Schedule of Deliverables for EE465

Created by Prof. Eric Salt and Prof. Ha Nguyen
in January 2015 for the first offering of EE465
and
Maintained by Prof. Eric Salt and Prof. Ha Nguyen
from January 1, 2015 to present

```
Revision Hisory:
Revised spring/summer 2015: Major revisions based on feedback
                            from the class of 2015.
Revised December 2016:      Major revisions based on feedback
                            from the class of 2016.
Revised Jan 3, 2017:  Corrected typos
Revised Jan 9, 2017:  Added the matlab code used in the
                      discussion for deliverable 1.
                      Also changed the length of RC and SRRC
                      filters used in deliverable 1.
Revised Jan 10, 2017: Revised method for steering discussions
                      for deliverable 1.
                      The original method of discussing a list
                      of fairly pointed questions that were provided
                      in advance did not seem to be working that well.
                      That has been changed to listing a few broader
                      topics for discussion.
Revised Jan 13, 2017: Added detail to the specification for
                      deliverable 1 so that it can be designed,
                      built and tested.
```

Acknowledgements

The creators of this document must thank the classes of 2014-2015 and 2015-2016 for their very considerate and well thought out suggestions for improving the class. The creators hold both classes, especially the class of 2014-2015, in high regard for being very understanding and maintaining a positive attitude while fighting through a class with organizational growing pains.

The creators would also like to thank Vecima Networks and in particular Dr. Brian Berscheid for their behind-the-scenes help.

Perspective

From an engineering project point of view, the objective is to design, build and test a 16-QAM modulator and 16-QAM demodulator for a CATV system in piece-by-piece fashion over 12 weeks. The course has a 3L-3P classification, which has the following student workload implications:

1. The students are expected to spend 3 hours per week in lectures (in this class *lectures* are *in-class discussions*).

2. The students are expected to spend 3 hours per week working on assignments.

3. The students are expected to spend 3 hours per week working in the laboratory and another 1.5 hours per week outside the laboratory (split between preparation and post-lab report writing).

The big picture implication is that the students are expected to spend about 7.5 hours per week outside of the lectures. Over a 12-week period this amounts to 90 hours of work outside the lectures.

This class is a design class and does not fit the standard class template where typically 12 assignments and 12 labs are issued at weekly intervals. Furthermore, the assignments and labs are bonded with the outcome being a few well-specified deliverables. The students are expected to spend 7.5 hours per week designing, building and testing circuits. The proportion of time spent in the lab to time spent outside the lab will vary from week to week.

The class has five milestones, each of which having predefined deliverables in the form of a working circuit. The timeline will vary from year to year, but the students will have to demonstrate that the circuit associated with each milestone works properly.

Project Overview

The general block diagram in Figure 1 provides an overview of the CATV communication system that will be designed and built. Its operation will be described in the lectures.

Deliverable 1

This deliverable was inserted at the request of the class of 2015-2016. They believed introducing/reviewing the material and tools needed to design, build and test the original deliverables (now deliverables 3 through 5) would significantly reduce the design, build and test time and, in the end, save the students time. Many in the class of 2015-2016 stated they had difficulty getting started and did not do much, if anything at all, in the first three weeks of the class. They strongly suggested moving the workload of the class forward by introducing a lab exam of some sort as an early deliverable.

Deliverable 1 is a lab type exam that will be held on the third (or possibly the fourth) week of classes. It will have two components:

1. Contributions made during the in-class discussions. A list of topics for the in-class discussions will be provided by the instructor. The topics may be posed as questions. Students will be chosen to lead the discussion for a segment of each lecture. Once all the students have lead a segment of a lecture a grade for "in-class" discussion will be assigned and will make up part of the deliverable 1 lab grade.

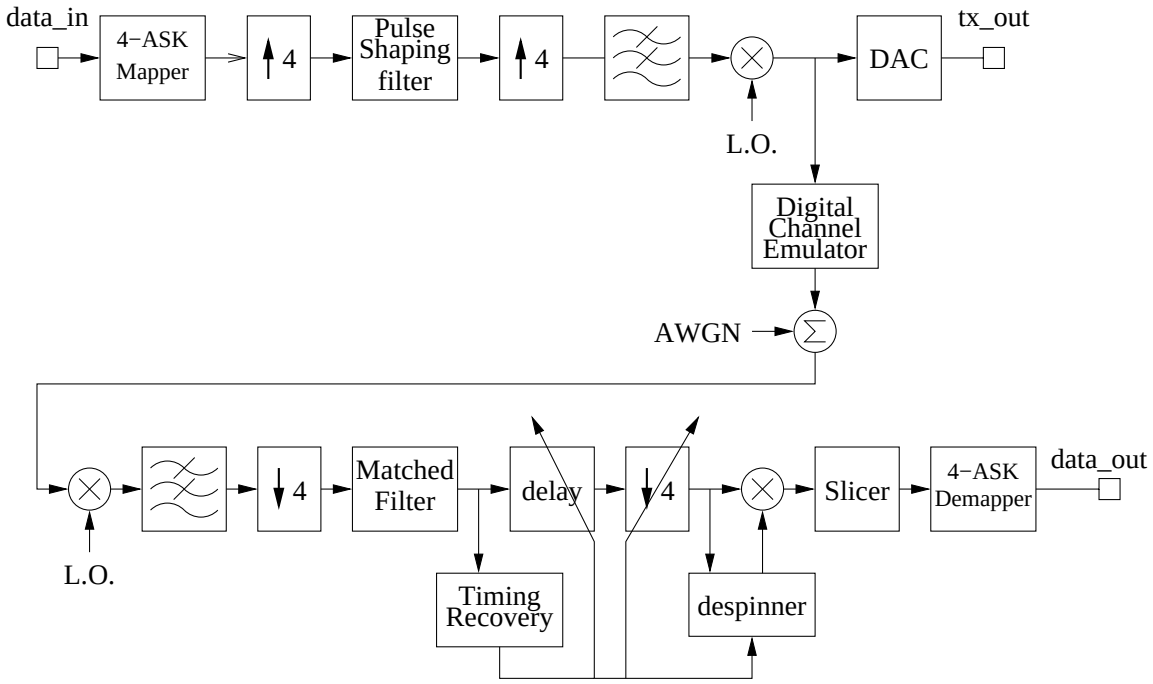


Figure 1: A high level block diagram for one channel (in-phase or quadrature) of a 16-QAM CATV communication system.

2. Performance of the circuit built by the student. The students will be asked to modify the circuit they were required to design, build and test and then demonstrate the modified circuit meets the specification to the lab instructor.

The students will design, build and test two length-17 square-root raised cosine (SRRC) filters: one for the transmitter and the other for the receiver. The transmit filter should be constructed with the lab exam in mind where the students could be asked to adjust the filter coefficients (through parameter selection and windowing) to sacrifice inter-symbol interference (ISI) for improved (i.e., less) out-of-band power.

The students are also required to build a multiplierless transmit SRRC filter suited for 4-ASK (i.e., in-phase component of 16-QAM).

The list of deliverable are:

1. One RCV SRRC filter (see specification below).

2. One TX SRRC filter that uses multipliers (see specification below).

3. One TX SRRC filter that does not use multipliers (see specification below).

The design must meet the specification given below. The quality of the design will be based on the MER measured at the output of the RCV filter.

Specification for Deliverable 1

1. Two length 17 Square Root Raised Cosine (SRRC) filters are to be built: One called the TX filter, which is the filter in the transmitter, and the other called the RCV filter, which is the filter in the receiver.
2. All coefficients in the filters must be less than or equal to 18 bits.
3. The input and output for each of the two filters can be at most 18 bits.
4. The sampling rate for both filters is $N_{sps} = 4$ times the symbol rate. The subscript `sps` in N_{sps} signifies samples-per-symbol.
5. The RCV filter is to have a roll-off factor of $\beta = 0.25$.
6. The impulse response of the RCV filter is to be the infinite SRRC response truncated to a length of 17. Just to be clear, the center 17 coefficients of the infinite impulse are used so in effect the infinite impulse response is rectangularly windowed about its center value.
7. The TX filter is limited to a length of 17. There is no restriction on the coefficients, although in the end they are sure to resemble those of a SRRC filter with a roll-off of 0.25.
8. The stop band of the TX filter starts at 0.2 cycles/sample and runs to 0.5 cycles/sample.
9. The magnitude response at all frequencies in the stop band of the TX filter must be 62 dB below the DC response.

The details of the lab exam that will test the quality of the design and the ability of the students to change their designs will be discussed in class.

Lesson Plan for Deliverable 1

The lesson plan for the in-class discussion for deliverable 1 is given below.

First it is pointed out that the first three-hour lab slot after the first lecture time will be treated as 3 hours of lecture time and used for in-class discussions. This means all students are required to attend the first 3-hour lab slot after the first lecture time. The room used for the in-class discussion that occupies the lab slot will be announced in class.

In the first lecture class a Matlab script will be provided. That matlab script will be expanded and modified to help answer questions that arise in the discussions.

The Matlab script provided for the first lecture generates the coefficients for a finite-length raised cosine (RC) filter with a 6 dB bandwidth of $F_s/(2N_{\text{sps}})$ and also the coefficients for a finite-length SRRC filter with a 3 dB bandwidth of $F_s/(2N_{\text{sps}})$, for $N_{\text{sps}} = 4$. Here N_{sps} means the number of samples per symbol.

This Matlab script also generates the theoretical power spectra for the infinite-length RC and SRRC filters.

Students will be selected to lead the discussion for a segment of the lecture. Depending on the topics and how easy it will be to transition from one lead student to another lecture can be broken into three segments, each of which will be lead by a different students. These lead students are to act mainly as moderators of class discussion making sure everyone who would like to get involved in the discussion gets a chance to do it.

If class decides it needs some graphical or numerical information to be generated by a Matlab script, then, providing script can be written quickly, it will be written in class by the lead student and the results displayed on the screen. If the script is difficult to write or is too lengthy, then with the support of the class the lead student may request the script be written after class by the instructor or the teaching assistant and have the results displayed in the next lecture.

The Matlab script that will be used and perhaps expanded during the discussions for deliverable 1 is given below:

```
clear all

% parmeters for the filters
 N_sps = 4;   % number of samples per symbol
 beta = 0.25; % roll off factor, script requires
              % beta be less than 1
 N_rc = 33;   % length of the impulse response
              % of the raised cosine filter
 N_srrc = 17; % length of the impulse response of the
              % square root raised cosine filter
 F_s = 1;     % sampling rate in samples/second
 f_6db = 1/2/N_sps; % 6 dB down point in cycles/sample
 F_6db = F_s * f_6db; % 6 dB down point in Hz

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time and frequency vectors
%
df = 1/2000;   % frequency increment in cycles/sample
f = [0:df:0.5-df/2]; % cycles/sample; 0 to almost 1/2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  magnitude response for RC and SCCR filters
Hrc_f = zeros(1,length(f)); % reserve space for
           % magnitude response of rc filter
Hsrrc_f = zeros(1,length(f)); % reserve space for
           % magnitude response of srrc filter
f1 = find(f < f_6db*(1-beta)); % indices where
                          % H_f = 1
f2 = find( (f_6db*(1-beta)<= f) & ( f <=...
          f_6db*(1+beta))); % indices where
                          % H_f is in transition
f3 = find(f > f_6db*(1+beta)); % indices where
                          % H_f = 0
Hrc_f(f1) = ones(1,length(f1));
```

```
Hrc_f(f2) = 0.5+0.5*cos(pi*(f2-f2(1))/(length(f2)-1));
Hrc_f(f3) = 0;

Hsrrc_f = sqrt(Hrc_f);

figure(1);
     plot(f,Hrc_f,'r', ...
          f,Hsrrc_f,'--b','LineWidth',2);
     xlabel('frequency in cycles/sample')
     ylabel('|H_{rc}(e^{2\pif})| and |H_{srrc}(e^{2\pif})|')
     grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% find and plot the impulse responses

h_rc=firrcos(N_rc-1,F_s/8,beta,F_s,'rolloff');
    % impulse response of rc filter
h_srrc=firrcos(N_srrc-1,F_s/8,beta,F_s,'rolloff','sqrt');
    % impulse response of rc filter

figure(2)
plot(0:N_rc-1,h_rc,'r*', ...
     0:N_srrc-1,h_srrc,'bd', 'MarkerSize',8);
ylabel('h_{rc}[n] and h_{srrc}[n]');
xlabel('n');
grid;

% Find and plot the frequency repsonses of the
% finite length RC and SRRC filters

H_hat_rc = freqz(h_rc,1,2*pi*f);
H_hat_srrc = freqz(h_srrc,1,2*pi*f);

figure(3)
plot(f,abs(H_hat_rc),'r', ...
     f,abs(H_hat_srrc),'--b','LineWidth',2);
ylabel('H_{hat}(\Omega) for RC and SRRC');
xlabel('\Omega');
grid;
```

The questions/topics of discussion for deliverable 1 are given below:

1. At the transmit end of a communication system one symbol is generated every N_{sps} samples (for this discussion $N_{\text{sps}} = 4$ samples per symbol), with zeros inserted between symbols. Of course this sample sequence is filtered and then transmitted and then received and then filtered again. The filters must be designed so that every Nsps sample of the output of the filter in the receiver is the data that was transmitted.

   Why is this so? Why are two filters used? Why do you need a filter in the transmitter? Do the filters need to be special in some way?

2. The impulse response of a raised cosine filter is special. What is special about it? The impulse response is given below

$$h[n] = \begin{cases} \frac{\pi}{4}\text{sinc}\left(\frac{1}{2\beta}\right), & n = \pm\frac{N_{\text{sps}}}{2\beta} \\ \text{sinc}\left(\frac{n}{N_{\text{sps}}}\right)\frac{\cos\left(\frac{\pi\beta n}{N_{\text{sps}}}\right)}{1-\left(\frac{2\beta n}{N_{\text{sps}}}\right)^2}, & \text{otherwise,} \end{cases}$$

   where β is the roll-off factor.

3. What is critical about the 6 db bandwidth of the cascade of the transmit and receive filters and how does that relate to N_{sps}?

4. The Fourier transform of the product of two sequences is $1/(2\pi)$ times the circular convolution of the Fourier transforms of the two sequences with the interval of the circular convolution being 2π. What does that mean? What is the graphical interpretation of circular convolution? The Matlab function `cconv(x,y,N)` computes the circular convolution of two sequences. Can this be used to find the convolution of two continuous waveforms? Is `cconv(x,y,N)` times the sample spacing (sample spacing could be Δt or Δf depending on the domain in which the convolution is taken) a Reimann approximation to an analog convolution?

5. If the impulse response of filter is multiplied by another sequence, say a window of some sort, what happens in the frequency domain. To start the discussion suppose the impulse response of an ideal low pass filter with bandwidth $F_s/8$ is multiplied by the Fourier transform of the positive half of a cosine that has zero crossings at $\pm F_s/16$. Is the resulting sequence the impulse response of a raised cosine filter, if so what is the roll-off factor.

   Will every 4th sample in the resulting impulse response be zero?

   Will every 4th sample in the result be zero for any type of window?

6. If an RC filter is windowed by a Hamming window to make the impulse response finite, say a length of 33, will every 4th sample still be zero? Why or why not?

   What does windowing do the transition band and what happens to the out-of-band power, i.e., the stop band power.

7. In theory an RC filter is equivalent to two SRRC filters in cascade. Are two length-17 rectangular windowed SRRC filters in cascade the same a one length 33 RC filter? What is the difference?

8. Define inter-symbol interference. Can it be measured from the impulse response? Is the rms level of ISI proportional to the amplitude of the signal?

9. Define MER. Is it a better measure of filter performance than ISI? Is MER, expressed in dB, given by

$$10\log\frac{\mathrm{P_{rcvd\_data}}}{\mathrm{P_{ISI}}}?$$

   What is $P_{\mathrm{rcvd\_data}}$ w.r.t. P_{ISI} in dB?

10. What is an easy way to measure the ISI generated by two length-17 SRRC filters in cascade? Will windowing the transmit filter, say by a hamming window, increase the system ISI? Is so why is the transmit filter windowed.

11. Can the ISI caused by windowing filter be minimized by adjusting the roll-off factors and/or the 3 dB bandwidths in the SRRC filters in the transmitter and receiver. If so, how would one find the optimum adjustments?

12. What is an eye diagram and could one be constructed in Matlab? Can an exact value of ISI be obtained from an eye diagram? Can an appreciation for the amount of ISI in the system be gleaned from an eye diagram?

13. After filters are constructed in Verilog, their impulse responses need to be verified. The filters can be quite long so hand checking the value of $h[n]$ for every n where $h[n] \neq 0$ can be tedious. Is there any easy way to check the impulse response for typographical errors (not to verify it meets spec). Would looking at the output to a DC input be helpful? Would looking at the output to an input of alternating $+1$ and -1 be useful? Could such a quick check be implemented in ModelsimAltera. Could such a quick check be implemented in hardware using SignalTap.

14. Is it possible to import data from ModelsimAltera to Matlab? If so could that be helpful in debugging filters described in Verilog?

15. 16-QAM is generated using two 4-ASK signals that are filtered with separate SRRC filters, up-converted with quadrature sinusoids and then combined at IF. The input to each of the two filters in the transmitter consists of a data sample followed by $N_{\mathrm{sps}} - 1$ zero samples. This means the output of just one of every N_{sps} multipliers is non-zero.

    Is there any way to share one multiplier among N_{sps} taps? Is there any way to reduce the number of registers used in the filter by a factor of N_{sps}?

16. Since the input to each of the two SRRC filters in the transmitter is a 4-ASK signal, the inputs can have only 1 of 4 values, which, for example, could be -3, -1, 1, or 3 every fourth sample. Is there any way to design the SRRC filter in the transmitter so that no multipliers are used, i.e., build a multiplierless filter.

Deliverable 2

This deliverable took effect in the 2015-2016 year. It has been included at the request of the class of 2014-2015 and most notably Conor Kerslake.

The students are given a working pulse shaping filter and a matched filter. They have to build the necessary circuits to measure the MER. They also have to look at the output of the pulse shaping filter on a spectrum analyser and measure the ratio of the out-of-band power to the transmitted signal power in the critical bands, referred to as OB1, OB2 and Adjacent Channel 2.

A block diagram showing how the tandem of the pulse shaping filter and the matched filter are to be tested is given in Figure 2.

The circuits that have to be built and the tasks that have to be done in preparation for measuring the MER are given below:

1. Design, build and test a clocking system that allows circuits to be clocked at one of the three rates: 25 Msamples/second, 6.25 Msamples/second and 1.5625 Msamples/second.

   There are two ways to design multi-rate clocking systems: One way is to clock all registers with the high rate clock and use the clock enable input on the register to control the rate at which it is being clocked. The other way is to generate three different clocks that run at three different frequencies and use the appropriate clock to clock each register.

   The first method consumes more power and generates more heat so is generally avoided in industry. However, it is more straight forward and flexible so is recommended for the design done in this class. A circuit that can be used for both methods is shown in Figure 3. For this lab the task is to build a circuit that generates:

   (a) a clock called `sys_clk` that is half the frequency of input clock, `clock_50`, as defined in Figure 3.

   (b) a 4-bit sequence called `clk_phase` that indicates the phase of the symbol clock, i.e., `sym_clk` as defined in Figure 3.

   (c) enables for the symbol clock and sample clock called `sym_clk_ena` and `sam_clk_ena`, respectively, as defined in Figure 3.

2. Design, build and test a maximum length 22-bit LFSR.

3. Design, build and test a circuit that estimates the average magnitude of the decision variable. Since the optimum reference level for the slicer is related to the average magnitude of the decision variable by a multiplicative constant, the slicer reference level can be set from an estimate of the average magnitude of the decision variable.

4. Design, build and test a circuit that estimates the average error in the decision variable.

5. Design, build and test a circuit that estimates the average power of error in the decision variable.

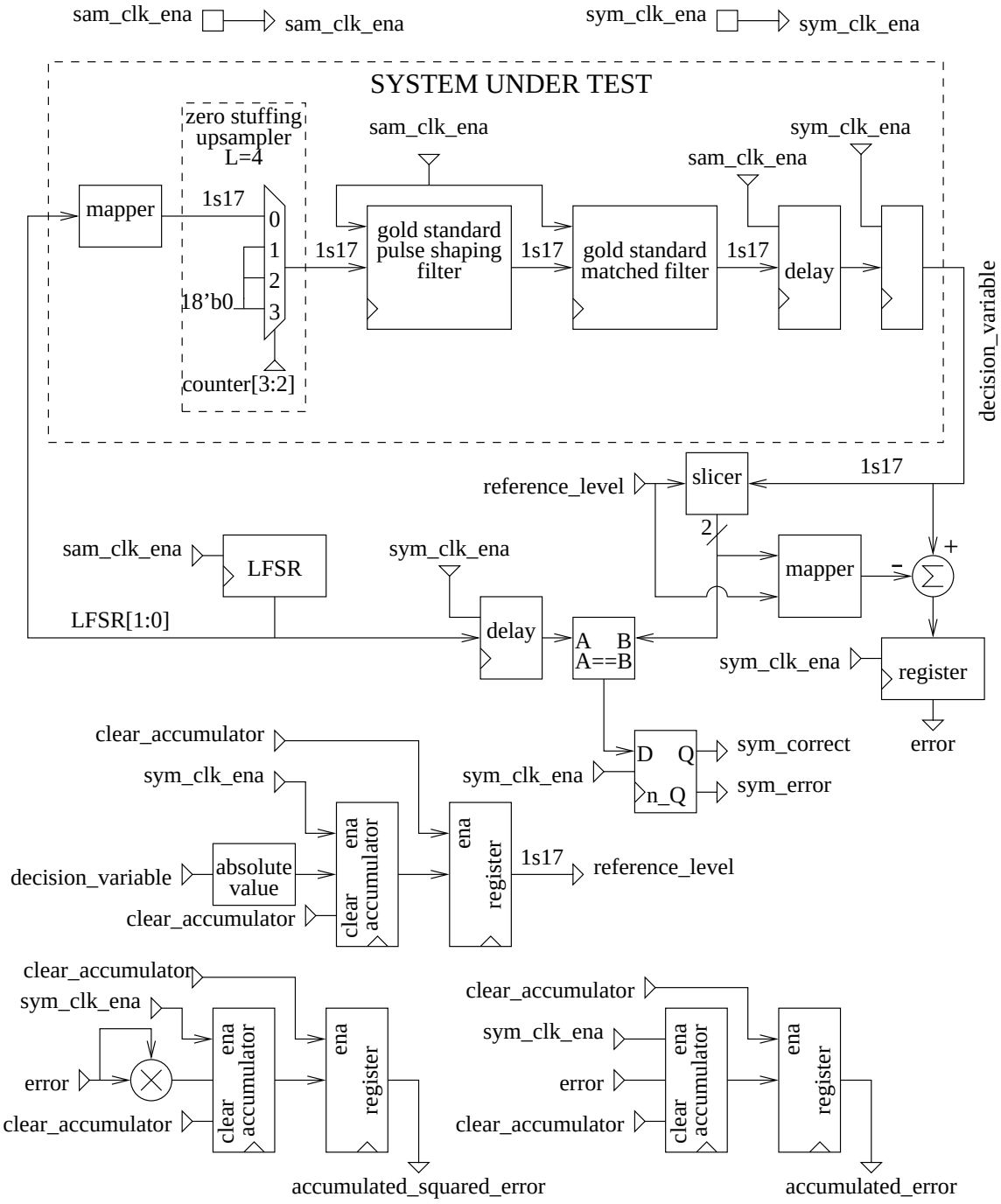
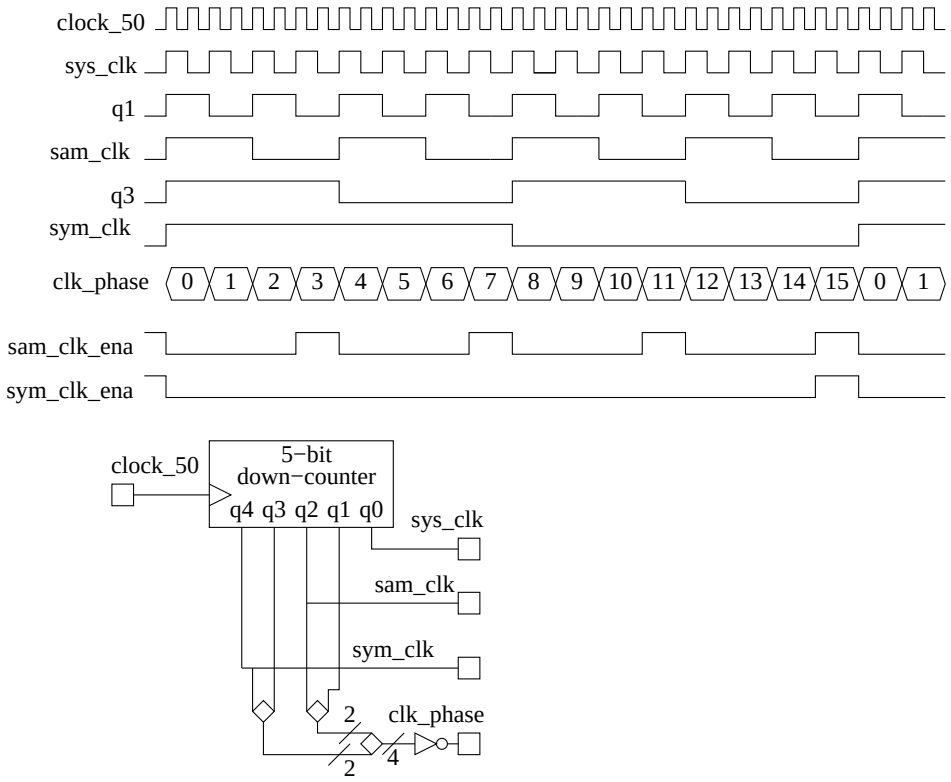


Figure 2: The circuits and system used as a reference in the measurement of the performance of a practical QAM transmitter and receiver.




```
# the .sdc file should contain the following clock constraints
create clock −name "input_clock" −period 20.000n [get_ports clock_50]
create_generated_clock −divide_by 2 −source [get_ports clock_50] −name sys_clock  [get registers {sys_clk}]
create_generated_clock −divide_by 8 −source [get_ports clock_50] −name sam_clock  [get registers {sam_clk}]
create_generated_clock −divide_by 32 −source [get_ports clock_50] −name sys_clock  [get registers {sym_clk}]
derive_pll_clocks −create_base_clocks
derive_clock_uncertainty
```

Figure 3: A circuit for generating clocks directly or enables to be used with the system clock.

This estimate together with the estimate of the average error can be used to estimate the MER due to inter-symbol interference (ISI). The ISI power is given by the average power in the error minus the average value of the error squared.

The MER can be calculated in a variety of ways. One of which is

$$\text{MER} = 10\log\left(\frac{\text{average power in decision variable}}{\text{ISI power}}\right)$$

Deliverable 3

The deliverables are:

1. A gold standard for the pulse shaping filter for a CATV 16-QAM modem.
2. A gold standard for the matched filter for a CATV 16-QAM modem.
3. A practical cost effective pulse shaping filter.

The specifications for the three deliverables are given below.

Specifications of the Gold Standard Pulse Shaping Filter

1. The pulse shaping filter must run at 4 times the symbol rate, i.e., 4 samples per symbol. This means the clock used for the filter has a rate 4 times that of the symbol clock, The clock used to clock the filter is referred to as the sampling clock and is denoted `sam_clk`.
2. The sampling rate is 1/4 the rate of the system clock. The system clock, referred to as `sys_clk`, is to run at 25 Msamples/second and the sampling rate is to be 6.25 Msamples/second.
3. The modulation must be 16-QAM.
4. The nominal roll-off factor for the pulse shaping is $\beta = 0.12$
5. The channel bandwidth at RF is $\frac{(1+\beta)}{\texttt{samples/symbol}} \times \texttt{sampling rate} = \frac{(1+0.12)}{4} \times 6.25$ Msam/sec = 1.75 MHz
6. The out of band attenuation must meet certain specifications in the bands defined in Figure 4.
   (a) The power transmitted in either of the two OB1 bands, which are the 220 kHz bands that border the signal channel, must be at least 58 dB below the power in the signal. The signal power is the total power transmitted in the signal channel.
   (b) The power transmitted in either of the two OB2 bands, which are the two 1.53 MHz bands defined in Figure 4, must be at least 60 dB below the power in the signal.
   (c) The power transmitted in either of the two 1.75 MHz bands labelled adjacent channel 2 in Figure 4, must be at least 63 dB below the power in the signal.

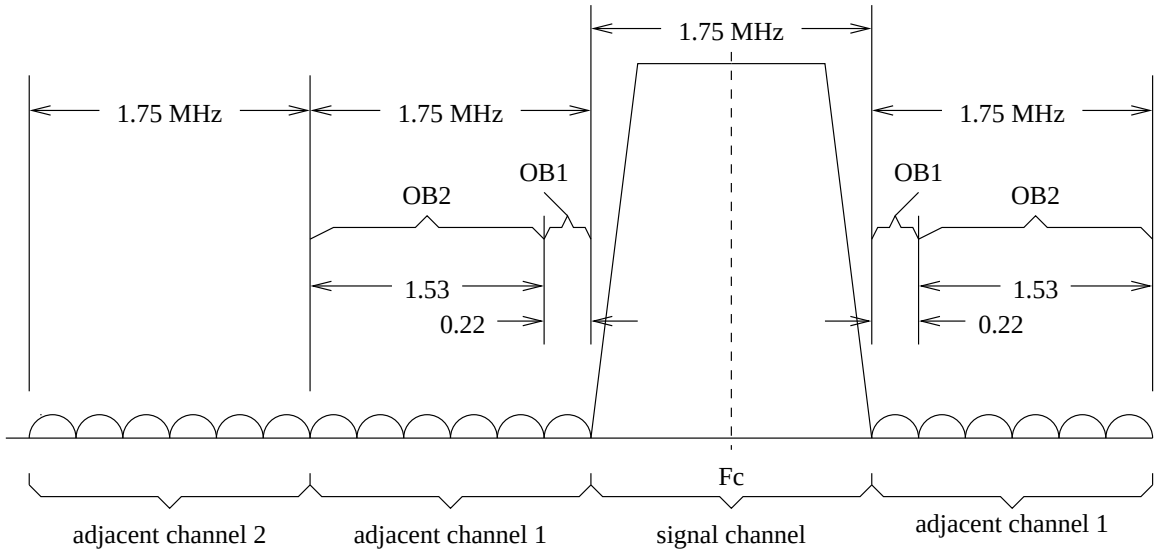


Figure 4: Illustration of out-of-band emission channels.

7. In the absence of channel distortion and AWGN, the MER (modulation error ratio) must be greater than 50 dB when the gold standards for both the pulse shaping filter and matched filter are used. The MER is defined by

$$\text{MER} = \lim_{N\to\infty} 10\,\log_{10}\left(\frac{\sum_{n=-N}^{N}(I^2[n]+Q^2[n])}{\sum_{n=-N}^{N}[(I[n]-\widehat{I}[n])^2+(Q[n]-\widehat{Q}[n])^2}\right),$$

where $I[n]$ and $Q[n]$ are the ideal values of the decision variable and $\widehat{I}[n]$ and $\widehat{Q}[n]$ are the actual values of the decision variable.

Specifications of the Gold Standard Matched Filter

For the purposes of this class the specifications for the gold standard for the matched filter are identical to the specifications of the gold standard for the pulse shaping filter.

Specifications for the Practical Pulse Shaping Filter

The practical pulse shaping filter has the same specifications as the gold standard except when the specifications given below are in conflict. The specifications given below supersede the specifications for the gold standard.

1. In the absence of channel distortion and AWGN, the MER must be greater than 40 dB *when paired with the gold standard for the matched filter.*

2. The number of 18x18 multipliers used in the implementation must be less than 10.

Deliverable 4

Deliverable 4 is the up-sampler and up-converter in the transmitter and the down-converter and down-sampler in the receiver. A block diagram that illustrates what is meant by up/down sampler/converter is shown in Figure 5. For this design, the carrier frequency of the signal channel is to be set at $F_c = 6.25$ MHz.

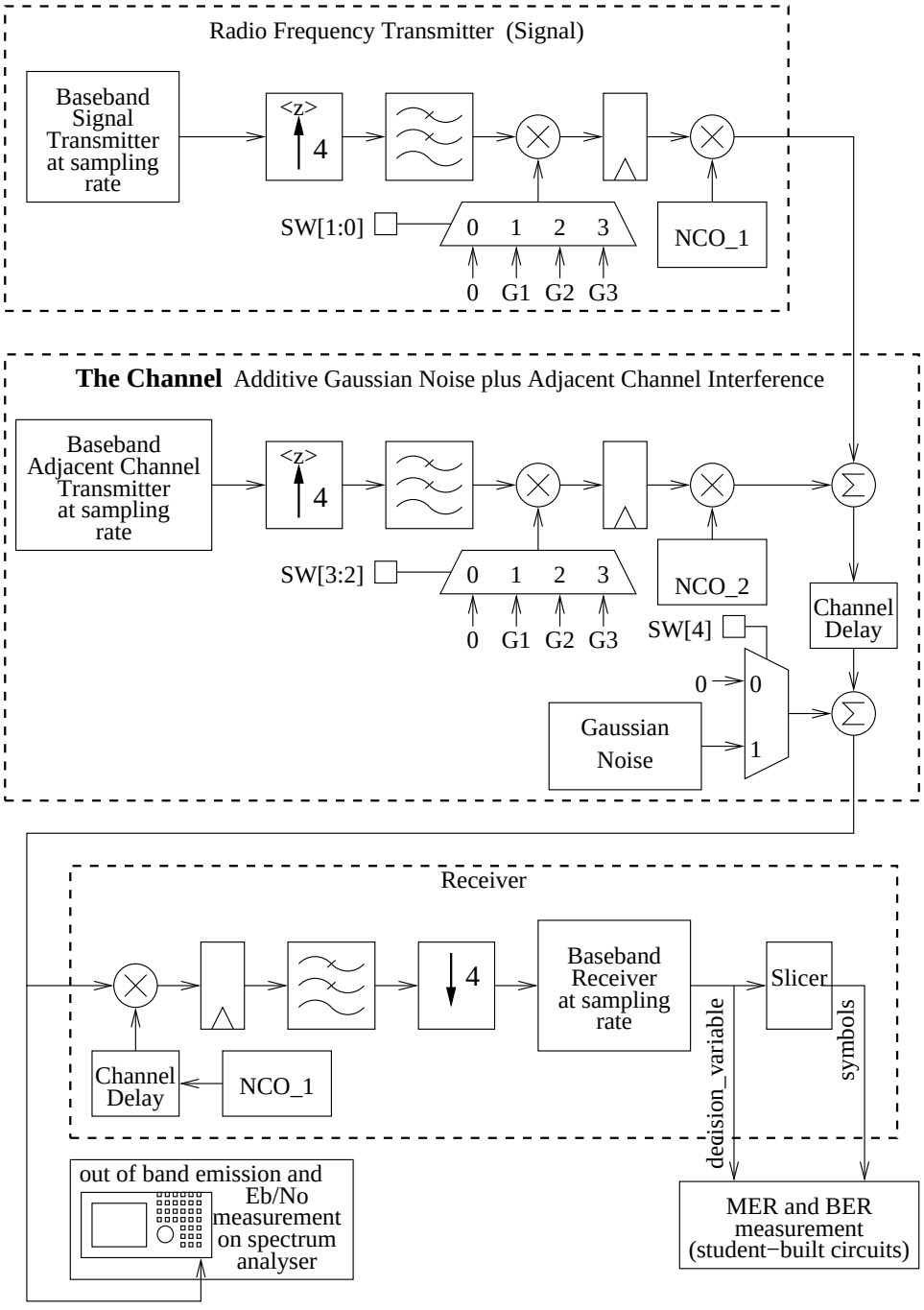


Figure 5: Block diagram showing the up/down sampling and up/down conversion.

The adjacent channel interference generator is simply the transmitter with the up-converter NCO set to an appropriate carrier frequency.

The specifications are given below:

1. In the absence of AWGN and channel distortion, with the practical pulse shaping filter and the gold standard matched filter employed in the transmitter and receiver, respectively, the process of up-sampling, up-converting, down-converting and down-sampling must not degrade the MER by more than 1 dB.

2. The total number of multipliers is limited to 14 or less.

Testing:

1. The Gaussian noise generator is supplied by the instructors. It has an 18-bit signed output with an average power that is 24 dB below full scale. That is, the power in the noise in units dBm is

$$P_{\text{noise\_total\_in\_dBm}} = P_{\text{full\_scale\_in\_dBm}} - 24 \text{ dB},$$

where

$$P_{\text{full\_scale\_in\_dBm}} = 10 \log \left(\frac{s^2}{1 \text{ mW}} \right),$$

where s is the worth of the most significant bit.

The power spectral density of the noise in W/Hz is given by

$$N_{0\_\text{in\_W/Hz}} = \frac{P_{\text{noise\_total\_in\_Watts}}}{F_s/2 \text{ cycles/sample}},$$

where F_s is the sampling rate in units samples/second. The power spectral density in dBm/Hz is given by

$$\begin{aligned} N_{0\_\text{in\_dBm/Hz}} &= 10 \log \left(\frac{\left(\frac{P_{\text{noise\_total\_in\_Watts}}}{F_s/2 \text{ cycles/sample}} \right)}{1 \text{ mW/Hz}} \right), \\ &= 10 \log \left(\frac{P_{\text{noise\_total\_in\_watts}}}{1 \text{ mW}} \right) - 10 \log \left(\frac{F_s/2 \text{ cycles/sample}}{\text{Hz}} \right) \\ &= P_{\text{noise\_total\_in\_dBm}} - 10 \log \left(\frac{F_s/2 \text{ cycles/sample}}{\text{Hz}} \right) \end{aligned}$$

2. The signal-to-noise ratio (SNR) is measured at RF. This is accomplished by measuring the powers in the signal and noise separately and then taking the ratio. The signal power is measured from $F_c - (1+\beta)R_s/2$ to $F_c + (1+\beta)R_s/2$ with the noise turned off, where F_c is the frequency of the carrier, i.e., the center of the signal's spectrum, and R_s is the symbol rate. The noise power is measured across the *noise bandwidth* of the pulse shaping/matched filter with the signal turned off. This translates to measuring the power in the noise from $F_c - R_s/2$ to $F_c + R_s/2$ with the signal turned off.

Both the power in the signal and the power in the noise can be measured on the spectrum analyzer using the marker function "Band interval power".

The SNR is calculated as follows. First, the power in the signal, denoted P_s, is equal to the energy per symbol, denoted E_s times the symbol rate, denoted R_s. Second, the power in the noise is equal to the noise power spectral density, denoted $N_{0\_in\_W/Hz}$, times the noise bandwidth, which is R_s cycles/symbol. It then follows that the SNR is given by

$$\text{SNR} \;=\; \frac{P_s}{P_n} = \frac{E_s\, R_s}{N_{0\_\text{in}\_\text{W/Hz}}\, R_s \text{ cycles/symbol}} = \frac{E_s \text{ symbols/cycle}}{N_{0\_\text{in}\_\text{W/Hz}}}$$

3. The three values of attenuation/gain in the transmitter must be chosen to achieve three specific values of E_b/N_0: 7.88 dB, 10.52 dB, 12.20 dB. In linear scale the equivalent values are 6.14, 11.28, 16.61. With correctly chosen values, assuming Gray code mapping, the probabilities of error should be 10^{-2}, 10^{-3} and 10^{-4}, respectively.

   The attenuation/gain that achieves a specified value of E_b/N_0 can be calculated using the equations above. Of course E_s can be converted to E_b by $E_b = E_s/2$.

4. The bit error rate can be measured in a few different ways. There is a fairly simple way that does not involve accessing the symbols sent by the transmitter. Obviously, accessing the symbols in the transmitter, delaying them and comparing them to the symbols received is impossible to do in practice. A circuit that measures BER without a "data connection" between the transmitter and receiver is illustrated in Figure 6.

   The circuit of Figure 6 is a bit tricky in that it runs at the bit rate, which is twice the symbol rate and half the sampling rate. It also requires the LFSR used in the transmitter be identical to the one shown in Figure 6. That is to say the serialized data leaving the parallel-to-serial converter in Figure 6 must be a delayed version of (but otherwise identical to) the output of the LFSR that generates the symbols in the transmitter's LFSR.

   The principle of operation is quite straight forward. Once timing is found and the decision variable is identified, the momentary switch shown in Figure 6 is pressed to put the error measurement circuit in motion. Pressing the momentary switch sets the "initialize" flip/flop, which positions the data selector at the input to the LFSR to select the serialized data from the slicer. At the same time it sets all the registers in the LFSR to "1". When the momentary switch is released the serialized data from the slicer is loaded into the LFSR until the "initialize" flip/flop is cleared. When the "initialize" flip/flop is cleared the data selector switches to select the feedback network.

   The parallel-to-serial converter takes the 2-bit symbols from the slicer, which obviously runs at the symbol rate, and produces a single bit output at the bit rate, which is obviously twice the symbol rate. Once the receiver's LFSR is filled, assuming no error is the data, it will contain a pattern identical to one that was in transmitter's LFSR a little while ago.

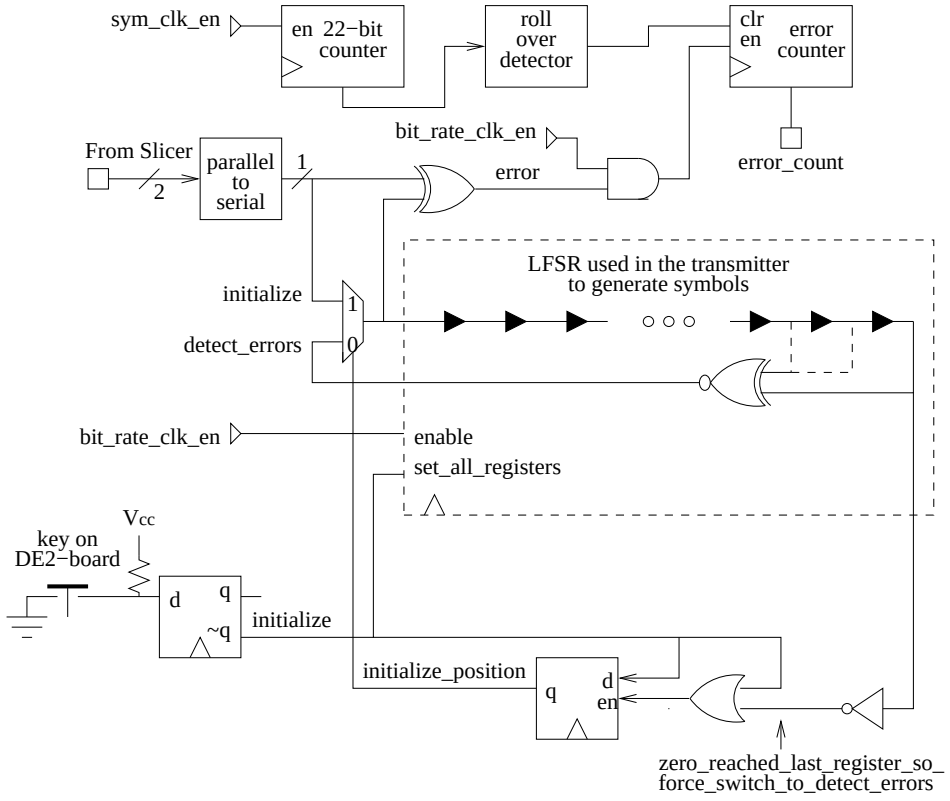


Figure 6: A simple circuit to calculate BER.

As previously mentioned, pressing the momentary switch sets all registers in the LFSR to "1". This is done so that when a "0" finally reaches the last register in the chain of shift registers it acts as a flag to indicate that LFSR is filled with data. When a "0" finally reaches the last register in the LFSR, the "initialize" flip/flop is cleared and the MUX at the input to the LFSR changes to select the feedback network. The LFSR now functions as an LFSR identical to the one in the transmitter that is synchronized to the serialized data.

The output of the now free-running LFSR is compared to the serialized data coming from the slicer with an exclusive-OR. If the output of the exclusive-OR is "1" an error has occurred. The errors are accumulated in a counter that runs at the bit rate. The "error counter" is periodically reset with period 2^{22} symbols. The error rate is obtained by reading `error_count` in signal tap just prior to it being cleared and then dividing its value by 2×2^{22} to get the bit error rate.

5. Measure E_b/N_0 and the corresponding BER. Compare with the theoretical calculation of the BER according to $P[\text{bit error}] = \frac{3}{4}Q\left(\sqrt{0.8\frac{E_b}{N_0}}\right)$.

6. Measure the MER at different settings of E_b/N_0, one of which is in the absence of noise, i.e., $E_b/N_0 = \infty$.

Deliverable 5

Deliverable 5 is the system described by deliverable 4 with two additional features, both of which are for a packet-based system. The two additional features are timing recovery and slicer reference level recovery.

Timing Recovery

To add packet based timing recovery to the system requires the design of two circuits: a packet generator that resides in the transmitter and a timing recovery circuit that resides in the receiver.

Before a packet generator can be designed the structure/format of a packet must be defined. For the system designed here a packet consists of a 27 symbol preamble followed by an infinite length payload. Normally the payload would have a fixed length that would either be known *apriori* or would be transmitted in a special field in the preamble. For this class the packet ends when an external "reset" signal is made active. The "reset" is generated by pressing `key[3]` on the DE2-115 board.

The transmission of a packet begins with the transfer of the first symbol of the preamble, which is the first symbol in the cyclic prefix, to the mapper (or to the pulse shaping filter if it is multiplier less). This transfer happens on the first positive clock edge of `sys_clk` that occurs after the reset signal goes low and `sym_clk_en` is active.

The 27-symbol preamble has three components: `cyclic_prefix`, `unique_word` and `cyclic_suffix`. The entire preamble is generated by a circuit described by the Verilog HDL below:

```
always @ (posedge sys_clk)
   if (reset == 1'b1)
       LFSR_4 = INITIAL_VALUE;  // INITIAL_VALUE is localparm
   else if (sym_clk_en == 1'b1)
       LFSR_4 = {LFSR_4[2:0], LFSR_4[3]^~LFSR_4[2]};
   else
       LFSR_4 = LFSR_4;

always @ *
   preamble_sequence = LFSR_4[3];
```

The local parameter `INITIAL_VALUE` is chosen so that `LFSR_4[3]` first generates the cyclic prefix, followed by the unique word, followed by the cyclic suffix. The preamble pattern to be generated is

```
preamble_sequence = suffix :   unique word    : prefix
preamble_sequence = 110111__0000_1010_0110_111__000010
                    |                                 |
                last sent                        first sent
```

The algorithm for switching from the preamble to the payload is described by the Verilog HDL below:

```
always @ (posedge sys_clk)
   if (reset == 1'b1)
       preamble_counter = 5'd0;
   else
       if (sym_clk_en == 1'b1)
          if (preamble_counter == 5'd27)
              preamble_counter = preamble_counter;
          else
              preamble_counter = preamble_counter + 5'd1;
       else
          preamble_counter = preamble_counter;

always @ (posedge sys_clk)
   if (sym_clk_en == 1'b1)
        if (preamble_counter == 5'd27)
            TX_symbol = random_data; // preamble finished so transmit payload
        else if (preamble_sequence == 1'b0)
            TX_symbol = MOST_POS_CONSTELLATION_POINT;
        else
            TX_symbol = MOST_NEG_CONSTELLATION_POINT;
    else
        TX_symbol=TX_symbol;
```

There are many ways to design a timing recovery circuit. To ease the complexity of the task a receiver architecture that includes timing recovery is given in Figure 7.

To further ease the task of designing, building and testing a timing recovery circuit a block diagram for a timing recovery circuit is given in Figure 8. The notation in one place in Figure 7 may be confusing. The dashed-line switch is meant to show the `peak_detection_pulse` depending on either `pattern_match` or `persistent_patten_match`. It is also pointed out that the circuit that generates `persistent_patten_match` is an example. If `persistent_patten_match` is used the persistence need not be 4.

There are two additional specifications that impact the timing recovery circuit.

1. A reset pulse of one system clock period duration is issued to the receiver. This pulse is a time critical pulse that would normally be generated by the MAC layer. The reset pulse is time critical in that it must coincide within tight limits of the arrival of the first symbol at the receiver. To be more specific, the reset pulse in the receiver occurs a fixed time after the "start of packet" reset pulse is issued in the transmitter. The time difference between pulses is the time from when a symbol enters the pulse shaping filter in the transmitter to the time the main tap of the response of the pulse shaping to that symbol reaches the receiver. This time depends on the length of the pulse shaping filter and the delay through the channel, which is unique to every customer. The critical time is measured during a initialization process known as "ranging" and stored in the MAC layer.

   For this project the reset pulse for the receiver will not be supplied by the MAC layer

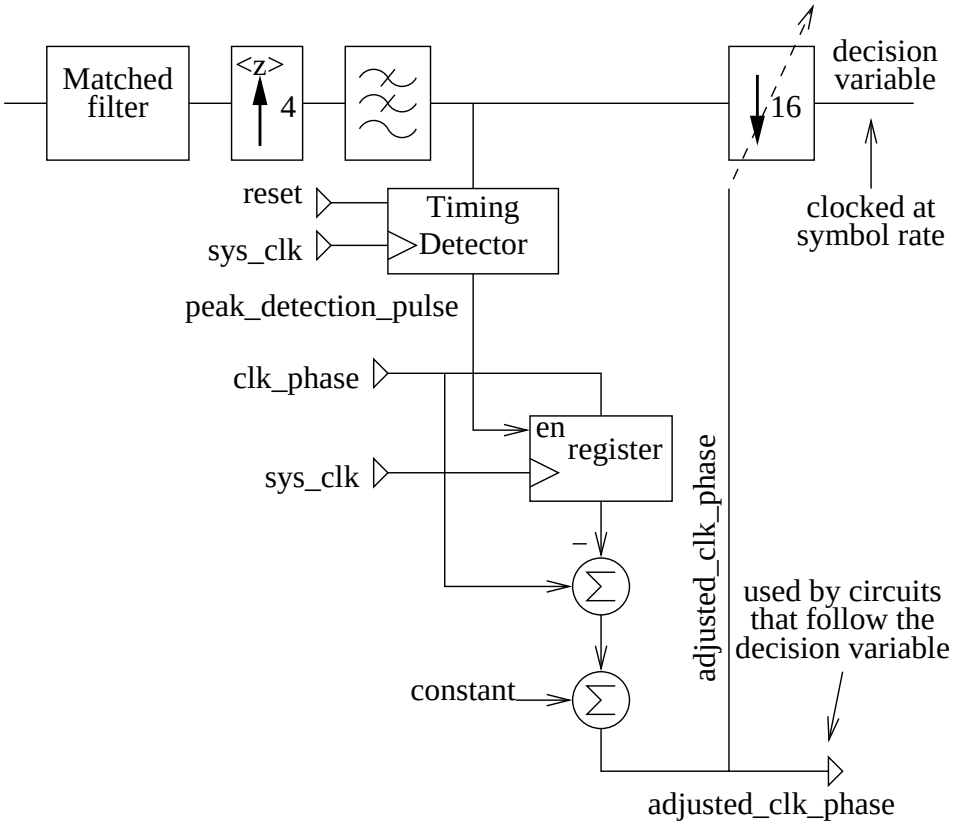


Figure 7: The architecture of a QAM receiver with timing recovery.

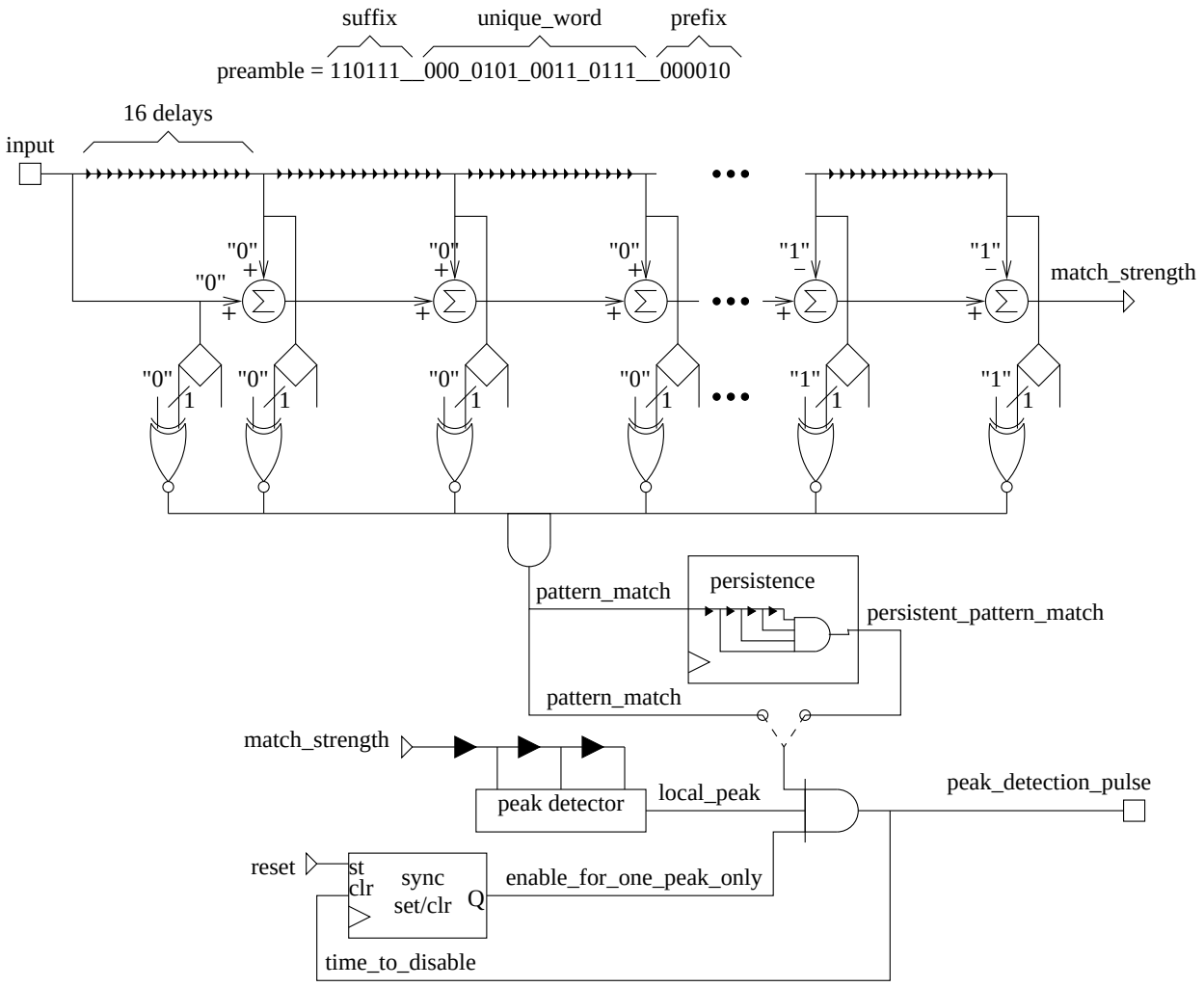


Figure 8: A block diagram of a unique word based timing recovery circuit.

nor by the instructors. It will be designed and build along with the receiver by the student.

For this project the reset pulse (for the receiver) will coincide with the first symbol of the cyclic prefix reaching the timing recovery circuit.

The pulse width and tolerance on the time of issuance of the reset pulse in the receiver is specified by:

The pulse is a positive pulse with a duration of one system clock period.

The pulse is to be issued a fixed time after the "start of packet" pulse in the transmitter such that under conditions of no channel delay the negative edge of the pulse will go low sometime after (or coincident with) the time the main tap of the first symbol transmitted reaches the timing recovery circuit and before the second last symbol in cyclic prefix reaches the timing recovery circuit.

It is pointed out that during the testing that is done in the lab exam a channel delay of any value up to 1 symbol will be inserted between the transmitter and receiver.

2. The timing recovery circuit must find the correct timing to within $\pm 1/32$ of a symbol time when there is no channel noise.

Note: The reset pulse arrives before the pipe line in the timing recovery circuit is filled with symbols. If the reset pulse is used for the reset in Figure 8 there will be a chance the unfilled pipeline will mimic the unique word and `peak_detection_pulse` **occur prematurely. Perhaps the reset in Figure 8 should be a delayed version of the reset pulse, i.e., delayed by 15 symbols.**

Slicer Reference Recovery

The reference level for the slicer must be recovered from the received data. In a packet based system a good estimate of the reference level must be recovered from the preamble (i.e. before the payload is received). Normally, the reference level is first estimated from during the preamble using the unique word and then continuously improved using the payload data. The initial estimate is normally based on the output of the peak detector, which depends on length of the unique word. The length of the unique word thereby determines the quality of the initial estimate of the reference level. The unique word is chosen long enough to give a reasonably good estimate, but, for reasons of efficiency, not so long that it gives a very good estimate. The error in the initial estimate is reduced over time using the data in the payload.

The decision variable, in the absence of noise, can be one of only 4 values: $\pm 1.5 v_{\text{ref}}$ or $\pm 0.5 v_{\text{ref}}$, where v_{ref} is the reference level for both the slicer and mapper. Only symbols that map to $\pm 1.5 v_{\text{ref}}$ are used in the preamble. Since there are 15 symbols in the unique word

and all symbols map to $\pm 1.5 v_{\text{ref}}$ the output of the peak detector is

$$\begin{aligned} v_{\text{pk\_det}} &= \sum_{n=0}^{14} \left(\frac{3}{2} v_{\text{ref}} + v_{\text{nse}}[n] \right) \\ &= \frac{45}{2} v_{\text{ref}} + \sum_{n=0}^{14} v_{\text{nse}}[n] \end{aligned}$$

where $v_{\text{pk\_det}}$ is the output of the peak detector and $v_{\text{nse}}[n]$ is the noise corrupting the n^{th} decision variable. An estimate of the decision variable is obtained by multiplying $v_{\text{pk\_det}}$ by $2/45$ to get

$$\widehat{v}_{\text{ref}} = v_{\text{ref}} + \frac{2}{45} \sum_{n=0}^{14} v_{\text{nse}}[n],$$

where $\widehat{v}_{\text{ref}}$ is an estimate of the correct reference level, which is v_{ref}. The noise corrupting $\widehat{v}_{\text{ref}}$ is $(2/45) \sum_{n=0}^{14} v_{\text{nse}}[n]$. It does not have a DC component, therefore its AC component has power

$$\begin{aligned} P_{\text{ref\_nse}} &= \overline{\left(\frac{2}{45} \sum_{n=0}^{14} v_{\text{nse}}[n] \right)^2} \\ &= \left(\frac{2}{45} \right)^2 \overline{\left(\sum_{n=0}^{14} v_{\text{nse}}[n] \right)^2} \\ &= \left(\frac{2}{45} \right)^2 \sum_{n=0}^{14} \overline{v_{\text{nse}}^2[n]} \\ &= \left(\frac{4}{2025} \right)^2 (15 \times \sigma_n^2) \\ &= 0.0296 \sigma_n^2 \end{aligned}$$

where $\sigma_n^2 = \overline{v_{\text{nse}}[n]^2}$ is the power in the noise corrupting the decision variable.

The data in the payload can be used to reduce the power in the noise corrupting $\widehat{v}_{\text{ref}}$ even further. There are, however, two significant differences between the preamble and the payload. The symbols in the preamble are known *apriori* where as the symbols in the payload are not and the symbols in the preamble are never mapped to constellation points as $\pm v_{\text{ref}}/2$ whereas the symbols in the payload are.

If the decision variable holds a constellation point at $\pm 0.5 v_{\text{ref}}$, then its absolute value has to be multiplied by 2 to get an estimate for v_{ref}. Whereas, if the decision variable holds a constellation point at $\pm 1.5 v_{\text{ref}}$, then the absolute value has to be multiplied by $1/1.5 = 2/3$ to get an estimate for v_{ref}. This means the constellation point must be ascertained before the decision variable can be used to estimate v_{ref}.

The constellation points for the payload can be ascertained using the reference $\widehat{v}_{\text{ref}}$ obtained in the preamble for the slicer. While $\widehat{v}_{\text{ref}}$ is not perfect it is good enough to categorize the symbols for purposes of improving the estimate $\widehat{v}_{\text{ref}}$ obtained in the preamble.

An interesting observation can be used to simplify the math. Summing the absolute values of two decision variables, where one represents a symbol mapped to $\pm 1.5 v_{\text{ref}}$ and the other represents a symbol mapped to $\pm 0.5 v_{\text{ref}}$ produces $((1.5 v_{\text{ref}} + v_{\text{nse}_1}) + (0.5 v_{\text{ref}} + v_{\text{nse}_2}))$, which is equivalent to $2 v_{\text{ref}} + v_{\text{nse}_1} + v_{\text{nse}_2}$. An estimate of the reference level is obtained by dividing the sum by two.

Using this approach an estimate for v_{ref} can be obtained by summing the absolute values of N decision variables taking care that exactly half of the N terms in the sum represent a constellation point at $\pm 0.5 v_{\text{ref}}$ and exactly half represent a constellation point at $\pm 1.5 v_{\text{ref}}$. Then v_{ref} is estimated by

$$\widehat{v}_{\text{ref}} = \frac{N v_{\text{ref}} + \sum_N v_{\text{nse}}[n]}{N},$$

where $\widehat{v}_{\text{ref}}$ is an estimate of v_{ref}. Choosing N to be a power of 2 makes the division a simple shift.

The power in the AC noise that corrupts $\widehat{v}_{\text{ref}}$ is given by

$$\begin{aligned} P_{\text{ref\_nse}} &= \overline{\left(\frac{\sum_N v_{\text{nse}}[n]}{N} \right)^2} = \frac{1}{N^2} \sum_N \overline{v_{\text{nse}}^2[n]} \\ &= \frac{1}{N^2} N \sigma_n^2 = \frac{\sigma_n^2}{N} \end{aligned}$$

Note that for $N = 32$ the noise power $0.0312 \sigma_n^2$, which is nearly the same as the power in the noise corrupting the initial estimate that was based on the symbols in the unique word.

A hardware efficient algorithm for recovering the reference level is given below:

Slicer Reference Level Recovery Algorithm

The initial estimate for the slicer reference is 2/45 of the peak value of the peak detector. This value is loaded into a register named `slicer_reference` that holds the current estimate of the slicer reference level. The register `slicer_reference` is also referred to a $\widehat{v}_{\text{ref}}$.

The slicer reference recovery circuit has four inputs: (1) a clock, (2) a reset pulse, (3) the absolute value of the decision variable, which is denoted $x_{\text{abs\_DV}}[n]$ and (4) a binary variable named $c[n]$, which is derived from $x_{\text{abs\_DV}}[n]$ as follows: if $x_{\text{abs\_DV}}[n] \geq \widehat{v}_{\text{ref}}$ then $c[n]$ is assigned a value of 1 otherwise it is assigned a value of 0.

The slicer reference recovery circuit has two outputs: (1) a clock enable named `clk_ena` and (2) an updated estimate of the reference level, which is named $\widehat{v}_{\text{ref\_out}}$.

The slicer reference recovery circuit has an accumulator named `ref_lvl_accumulator`, two counters named `counter_0` and `counter_1` and a register named `counter_milestone`.

The algorithm can be described by the logic that is applied to each new sample of the decision variable. That logic is described below:

1. On clock edges that occur while the reset is high:

   Load counters `counter_0` and `counter_1` with 16.

   Load `ref_lvl_accumulator` with 32 times $\widehat{v}_{\text{ref}}$.

   Load `counter_milestone` with 32.

2. If (`counter_0 == counter_milestone`) and (`counter_1==counter_milestone`) then

   Set `clk_ena = 1`.

   Assign $\widehat{v}_{\text{ref\_out}}$ the value in `ref_lvl_accumulator` divided by (2×`counter_milestone`).

   Load `counter_milestone` with 2×`counter_milestone`.

3. If ($c[n] == 0$) and (`counter_0`≠`counter_milestone`) then add $x_{\text{abs\_DV}}[n]$ to `ref_lvl_accumulator` and increment `counter_0`.

4. If ($c[n] == 1$) and (`counter_1`≠`counter_milestone`) then add $x_{\text{abs\_DV}}[n]$ to `ref_lvl_accumulator` and increment `counter_1`.