Lab 8 - Bit Error Rate Measurement

Introduction

In this lab you will use a Field Programmable Gate Array (FPGA) to build a circuit to measure Bit Error Rate (BER).

The BER is the fraction of bits that are received incorrectly and is a common performance metric for digital communication systems.

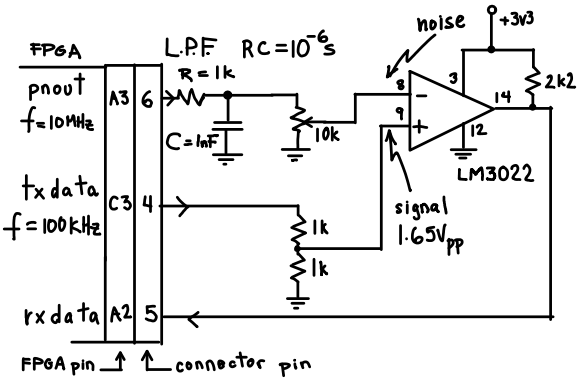
Interference, distortion, crosstalk, and various types of noise can all cause bit errors. In this lab we will measure the BER of an NRZ link that has Gaussian noise added to it. This is the so-called the Additive White Gaussian Noise (AWGN) channel. We will compare our experimental BER results to the BER results predicted by the theory covered earlier in the course.

Small- and medium-volume communication products often include programmable logic devices called FPGAs. FPGAs consist of thousands of simple logic elements whose interconnections and functionality can be configured to perform arbitrary digital logic functions. Pre-designed "Intellectual Property" (IP) building blocks such as microprocessors, communication interfaces, and signal processing functions can be included in FPGA designs.

In this lab you will use a *DE-0 Nano* FPGA "Demonstration and Education" board that contains an Altera EP4CE22F17C6N FPGA. This FPGA has 22,000 logic elements, 154 I/O pins and 600k of RAM memory. The board includes some simple I/O peripherals including 8 LEDs, two push-buttons and two "header" connectors with 40 general-purpose I/O pins each. The Quartus II FPGA design software will be used to synthesize (compile) the design and program (configure) the FPGA.

Circuit Description

The hardware used to implement the AWGN channel is shown below:



`pnout` is a logic-level (0 to 3.3V) 10 MHz Pseudo-Random Bit Sequence (PRBS) with a period of $2^{31}-1$ bits. It is low-pass filtered by an RC filter with a 1 μs time constant. This produces an analog voltage that varies in a pseudo-random manner according to the bit pattern and has a probability density function that is approximately Gaussian. The level of the noise signal can be adjusted with a 10k pot acting as a variable voltage divider.

`txdata` is a 100 kbps (kHz) PRBS data signal with a period of $2^{10}-1$ bits.

The data and noise signals are input to an LM3022 comparator. The comparator subtracts the noise signal from the data and outputs a high logic level if the result is greater than zero. Thus the noise causes an error when the data signal is positive and the voltage of the noise signal exceeds the voltage of the data signal.

The FPGA compares the transmitted and received digital data (`txdata` vs `rxdata`) to detect and count bit errors. Bits 0, 4, 8, 12 and 16 of a 17-bit error counter are displayed on the board's LEDs so you can tell when the error count reaches certain values (1, 16, 256, 4096 and 65536 respectively).

Pre-Lab Procedure

Before the lab you should complete the FPGA design and synthesize (compile) it to an FPGA programming (`.sof`) file using Quartus II.

Download the partial solution `.zip` file from the course web site. The zip file includes a project file (`lab8.qpf`), a block diagram (schematic) file (`lab8.bdf`), a settings file (`lab8.qsf`) and four pairs of files (a `.qip` and a `.vhd` file for each of the `pnclock`, `dataclock`, `shiftregister`, and `errors` modules).

The settings file contains the FPGA device type and the pin assignments so that the synthesizer knows which FPGA pins should be used for the signals in your design. I/O pins are defined for the LEDs (`LED[7..0]`) and the reset button (`reset`) on the board. The PN output (`pnout`), data output (`txdata`) and the data input (`rxdata`) are connected to GPIO0 connector pins shown above and described below.

The 10 MHz PN PRBS clock, `pnclock`, is derived by dividing down the 50 MHz oscillator on the board. It has an output, `pnclockN`, that is delayed by half a clock period and drives the data PRBS generator. The time offset minimizes digital switching noise when the data signal is sampled.

The data clock signal `dataclock` also has a delayed version, `samplingclock`, that is used to sample the data in the middle of the bit period.

There are two 31-bit shift registers that shift left (LS to MS bit) and are used to implement the LFSRs. Each shift register has a serial input, `shiftin` and each bit of the shift register is available in parallel, on the `datasr` and `pnsr` buses. You will have to complete the design by making connections from the correct shift register outputs to the XOR gate inputs. Connections are made by labeling[1] a conductor with a signal name. For example, labeling a conductor with `pnsr[8]` would connect it to the 9'th storage element of the `pnsr` shift register.

The partial solution also has a a 17-bit counter with a synchronous count enable input. The counter only counts up if the error (`error`) input is asserted at the rising edge of the clock. Bits 0, 4, 8 12 and 16 of the counter are connected to `LED[0]` through `LED[4]` so you can tell when the error count reaches 1, 16, 256, 4096 and 65536. The synchronous clear input, `sclr`, is connected to the `reset` signal.

The `reset` signal is driven by (inverted) pushbutton `KEY0`. It has two functions: it resets the error counter to zero and initializes the LFSRs to all ones. This is necessary because on power-up the LFSRs are initialized to zeros. This means **your circuit will not generate noise or data until it has been reset!**

For testing purposes the PLL locked status is copied to `LED[7]` and the reset input is copied to `LED[6..5]`.

The `error` signal is only asserted if `rxdata` differs from `txdata` and `txdata` is high (only errors on 1's are counted).

The supplied solution will not work properly because the shift register taps are not correct. You will have to look up the correct taps to use to achieve the desired PRBS length and select the appropriate shift register bits as inputs to the XOR gates.

Synthesize the modified design to make sure there are no errors. Make sure you have the files in the project directory available for the lab (put them on your H: drive and/or on a flash drive).

Submit a PDF file to the dropbox on the course web site containing your name, ID, lab number, date and a screen capture of the modified portion of your schematic.

Lab Procedure

Assemble the AWGN Channel

Connect a USB cable from the PC to the USB connector on the FPGA board. The blue power LED should light and the LEDs should show the default "breathing" demo.

Run the Quartus II version 13 software. Open your project file (File/Open Project). You may want to work off your `H:` drive to avoid losing work.

Synthesize your design (Processing/Start Compilation). Correct any errors and repeat until your design synthesizes without errors.

Run the programmer utility (Tools/Programmer), make sure the USB-Blaster programmer is detected and press Start to program the FPGA with your design. Press the KEY0 button to reset the FPGA and turn on LEDs 5 and 6. Disconnect the USB cable to power-down the board.

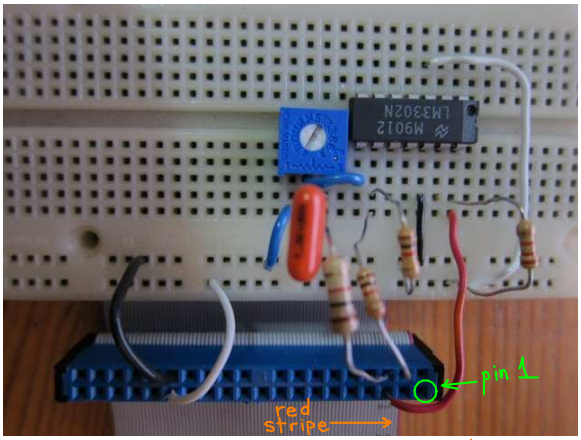
Connect a supplied 40-pin ribbon cable to the GPIO0 connector on the FPGA board (see photo below).

Build the AWGN channel circuit on your solderless prototyping board and make the connections to the ribbon cable connector as shown below. Obtain 3.3V

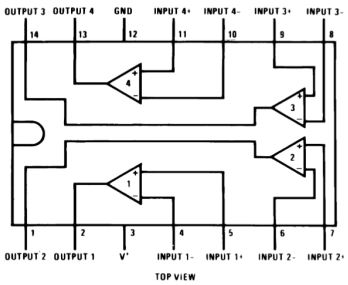
[1] Right-click and select Properties.

from the ribbon cable connector pin 29 and ground from pin 30.

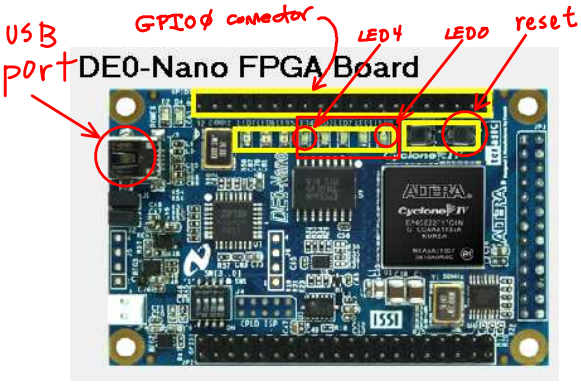
The following photograph shows how connections can be made from the ribbon cable connector to the solderless prototyping board by treating the connector as if it were another prototyping board:



The pinout of the LM3302 is as follows:



The following photograph shows the LEDs and pushbuttons on the board:



You will use the `reset` pushbutton (labelled `KEY0` on the board) to reset the error counter. LEDs labelled LED0 through LED4 turn on when the error counts reach 1, 16, 256, 4096 and 65536 errors respectively.

The following diagram labels the connection points on the ribbon cable connector with pin 1 (red conductor) on top. The letter-number pairs are the row and column respectively on the 256-pin (16×16) BGA (ball-grid array) FPGA package; numbers beginning with zero (0) are signal names used in the the DE-0 documentation (prefixed with `GPIO_`), the other numbers are the 40-pin "header" connector numbers. For example, the signal on FPGA pin A2 has signal name GPIO_02 and is on pin 5 of the connector (third pin from the top on the right side).

D3	00	2	1	0_IN0	A8
C3	01	4	3	0_IN1	B8
A3	03	6	5	02	A2
B4	05	8	7	04	B3
B5	07	10	9	06	A4
GND	–	12	11	–	5V
D5	09	14	13	08	A5
A6	011	16	15	010	B6
D6	013	18	17	012	B7
C6	015	20	19	014	A7
E6	017	22	21	016	C8
D8	019	24	23	018	E7
F8	021	26	25	020	E8
E9	023	28	27	022	F9
GND	–	30	29	–	3.3V
D9	025	32	31	024	C9
E10	027	34	33	026	E11
B11	029	36	35	028	C11
D11	031	38	37	030	A12
B12	033	40	39	032	D12

Connect one 'scope channel to the non-inverting (data) input of the comparator and one to the inverting (noise) input. Enable the 'scope's average and RMS voltage measurements for the noise channel and enable measurement statistics (see Measurement Hints section below).

BER Measurements

Adjust the noise power until you get approximately one change every two seconds on the least-significant LED (`LED[0]`). Press the reset button and measure the time it takes for the next most-significant LED to turn on (about 30 seconds). This is the time it took to count 16 bit errors. Multiply this time by the data

rate (50,000 bits/second since only '1' bits are being checked) to get the number of bits checked.

Measure the high-level signal voltage using the measurement cursor. Measure the average noise voltage and the noise variance (see Measurement Hints below). Compute the predicted BER using the equation:

$$P_e = \frac{1}{2}\text{erfc}(\frac{v}{\sigma\sqrt{2}})$$

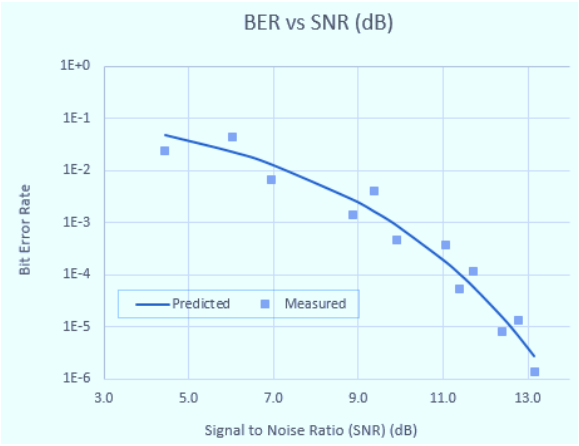
where v is the positive signal voltage (approximately 1.65) minus the average noise voltage, and σ is the standard deviation of the noise (this is the noise RMS voltage if the signal is AC coupled; see below). Compare your measured and predicted BER values.

Enter your measurements into the spreadsheet[2]. It should include columns for the error count, elapsed time, measured BER, signal voltage, average noise voltage, noise variance and predicted BER.

Repeat for the next most significant LED (256 errors): increase the noise power so the blink rate on `LED[1]` is about one every two seconds and repeat the BER and SNR measurement and the BER prediction calculations.

Repeat for the remaining LEDs so that you have four measured and four predicted BER values.

Insert a graph (chart) in your spreadsheet comparing the measured and predicted BERs. The BER should be plotted on a logarithmic scale and the SNR (ratio of signal voltage to square root of the variance) should be in dB as shown below (your values may be different):



[2]It is good practice to make a permanent record of your measurements in a lab notebook.

Measurement Hints

You will need to measure the signal and noise powers carefully to get useful results because the BER changes quickly over a narrow range of SNRs. Use the following guidelines when making measurements:

- set the voltage offset and gain to get the largest waveform display that does not clip the signal
- turn on bandwidth limiting (20 MHz) and use a sampling rate that avoids aliasing (auto sample rate with about 5μs/division sweep rate for this lab)
- use the 'scope's measurement feature along with measurement statistics. Average at least 20 measurements and continue until the average stops changing.
- measure the variance of the noise voltage by measuring the RMS voltage of the AC-coupled input. AC coupling removes the DC (average) component so that the RMS value is equal to the variance[3]. The average voltage must, of course, be measured with DC coupling or it will be zero.

Report

Submit a report in PDF format containing:

- the identification asked for in Lab 1
- the schematic of your final LFSR PRBS generator including any corrections you made in the lab
- the spreadsheet showing your predicted and measured values
- the graph showing the predicted and measured BER versus SNR

[3]The variance is the second central moment: $\sqrt{\overline{x^2} - \overline{x}^2}$.