FPGA-Based Error Rate Measurement

Introduction

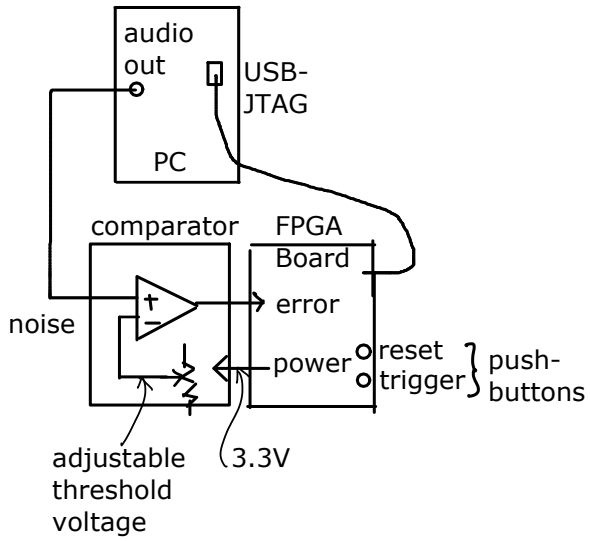
A Field-Programmable Gate Array (FPGA) is a logic device that can be programmed to implement any desired digital logic function.

FPGAs are useful for implementing small-volume products and to set up test fixtures, particularly when the signal frequencies are too high to allow processing in software.

Logic analyzers are instruments that recognize patterns in digital signals and capture and display them. FPGAs can be configured to include logic analyzer features. This approach is often more flexible than using stand-alone logic analyzers.

In this lab you will use an FPGA to implement a circuit that measures the error rate that would be experienced by a communication system affected by Gaussian noise. The test circuit will include one clock and two counters. The values of the counters will be read using a logic analyzer embedded in the FPGA design.

A block diagram of the test setup is shown below:



A Matlab script is used to generate samples of uncorrelated ("white") zero-mean Gaussian noise that are saved in a file. This noise waveform is output by the PC's sound card and applied to one input of comparator. A threshold voltage set by a potentiometer is applied to the other comparator input. The comparator outputs a high (H) level when the noise voltage exceeds the threshold voltage. The output of the comparator is output to a digital input on an FPGA. You 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 FPGA is configured with a 100 kHz clock driving two counters. One counter counts up on every clock cycle, the other counts up only when the output of the comparator is high. The ratio of the two counter values is thus the ratio of time that the signal exceeds the threshold.

In a baseband communication system using bipolar NRZ signaling with a noise margin equal to the threshold voltage, this ratio will be equal to be long-term average fraction of bits received in error which is, by definition, the probability of error.

The FPGA board has two pushbuttons. One is used to reset both counters. The other is used as a trigger by the embedded logic analyzer. The comparator power is supplied by 3.3VDC from the FPGA board. The FPGA is programmed over a JTAG interface plugged into one of the PC's USB ports. The JTAG interface is also used for the logic analyzer interface.

Noise Signal

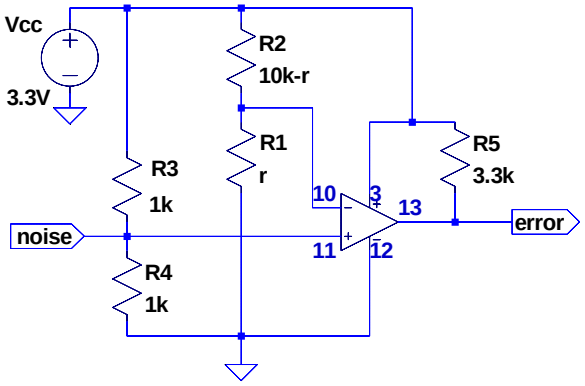
The Matlab `randn(n,1)` function can be used to generate a column vector with `n` Gaussian (normally-distributed) random numbers with $\mu = 0$ and $\sigma = 1$. The function `fopen()`, `fwrite()` and `fclose()` can be used to create a waveform file. For example, the statements:

```
x=4000*randn(1e6,1);
of=fopen('lab7.dat','w');
fwrite(of,x,'int16');
fclose(of);
```

create a 1-million sample file of normally-distributed random 16-bit[1] integers with zero mean and a standard deviation of 4000. This file can be imported into Audacity as in previous labs. If the sample rate is set to 48000 Hz the noise will have a bandwidth of 24 kHz and a duration of about 20.8 seconds.

Threshold Comparator Circuit

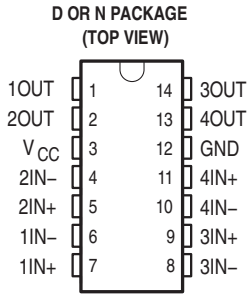
The following schematic shows the circuit that compares the input audio signal to a threshold:



Resistors R1 and R2 with values r and 10k $-$ r are used instead of a 10 k potentiometer so that the same schematic can be used for simulation. Any convenient value can be used instead of 10 k.

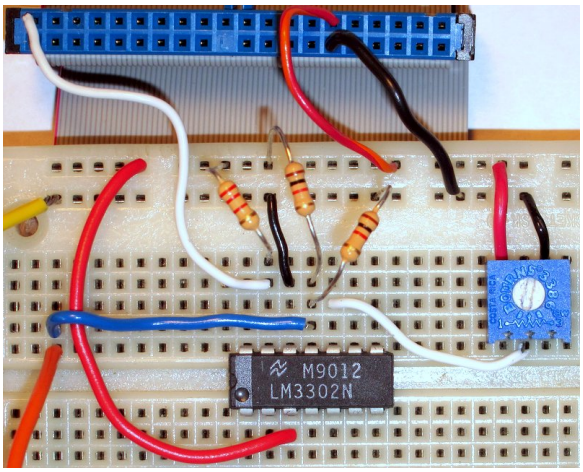
Resistors R3 and R4 are used to raise the average voltage of the (AC-coupled) noise input since the minimum comparator input voltage is -0.2V.

Note that the logic inputs to the FPGA *must* be 3.3V (not 5V). You can substitute any comparator that operates at 3.3V and has an open-collector output (the LM339 has the same pin-out as the LM3302 shown). The pin-out of the LM3302 is:



[1] Integers on an x86 architecture PC are stored in little-endian format.

The following diagram 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 marks the side of the ribbon cable connected to pin 1. The audio cable yellow wire is ground and the orange and violet are the audio signal.

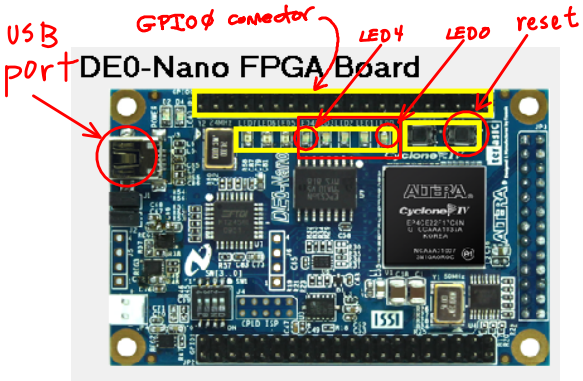


FPGA Board

The connections to the FPGA are made through a supplied 40-pin ribbon cable plugged into the GPIO0 connector on the FPGA board (see photo below).

Obtain 3.3V from the ribbon cable connector pin 29 and ground from pin 30 for the comparator circuit.

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.

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

FPGA Configuration

Figure 1 shows the schematic of the bit and error counters.

The procedure to design the error rate measurement circuit and configure the FPGA using the (free) Quartus II software is as follows:

- start Quartus II Web Edition (64 bit)
- File -> New -> Quartus II Project
- create or select a folder, project name (e.g. lab7), the Cyclone IV E Family, and the EP4CE22F17C6 device, and leave other settings set to defaults
- select Assignments -> Import Assignments -> DE0-Nano.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 file.
- similarly, define a clock component by selecting Library -> Basic Functions -> Clocks; PLLs and Resets -> PLL -> ALTPLL
- specify a name (e.g. clock), a 50 MHz input, no additional inputs or outputs and a single 100 kHz clock output (c0)
- use the "Symbol Tool" and insert two counters and one clock from the Project library, one Vcc constant (to enable the bit counter) and one inverter (not) as shown in the schematic. Give each component instance a unique label (e.g. bits, errors, clock).
- use the "Pin Tool" to insert the four output and four input ports (50 MHz clock, error, reset and trigger) labeled with the exact pin names shown in the schematic[2] Although the pushbutton and counter outputs are only used by the logic analyzer, they must be connected to output pins so that they are not optimized away. The error and trigger pushbuttons also drive two LEDs to verify that the FPGA has been properly configured.
- use the "Bus", "Node" and "Selection" tools to connect the components as shown in the schematic. Note that the bus pin numbers in the schematic below should be `[31..0]`.

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

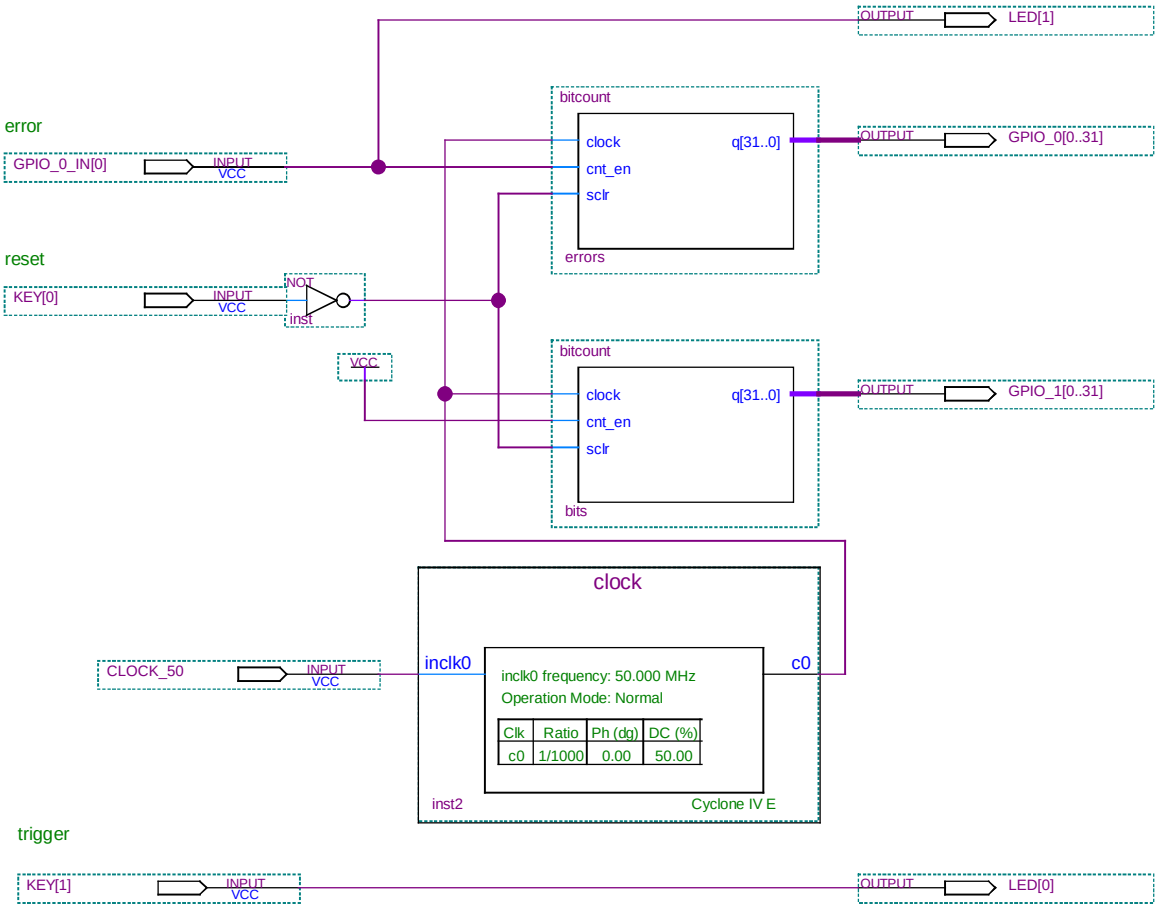


Figure 1: Bit and Error Counters.

- 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
- under SOF select the correct `.sof` (FPGA serial programming) file 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 and add the CLOCK_50 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], GPIO_O, GPIO_1, KEY[0] and KEY[1] pins to the list of signals being monitored in the Setup tab
- right-click on the trigger condition for KEY[1] (trigger) and select a falling edge
- recompile and reprogram the FPGA
- click on "Autorun Analysis" to start the logic analyzer
- each time you press on the trigger button the logic analyzer should display the error and bit counts in the Data tab
- you can change the display format to decimal to make it easier to calculate the error rates.

Data Analysis

Figure 2[3] shows the shape the Gaussian probability density function (pdf). The x-axis is labelled with the threshold voltage normalized by the noise RMS voltage (σ) as well as the area under the curve to the left of that voltage (the cumulative distribution function, or CDF). For example, if the threshold voltage was set to the mean noise voltage plus two times the RMS voltage (2σ) we would expect that about 97% of the time the noise signal would be below the threshold.

The oscilloscope and DMM can be used to measure the threshold voltage and the mean and RMS noise voltages. From these the BER can be predicted. The logic analyzer can be used to read the counter values and their ratio is the measured BER. The predicted and measured BER values can then be plotted against each other for a range of SNR values. 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 Quartus II to create the design shown in Figure 1. Check that you can program the FPGA and that pushing the trigger and reset buttons give you the expected results in the logic analyzer display.

Build the comparator circuit and connect it to the FGPA board using the supplied ribbon cable. Check that the comparator output responds as expected as you change the threshold voltage. Check that the error count as displayed by the logic analyzer responds as expected.

Generate the Gaussian noise samples and output the waveform from the PC audio output using Audacity and in previous labs. Connect the audio signal to the comparator input and check that the LED connected to the `error` input responds as expected.

Use the 'scope and DMM to measure the DC and RMS voltages at the comparator and use the logic analyzer to measure the BER.

[3] From the Noisecom Application Note "Noise Basics."

Lab Report

Submit the usual identification information and the following results:

- your FPGA block diagram (schematic)
- a spreadsheet showing the calculation of the predicted and measured BER. Set the threshold voltage to approximately $\mu + k \times \sigma$ for $k = 0, 1, 1.5, 2, 2.5, 3, 4$ where μ is the average noise voltage and σ is the noise (AC) RMS voltage. In your spreadsheet you can use the erfc() function to compute the predicted error rate:

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

- a plot showing both predicted and measured BER
- a plot of the first 100 samples of the noise waveform

Figure 2: Gaussian density function and values of the cumulative distribution.

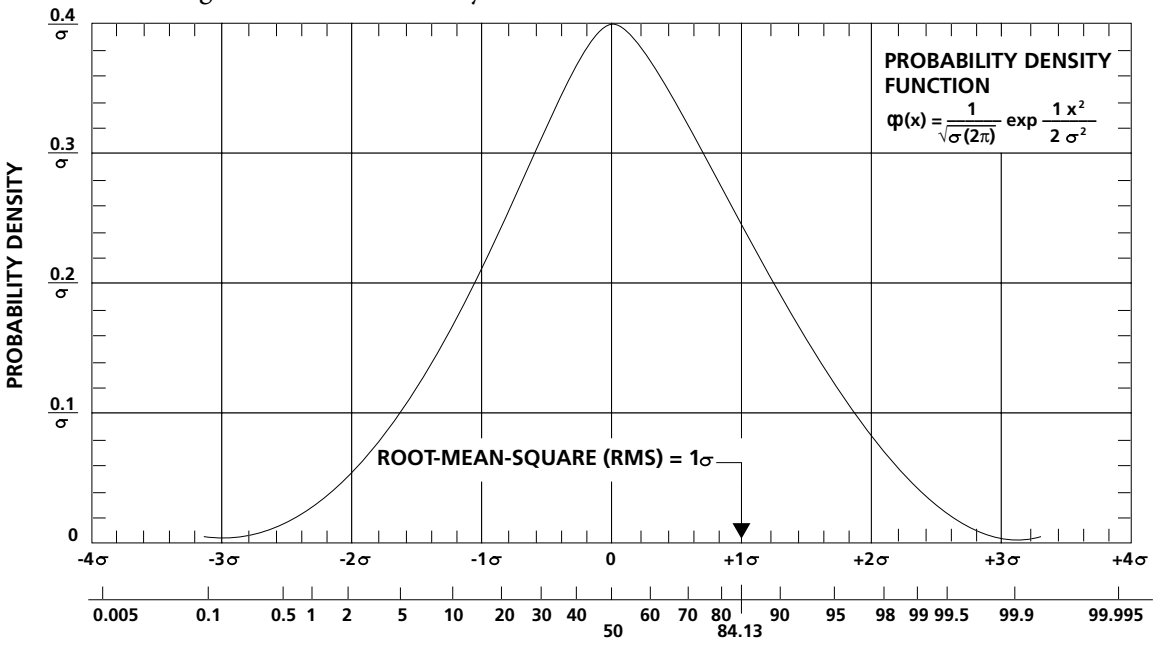


Figure 1. Gaussian Voltage Distribution