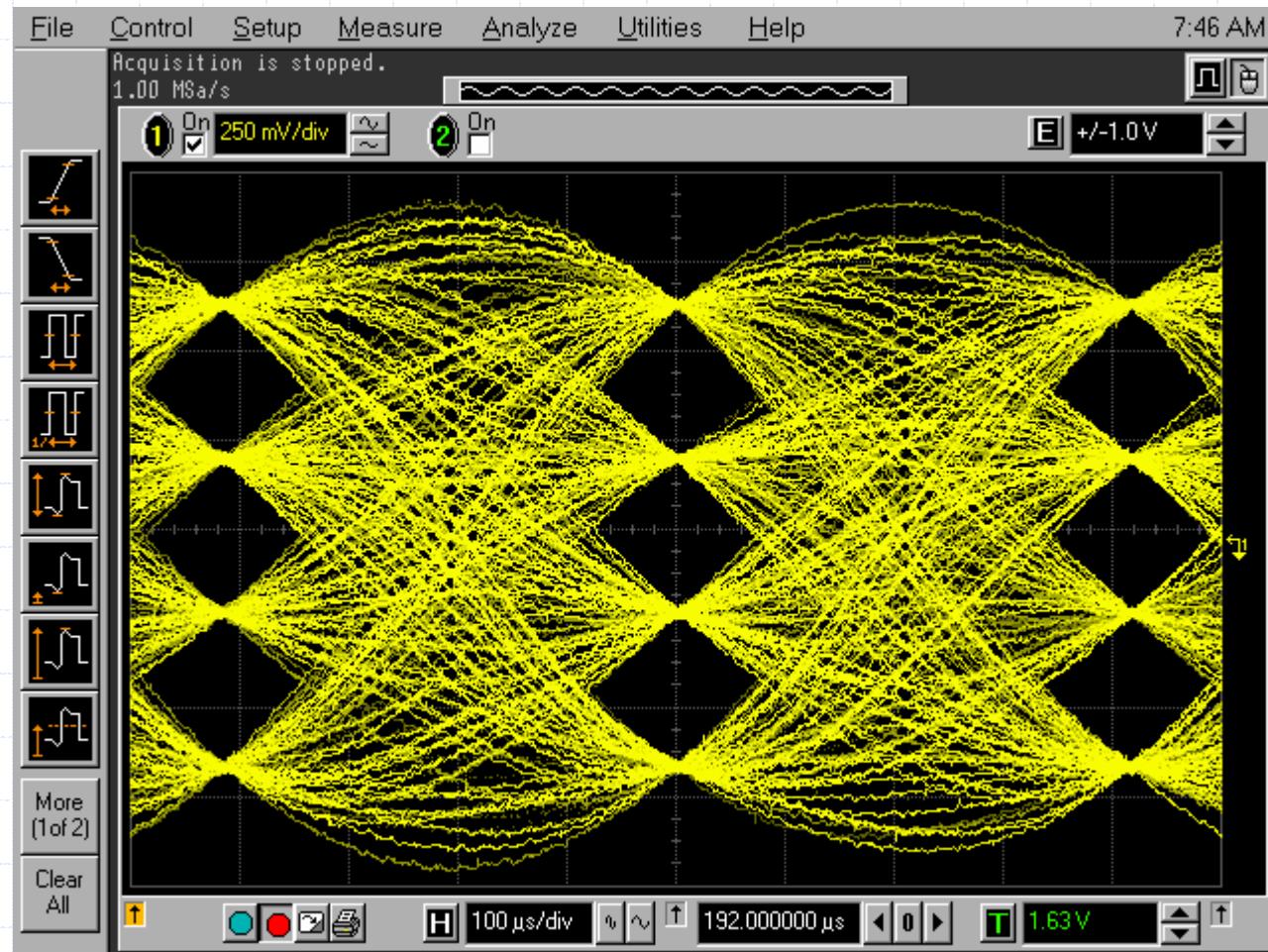
# Sine and Cosine Sampling



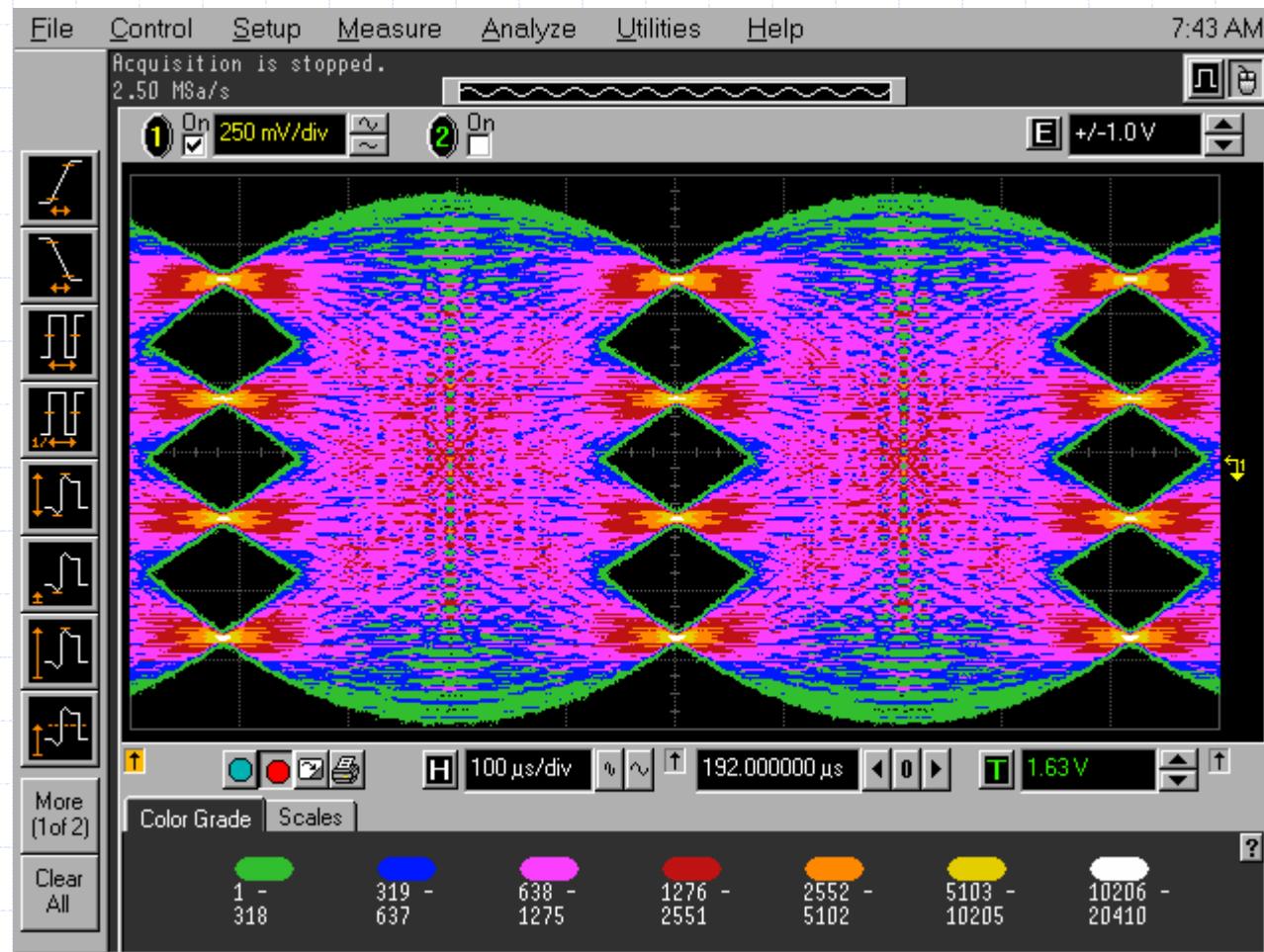
# Sampling Summary

- ◆ Cosine degenerates into +1, 0, -1, 0
- ◆ Sine degenerates into 0, +1, 0, -1
- ◆ So, don't even bother to do the multiplication!
- ◆ The mixer's output becomes
  - I, Q, -I, -Q, . . .

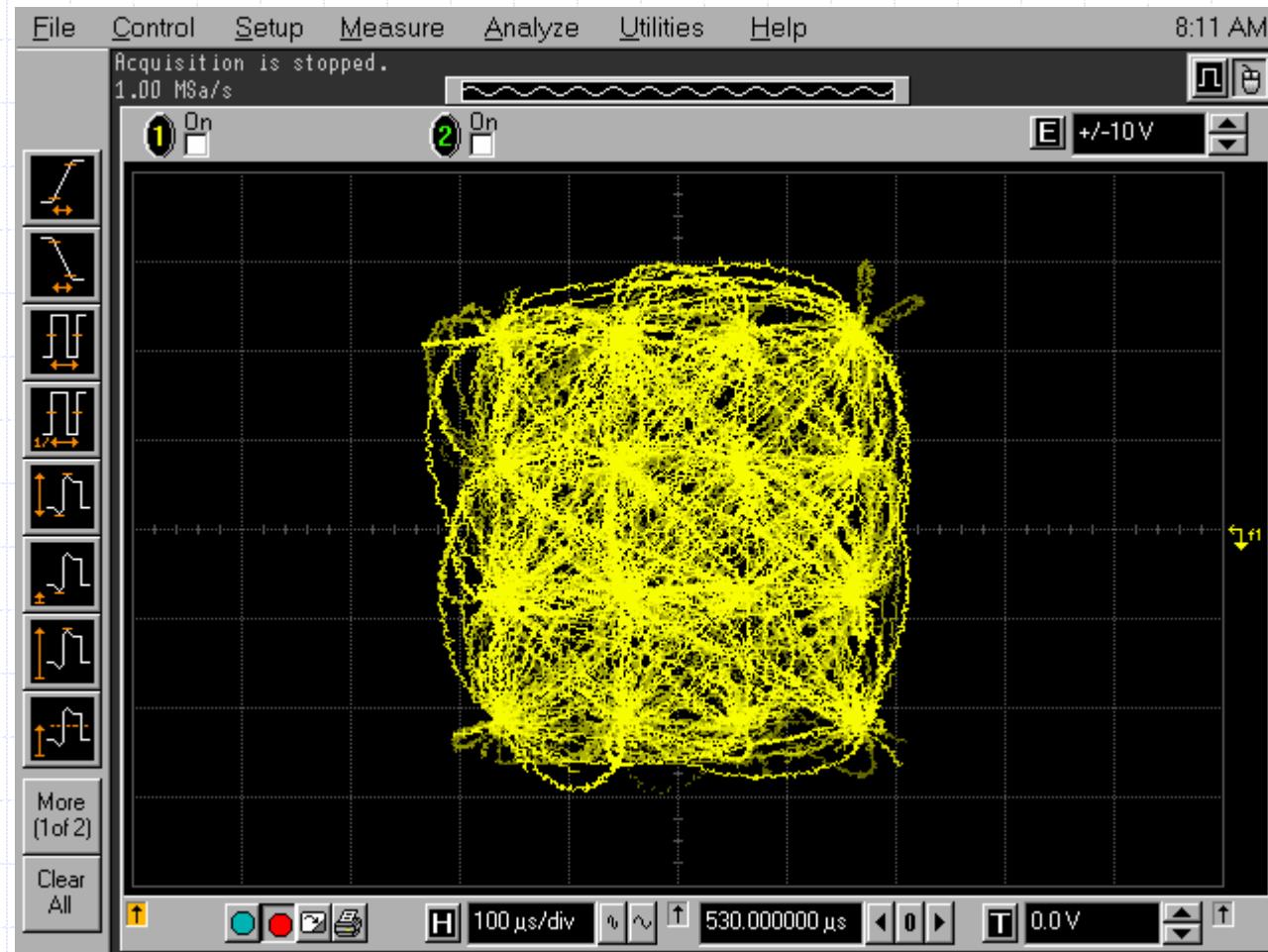
# System Performance – part 1



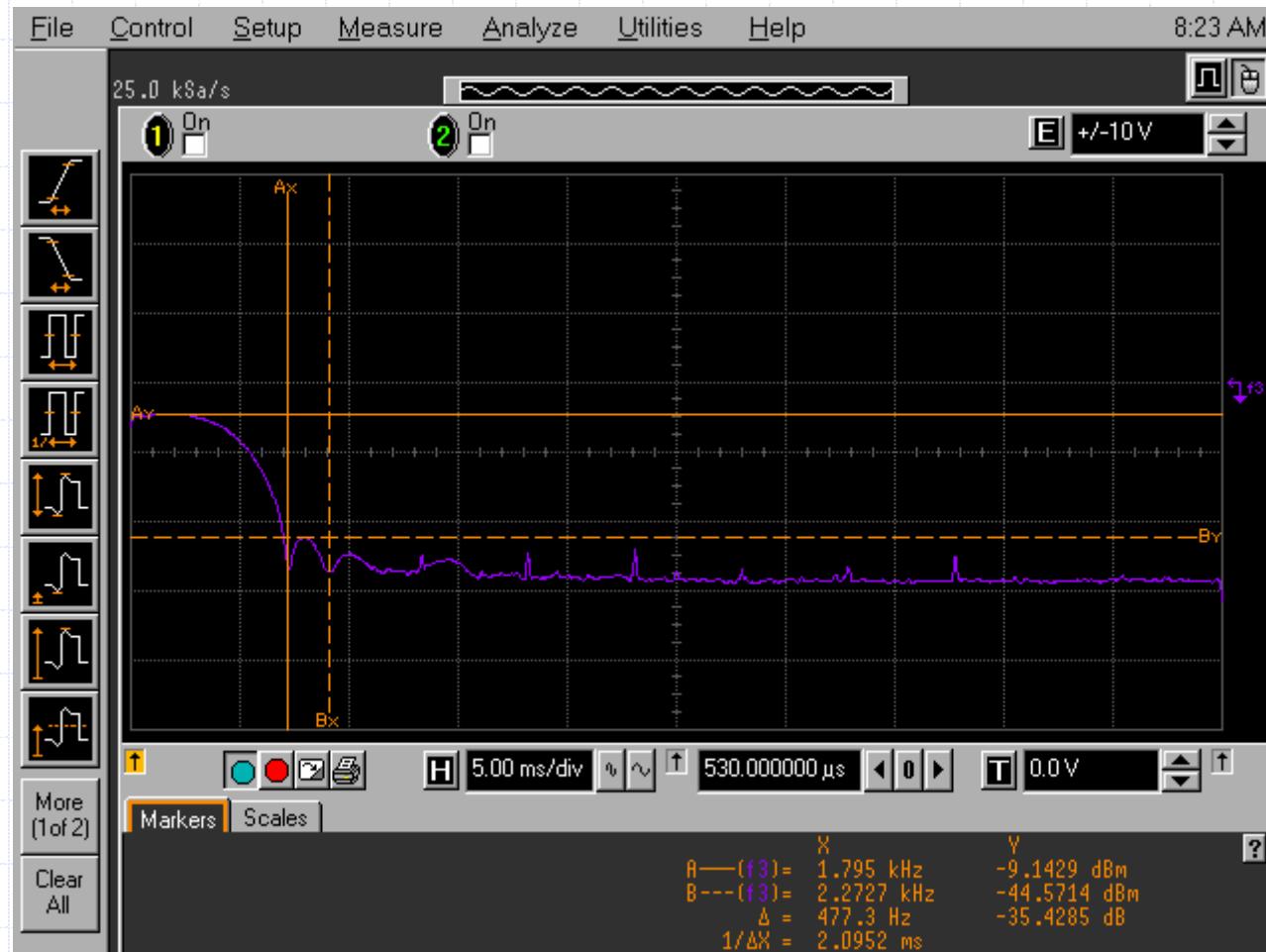
# System Performance – part 2



# System Performance – part 3



# System Performance – part 4



# Real-time demonstration

Demo Time!

Demo Time!

Demo Time!

Demo Time!

# Comments

## ◆ A few comments/considerations

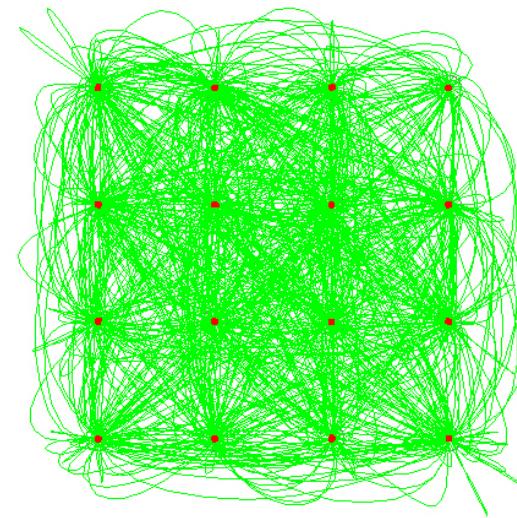
- Table-based modulation/mappers
  - ◆ Once you get one modulation scheme working, others schemes can be straight-forward
- Scale factor for the system's output
- Recalculation of filter coefficients
  - ◆ based on the roll-off factor,  $r$
- Impulse modulation isn't that expensive

# Comments and Conclusions

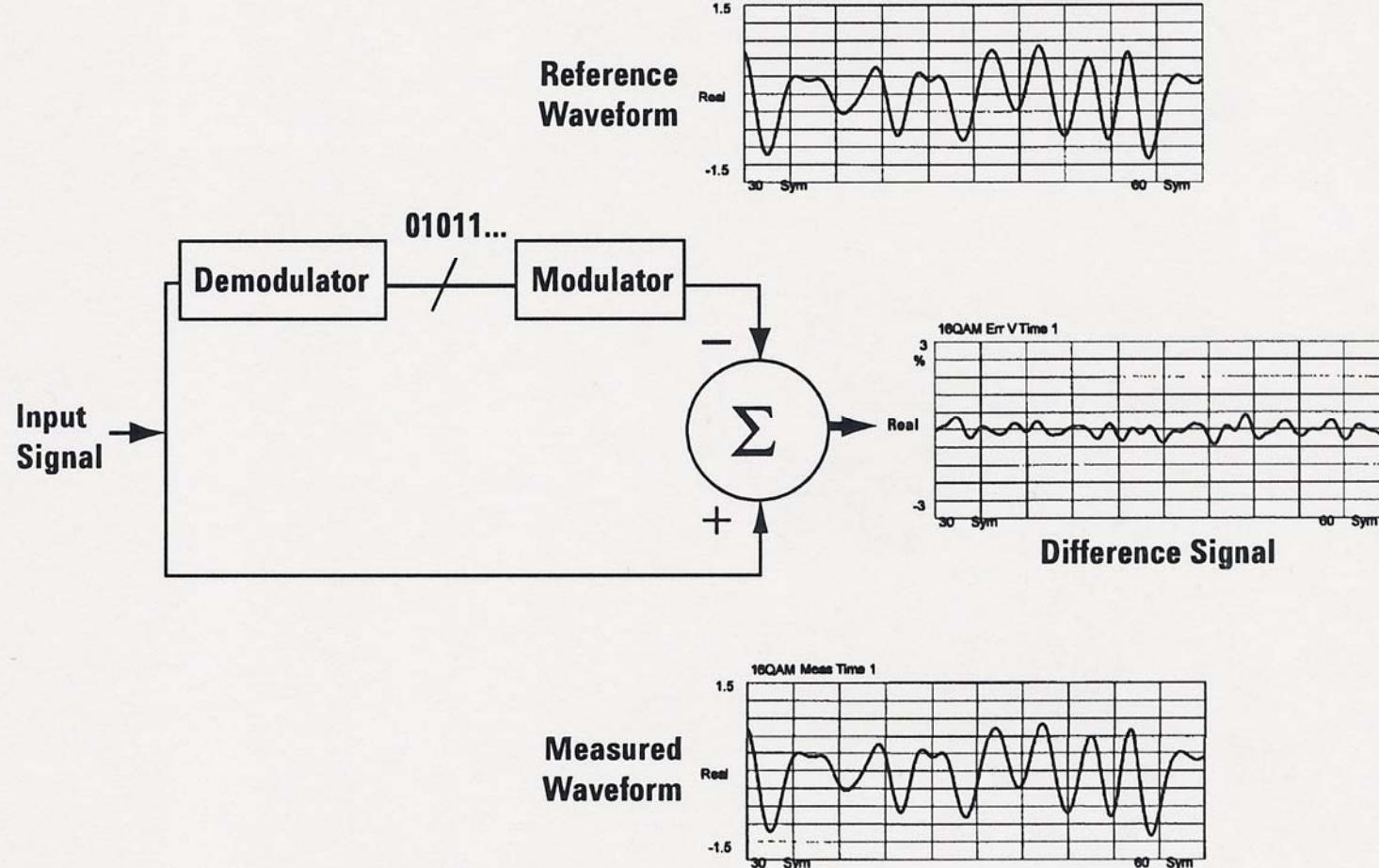
- Modulation –  $\sin\left(n \frac{\pi}{2}\right)$  and  $\cos\left(n \frac{\pi}{2}\right)$
- CODEC pin for symbol synchronizing pulses
- Serial port issues
  - ◆ ISRs must be allowed to complete
- Output switch
  - ◆ Selectable outputs without a code rebuild
  - ◆ Windows-based Apps are now possible

# Implementing Spectrally Efficient Digital Transmitters Using DSPs

Questions ???



# Error Vector Magnitude (EVM)



# Interpretation of EVM

