Hardware Discrete Channel Emulator

Emmanuel Boutillon, Tang Yangyang, Cédric Marchand, Pierre Bomel

Hardware Discrete Channel Emulator

Emmanuel Boutillon, Yangyang Tang, Cédric Marchand and Pierre Bomel
Université Européenne de Bretagne, Lab-STICC, CNRS, UBS BP 92116 Lorient, France
{first-name}.{last-name}@univ-ubs.fr

ABSTRACT

In this paper, the emulation environment named Hardware Discrete Channel Emulator (HDCE) has been developed as a coherent framework to emulate on a hardware device (FPGA as the implementation platform in the verification) and simulate on a computer the effect of an Additive White Gaussian Noise (AWGN) in a base band channel. The HDCE is able to generate more than 180 M samples per second for a very low hardware cost, which has been achieved in an efficient architecture. Using the HDCE, the performance evaluation of a coding scheme for a BER of 10-9 requires only one minute of emulation time.

KEYWORDS: Channel emulation, Additive White Gaussian Noise, BER/FER performance evaluation, Monte-Carlo simulation, Synthesis, VHDL.

1. INTRODUCTION

Advanced digital wireless communication systems often require an appropriate trade-off between complexity and performance of an efficient iterative decoder design. In practice, from the floating point to the fixed-point hardware description, many parameters (reliability message length, digital wordsize, rounding and quantization operations, etc.) should be jointly optimized. However, these parameters interact in a non-linear way and the selection of the optimal algorithm is a very high time-consuming task. Usually, the formal expression of Bit Error Rate (BER) or the Frame Error Rate (FER) expressed in [1] has been used to predict the performance of the system. The conventional solution is the Monte-Carlo simulation that evaluates the BER, which gives an estimation of the error correcting capability of the decoder.

The Monte-Carlo simulation method is traditionally performed by software programs. With this approach, a FER around 10-8 requires one or two weeks of simulation. To speed up these very long simulations, some software approaches are proposed, such as, a reduced Monte-Carlo simulation method [2] that re-runs only erroneous codewords obtained from an initial "classical" Monte-Carlo simulation. We then propose a technique called the distance-based method which is based on the direct evaluation of a distance between the soft output of the sub-optimal decoder and the soft output of a reference decoder [3]. Although these methods reduce the simulation time, the software based execution (for instance, executing applications on a conventional CPU cluster) is still infeasible due to the high power consumption and physical space cost. Consequently, we have turned our attention to the hardware accelerator based simulation.

In our previous works [4, 5, 6], as well as in the works of Dong-U Lee et al. [7, 8], the hardware emulation leads to a significant speed-up factor (from a few hundreds to a few ten-thousands) in terms of simulation time. Those designs are based on the realization of a normal random variable by using the Box Muller method [9], the Wallace method [10], or Box Muller and Centre-Limit theorem. The normalized variable is then multiplied by the variance and quantified. In this paper, the proposed approach is to directly generate the quantified Gaussian variable by the application of the alias method [11]. We denote the conception of a hardware channel emulator, called HDCE for Hardware Discrete Channel Emulator. The HDCE aims at emulating the effect of a base band discrete channel and has several properties: high accuracy, high speed (more than 180 M samples per second) and also the capacity of seamless switching from hardware emulation to software simulation and vice-versa. The last feature will allow us to combine a reduced Monte-Carlo simulation method [3] and hardware emulation, thus giving a set of very efficient and complementary methods to perform the optimization of decoders. Since the Gaussian channel is widely employed as the universal discrete channel, the proposed HDCE has been dedicated for an Additive White Gaussian Noise (AWGN) channel. Even though, the HDCE still satisfies different discrete channel models. It only requires its user to focus on the adaptation of distributions for the channel specification.

This paper is divided into seven sections. First, the discrete Gaussian channel model is depicted in Section 2. Then, the method of Gaussian random variable generation is presented in Section 3. Section 4 presents the architecture of the HDCE. The applications of the HDCE are showed in Section 5. In Section 6 we demonstrate the experimental result of the HDCE and give the conclusion in the final section.

2. GAUSSIAN CHANNEL MODEL

According to the Noisy Channel Coding theorem developed by C. E. Shannon in 1948 [12], a binary source generates bits that are encoded, modulated and affected by a noise in the channel. In our design, we directly consider the AWGN channel combined with the Analog-to-Digital Converter (ADC), as shown in Figure 1.

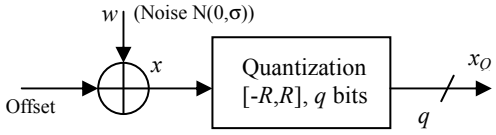

Figure 1. Model of the Channel

A signal of amplitude O (Offset) is sent through a Gaussian channel. The received signal is then $x = O + w$, where w is a realization of a Gaussian random variable of zero mean and variance σ, derived from Normal distribution theorem, the probability of an observation received from a Gaussian channel is,

$$P(X=x)=\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(x-O)^2}{2\sigma^2}}$$

Signal x is then quantized on q bits according to the following three equations:

$$x_Q = sat\left(floor(\frac{x}{R}(2^{q-1}-1)+0.5), 2^{q-1}-1\right) \quad (1)$$

where $sat(a, b) = a$ while a belongs to [$-b$, b], otherwise, $sat(a, b) = sign(a)\times b$. R is the interval dynamic range of the quantization (data are quantized between [$-R$, R]). According to (1), with $-2^{q-1}+1 < a < 2^{q-1}-1$, if $x_Q = a$, then:

$$a \le \frac{x}{R}(2^{q-1}-1)+0.5 < a+1 \quad (2)$$

$$\frac{R}{2^{q-1}-1}(a-0.5) \le x < \frac{R}{2^{q-1}-1}(a+0.5) \quad (3)$$

Thus, $P(xQ = a)$ for $-2^{q-1}+1 < a < 2^{q-1}-1$ can be expressed as:

$$P(x_Q=a)=\frac{1}{\sqrt{2\pi}\sigma}\int_{(a-0.5)\Delta}^{(a+0.5)\Delta} e^{\frac{-(\tau-O)^2}{2\sigma^2}}d\tau \quad (4)$$

where $\Delta = \frac{R}{2^{q-1}-1}$.

Using $u=\frac{\tau-O}{\sqrt{2}\sigma}$, we obtain

$$P(x_Q=a)=\frac{1}{\sqrt{\pi}}\int_{low(a)}^{high(a)} e^{-u^2}du=\frac{1}{2}\left(erf(high(a))-erf(low(a))\right) \quad (5)$$

where $low(a)=\frac{(a-0.5)\Delta-O}{\sqrt{2}\sigma}$, $high(a)=\frac{(a+0.5)\Delta-O}{\sqrt{2}\sigma}$

and $erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$

For a = amin = -2q-1+1, we obtain:

$$P(x_Q=a_{\min})=\frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{(-2^{q-1}+1.5)\Delta} e^{\frac{-(\tau-O)^2}{2\sigma^2}}d\tau \quad (6)$$

$$P(x_Q=a_{\min})=\frac{1}{\sqrt{\pi}}\int_{-\infty}^{high(a_{\min})} e^{-u^2}du=\frac{1+erf(high(a_{\min}))}{2} \quad (7)$$

Similarly, for a = amax = 2q-1-1, we obtain:

$$P(x_Q=a_{\max})=\frac{1}{\sqrt{\pi}}\int_{low(a_{\max})}^{+\infty} e^{-u^2}du=\frac{1-erf(low(a_{\max}))}{2} \quad (8)$$

We propose a component to directly generate the quantized version of the Probability Density Function (PDF). The Gaussian random variable generator is presented in the following section.

3. GAUSSIAN RANDOM VARIABLE GENERATION

The main aspect of the Gaussian channel emulator is the generation of Gaussian random variable. The method is conventionally achieved by the transformation of a uniformly distributed random variable in a non-uniformly distributed random variable. In the literature, four well-known methods are used: the Box-Muller method [9], the Ziggurat method [13], the Wallace method [10] and the combination of Box-Muller method and Centre-Limit theorem. Moreover, D. B. Thomas and W. Luk proposed a non-uniform random number generator that performs automatic customization of the distribution using a hybrid of the piecewise linear approximations and the alias method [14]. In our design, the alias method which is detailed in following is also employed for the Gaussian random variable generation.

3.1. The alias method

The alias method was initially proposed by A. J. Walker in [14]. Briefly speaking, this method allows to transform a uniform distribution *Pu* of natural numbers between [*0*, 2^{q+l}-1] in a non uniform distribution *Pn* of natural numbers between [*0*, 2^{q}-1], where for all elements *i* of [*0*, 2^{q}-1], *Pn(X = i)* is a multiple of $2^{-(q+l)}$. It can be described in two stages:
In the first stage, the uniform distribution *Pu* of integer between [*0*, 2^{q+l}*-1*] is split into two independent uniform distributions Ps_0 and *Prv*, where Ps_0 belongs to [*0*, 2^{q}*-1*] and *Prv* belongs to [*0*, 2^{l}*-1*];
In the second stage, a test is performed on *Prv*. while *Prv* < *Threshold*(Ps_0), then the system output is Ps_0. Otherwise, the system produces an alternative value given by Ps_1 = *alias*(Ps_0). The output distribution is then defined by the value of the two tables *Threshold_table* (size $2^{q} \times l$-bit words) and *alias_value* (size $2^{q} \times q$-bit words). As the two tables are always used at the same address Ps_0, they can be merged into a generic *alias_table* of 2^{q} words of size $l+q$ bits.

In a more general case, we obtain,

$$P_{x=a} = \frac{threshold(a) + \sum_{p=0}^{2^q-1} T(alias(p)=a).(2^l - threshold(p))}{2^{q+l}} \quad (9)$$

where *T(alias(p)=a)* equals *1* while *alias_value(p) = a*, *0* otherwise. From the alias table, the PDF of the generated random variables can be computed.

3.2. Generation of the alias table

The construction of the alias table to obtain a given PDF is not a trivial problem and we have already tried several methods to find an optimal solution. The proposed method separates the problem of quantification of the initial PDF and the problem of the construction of the alias table.

Let *X* be a random variable taking its values in the set [*0*, 2^{q}-1], *X* is characterized by its PDF *{P(X=i), i = 1...2q-1}*. The first step is to quantize the PDF of *X* with $q+l$ bits of precision to obtain an approximated random variable $\hat{X}$. The direct quantization gives:

$$P(\hat{X} = i) = 2^{-(q+l)} \left\lfloor P(X = i) 2^{q+l} + 0.5 \right\rfloor,$$

where $\lfloor x \rfloor$ represents the highest integer smaller than *x*.

Unfortunately, with this method, we have no guaranty of $\sum_{i=0}^{2^q-1} P(\hat{X} = i) = 1$. In fact, due to quantization effect, the sum can have p quantums $\delta = 2^{-(q+l)}$ below or above *1*. In this case, a post-processing is performed to increase or decrease δ in such a way that the summation of *p* probability values remains at value *1*. The *p* values are chosen to minimize the quantization error.

4. ARCHITECTURE

The architecture of the HDCE is composed of two blocks, one to generate the pseudo-random uniform variable using Linear Feedback Shift Register (LFSR) [15], the other to

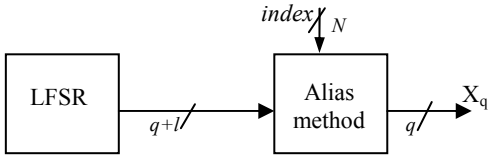

Figure 2. Generation of a sample

translate the uniform random variable into a non uniform random variable using the alias method. To obtain a sample of a particular channel realization, as shown in Figure 2, several parameters should be indicated to the hardware,
- The number of quantization bits *q* of the received signal.
- The internal precision *l* of the alias table.
- The number *N* of different PDF implemented.
- The *index* of the PDF required to generate the sample.

Architectures of these blocks are described in more details in the following subsections.

4.1. LFSR

To simplify the design of the LFSR , we used a size-*63* LFSR that has a Repetition Period (RP) of *RP=263-1*. Using the HDCE at a frequency of *100* MHz, the time before a repetition is then equal to:

$$\frac{2^{63}-1}{10^8} \approx \frac{8.10^{18}}{10^8} = 8.10^{10}\,\text{s} = 2400\ \text{years} \quad (10)$$

Note that $q+l$ *63*-bit LFSR works simultaneously to generate $q+l$ binary RV, the initial state of the $q+l$ LFSR should be different, and ideally, uniformly spread among the RP of the LFSR.
Let *S(k) (62 DOWNTO 0)* be the state of the LFSR at time *k*, than, at time *k+1*, the state *S(k+1)* will be:

$$\begin{cases} S(k+1)(0) = (S(k)(62) \oplus S(k)(61)) \\ S(k+1)(62:1) = (S(k)(61:0) \end{cases} \quad (11)$$

For more information on LFSR, please refer to [16]. A simplified architecture of the LFSR is given in Figure 3.

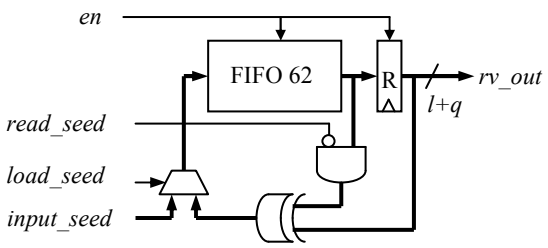

Figure 3. Architecture of the LFSR

The signal en allows enabling the use of the First In First Out (FIFO) (62) and the final register. When *en = 0*, the contents are frozen. The signal *load_seed* allows entering a new seed in the LFSR. The multiplexer controlled by signal *load_seed* inputs (S(61) XOR S(62)) to the FIFO in the normal mode (i.e. *load_seed = 0*). In the *load_seed* mode (*load_seed=1*), it inputs directly the input *input_seed*. In this later mode, the LFSRs are then reinitialized with the new seed in exactly 63 cycles.

Finally, the signal *read_seed* allows to feedback directly FIFO (62) to the input of the FIFO. Thus, in 63 cycles, the 63 values of the LFSR are sent to the output via *rv_out*. Moreover, the final state after 63 cycles is equal to the initial state: the data have only performed a cyclic rotation.

4.2. Direct computation of the LFSR seed

To replay the simulation, the rehabilitation of former random variable is needed. Thus, here, we present an algorithm for the direct computation of the LFSR seed.

The LFSR is a linear structure, i.e. if $S(k) = A(k) + B(k)$, then, for all p, $S(k+p) = A(k+p) + B(k+p)$. The state of the LFSR can be written as:

$$S(k) = \sum_{i=0}^{62} S(k)(i)U_i(0) \qquad (12)$$

where $S(k)(i)$ is the i^{th} component of binary vector $S(k)$ and $U_i(0)$ is an unitary vector of size 63 that contains a single non zero value at position i considered at time $k=0$, i.e., $U_i(0)(j) = 0$ if $j \neq i$ and $U_i(0)(i) = 1$.

By using the linearity of the LFSR structure, for any $p>0$, we can directly compute the state $S(k+p)$ of the LFSR at time $k+p$ as a function of the vector $U_j(p)$. Then, the computation of $S(k+p)$ requires only the XOR of 63 vectors.

The question that now arises is how to compute $U_j(p)$ in a simple way. Let us first assume that p is a power of 2, i.e., $p=2k$. For $k=0$, $U_j(1)$ is computed using (13). Then, by recursion, assuming that the $U_j(2k)$, j=0..62 are known for a given k, then, for j=0 ... 62 , $U_j(2^{k+1})$ are computed using:

$$U_j(2^{k+1}) = U_j(2^k + 2^k) = \sum_{i=0}^{62} U_j(2^k)(i)U_i(2^k) \qquad (13)$$

In the general case, $p = \sum_{k=0}^{62} a_k 2^k$. Let $p(r)$ be the partial sum $p(r) = \sum_{k=0}^{r} a_k 2^k$, then $U_j(p(r+1))$ can be computed recursively from $r = 0$ to 61:

If $a_{k+1}=0$, then $U_j(p(r+1)) = U_j(p(r))$

If $a_{k+1}=1$, then $U_j(p(r+1)) = \sum_{i=0}^{62} U_j(p(r))(i)U_i(2^{k+1})$

To sum up, it is possible, at a cost of $63^2 = 4048$ 63-bits XOR, to compute the $S(k+p)$ for any $S(k)$ and any p. This algorithm facilitates the computation of the LFSR seed with a given number of cycles and an initial seed.

4.3. Alias method

With the generation of a pseudo-random uniform variable thanks to the LFSR, the translation of the uniform into a non uniform random variable will be accomplished with the alias method. After the construction of the alias table as explained in Section 3, the alias method is simple and can be expressed in the following few pseudo-code words.

```
Input: two uniform random variables S0 (on q
       bits) and rv (on l bits).
Output: "alias(S0)", while threshold(S0)< rv,
        "S0", otherwise.
```

To allow for a high clock frequency, even when $l = 32$ (i.e. if a 32-bits comparator is required), we have defined a pipelined structure which is shown in Figure 4.

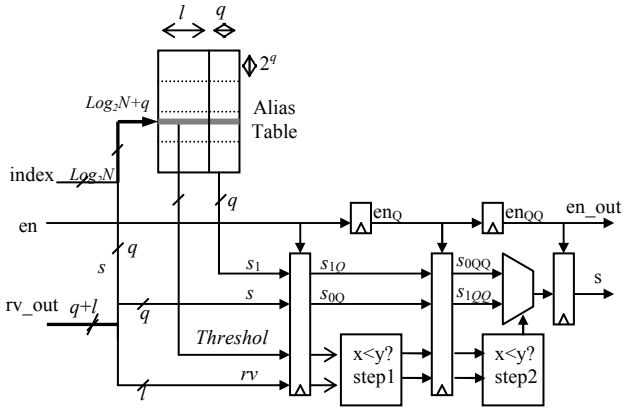

Figure 4. Architecture of the Alias Method

Firstly, the signal *index* denotes the index of the PDF required to generate the sample, and the signal *rv_out* is the uniform variable output of LFSR. The signal *en* follows the pipeline in order to be synchronized with the output of the HDCE. The pipeline structure consists of three phases: the two variables, *threshold* fetched from alias table and *rv_out*, are registered in the first phase; then, the second phase executes the comparison in every *8* bits; at last, the determination will be done. The reconfiguration of the distributions in the hardware is carried out by re-writing the alias table. It allows the system to play several testing scenarios without the need of synthesis, place-route and configuration of the FPGA. In our design, the alias tables are stocked as RAM with size of $(l+q)\times 2^q$.

5. Applications

In this section, we first present two practical applications where the HDCE has been employed as a tool for the design and the test of Low Density Parity Check (LDPC) decoders. We then present other possible application scenarios.

5.1. Test of an LDPC code architecture

Due to the linearity of the LDPC decoder, we first consider an 'all zero codeword' with a Binary Phase-Shift Keying (BPSK) transmission on an AWGN channel. First, we pre-compute an alias table for each required SNR value. These alias tables are stored in a ROM and the index signal addresses the alias table corresponding to the required SNR values.

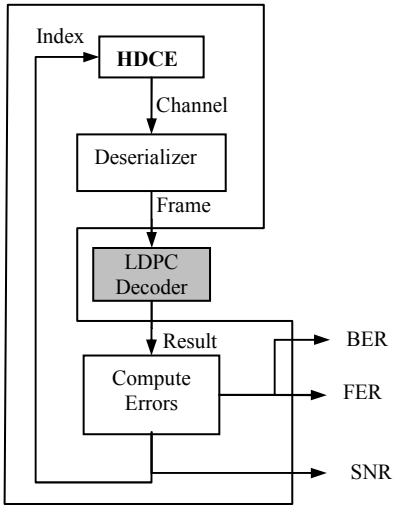

Figure 5. All Zero Codeword Model

Figure 5 shows the all zero codeword model for the test of FER or/and BER. In the compute errors block linked with the decoder output, each non-zero value is counted as a bit error. With the bit error, the frame error is easily deduced and the index (SNR) is incremented when a maximum number of frame errors is reached. This block is low cost and easy to design. The test patch (the HDCE and the compute errors block) can be included as a part of a decoder chip or IP for built-in SNR estimation and testing purposes at a low area cost. It would take less than 2 % of the area of a DVB-S2 LDPC decoder [17].

In the second application, the goal is to test the performance of the LDPC codes in a communication model. In Figure 6, the codeword from the encoder is sent serially to emulate a BPSK modulation on the AWGN channel. This bit is concatenated with the SNR value to address the correct alias table. One alias table is generated for each signal value (-1 and +1) and for each SNR value, e.g. SNR from 0 to 2dB by step of 0.1, with a BPSK transmission requires 20×2 alias table. The FIFO stores the source words until the codeword is decoded. Then the compute errors block compares the two words and deduces bit errors. The same test model can be used for a non binary LDPC codes.

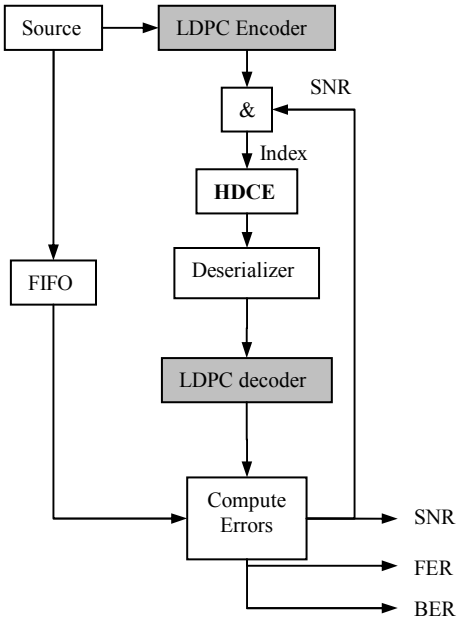

Figure 6. Communication System Model

5.2. Other applications

The previously described models have been used to test the LDPC encoder and decoder, but it also suits any other error control codes.

Additionally, in terms of channel emulation, channels other than AWGN can be also emulated. The Rayleigh fading channel is a reasonable model to simulate the effect of heavily built-up urban environments on terrestrial wireless communication. Usually, the Rayleigh channel can be built from two uncorrelated variables: Y= R.X + G where R is a variable with Rayleigh distribution, G is a variable with Gaussian distribution, X the transmitted signal and Y the received signal.

The HDCE can be used to emulate at high frequency a Poisson distribution, exotic distribution, or any required discrete distribution. It is also possible to manage on the fly distribution evolution with the use of the index as shown in the applications.

6. EXPERIMENTAL RESULT

6.1 Complexity & Performance of the HDCE

The synthesis of the HDCE has been completed for different FPGA targets. Table 1 provides the result obtained for two platforms of Xilinx kits. The LFSRs have been assigned in dual port RAM in terms of Look Up Tables (LUT) by the synthesis tool. Substantially, the complexity is evaluated as the number of LUT.

Table 1. Complexity & Performance of the HDCE for Several Sets of Parameters

Virtex2P Xc2vp2	LUTs as logic	LUTs as RAM	Utilization Of LUTs	F_{clock} max (Mhz)
N=1, q=3, l=16	94	114	7%	300
N=2, q=6, l=16	130	440	20%	294
N=2, q=6, l=32	224	760	35%	284
N=8, q=6, l=32	344	2584	104%	232
N=64, q=6, l=24	1194	15500	592%	183

Virtex5 Xc5vlx30	LUTs as logic	LUTs as RAM	Utilization Of LUTs	F_{clock} max (Mhz)
N=1, q=3, l=16	49	57	1%	487
N=2, q=6, l=16	49	110	1%	441
N=2, q=6, l=32	82	190	1%	441
N=8, q=6, l=32	124	646	4%	441
N=64, q=6, l=24	398	3870	22%	441

Basically, the maximum frequency of the HDCE is almost independent of its configuration. The complexity of the HDCE depends on the parameters. In this case, if the parameters lead to a high value of LUT used as RAM, it is possible to impose directly the embedded RAM of the FPGA, or eventually, an external RAM.

7.2. Accuracy of the HDCE

The evaluation of the reliability of a Gaussian channel emulator mainly stems from the trivial difference between empirical PDF and standard PDF. Nevertheless, observing from a novel standpoint, the precision of the tails of the Gaussian distribution is a good indicator of accuracy. Thus, we evaluate our emulator through the difference of the tails. In table 2, we give the percentage of the difference to a reference standard PDF of a Gaussian distribution (0, 1) for x = 2σ, 3σ, 4σ and 5σ with various l value. The Ref. PDF represents the percentage of standard probability in vary x, also, the Diff. denotes the difference percentage to the Ref. PDF.

Table 2. The Difference Percentage of the Tails

x		2σ	3σ	4σ	5σ
Ref. PDF (%)		6	0.6	0.023	3e-4
Diff. (%)	l=10	0.02	0.06	7	100
	l=16	2e-4	0.006	0.25	12
	l=24	1e-6	3e-5	3e-4	0.0044

7. CONCLUSION

In this paper, we have presented a tool called HDCE for Hardware Discrete Channel Emulator which allows emulating the effect of a base-band Gaussian channel directly in hardware. The HDCE was presented in 4 steps. First, we derived the theoretical model of the HDCE, i.e., we have mathematically defined the expression of the PDF of the output of the channel. Secondly, we explained how to transform a uniform distribution to a given non-uniform distribution thanks to the alias method. Subsequently, the architecture of the HDCE was exhibited by two blocks, the LFSR and the alias method. After which, the applications of the HDCE were demonstrated and it was explained that the HDCE was also compatible with different channel models, other than just the AWGN channel. Finally, we gave the evaluation of the HDCE observed by the trade-off between complexity and performance and the accuracy.

Overall, the proposed HDCE that has been achieved in an efficient architecture is able to emulate the effect of a base band Gaussian channel with several properties: high accuracy, high speed (more than 180 M sample per second) and also the capacity of seamless switching from hardware emulation to software simulation and vice-versa.

This last feature allows for the combining of a Reduced Monte-Carlo Simulation method and hardware emulation, thus, providing a set of very efficient and complementary methods to perform the optimization of decoders.

ACKNOWLEDGEMENTS

This research has been supported by the DAVINCI [18] project, where HDCE is used as the emulation environment.

REFERENCES

[1] J.G. Proakis, DIGITAL COMMUNICATION, 3rd ed., McGraw Hill International Editions, 1995.

[2] E. Boutillon, C. Douillard and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation Issues," Proc. IEEE, vol. 95, no. 6, pp. 1201-1227, 2007.

[3] A. Singh, A. Al-Ghouwayel; G. Masera and E. Boutillon, "A new performance evaluation metric for sub-optimal iterative decoders," IEEE Communication letter, vol. 13, no. 7, pp. 513-515, 2009.

[4] J. L. Danger, A. Ghazel, E. Boutillon and H. Laamari, "Efficient FPGA implementation of Gaussian noise generator for communication channel emulation," Proc. ICECS, vol. 1, pp. 366-369, 2000.

[5] A. Ghazel, E. Boutillon, J. L. Danger and G. Gulak, "Design and performance analysis of a high speed AWGN communication channel emulator," Proc. PACRIM, vol. 2, pp. 374-377, 2001.

[6] E. Boutillon, J. Danger, and A. Gazel, "Design of high speed AWGN communication channel emulator," Proc. AICSP, vol. 34, no. 2, pp. 133-142, 2003.

[7] D. Lee, W. Luk, J. Villasenor, G. Zhang and P. Leong, "A hardware Gaussian noise generator using the Box-Muller method and its error analysis," IEEE Trans. Computers, vol. 55, no.6, pp. 659-671, 2006.

[8] D. Lee, W. Luk, G. Zhang, P. Leong, and J. Villasenor, "A hardware Gaussian noise generator using the Wallace method," IEEE Trans. VLSI System, vol. 13, no. 8, pp. 911-920, 2005.

[9] G. Box and M. Muller, "*A note on the generation of random normal deviates*," Jour. AMS., vol. 29, pp. 610-611, 1958.

[10] C. Wallace, "Fast pseudorandom generators for normal and exponential variates," ACM Trans. Math. Software, vol. 22, no. 1, pp. 119-127, 1996.

[11] A. J. Walker, "An efficient method for generating discrete random variables with general distributions," ACM Trans. Math. Software, vol. 3, no. 3, pp. 253-256, 1977.

[12] C. E. Shannon, "*A mathematical theory of communication*," Jour. Bell Sys. Technical, vol. 27, pp. 379-423 and 623-656, 1948.

[13] G. Marsaglia and W. Tsang, "*The Ziggurat method for generating random variables*," Jour. Statistical Software, vol. 5, no. 8, pp. 1-7, 2000.

[14] D. Thomas and W. Luk, "Non-uniform random number generation through piecewise linear approximations," Proc. FPL, pp. 1-6, 2006.

[15] D. Knuth, THE ART OF COMPUTER PROGRAMMING, Addison-Wesley Press, vol. 2. Semi numerical Algorithms, 1998.

[16] S. W. Golomb, SHIFT REGISTER SEQUENCES, Aegean Park Press, Laguna Hills, 1982.

[17] S. Müller, M. Schreger, M. Kabutz, M. Alles, F. Kienle and N. When, "A novel LDPC decoder for DVB-S2 IP," Proc. DATE, pp. 1308-1313, 2009.

[18] http://www.ict-davinci-codes.eu/