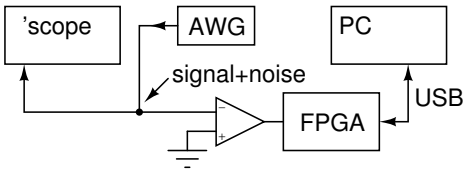
FPGA-Based Error Rate Measurement

Introduction

A Field-Programmable Gate Array (FPGA) is an IC that can be programmed to implement different digital logic functions. FPGAs are primarily used in low-volume products. But they are also useful for setting up custom tests and measurements.

Logic analyzers are instruments similar to oscilloscopes but for digital signals. They trigger on events in digital signals and then capture and display them. FPGAs can be configured to include a logic analyzer function. FPGAs with embedded logic analyzers are a powerful tool.

In this lab you will use an FPGA with an embedded logic analyzer to measure the error rate of data transmitted over a noisy channel using the following measurement setup:

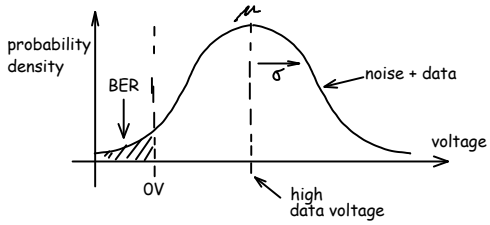


The AWG outputs a bipolar NRZ signal corrupted with additive white Gaussian noise (AWGN). A comparator compares the noisy signal to a decision threshold at 0 V. The FPGA monitors the comparator output and counts the total number of bits received and the number of bits received in error. The ratio of errors to bits is the bit error rate (BER).

Analysis of the error rate for bipolar NRZ and AWGN shows that the error rate is the same regardless of whether a high or low level is transmitted. In this lab we will only transmit a high data value. The transmitted signal is thus a zero-mean Gaussian noise signal with a positive offset equal to the high data signal level.

The signal plus noise is applied to the negative input of the comparator. When this input is lower than the positive input (0 V) the output is high, otherwise it is low. The comparator thus indicates if an error would be made if the receiver sampled the received signal at that time.

The diagram below shows the probability distribution of the signal plus noise:



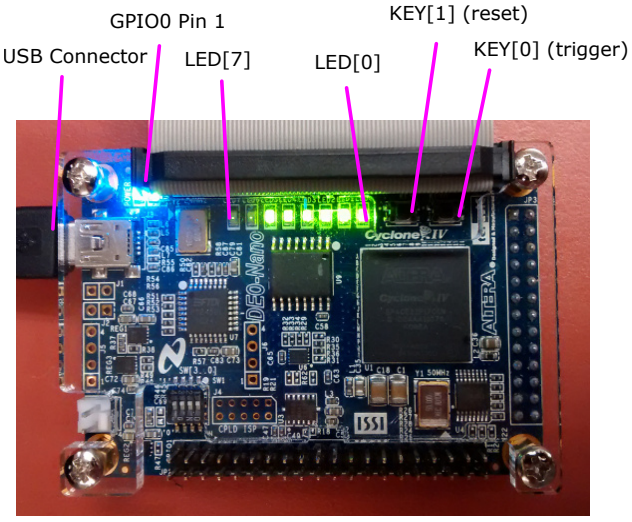
The signal voltage is the mean of the distribution (μ) and is measured as the average voltage using DC coupling. The standard deviation of the noise (σ) and is measured as the RMS voltage using AC coupling to eliminate the average (DC) component.

In this lab you will measure the BER for various signal to noise power ratios (SNRs) and compare your measurements to theoretical results.

FPGA-Based BER Measurement

We will use an Altera *DE-0 Nano* FPGA board that includes a Cyclone IV FPGA, 8 LEDs, two push-buttons and two connectors with 40 I/O pins each.

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



The FPGA board has two pushbuttons. You will use the left pushbutton, labelled KEY1, to reset the error counter. You will use the right button, labelled KEY0, to trigger the logic analyzer and display the current bit and error counts on the PC.

The FPGA is programmed over a "JTAG" interface connected to the PC over a USB port. The JTAG interface is also used to access the embedded logic analyzer.

The comparator circuit is connected using a 40-pin ribbon cable plugged into the GPIO0 connector on the FPGA board (see photo above). Note that the ribbon cable connector has a tab on one side which must fit in the slot on the board's plexiglass cover.

The following table shows the connections on the other end of the ribbon cable when the connector holes are facing up and pin 1 (red conductor) is along the top. Each square represents a position where you can insert a wire.

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

The outer columns with letter-number pairs show the row and column respectively on the 256-pin (16×16) BGA (ball-grid array) FPGA package; the middle columns with numbers beginning with zero (0), when prefixed with GPIO_, are the names to be used in Quartus for the signals on the 40-pin connnector; the numbers in the inner columns, next to the squares, are the connector pin 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). Look at the instructor's demonstration circuit if you have any questions.

FPGA Configuration

Figure 1 shows the schematic of the circuit to be programmed into FPGA. It includes a 100 kHz bit clock and two counters. One counter counts up on every clock cycle, the other counts up only when the comparator indicates an error.

The procedure to enter the design and configure the FPGA using the (free) Quartus Prime software is as follows:

- start Quartus Prime
- File -> New -> New Quartus II Project
- create or select a folder, project name (e.g. lab8), the Cyclone IV E Family, the EP4CE22F17C6 device, and leave other settings set to defaults
- select Assignments -> Import Assignments -> DE0-NanoX3525.qsf to assign FPGA pin numbers to the signals on the DE0-Nano board. This file is available on the course web site
- File -> New -> Block Diagram/Schematic File
- if necessary, select View -> Utility Windows -> IP Catalog to show the list of configurable intellectual property (IP) blocks
- select Library -> Basic Functions -> Arithmetic -> LPM_COUNTER
- define the counter component by specifying a name (e.g. bitcounter), VHDL source, 32-bit up-only counter with count enable and synchronous clear inputs. Enable generation of a symbol (.bsf) file and add the component to your project.
- similarly, define a clock component by selecting Library -> Basic Functions -> Clocks; PLLs and Resets -> PLL -> ALTPLL

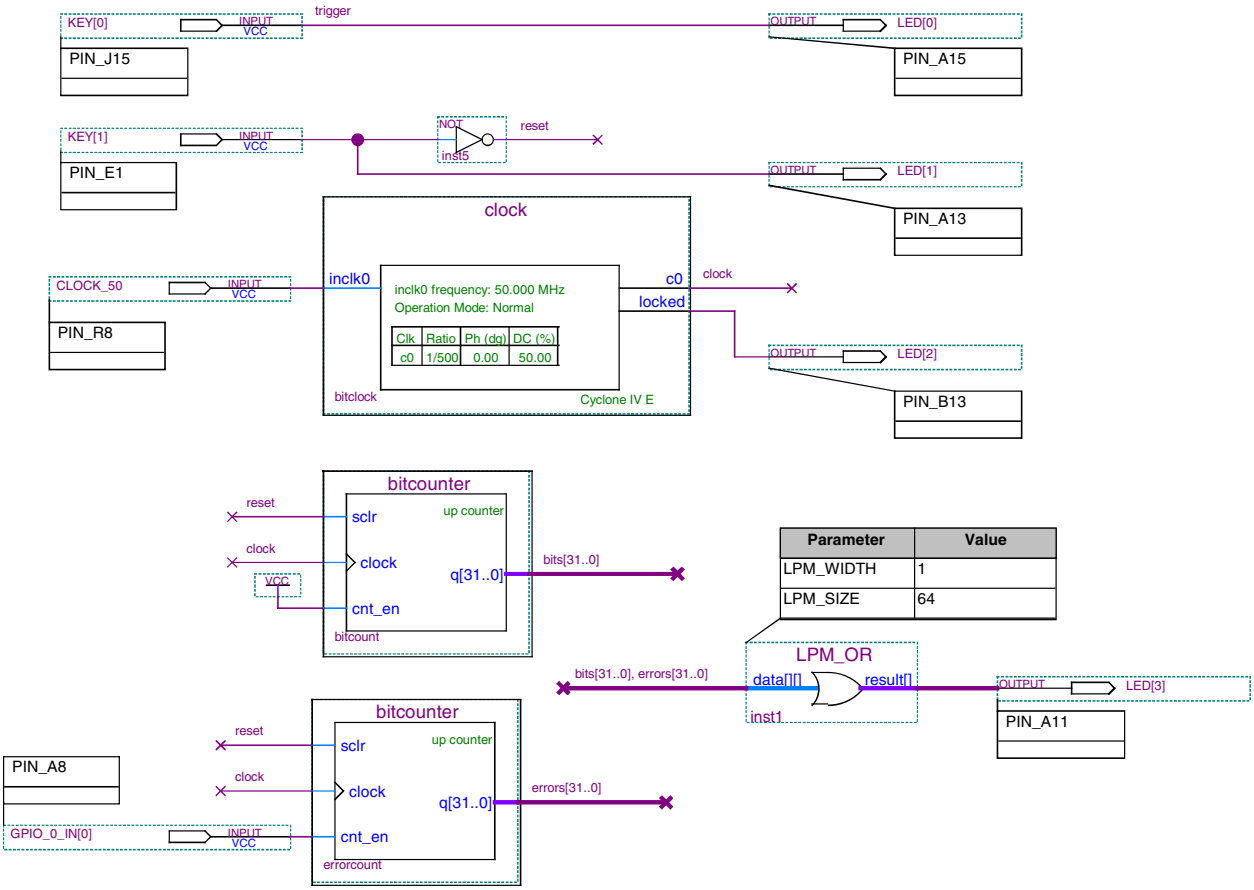


Figure 1: FPGA schematic showing the clock, bit counter and error counter.

- specify a name (e.g. clock), a 50 MHz input, locked and a 100 kHz clock outputs (c0), and no additional inputs or outputs. As before, generate a symbol (.bsf) file and add it to the project. The ports for the clock and counter symbols should match those shown in Figure 1.

- use the "Symbol Tool" and insert two counters and one clock from the Project library as shown in the schematic. Give each component instance a unique label (e.g. bitcount, errorcount, bitclock) by changing the inst(ance) name in the symbol.

- use the "Symbol Tool" and insert the required Vcc symbol (to enable the bit counter), and one 64-input OR gate (Megafunctions -> gates -> `lpm_or`) as shown in the schematic. *Hints: Use the search box to find components. It is often easier to give signals names (right-click -> Properties) rather than connecting them.* Assign names to the bit and error count signals (right-click, Properties -> Name) so the logic analyzer can select them.

- use the "Pin Tool" to insert the output and input ports labeled with the exact pin names shown in the schematic[1] Since the counter outputs are only used by the logic analyzer, they must drive an output (through the 64-input OR gate) so that they are not optimized away. The error and trigger pushbuttons drive two LEDs to verify button pushes and that the FPGA has been programmed.

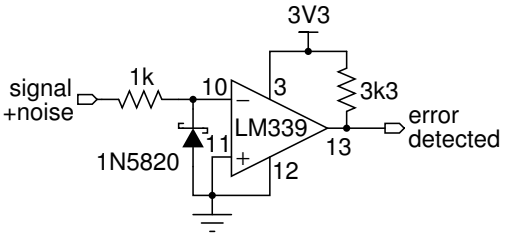
- use the "Bus", "Node" and "Selection" tools to connect the components as shown in the schematic. Note that the bus ranges in the schematic below should be `[31..0]`.

[1]Signal names must match the symbol names in the DE0-NanoX3525 settings file or else they will not be connected to the correct FPGA pins.

- click on the "Start Compilation" icon. Correct any errors and recompile as necessary.
- connect the FPGA to the PC's USB port and select Tools -> Signal Tap II Logic Analyzer
- under JTAG Chain Configuration -> Setup select USB Blaster
- click on ... and select the correct `.sof` (FPGA serial programming) file (it's in the `output_files` folder)
- click on the download ("Program Device") icon to make sure the device is recognized and can be programmed.
- under Signal Configuration use the node finder (...) to list all pins using the SignalTapII:pre-synthesis filter and add `clock:bitclock c0` as the clock
- in the logic analyzer window double-click in the indicated area to add signals. As before, list the pins and add the GPIO_0_IN[0] (error detected), KEY[0] (trigger) and KEY[1] (reset) pins to the list of signals being monitored in the Setup tab. Add the bit and error count signals.
- right-click on the trigger condition for KEY[0] (trigger) and select a falling edge
- save the SignalTap file, add it to the project, recompile and reprogram the FPGA
- click on "Autorun Analysis" to start the logic analyzer
- the logic analyzer should display the error and bit counts in the Data tab each time you press on the trigger button. Change the display format to unsigned decimal. You may need to zoom in to see the bus values.

Threshold Comparator

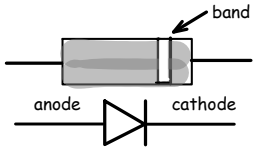
The schematic of the comparator circuit is shown below:



Power for the comparator is supplied by 3.3 VDC from the FPGA board. Connect to 3.3V on ribbon cable connector pin 29 and ground on pin 30.

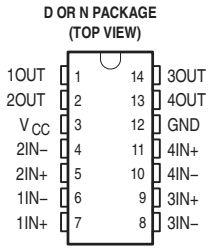
The comparator output is open-collector so a pull-up is required.

The Schottky diode prevents the input going more negative than about -0.3 V to prevent damage to the comparator which cannot tolerate negative input voltages. The white band marks the cathode. The diode *must* be connected with the correct polarity or instead of protecting the comparator from negative voltages it will prevent positive voltages from being applied to the comparator input and could result in damage to the comparator.



The resistor values are not critical – values between 1 k and 4.7 k should work well.

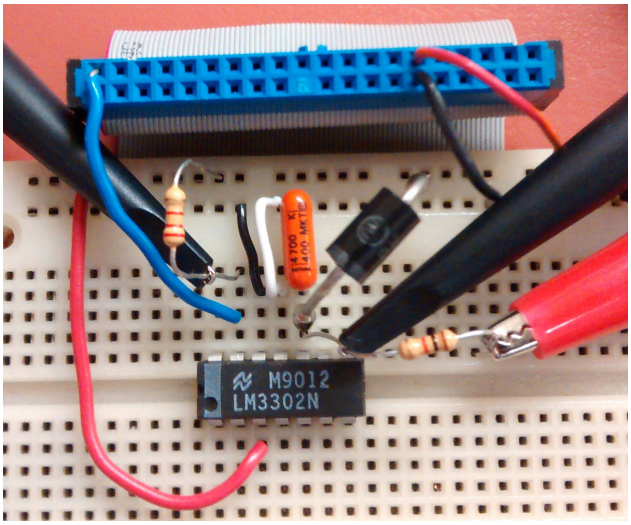
You can substitute an LM3302 comparator. Their pin-out is:



Assemble the circuit on a prototyping board and make connections to the cable that connects to the FPGA board. The red stripe marks the side of the ribbon cable connected to pin 1.

The following photograph shows how the circuit can be assembled on a prototyping board and connections made to a cable that connects to the FPGA board. The red stripe (on the left of the photograph

below) marks the side of the ribbon cable connected to pin 1. The capacitor (orange) shown in the photograph should be omitted.



Note: The FPGA will be damaged by 5 V signals. External power supplies must not be used during this lab.

50% of your lab mark will be deducted if you turn on or hook up a power supply during this lab!

Gaussian Noise Generation

The Arbitrary Waveform Generator (AWG) will output samples of a zero-mean pre-computed AWGN signal. The offset voltage will be configured manually.

Download the `berlabnoisegen.m` file from the course web site and run the command `berlabnoisegen` in Matlab or Freemat to generate the file `awgn.raf`. Set the AWG for Arb operation and load the `.raf` file using the instructions in a previous lab. Select SRate (sample rate) mode and set the sample rate to 200 kHz.

Data Analysis

The oscilloscope and DMM can be used to measure the signal (average-DC coupled) and noise (rms-AC coupled) voltages at the negative comparator input. From these, which correspond to μ and σ of the signal's Gaussian probability distribution, the BER can be predicted.

The logic analyzer can be used to read the bit and error counter values. Their ratio is the measured BER. The predicted and measured BER values can then be plotted against each other.

For communication systems the BER is usually plotted as a function of the signal to noise ratio (SNR) in dB instead of against the normalized threshold ($t = \frac{\mu}{\sigma}$). When the BER in log units is plotted against the SNR in dB the result is the familiar "waterfall" curve showing a rapid reduction in BER with increasing SNR.

Procedure

Use the instructions above to enter the design shown in Figure 1 and program the FPGA. Build the comparator circuit and connect it to the FGPA board using the supplied ribbon cable. Connect the output of the AWG to the comparator circuit.

Look at the comparator input and output with a 'scope and adjust the AWG amplitude and offset to verify that the comparator operates as expected. Then check that the bit and error counts as displayed by the logic analyzer are changing as expected.

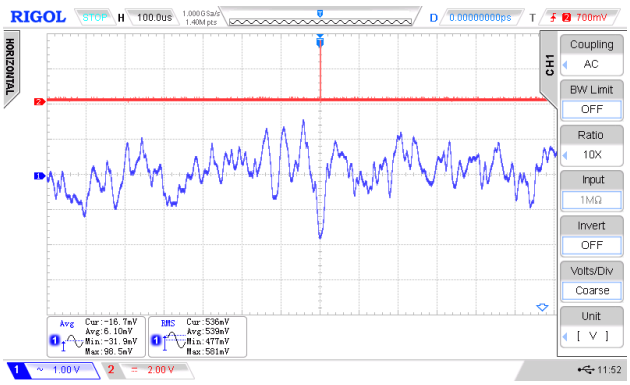
Set the AWG offset (signal level) to 1 V and verify this by measuring the DC (average) voltage with the DMM. Measure the RMS voltage at the comparator input. Adjust AWG output amplitude to obtain the desired SNR. Record the signal voltage and noise RMS voltage. Reset the error and bit counts by pressing KEY1 and trigger the logic analyzer by pressing KEY0 to view the bit and bit error counts. Wait until the error count reaches 10 or more errors to get a reasonably accurate estimate of the BER. Record the numbers in your spreadsheet. Repeat for each of the SNRs in your table.

Compare your measured and predicted results and try to identify the cause of any differences. Note the following:

- the noise waveform only contains 1 million samples. It may not contain any samples with large voltages and you may not see any errors at high SNRs.

- the 'scope's RMS measurements appear to be a function of the sweep rate and may not be reliable (check using the DMM).

Save a 'scope screen capture showing the signal plus noise input to the comparator and the error-detect output of the comparator:



Pre-Lab Report

Create the spreadsheet you will use to record your data. You will be measuring the signal voltage (DC, average voltage), the noise voltage (AC, RMS voltage), the number of errors received and the number of bits received. From these you will compute the SNR (in dB), the predicted BER and the measured BER. For example:

signal voltage (V DC)	noise voltage (V AC RMS)	SNR (dB)	predicted BER	errors	bits	measured BER
1.000	0.500	6.0	2.3E-2	15270	689790	2.2E-2

You can use the complementary error function (`erfc()`) to compute the predicted error rate:

$$P(x < t) = \frac{1}{2}\text{erfc}\left(\frac{t}{\sqrt{2}}\right)$$

where t is the normalized threshold (the average signal voltage divided by the noise RMS voltage).

Fill in the spreadsheet with a signal voltage of 1 V (you may need to change this in the lab) and RMS noise voltages that will result in SNRs of between 6 and 12 dB in steps of about 1.5 dB.

Create a chart similar to the one below showing the predicted BER for this range of SNRs (you will add the measured curve in your final report).

Submit a PDF file containing the usual identification information and a printout of your spreadsheet including the graph of predicted BER vs SNR.

To save time you may enter the FPGA design before the lab. In this case you should save it as a project archive rather than a collection of files. Use Project -> Archive Project to create the `.qar` archive file and Project -> Restore Archived Project to restore the project directory from the `.qar` file.

Lab Report

Submit the usual identification information and a report in PDF format containing the following:

- your FPGA block diagram (schematic)
- the 'scope screen capture showing the noise waveform including the measured average and RMS voltages
- a table from your spreadsheet showing the calculation of the predicted and measured BER. The BER values should range from about 10^{-1} to about 10^{-6}.
- a plot showing both predicted and measured BER formatted as follows: