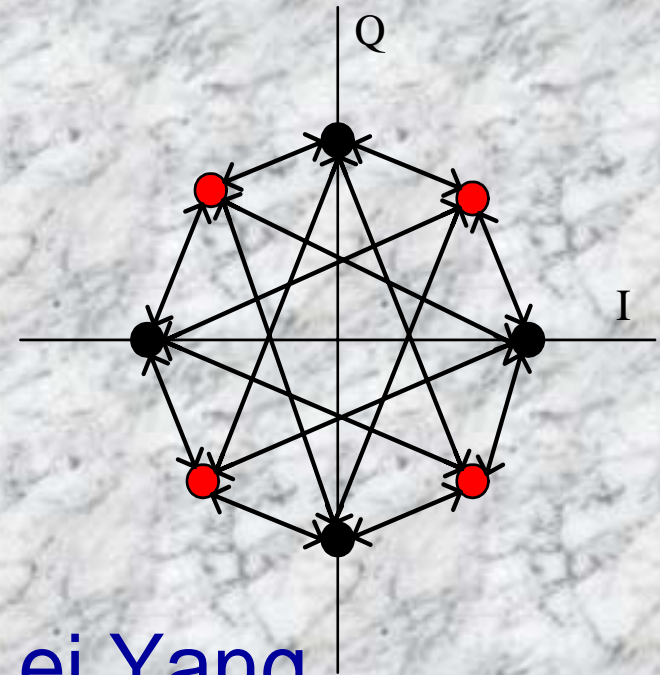
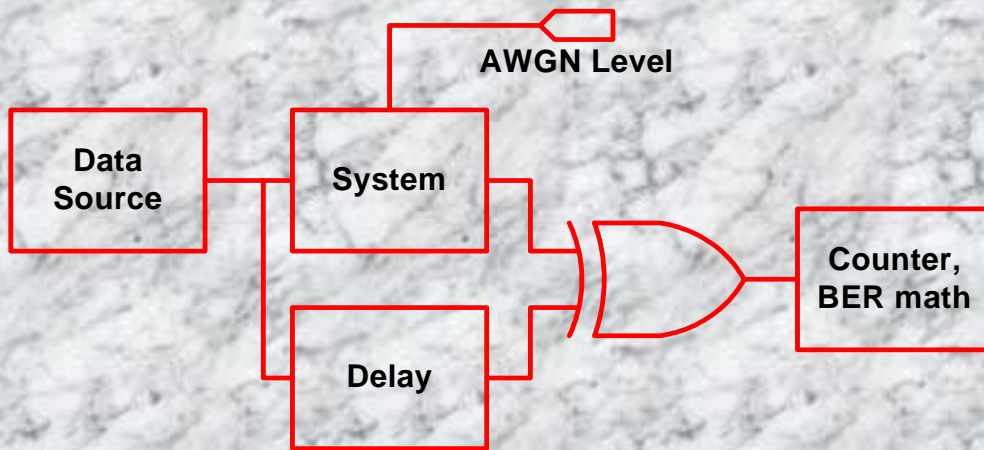


VHDL-AMS Simulation of RF Mixed-Signal Communication Systems



Erik Normark, Lei Yang,
C. Wakayama, P. Nikitin, R. Shi
University of Washington,
MSCAD Lab



Outline

- Background and Motivation
- Simple BPSK Model
- $\pi/4$ DQPSK Model
- Summary and Conclusions

Outline

- Background and Motivation
- Simple BPSK Model
- $\pi/4$ DQPSK Model
- Summary and Conclusions

Design of Mixed Signal Systems

- Increasing demand for System-On-Chip
 - RF, analog, digital circuits all on one chip
 - Fast time-to-market issues
- The designer want to simulate RF, analog,baseband part simultaneously.
- Simulatie high frequency RF system ,analog system and digital components is computationally complex
- VHDL-AMS:
 - facilitates mixed design approach
 - The co-simulation of SOC system maybe not so accurate, but it gives the designers insight how the system works.

Motivation

- Create a mixed-signal, system-level model of a high-frequency transceiver in VHDL-AMS
- Ability to measure system performance
Through Bit-Error-Rate (BER) analysis
- Compare results of VHDL-AMS simulations with other available mixed-signal modeling environments.

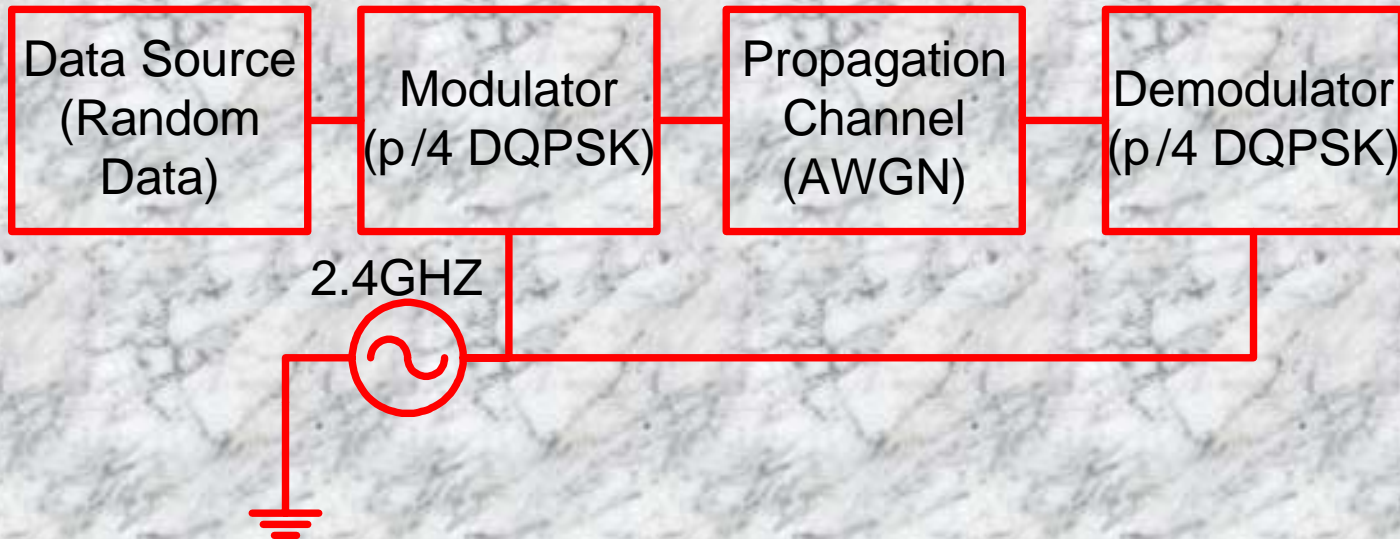
Design Tools

- **ADVance-MS (ADMS, Mentor Graphics)**
 - Compiler and simulator for VHDL, VHDL-AMS, Verilog, Verilog-A, SPICE, C
 - Supports most of VHDL-AMS standard
- **Agilent ADS (Hewlett Packard)**
 - Commercial RF design environment for system-level design modeling and simulation

Outline

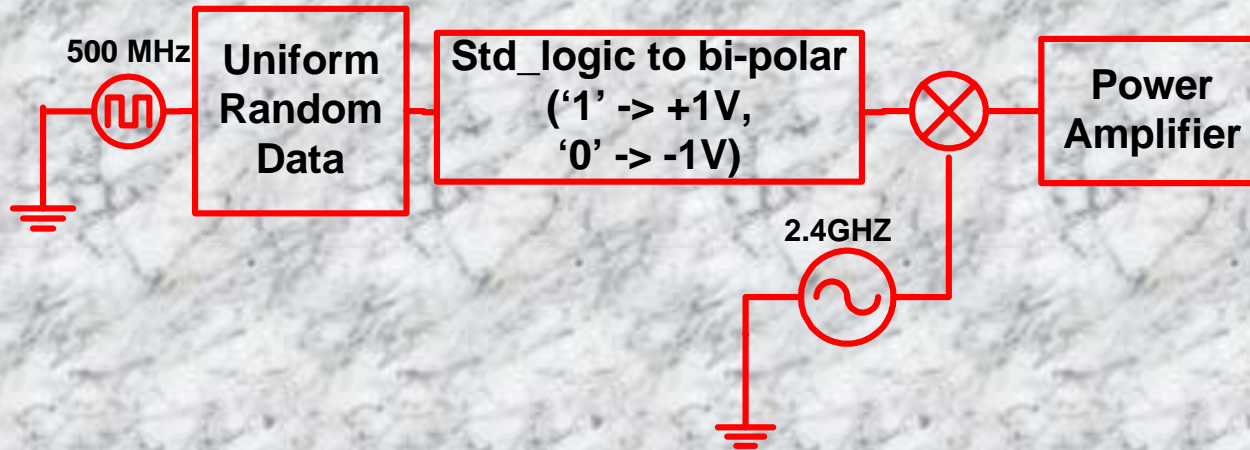
- Background and Motivation
- Simple BPSK Model
- $\pi/4$ DQPSK Model
- Summary and Conclusions

BPSK System



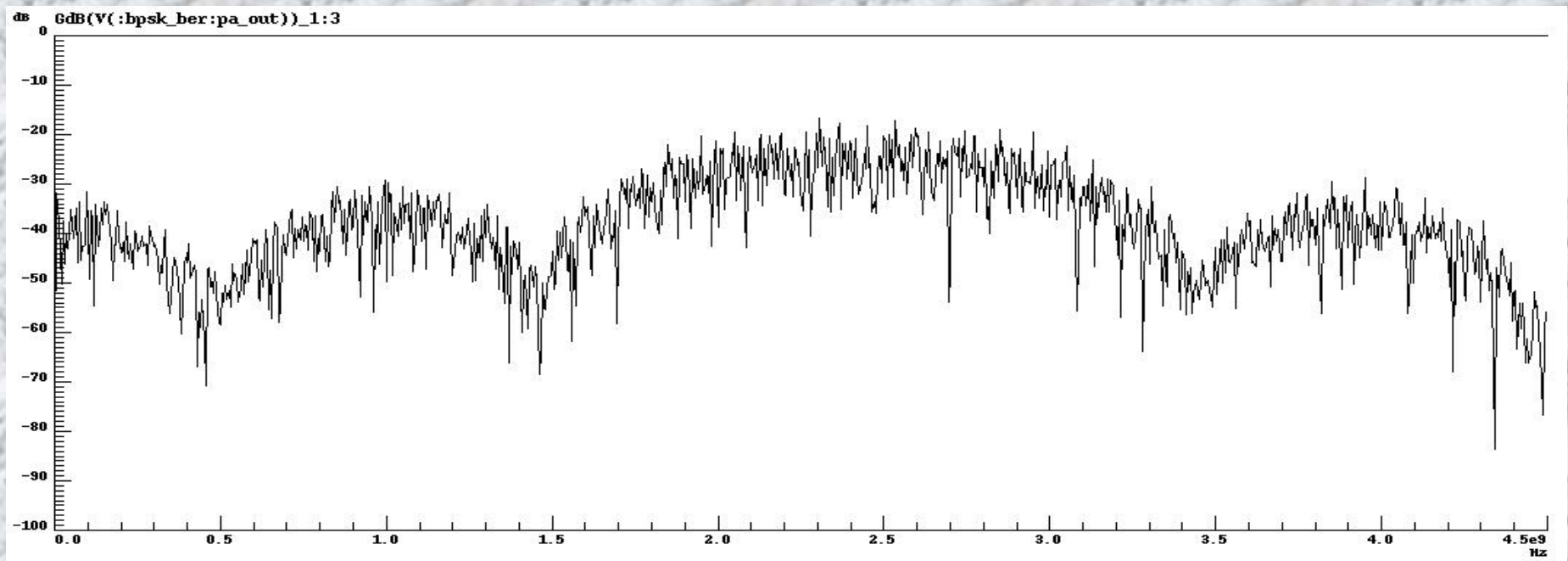
- Ideal System Architecture
 - Transmitter
 - Noisy Channel
 - Receiver
 - BER Calculation
- Evaluate system performance via comparison to theoretical BER calculation

BPSK : Transmitter



- Modulate data by shifting phase of oscillator between $\pm 180^\circ$

BPSK Transmitted Spectrum



BPSK : Propagation Channel



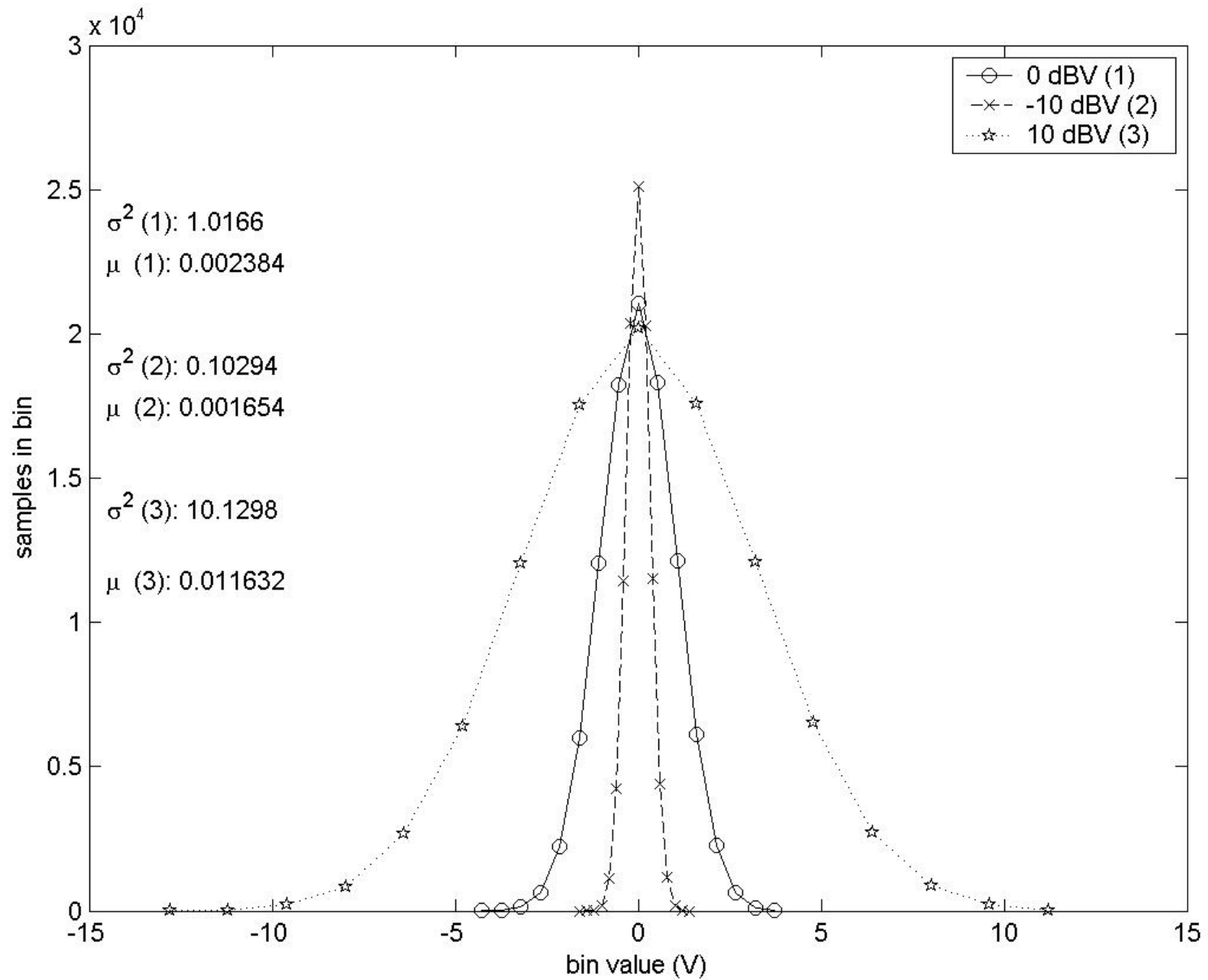
- Basic channel with variable Additive White Gaussian Noise Power
- Box-Muller transformation of two uniform, independent random variables

AWGN Generator Process

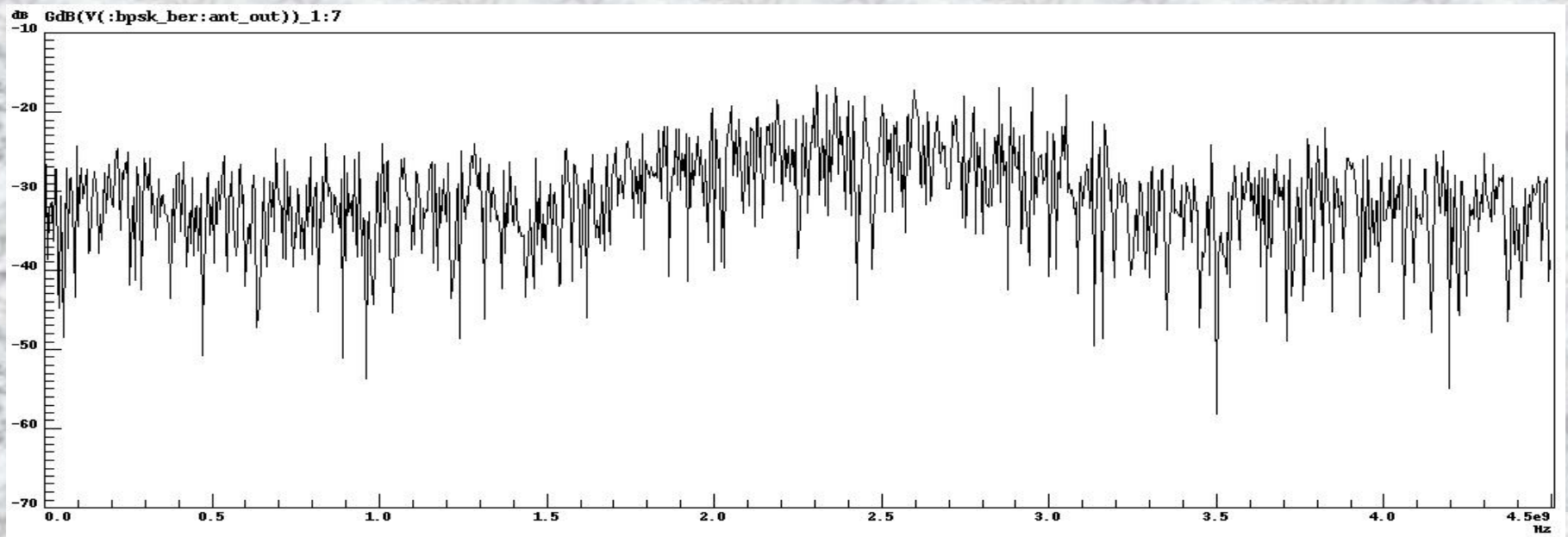
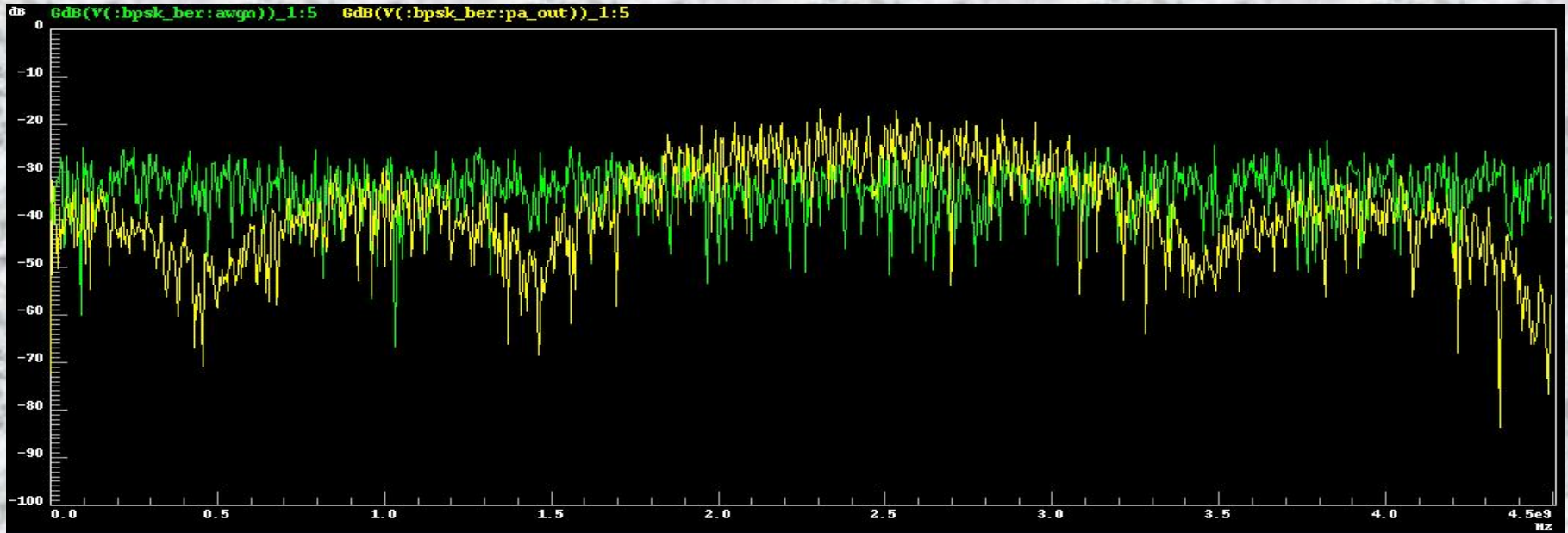
```
noise_calc : process (noise_s)
  variable s1 : positive := seed1;
  variable s2 : positive := seed2;
  variable x1,x2 : real;  -- Uniform random variables
begin
  UNIFORM(s1,s2,x1); -- create two uniform variables
  UNIFORM(s1,s2,x2);
  -- create Gaussian variable using Box-Muller method
  noise_s <= SQRT(-2.0*LOG(x1))*COS(2.0*MATH_PI*x2)
after rate;
end process noise_calc;

vo == 10.0** (level/20.0)*noise_s;
```

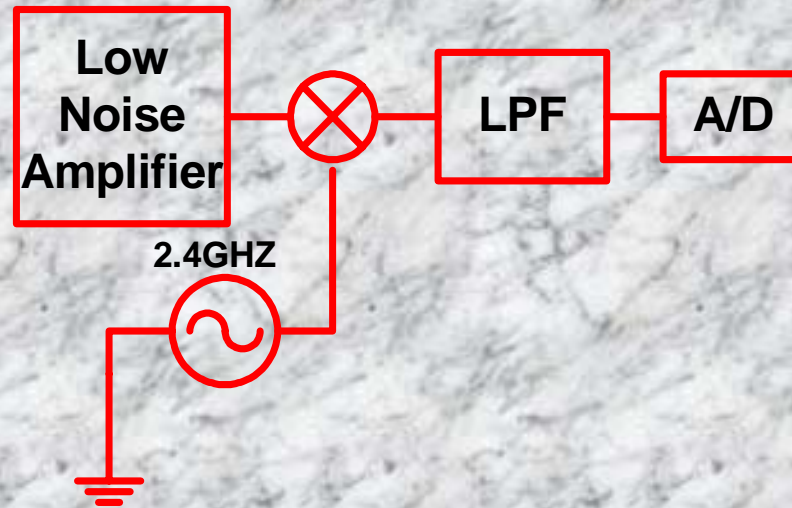
AWGN Testing



BPSK : Received Spectrum

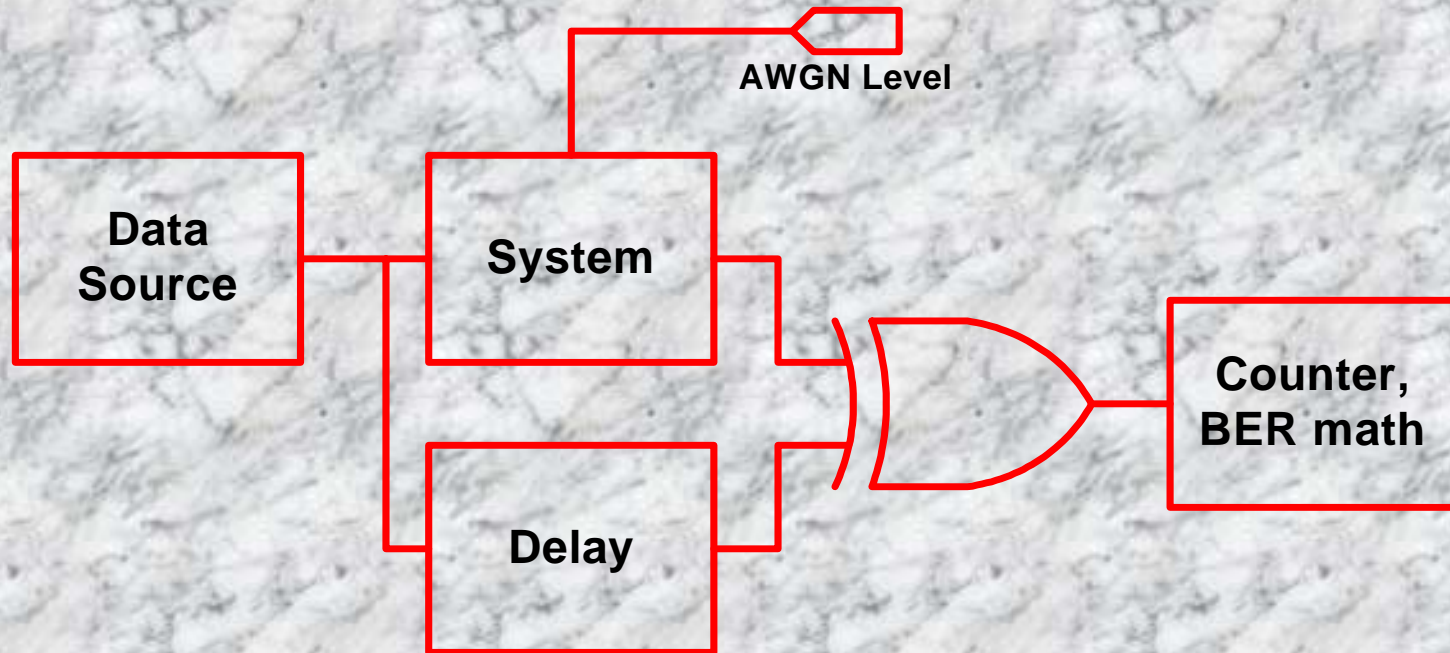


BPSK : Receiver



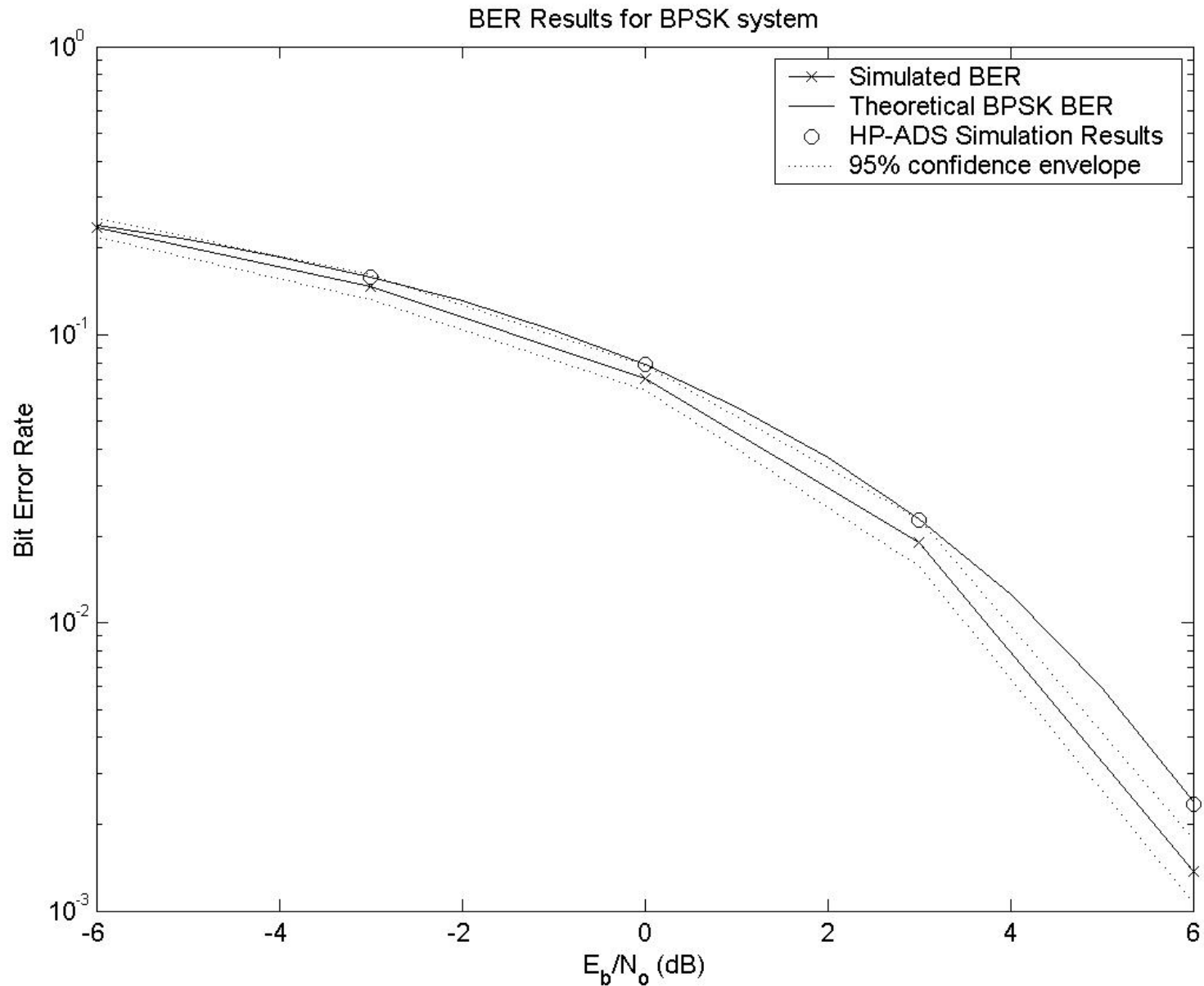
- LNA, Mixer, LPF, A/D

BER Calculation



- Good estimate of system performance in the presence of noise
- Used Monte Carlo method to measure BER

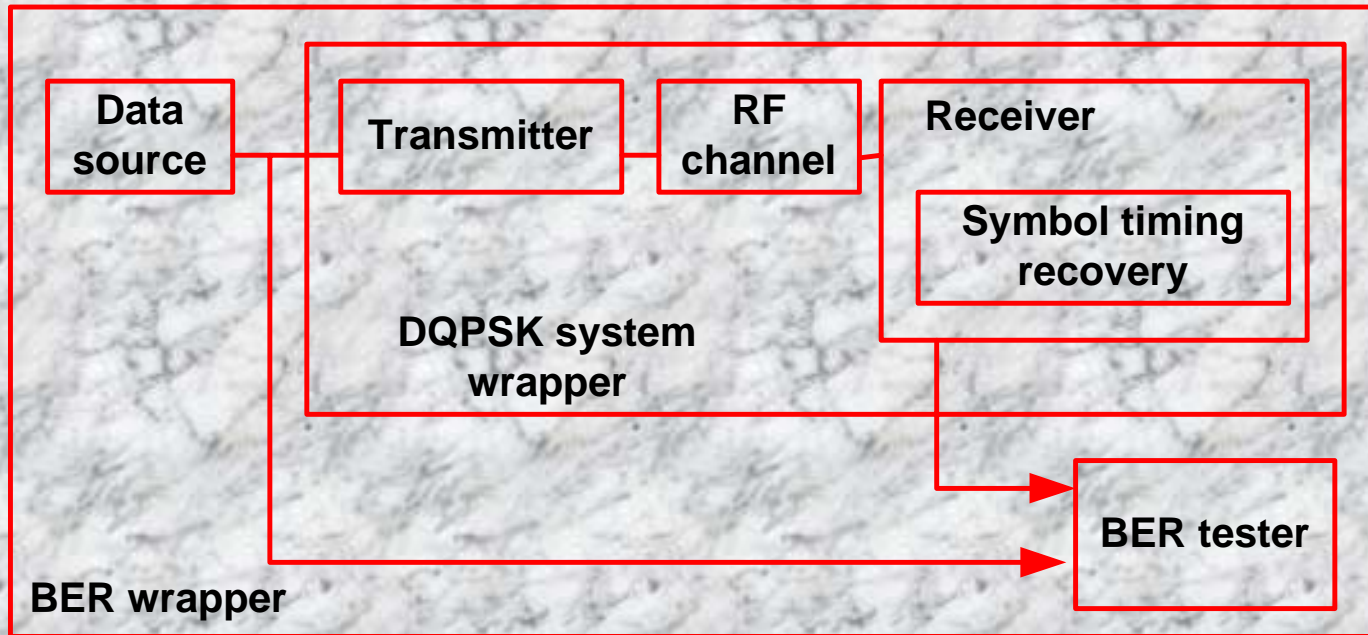
BPSK : Results



Outline

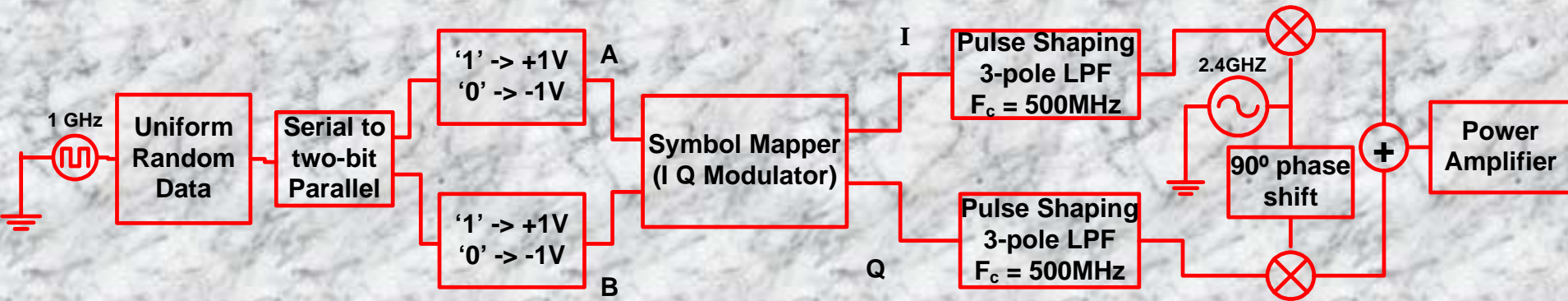
- Background and Motivation
- Simple BPSK Model
- $\pi/4$ DQPSK Model
- Summary and Conclusions

Basic $\pi/4$ DQPSK System



- Ideal System and Architecture
- Standard for US and Japanese cell phones
- Less complex receiver implementation
- Better spectral characteristics than QPSK, BPSK

$\pi/4$ DQPSK : Transmitter

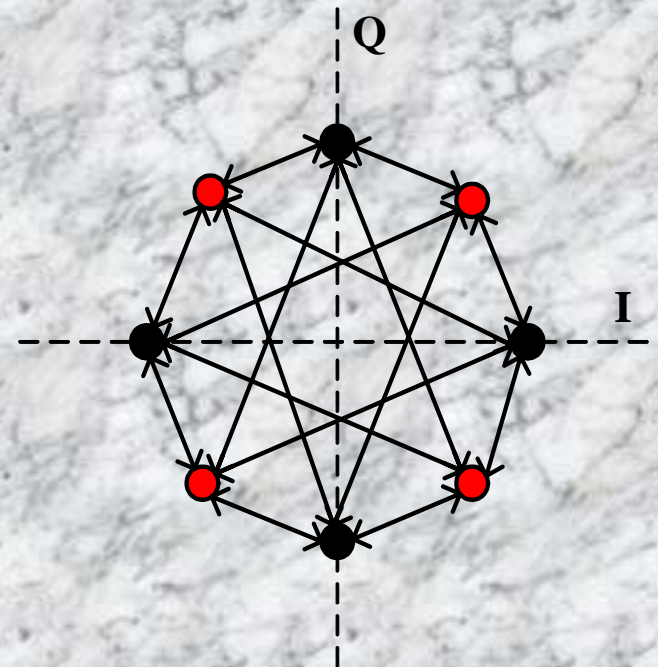


- 2-bit parallelize data (symbol)
- Map Symbols to phase change relative to previous symbol
- Pulse Shape filter
- Up-convert and amplify

Signal Constellation

- Directly map a pair of input bits onto relative phases ($\pm \pi / 4, \pm 3 \pi / 4$)

A_k	B_k	$\Delta \theta$
0	0	$\pi / 4$
1	0	$3 \pi / 4$
1	1	$-3 \pi / 4$
0	1	$-\pi / 4$



Symbol Mapping

When up-converted, each symbol can be represented as:

$$S_k = A \cos(\omega_c t - (\theta + \Delta\theta))$$

$$S_k = A \cos(\theta + \Delta\theta) \cos(\omega_c t) + A \sin(\theta + \Delta\theta) \sin(\omega_c t)$$

If we use I_k in-phase symbol:

$$I_k = A \cos(\theta + \Delta\theta)$$

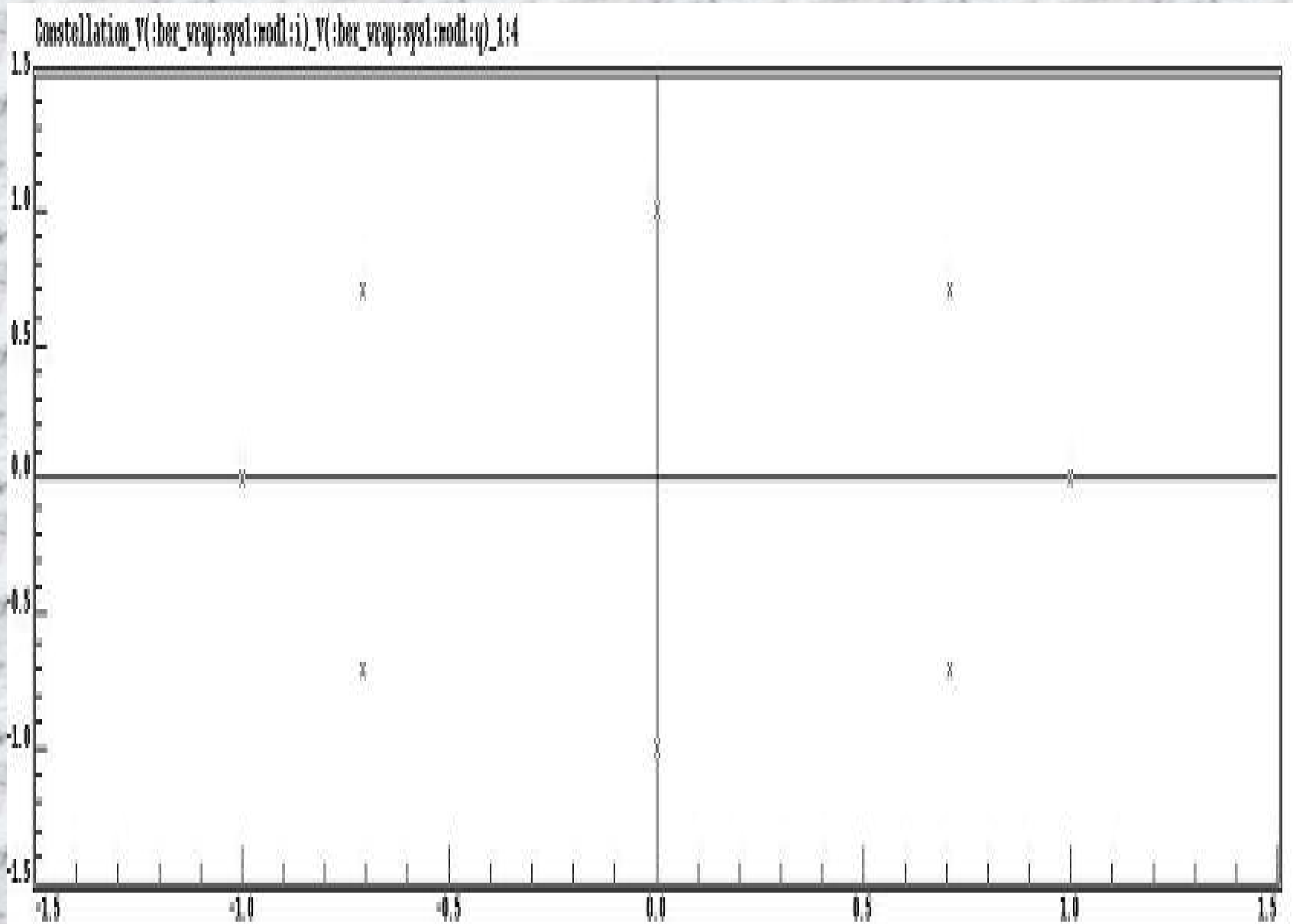
$$I_k = A \cos(\theta) \cos(\Delta\theta) - A \sin(\theta) \sin(\Delta\theta)$$

Similarly :

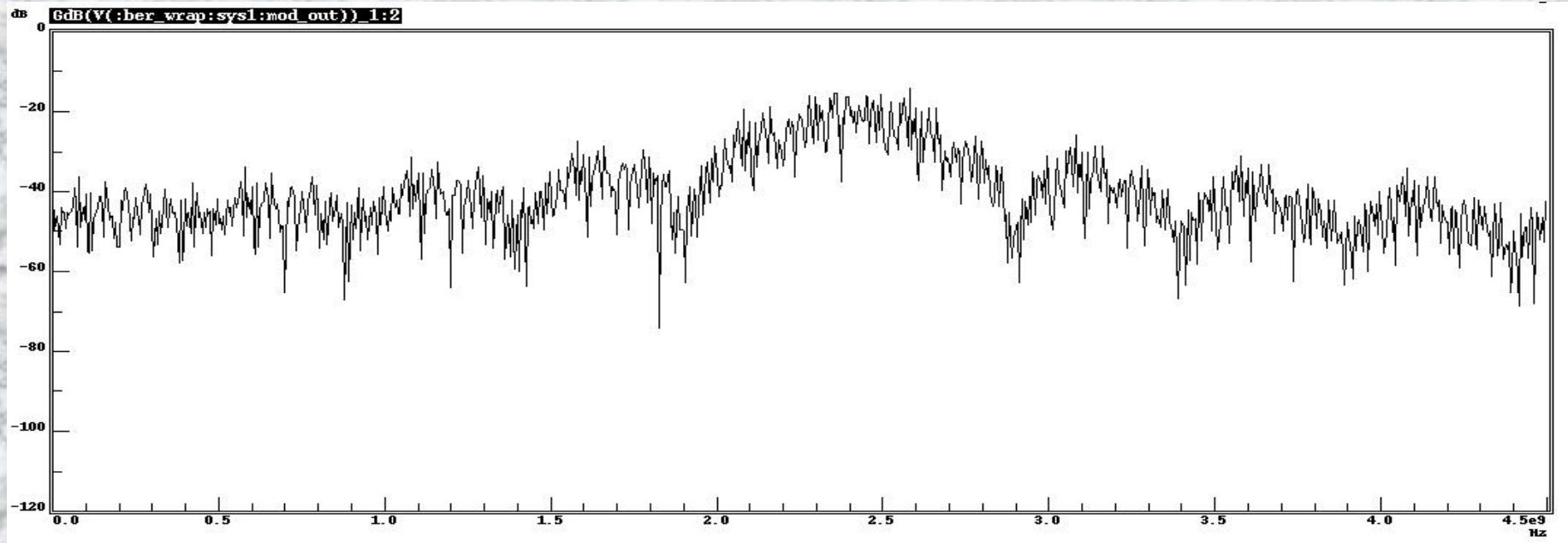
$$Q_k = A \sin(\theta + \Delta\theta)$$

$$Q_k = A \sin(\theta) \cos(\Delta\theta) + A \cos(\theta) \sin(\Delta\theta)$$

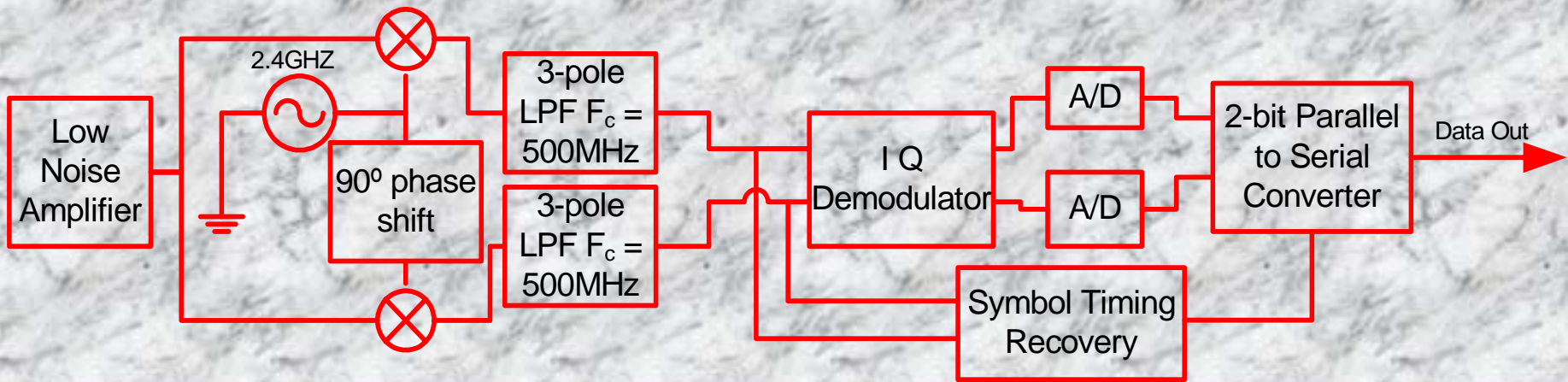
Transmitted Constellation



Transmitted Spectrum

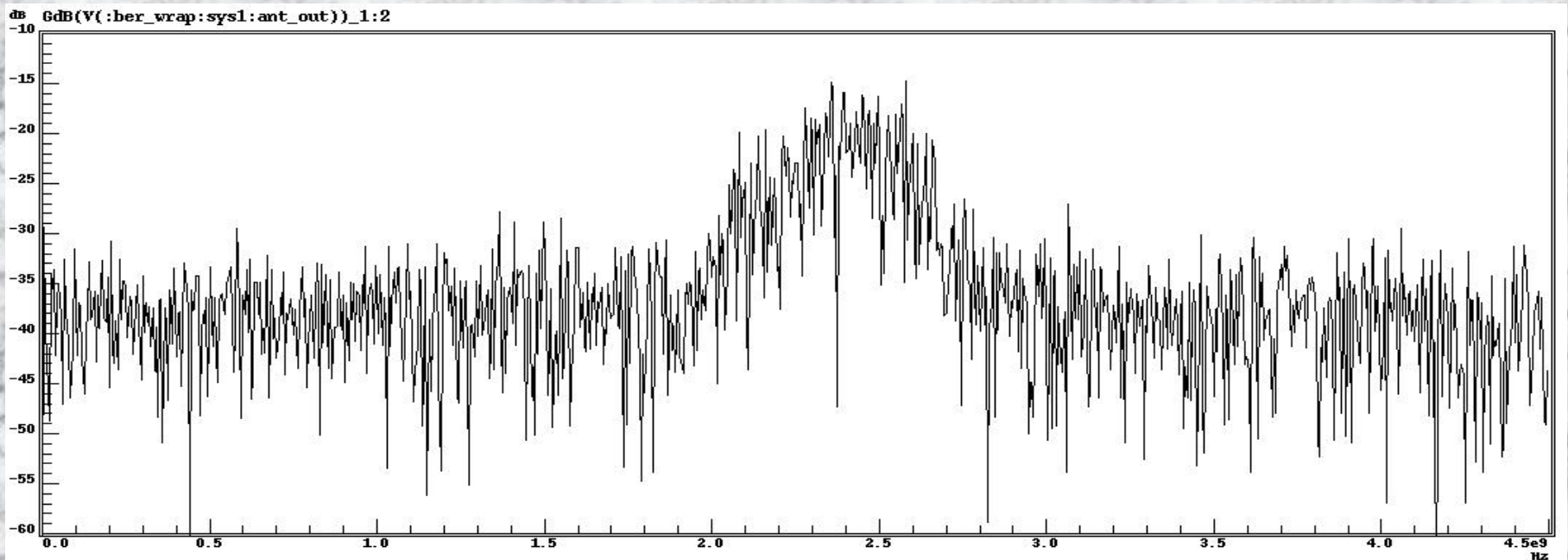
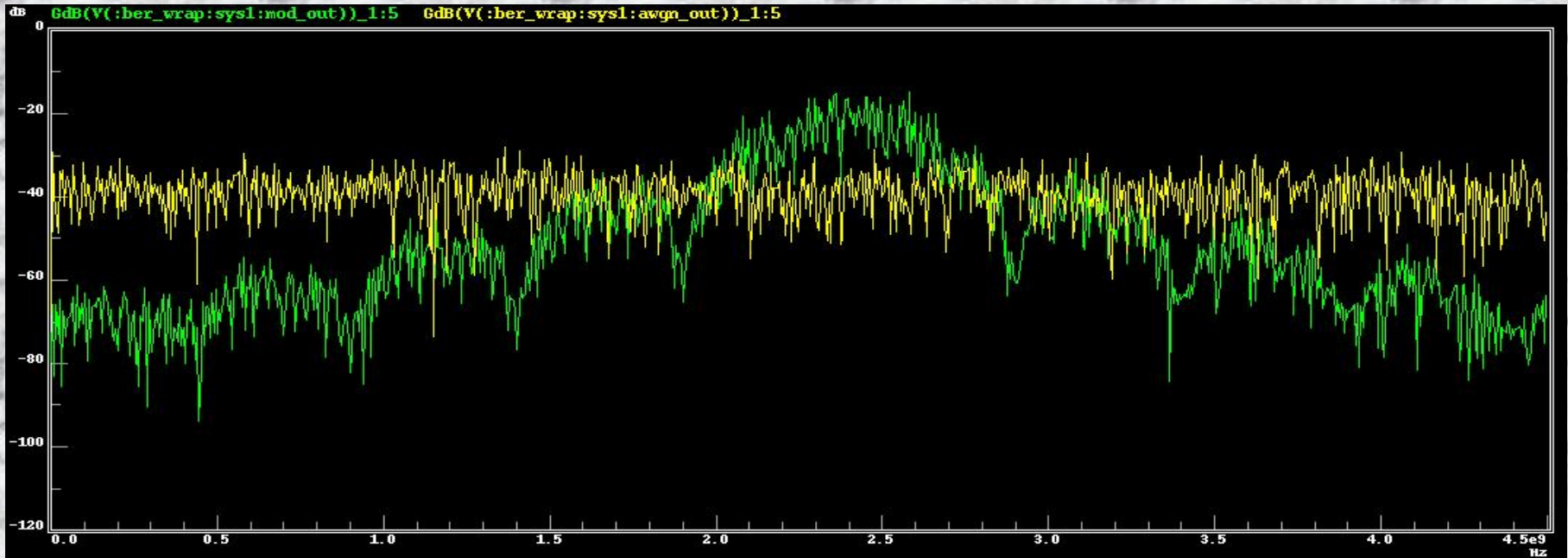


$\pi/4$ DQPSK : Receiver

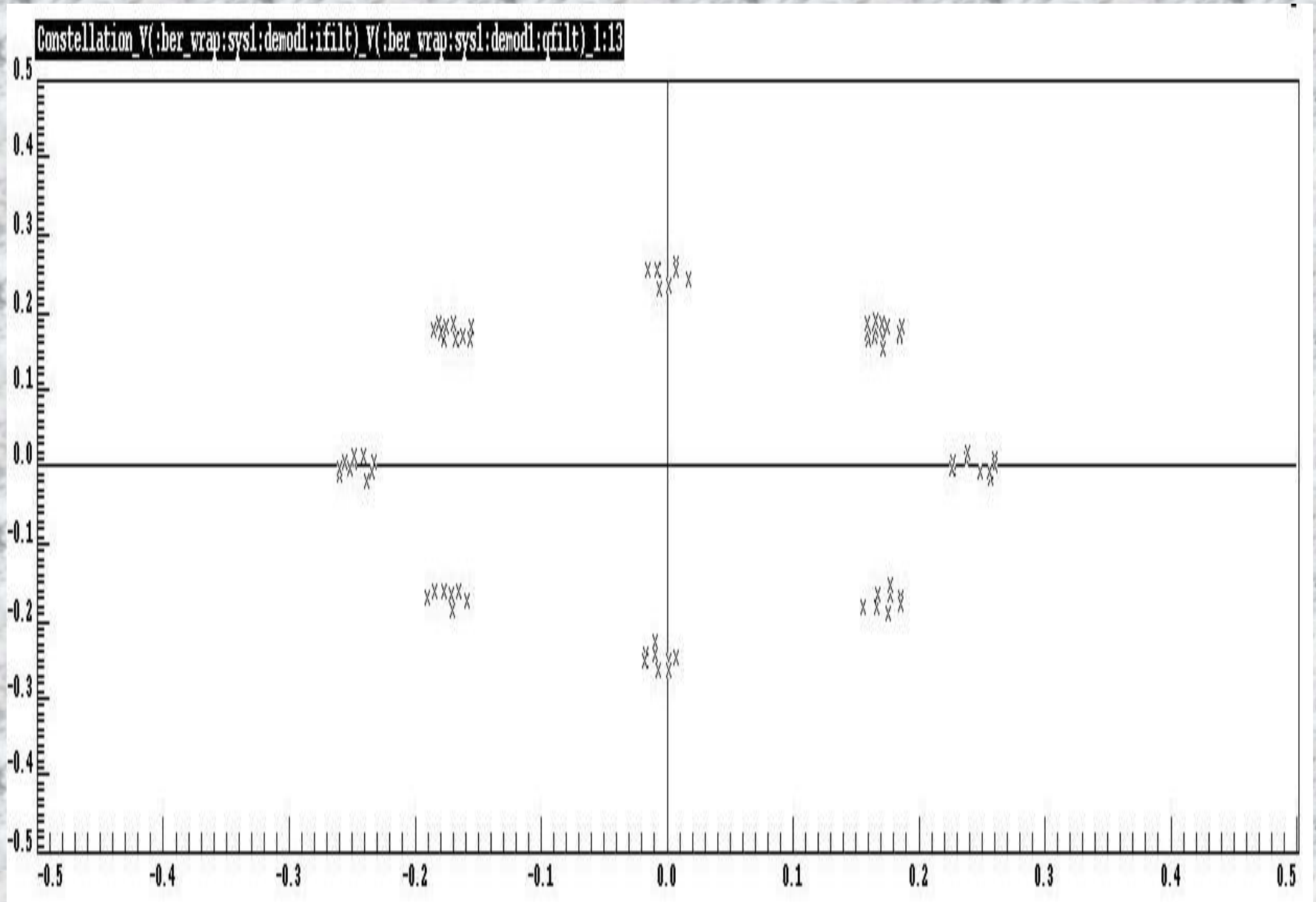


- Four Steps:
 - Amplify and down-convert
 - Filter
 - Demodulate and recover symbol clock
 - Digitize and Serialize

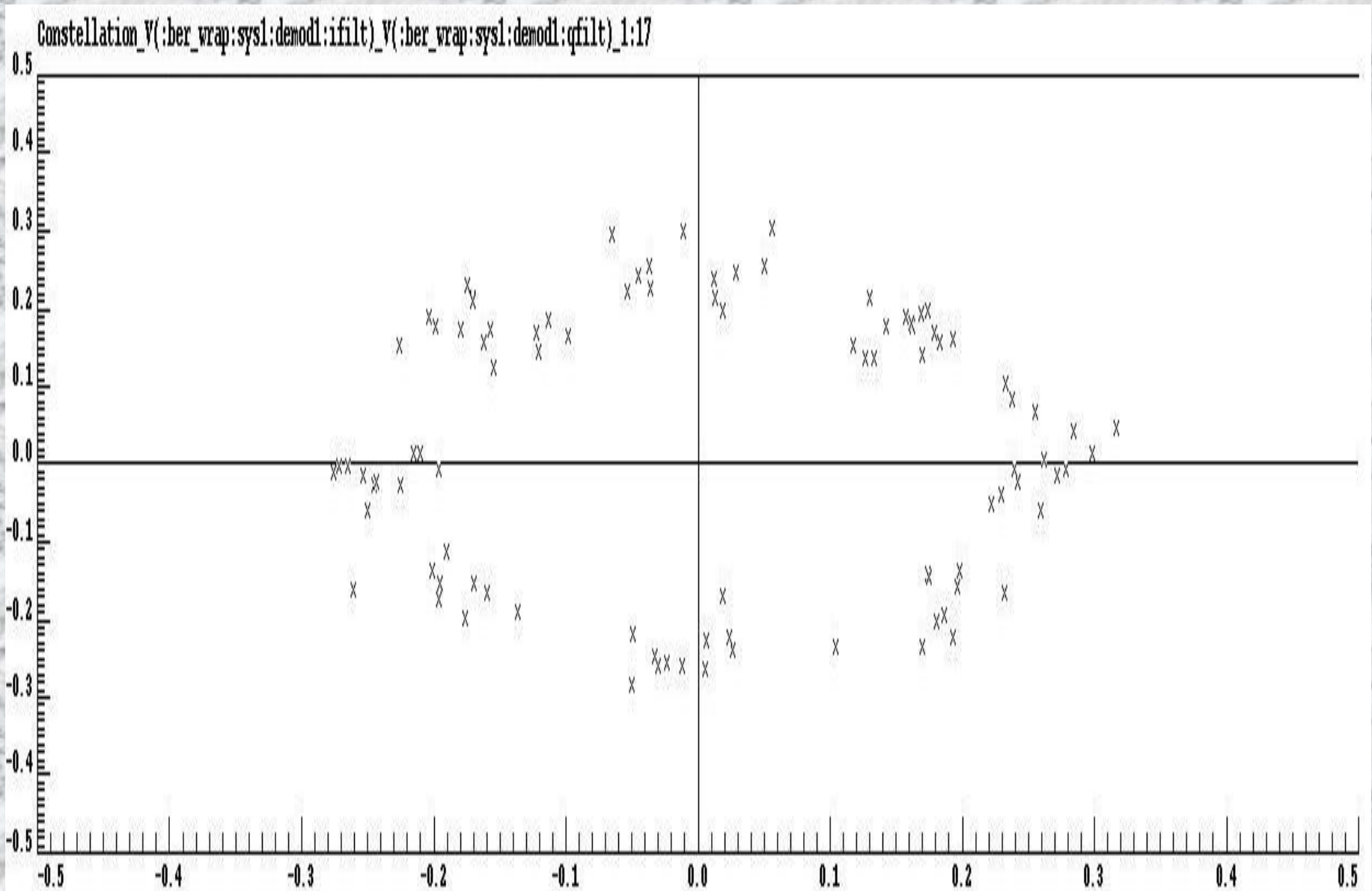
Received Spectrum



Received Constellation: Low Noise



Received Constellation: High Noise



IQ Demodulation

Estimation for the received A_k and B_k is using:

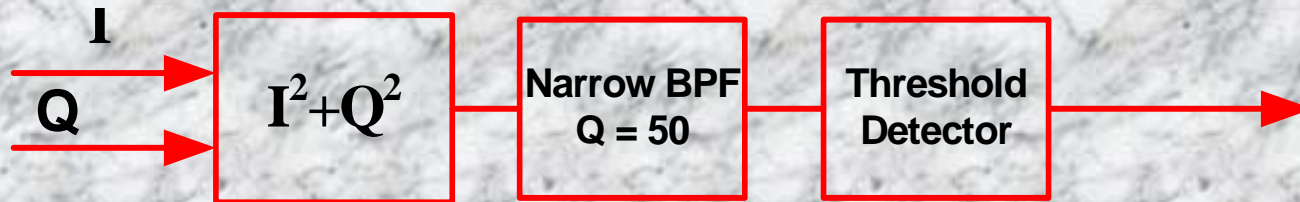
$$\text{sign}(\cos(\theta_k)) = \text{sign}(Q_k Q_{k-1} + I_k I_{k-1})$$

$$\text{sign}(\sin(\theta_k)) = \text{sign}(Q_k I_{k-1} - I_k Q_{k-1})$$

If $(\cos \theta_k > 0)$ then $A_k = 1$, else $A_k = 0$

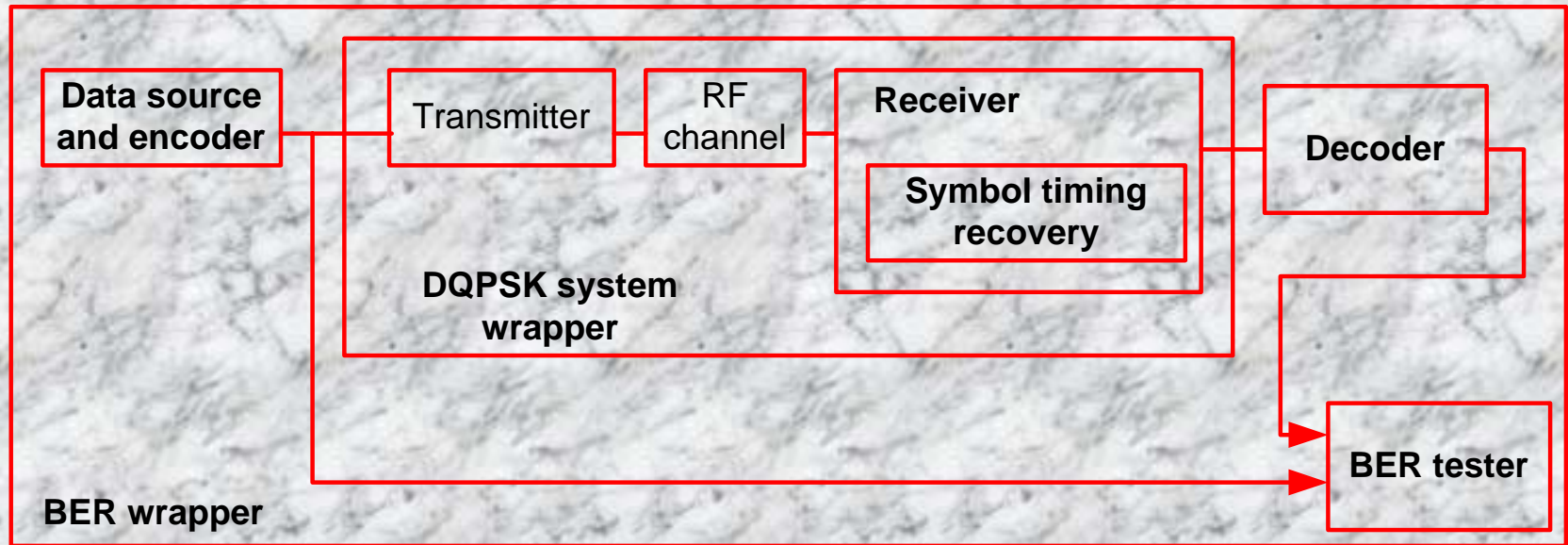
If $(\sin \theta_k > 0)$ then $B_k = 1$, else $B_k = 0$

Symbol Timing Recovery



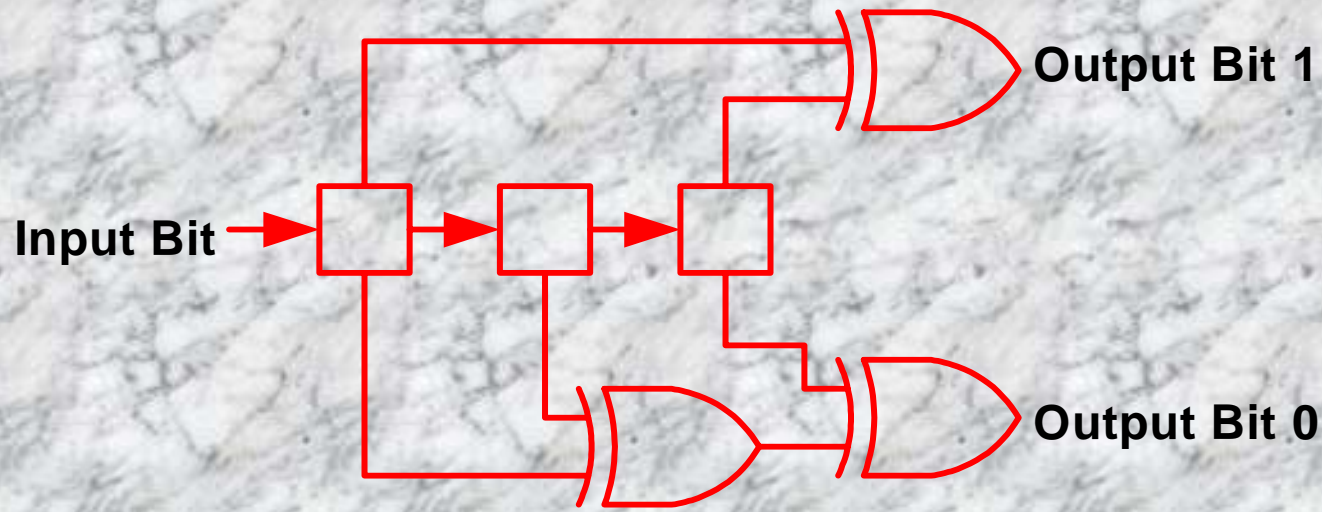
- Squaring and adding I, Q channels produces tone at symboling frequency
- High-Q BPF isolates tone
- Threshold detector creates `std_logic` clock at symbol frequency

$\pi/4$ DQPSK : Viterbi Encoder / Decoder



- Added a simple rate 1/2 Viterbi encoder
 - Decreases BER
 - Half clock rate and removal of serial to parallel conversion

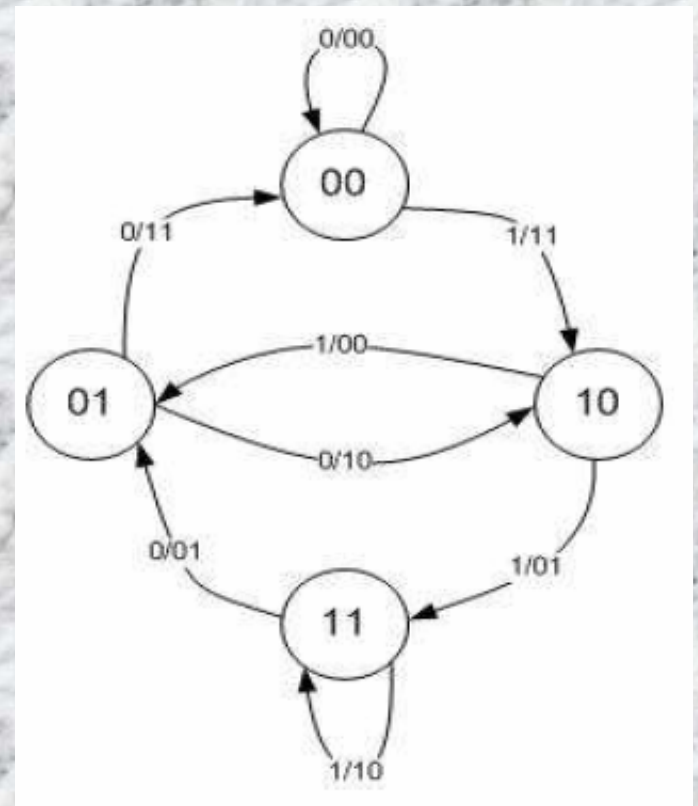
Viterbi Encoder



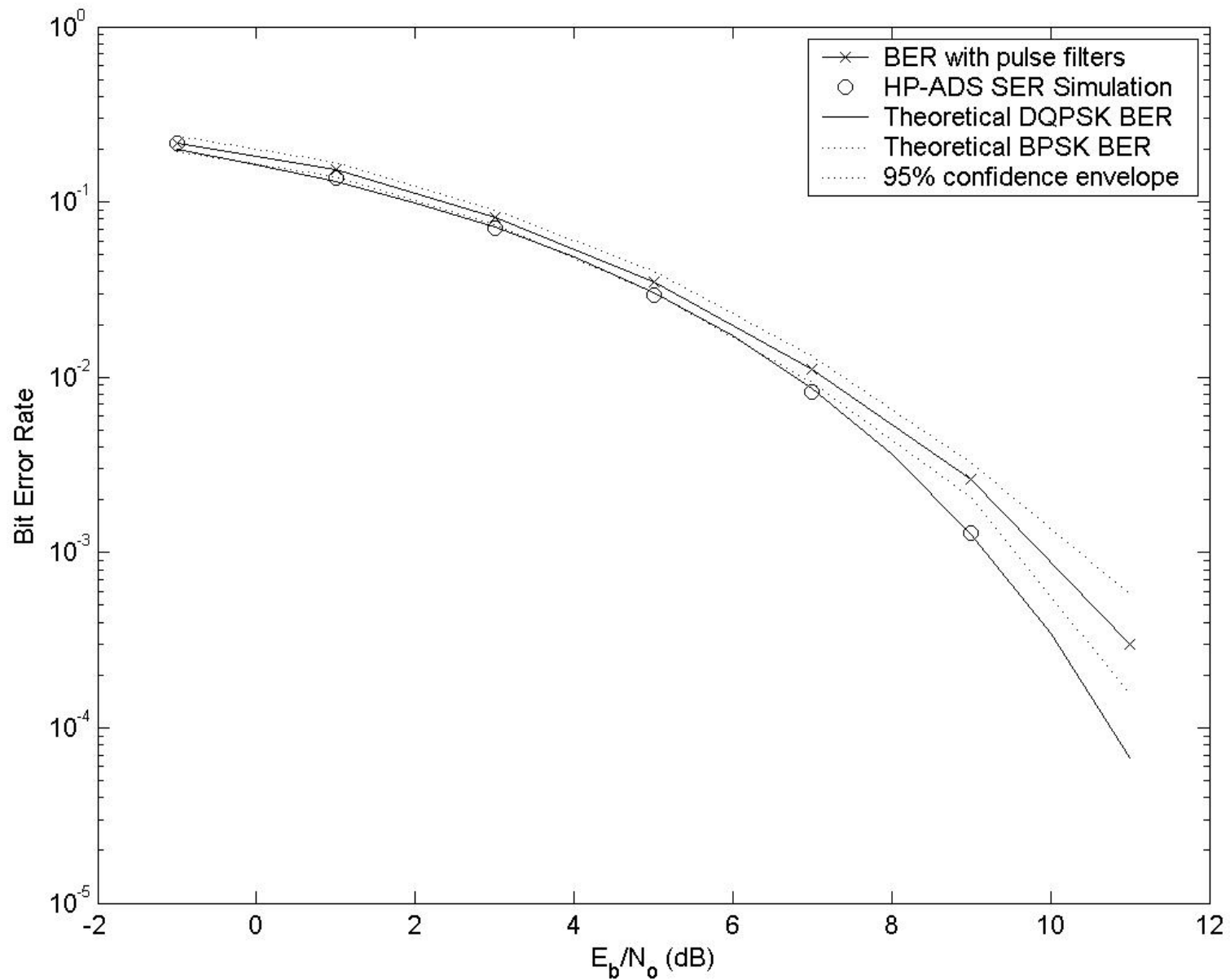
- Rate 1/2 encoder ($K=3$)
- Operates on 3 input bits and two bits from cleared register
- Produces specific 10 output bits

Viterbi Decoder

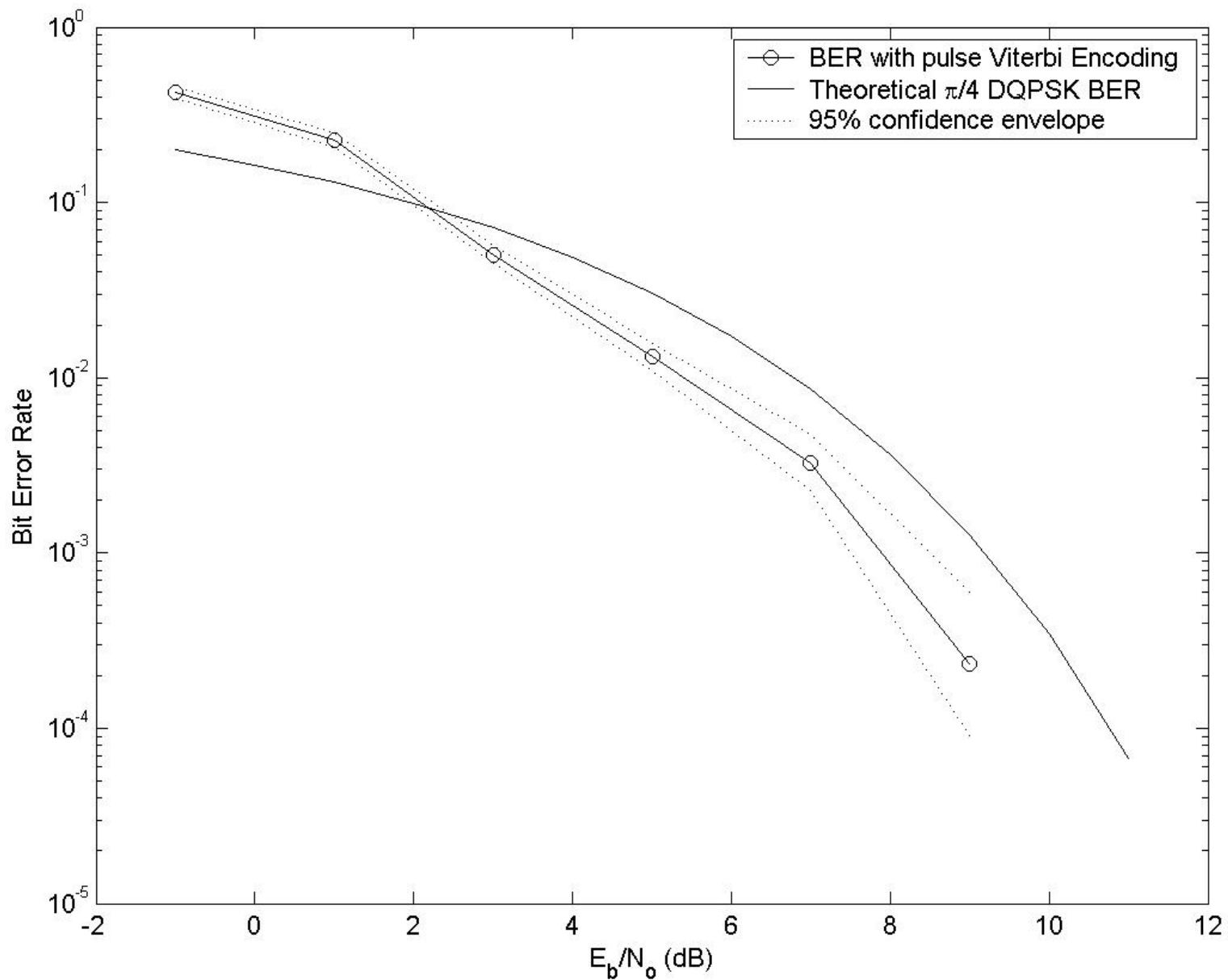
- Implemented as state machine
- Makes decision to get the max possibility correct bits
- Less complex but less error tolerant



BER DQPSK System without Encoder/Decoder



BER With Viterbi Encoder



Outline

- Background and Motivation
- Simple BPSK Model
- $\pi/4$ DQPSK Model
- **Summary and Conclusions**

Summary of Results

- Motivation of the project
- BPSK design example
 - Similar results to theoretical and HP-ADS
 - Verified noise modeling technique
- $\pi/4$ DQPSK design
 - BER closely matches Agilent ADS and theoretical curves
 - Increased model complexity with encoder / decoder
 - Verifies that complete system modeling can be easily performed in VHDL-AMS

How Ideal?

- Oscillator:

```
V==10**(A/20.0)*cos(math_2_pi*f*now + Ph);
```

- PA and LNA:

```
vo == vi*10**(gain/20.0);
```

- Mixer:

```
vout==v1 * v2;
```

Example

Transmit: 00 11 11 01

