Development of a Wideband PLC Channel Emulator with Random Noise Scenarios

Ann Dulay, Ryan Sze, Aileen Tan, Roderick Yap, Lawrence Materum
Gokongwei College of Engineering, De La Salle University, Manila 1004, Philippines
ann.dulay@dlsu.edu.ph

***Abstract*—Channel emulators are an integral part of the test equipment that offers a more practical approach to testing new communication devices. It is imperative though to develop the emulator such that it best represents the channel. For PLC channel emulator, the channel representation can be either top-down or bottom-up. In this paper, the top-down characterisation and reference channels are used. In this approach, statistical measurements of the characteristics of the power line were conducted, and the closest mathematical representation is presented. The emulator operates in the frequency domain utilising 4096 transform points for the FFT process and 14 fractional bits for fixed point presentation. This number of bits allowed the emulator to sufficiently generate an average of 0.4% error between the software simulation results and the hardware test results. The input signal is converted to an LVDS signal by the FMC151 which serves as the AFE of the emulator. Two linear regulators block are used to convert both the negative and positive values of the input signal. The random generation of noise reduced the taxing efforts of adding different combinations of noise thus providing ease in focusing on the analysis of the resulting waveform.**

***Index Terms*—PLC; BBPLC; Emulator; Random Noise Generation; FPGA.**

I. Introduction

Powerline communications is an emerging technology that uses the cables for transporting electricity as the communications medium [1]. Although it has been here since the 1900's, the renewed interest started with the introduction of Smart Grid [2]. One of PLC's advantages over other communications system is that it uses the same infrastructure as the grid. Hence, there is no need for additional wires (for wired communications) or the setting up of high power antennas (for wireless).

There are two types of PLC based on the frequency it covers, narrowband and wideband. The latter is used in this study. Wideband PLC uses the frequency range from 500 kHz to 10 MHz. The high-frequency range allows high data rate transmission which makes it useful for internet and video applications [3].

Several PLC channel characterisations show that PLC channel is frequency selective, time-varying, and marred by the noise coming from various sources [4]. Thus, PLC modem designers must develop a robust design. Testing of the modem though using the live power network is not only hazardous, it is also relatively expensive and impractical since the test is not repeatable. PLC channel emulator offers a more viable testing approach.

II. Prior Studies

The latest literature on PLC channel emulation is found in Table 1. Each of these emulators considers a specific channel model and PLC category in their implementation. It is notable that most emulator implementation uses the impulse response where the PC generates the filter coefficients. While [5] has the random noise generation feature similar to this study, it covers only a single channel model. The noise model presented in [6] and [7] assumes that the noise is fixed. In this study, the noise is generated randomly such that either none, one of them, all of them or other possible combinations of the different types of noise are added to the channel.

In this paper, the top-down characterisation and the reference channels presented in [8] are used. The statistical measurements of the characteristics of the power line were conducted, and the closest mathematical representation is presented in [9].

III. PLC Channel Emulator Elements

A. Analog Front End

The channel emulator requires an analogue signal as its input to truly emulate a hardware implementation of the whole system. With this, an ADC and a DAC are required before the input and output respectively. Selecting the proper ADC and DAC requires knowledge of the input frequency for the system. Both ADC and DAC should also compliment the FPGA system clock to synchronise both devices properly. The basic requirement for the sampling rate must be at least twice the highest fundamental frequency, following the Nyquist theorem.

The AFE used is the FMC151. FMC151 has an FPGA mezzanine card that is compliant with Virtex 6 and higher boards. It contains a dual-channel high-speed 14-bit ADC, dual-channel 16-bit DAC, and a clock tree within the module which is controlled by a PC using SPI connection (Ethernet). Not only does the FMC151 needs an SPI connection, but it must also have a reference design code to store the digitalised values inside the FPGA. This reference design code merely is used as mapping and synchronising code for the FMC151 to store the values properly. The reference code was modified to integrate the channel transfer function in between the analog interfaces. Figure 1 shows the FMC151 block diagram with its interconnection to the FPGA.

Table 1
PLC Channel Emulation Literature

Ref	Title	Implementation	Random noise generation
[10]	Flexible FPGA-based powerline channel emulator for testing MIMO-PLC, neighbourhood networks, hidden node or VDSL coexistence scenarios	FPGA	No
[11]	Analysis of the PLC channel statistics using the bottom-up random simulator	Simulation	No
[3]	Channel emulation of low-speed PLC transmission channels	FPGA	No
[12]	A top-down random generator for the In-Home PLC Channel	Simulation	No
[13]	Time-varying channel emulator for indoor powerline communications	FPGA	No
[7]	A new channel emulator for low voltage broadband power line communications	FPGA (Time domain)	yes

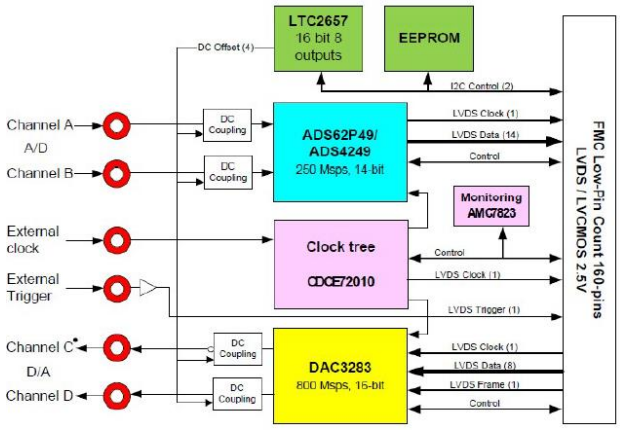


Figure 1: FMC151

B. Linear Regression

Linear regression is used to accurately recreate the ADC's binary coded signal, which is to be interpreted as fixed point by the FPGA. It is essential to take note of this for the FFT modules since the core interprets the amplitude of the signal as its fixed-point equivalent and cannot interpret the ADC's coded signal. To do this, the equivalent resolution of the ADC is calculated and used as the intervals for amplitude change, then the equivalent amplitude in fixed point is plotted against the samples needed by the FFT. Figure 2 shows the flowchart of the linear regression module as implemented in the FPGA.

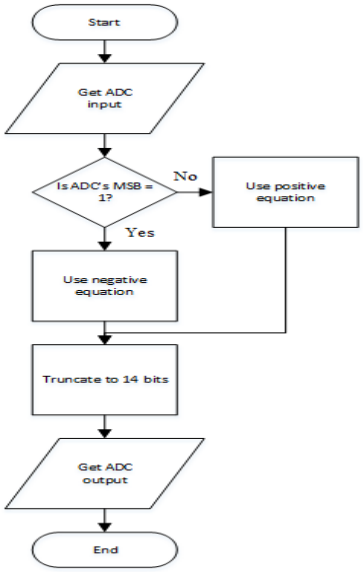


Figure 2: Flowchart of the linear regression module in Xilinx

Since the ADC operates on low voltage differential signalling (LVDS), its output code is represented in binary 2's complement [14]. Thus, two equations are needed to accommodate the positive coded values and negative coded values. The positive and negative equations are obtained by getting the equation of the line made from the plots mentioned earlier. Equations (1) and (2) are the fixed point equivalent equations for the positive and negative values respectively. Using the fact that negative coded values are represented as 2's complement of the positive value, it is easily distinguished with a 1 at its MSB. Finally, the output is then truncated before it is passed through the FFT module to fit the number of input bits.

$$Positive\ Equation: Y = 512x - 1279 \quad (1)$$

$$Negative\ Equation: Y = 512x + 29360128 \quad (2)$$

where Y represents the output of the linear regression, and x is the output of the ADC.

C. FFT/IFFT

The Fast Fourier Transform converts the time domain signal to the frequency domain signal. In this domain, channel transfer function is simply multiplied with the input signal.

The FFT and IFFT blocks are implemented using the built-in FFT core from Xilinx. The FFT core is an IP (intellectual property) core designed to perform the FFT or IFFT algorithm for an input using Equations (3) and (4) respectively.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{jnk2\pi}{N}} \quad k = 0, \dots, N-1 \quad (3)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{\frac{jnk2\pi}{N}} \quad n = 0, \dots, N-1 \quad (4)$$

The following specifications are included within the core:

- Number of points
- Target clock frequency
- Arithmetic type
- Architecture type
- Bit interface type

To properly use this core, the first parameter that needs to be set is the bit interface type. Within the FFT core, the designer can either use floating point or fixed point as the bit interface. Fixed-point representation is used here to quickly interpret the output data of the whole system while using minimal resources. The other design specifications are all based on the subcarrier spacing of current BPLC modem standard, *Homeplug AV,* which is 24.414 kHz. Therefore, the needed number of points must be enough to ensure the accuracy of the system while considering the 24.414 kHz spacing. Aside from this, the target clock rate is also determined by considering the Nyquist rate, ensuring an un-aliased signal. Since this study covers the frequency range up to 10 MHz, a 20 MHz minimum clock rate is necessary. In actual application, however, it is preferable to oversample to attain a high-fidelity signal at the output. Based on experimentation, the clock that satisfies the criteria is 100 MHz. From this value, the number of points needed is calculated by dividing the sampling frequency (target clock) with the subcarrier spacing as shown in Equation (5).

$$N = \frac{fs}{\Delta f} \tag{5}$$

where: fs = *target clock*; Δf = *subcarrier spacing*

The transform points are found to be 4096. The chosen arithmetic type of the core is unscaled to ensure maximum precision for the outputs. An unscaled output means that the FFT core output bits do not match the user input bits. This growth in the integer bits is carried to the output. Therefore, the FFT output bit, b_{xk}, results to 29 bits and is calculated using Equation (6).

$$b_{xk} = b_{xn} + log_2(FFT\ point\ Size) + 1 \tag{6}$$

where: $b_{xk} = FFT\ output\ \ bits$
$b_{xn} = input\ signal\ bits$

Lastly, the architecture type improves the latency or speed of the FFT core and is not directly related to emulation needs. There are four design choices, each with its corresponding speed and resource tradeoff. The pipelined streaming architecture is chosen since it offers the best speed compared to the others at the expense of using more resources. This type is optimal for the system to attain maximum accuracy.

D. Channel Transfer Function

The channel transfer function refers to the power line's channel frequency response expressed in an equation. The powerline channel used is the multipath model proposed by Zimmermann and Dostert [15]. Equation (7) gives the transfer function based on this model. Four different transfer function test models given in [8] were used as test channels for the emulator. These are generated by variations in the total distance and number of paths used. Figure 3 shows the different transfer functions based on Zimmermann's equation.

$$H(f) = \sum_{i=0}^{N} g_i . e^{-(a_0 + a_1 f^k) d_i} . e^{-j2\pi f \frac{d_i}{vp}} \tag{7}$$

The transfer function contains both real and imaginary components. The values for each are generated using the Euler's theorem as follows:

$$H(re) = \sum_{i=0}^{N} g_i e^{-(a0 + a1 f^k) d_i} . \cos(2\pi f \frac{d_i}{vp}) \tag{8}$$

$$H(im) = -\sum_{i=0}^{N} g_i e^{-(a0 + a1 f^k) d_i} . \sin(2\pi f \frac{d_i}{vp}) \tag{9}$$

All the transfer functions shown in Figure 3 are implemented in Xilinx as a look-up table or LUT. LUT's are similar to a storage register wherein the different values of the channel are stored in two registers – one for real and one for imaginary. These values are then called simultaneously to be multiplied with the FFT outputs.

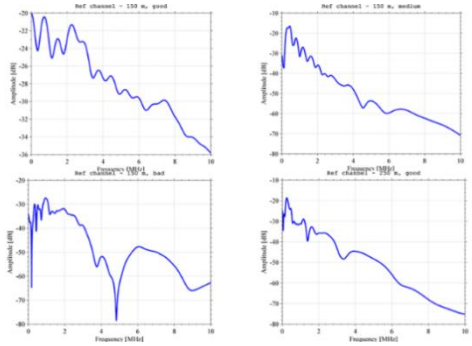


Figure 3: Four references channels based on distances and number of multi-paths

E. Multiplier Block

The multiplier block is designed using FOIL method. In this method, the rectangular form of the equation is used. This is simpler to implement in hardware than the polar form. Equation (10) shows the format of the two values.

$$(a + jb) * (c + jd) \tag{10}$$

where: a = Real value of FFT;
b = Imaginary value of FFT;
c = Real value of transfer function;
d = Imaginary value of transfer function.

Since the transformed signal contains negative values, this is considered in the allocation of bits. With 4096 transform points and a 16 bits input bit from the FIFO of the ADC of FMC151, the FFT outputs 29 bits. To make the multiplication simpler, the number of bits for the transfer function is set at 29 bits also. With this, the necessary bits to support multiplication would be twice the current number of bits.

Negative numbers are represented in 2's complement form; thus, when multiplying with negative numbers, overflow is possible unless an additional bit is allocated. Instead of allocating more bits to accommodate the overflow, the negative values are converted to positive, therefore keeping the bits predictable, then performing 2's complement to bring back its sign. The padded register, which is twice that of the original number of bits, is then truncated to fit the IFFT. The fractional bits sent to IFFT are maintained at 14 bits.

F. Noise Blocks

The noise present inside the PLC channel is characterised by three types, namely, coloured background noise, narrowband noise, and impulsive noise. The noise is added to the output of the multiplier block. The noise models for the background and impulsive noise were taken from [16] [17] while the narrowband noise is based on [18] and its equation is presented in Equation (11). All these noise values are stored as LUT. LUT was used for the noise blocks since the FPGA is limited regarding implementing high-level mathematical equations. Figure 4 and Figure 5 shows the background and impulsive noise respectively.

$$N_{ND}(f) = \sum_{k=1}^{N} A_k e^{\frac{(f-f_{o,k})^2}{2B_k^2}} \quad (11)$$

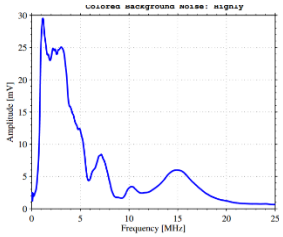


Figure 4: Coloured background noise simulated in MATLAB

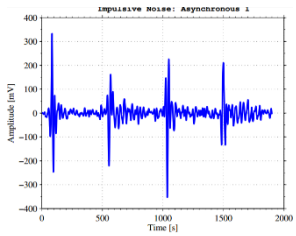


Figure 5: Impulsive noise simulated in MATLAB

G. Random Number Generator

The random number generator (RNG) is implemented using the concept of linear feedback shift register (LFSR) with slight modifications. LFSR's are easily implemented using simple exclusive OR logic gates for the taps and D-type flip-flops for the shift register [19]. By performing left shift and XOR operations, and tapping it on specific points, a random number is generated. In this study, the RNG developed uses a 32-bit vector with only three taps. Figure 6 shows the logic circuit.

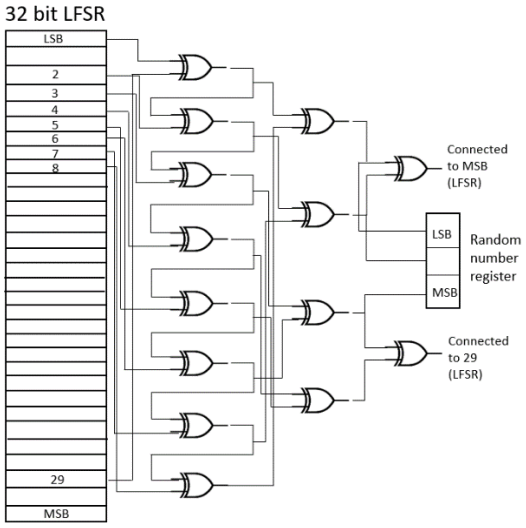


Figure 6: Random Number Generator with Three Taps

Initially, the RNG needs a constant state to generate succeeding values. This is done through simultaneously shifting of the register and performing the XOR operation for the taps [20]. The initial value can be mathematically represented as shown in Equation (12) and referred to as the "seed" [21].

$$Reg = \{X_0\ (BIT\ 31\ to\ BIT\ 0\} \quad (12)$$

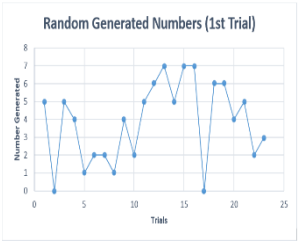
where: Reg = 32-bit vector register; X_o = 32 bit seed;

The series of vectors produced by the RNG is pseudo-random, meaning, the vectors possess the statistical property of randomness but not entirely random. This is because the values are only repeating in powers of 2. Thus, the values or the intervals can be computed using Equation (13) [22].

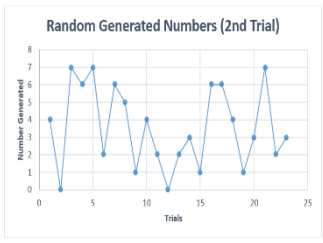
$$Interval = 2^n - 1 \quad (13)$$

In a truly random sequence, it should be occurring at varying intervals. Therefore, by making use of a maximum length sequence generator (MLSG) with k number of taps, better randomness can be achieved [23].

Test results for the random number generator are shown in Figure 7(a) and 7(b) and were observed to be random enough based on varying intervals.



(a) Random number generation plot for the first trial.



(b) Random number generation plot for the second trial

Figure 7: Random number generation with two different trials

IV. Results and Discussions

The test setup involves interconnection between the analogue front-end modules and the FPGA. Figure 8 shows the setup used to perform all the tests for the hardware part. The results from the actual hardware were captured through the oscilloscope. To properly test the hardware, module per module test were done to ensure accuracy. Before the actual hardware test, parameters are first simulated in Xilinx and gathered through iSim, a simulator specifically for ISE designs.

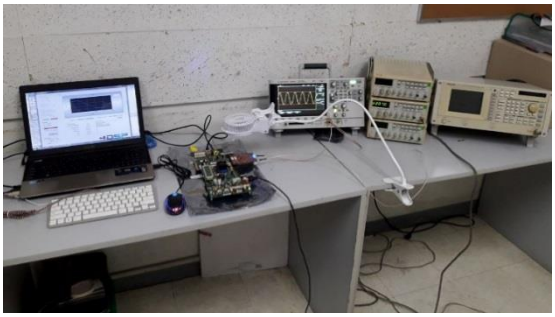
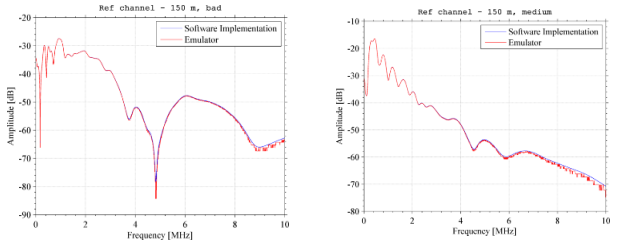


Figure 8: Complete test setup of the channel emulator

A. Transfer Function Implementation Accuracy Test

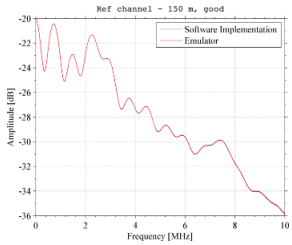
To properly emulate the channel, the channel transfer function was first simulated when converted to fixed-point representation. This is to assure accuracy when implemented

in the FPGA. Figure 9(a), 9(b), 9(c), and 9(d) shows all four transfer functions for the MATLAB simulation compared with fixed point converted transfer functions. Stored values are taken at 14 bits as its fractional part and 15 bits as the integer part, having a total of 29 bits. Different bits were tested, but due to hardware constraints, only 14 bits were allocated for the fraction. The transfer functions generated with 14 fractional bits implementation is already acceptable as Figure 9 suggests.

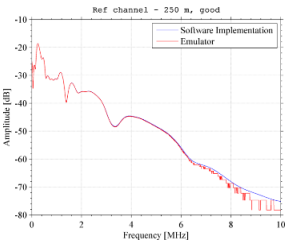


(a) 150 meters reference channel with bad characteristics.

(b) 150 meters reference channel with medium characteristics.

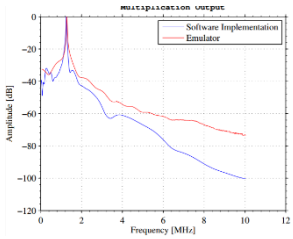


(c) 150 meters reference channel with good characteristics.

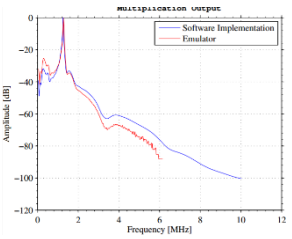


(d) 250 meters reference channel with good characteristics.

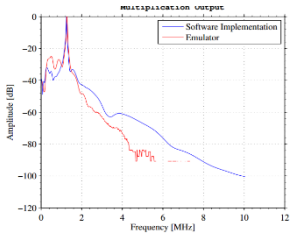
Figure 9: Comparison between simulation and hardware implementation of the transfer function



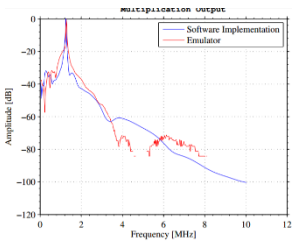
(a) 150 meters good multiplication output



(b) 150 meters medium multiplication output



(c) 150 meters bad multiplication output



(d) 250 meters good multiplication output

Figure 10: Multiplication Output of All four channel transfer functions

B. Algorithm Accuracy Test

The next module tested the multiplication algorithm which mimics the passing of the signal through the channel. Similar to the previous test, values were simulated in Xilinx for the linear combination of the regression block, FFT block, multiplier block and the channel block. Testing was all done for the four transfer functions with an input signal frequency of 1.23 MHz. Noticeable changes can be seen in the values that have extremely low amplitudes due to the number of bits used as stated earlier. Figure 10 shows the normalised magnitude of the multiplied functions for all four channel transfer functions. An average of 0.4 % error between the software simulation results and the Xilinx results is logged for all the reference channels. The critical value is that for the centre frequency, which basically is the input signal frequency. The other frequencies show very small values; hence, the very small values cannot be detected anymore by the emulator since it can only detect fractional bits up to 14 bits. This does not affect the overall performance of the emulator since it is only the input frequency that is required to have a significant magnitude.

C. Random Noise Selection Output and Hardware Results

For the final test setup, the full modules were implemented. Results were obtained from Matlab, Xilinx, and the actual hardware emulator. The selection of transfer functions was made user dependent. A simple multiplexer is used for the selector with the select lines acting as the user input. Table 2 summarises the combinations used in the test.

Table 2
Selector Input Combination to Output Certain Transfer Functions.

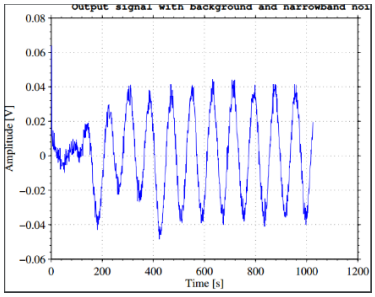
Binary Combinations	Channel Transfer Function Outputs
0000	No Transfer Function
0001	150 meters with good characteristics
0011	150 meters with medium characteristics
0111	150 meters with bad characteristics
1111	250 meters with good characteristics

On the other hand, the random number generator served as the random selector for the noise models to mimic the randomisation of noise present in a power line. Testing involved stopping the randomisation counter through a switch to select the needed noise combination. There was a total of 8 combinations for the noise models and are summarised in Table 3 with its corresponding binary input equivalent.

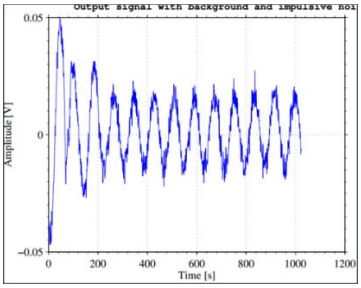
Table 3
Binary Combinations of the Random Noise Selector.

Binary Combinations	Noise Selected
000	No Noise
001	Impulsive Noise
010	Narrowband Noise
011	Background Noise
100	Impulsive + Background
101	Impulsive + Narrowband
110	Background + Narrowband
111	All Noise

The selection process is done by inserting the output from the random number generator into the multiplexer. The multiplexer contains the input noise models and outputs the selected noise combination. Figure 11(a) and 11(b) presents the simulated outputs of noise in iSIM.



(a) Channel emulator output with randomly selected noise (background noise + narrowband noise)

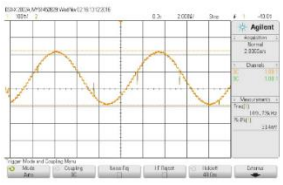


(b) Channel emulator output with randomly selected noise (background noise + impulsive noise)

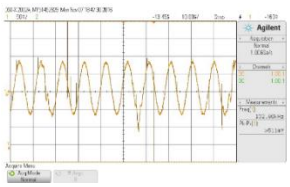
Figure 11: Channel emulator outputs for a 1.23 MHz input signal

In the actual hardware test, signals are applied using an RF generator. It goes through the AFE that converts it to the form read by the FPGA. Several tests were conducted utilising the RNG to generate the pattern that would select the type of noise as provided in Table 3. The selector switch that selects the channel transfer function is also utilised. Figure 12 shows output waveforms measured using a digital oscilloscope taken at the output of the DAC for four different scenarios. The results were compared with Matlab simulation and Xilinx, and their waveforms are comparatively proximate.

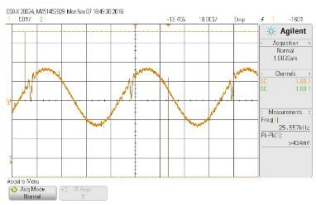
One difference seen in the hardware results is the effect of all three noise models to the signal. It was observed to have completely distorted the signal, and none of its remaining characteristics can be seen. Overall, the main hardware implementation has proven the effects of the channel are correlated with the input signals.



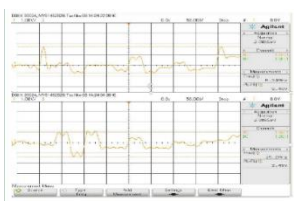
(a) Output signal with no transfer function used



(b) Output signal with transfer function and background noise.



(c) Output signal with transfer function and impulsive noise



(d) Output signal with transfer function and all noise models present

Figure 12: Hardware implementation of channel emulator results

D. Channel Parameter Results

To verify the channel's integrity, the signal to noise ratio (SNR) was obtained for the system. This test was conducted by directly connecting the system's output to a spectrum analyser which calculates the signal power. The input signal's power was computed as the reference signal power for all the tests. Noise power was obtained in a similar manner. This is made possible by removing all the modules except the noise models and the IFFT block and directly connecting it to the input for the DAC. SNR results are shown in Table 4.

Table 4
Signal to Noise Ratio for Input Signal with Noise Models

Output Signal	Signal to noise ratio
Signal with background noise	22.15 dB
Signal with impulsive noise	12.93 dB
Signal with narrowband noise	39.11 dB
Signal with all noise	-13.86 dB

The values measured were consistent with the degree of the effect of each noise. Impulsive noise proved to have the most significant effect of the three noise models which is consistent with its MATLAB simulation. This is followed by background noise and lastly the narrowband noise. Also, since the output of the system when all three noise models are present completely distorted the signal, the expected SNR would be below 0 dB.

V. Conclusion

The emulator presented in this paper successfully implemented the multipath models developed by Zimmermann using Virtex 6 FPGA and FMC151 AFE with channel selection and random noise generation. Since the broadband channel uses higher frequency range, it is critical to consider the memory that the overall design will occupy in the FPGA. A conservative 14 bits for the fractional part of the fixed-point representation of the input and the transfer function was used to economically include at least four reference channels and three noise models. A 0.4% percent difference between the simulated value and the hardware implemented transfer function is logged. The random generation of noise was also demonstrated and was successfully implemented.

Acknowledgements

The authors would like to thank URCO (University Research Coordination Office) of De La Salle University-Manila for funding this research.

References

[1] M. Bauer, W. Liu and K. Dostert, "Channel emulation of low-speed plc transmissiion channels," *Power line communications and its applications (ISPLC)*, pp. 267-272, 2009.

[2] S. Galli, A. Scaglione and Z. Wang, "For the grid and through the grid: The role of power line communications in the smart grid," *Proceedings of the IEEE*, pp. 998-1027, 2011.

[3] W. Zhu, X. Zhu, E. Lim and Y. Huang, "State-of-art power line communications channel modeling," *Procedia computer science, first international conference on information technology and quantitative management*, vol. 17, pp. 563-570, 2013.

[4] J. Anatory and N. Theethayi, Broadband power-line communications systems: theory and applications, WIT Press, 2010.

[5] Y. Zhao, X. Zhou and C. Lu, "A new channel emulator for low voltage broadband power line communication," in *2013 IEEE 10th International Conference on ASIC*, Shenzhen,, 2013.

[6] H. Çelebi, S. Güzelgöz, T. Güzel and H. Arslan, "Noise and channel statistics of indoor power line networks," in *2011 18th International Conference on Telecommunications*, Ayia Napa, 2011

[7] H. Gassara, F. Rouissi and A. Ghazel, "Narrowband stationary noise characterization and modelling for power line communication," *Communications and information technologies (ISCIT)*, 2013.

[8] H. Ferreira, L. Lampe, J. Newbury and T. Swart, Power line communications: theory and applications for narrowband and broadband communications over power lines, John Wiley & Sons Ltd, 2010.

[9] M. M. R. S P Majumder, "Performance analysis of a power line communication with fading over non-white impulsive noise channel," *Information Technology: Towards New Smart World (NSITNSW) 2015 5th National Symposium on*, vol. 57, pp. 1-5, 2015

[10] N. Weling, "Flexible FPGA-based powerline channel emulator for testing MIMO-PLC, neighborhood networks, hidden node or VDSL coexistence scenarios," *Power Line Communications and Its Applications (ISPLC)*, pp. 12-17, April 2011

[11] F. Versolatto and A. Tonello, "Analysis of the PLC channel statistics using a bottom-up random simulator," *Power line communications and its applications (ISPLC)*, pp. 236-241, March 2010.

[12] Tonello, F. Versolatto and B. Bejar, "A Top-Down Random Generator for the In-Home PLC Channel," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Houston, TX, USA, 2011.

[13] F. Cañete, L. Diez, J. Cortes, J. Sanchez-Martinez and L. Torres, "Time-varying channel emulator for indoor power line communications," in *Global Telecommunications Conference*, 2008.

[14] J. Albanus, "Coding schemes used with data converters," Texas Instruments Incorporated, Dallas, 2015.

[15] M. Zimmermann and K. Dostert, "A multipath model for the powerline channel," *IEEE Transactions on Communications*, vol. 57, pp. 553-559, 2002.

[16] "Working group on plc," [Online]. Available: http://www.plc.uma.es/channels.htm.

[17] J. Cortes, L. Diez, F. Cañete and J. Sanchez-Martinez, "Analysis of the indoor broadband power-line noise scenario," *IEEE Transactions on Electromagnetic Compatibility*, vol. 52, pp. 849-858, 2010.

[18] N. Andreadou and F. N. Pavlidou, "Modeling the noise on the OFDM power-line communications system," *IEEE Transacions on Power Delivery*, vol. 25, pp. 150-157, 2010.

[19] H. Lv, J. C. Fang, J. X. Xie and P. Qi, "Generating of a nonlinear pseudorandom sequence using linear feedback shift register," in *2012 International Conference on ICT Convergence (ICTC)*, Jeju Island, 2012.

[20] K. Saluja, *Linear feedback shift registers theory and applications*, Madison: University of Wisconsin-Madison, 1991.

[21] H. Pan, E. Hou and N. Ansari, "Enhanced name and vote separated E-voting system: an E-voting system that ensures voter confidentiality and candidate privacy," in *Security and Communication networks*, 2014.

[22] H. Pan, E. Hou and N. Ansari, "M-NOTE: A multi-part ballot based e-voting sytem with clash attack protecion," in *IEEE international conference on communications*, 2015.

[23] H. Lv, J. C. Fang, X. J. Xie and P. Qi, "Generating a nonlinear pseudorandom sequence using linear feedback shift register," in *International Conference on ICT Convergence (ICTC)*, Jeju Island, 2012.