G-DSP Engine

Simulación de Canal AWGN

Fase 2 — Documentación Técnica del TFG

G-DSP Team

Febrero 2026

Índice

1. Introducción

El modelado del canal de comunicaciones es un componente esencial en cualquier sistema de verificación de módem. En un escenario de laboratorio clásico, el ruido del canal se introduce externamente mediante un generador de ruido calibrado; sin embargo, para una demostración autónoma sobre FPGA — como la que persigue el proyecto G-DSP Engine— resulta imprescindible sintetizar dicho ruido *dentro* de la propia lógica programable.

El objetivo de la Fase 2 es implementar un **canal AWGN** (*Additive White Gaussian Noise*) que permita:

1. Inyectar ruido de distribución aproximadamente gaussiana a las señales I y Q procedentes del transmisor.
2. Controlar la potencia del ruido en tiempo real mediante un registro de magnitud (`noise_magnitude`), facilitando barridos de E_b/N_0 para curvas de BER.
3. Garantizar la independencia estadística entre los canales I y Q.

La cadena completa queda:

$$r_{I,Q}[n] = s_{I,Q}[n] + w_{I,Q}[n] \tag{1}$$

donde $s_{I,Q}$ es la señal transmitida (Q1.11) y $w_{I,Q}$ es el ruido gaussiano generado en hardware.

2. Justificación de Arquitectura: CLT frente a Box-Muller

Se evaluaron tres estrategias para la generación de muestras con distribución gaussiana en hardware:

2.1. Transformada de Box-Muller

La transformada de Box-Muller genera pares de variables gaussianas independientes a partir de dos variables uniformes $U_1, U_2 \in (0, 1)$:

$$Z_0 = \sqrt{-2\ln U_1}\,\cos(2\pi U_2), \qquad Z_1 = \sqrt{-2\ln U_1}\,\sin(2\pi U_2) \tag{2}$$

Coste de implementación. El cálculo de $\ln(\cdot)$ y $\sqrt{\cdot}$ en hardware requiere aproximaciones por CORDIC o tablas de consulta (LUT) almacenadas en BSRAM. Las funciones trigonométricas añaden un segundo bloque CORDIC. La estimación de recursos para la FPGA GW1NR-9 es:

Tabla 1: Coste estimado de Box-Muller en GW1NR-9.

Recurso	**Necesario**	**Disponible**
LUT4	> 1200	8640
BSRAM (18 Kbit)	2–4 bloques	26 bloques
DSP (MULT18)	2–4	10
Latencia	15–20 ciclos	—

El consumo de LUTs y BSRAM es significativo y compite directamente con los recursos necesarios para el filtro RRC, la cadena de sincronización y el pipeline HDMI. **Se descarta** esta opción.

2.2. Método del Zigurat

El método del zigurat es una técnica de rechazo (*rejection sampling*) que encapsula la PDF gaussiana en rectángulos apilados. Su principal inconveniente para hardware es la **latencia variable**: cada muestra requiere un número indeterminado de iteraciones del bucle de rechazo, lo que rompe el determinismo del pipeline. Se necesitaría un FIFO de desacoplamiento y lógica de control adicional. **Se descarta** por complejidad.

2.3. Teorema del Límite Central (CLT) — Opción seleccionada

El Teorema del Límite Central establece que la suma de N variables aleatorias independientes e idénticamente distribuidas (i.i.d.) converge en distribución a una variable gaussiana conforme N crece:

$$Z = \frac{\sum_{i=1}^{N} X_i - N\mu}{\sigma\sqrt{N}} \xrightarrow{d} \mathcal{N}(0,\,1) \quad \text{cuando } N \to \infty \tag{3}$$

donde $X_i \sim \text{Uniforme}[0, 2^{12})$, $\mu = 2^{11} - 0{,}5$ y $\sigma^2 = (2^{12})^2/12$.

Para $N = 16$, la calidad de la aproximación se evalúa mediante la curtosis en exceso:

$$\kappa_{\text{exc}} = \frac{\kappa_4}{\sigma^4} - 3 = \frac{-6/5}{N} = \frac{-6/5}{16} = -0{,}075 \tag{4}$$

Un valor de $|\kappa_{\text{exc}}| = 0{,}075$ frente a los 0 de una gaussiana pura indica que las colas de la distribución son ligeramente más ligeras. En la práctica, esta desviación es indistinguible para pruebas de BER con $E_b/N_0 \geq 4\,\text{dB}$.

Tabla 2: Coste estimado de CLT ($N = 16$) en GW1NR-9.

Recurso	**Estimación**	**Justificación**
LUT4	∼350–500	16 LFSRs + árbol de sumadores
BSRAM	0	Sin tablas de consulta
DSP	1 (MULT18)	Multiplicación por `noise_magnitude`
FF	∼400	16 registros LFSR + pipeline
Latencia	3 ciclos	Determinista, sin rechazo

Coste de implementación. El coste es una fracción menor de los recursos de la FPGA: sin BSRAM, sin CORDIC, sin latencia variable. **Se selecciona** esta arquitectura.

3. Fundamento Matemático

3.1. Distribución resultante de la suma de $N = 16$ uniformes

Sea X_i la salida de 12 bits sin signo del i-ésimo LFSR, con distribución uniforme discreta en $\{0, 1, \ldots, 2^{12} - 1\}$. La suma $S = \sum_{i=1}^{16} X_i$ sigue (para N grande) una distribución de Irwin-Hall, cuya PDF converge a la campana gaussiana.

Los parámetros de S son:

$$\text{Media:} \quad \mu_S = N \cdot \frac{2^{12}-1}{2} = 16 \times 2047{,}5 = 32760 \tag{5}$$

$$\text{Varianza:} \quad \sigma_S^2 = N \cdot \frac{(2^{12})^2}{12} = 16 \times \frac{16\,777\,216}{12} \approx 22\,369\,621 \tag{6}$$

Se centra la distribución sustrayendo $2^{15} = 32768$ (potencia de 2 más cercana a μ_S, implementable como simple inversión de bits), obteniéndose un valor con signo de 17 bits.

3.2. Relación entre `noise_magnitude` y SNR

Sea $M \in [0, 255]$ el valor del registro `noise_magnitude` y $w[n]$ la muestra de ruido a la salida del generador (formato Q1.11). El procesamiento interno es:

$$w[n] = \frac{(S[n] - 2^{15}) \times M}{2^{12}} \tag{7}$$

donde la división por $2^{12} = 2^4 \times 2^8$ corresponde a:

- $\div\, 2^4$: normalización de la suma (dividir entre $N = 16$ para obtener varianza unitaria referida a la escala uniforme).
- $\div\, 2^8$: escalado por el factor $M/256$.

La varianza de $w[n]$ resulta:

$$\begin{aligned} \sigma_w^2 &= \left(\frac{M}{2^{12}}\right)^2 \cdot \sigma_S^2 = \left(\frac{M}{4096}\right)^2 \cdot 16 \cdot \frac{(2^{12})^2}{12} \\ &= \frac{M^2}{4096^2} \cdot \frac{16 \cdot 4096^2}{12} = \frac{M^2 \cdot 16}{12} = \frac{4\,M^2}{3} \end{aligned} \tag{8}$$

Expresando en unidades Q1.11 (donde el fondo de escala es $2^{11} = 2048$), la varianza normalizada es:

$$\sigma_{w,\text{norm}}^2 = \frac{\sigma_w^2}{(2^{11})^2} = \frac{4\,M^2}{3 \times 2^{22}} = \frac{M^2}{3 \times 2^{20}} \approx \frac{M^2}{3\,145\,728} \tag{9}$$

Para la constelación 16-QAM normalizada a potencia unitaria ($P_s = 1$), la relación señal a ruido considerando ambos canales (I y Q, con ruido independiente) es:

$$\text{SNR} = \frac{P_s}{2\,\sigma_{w,\text{norm}}^2} = \frac{3 \times 2^{20}}{2\,M^2} \tag{10}$$

En decibelios:

$$\boxed{\text{SNR}_{\text{dB}} = 10 \log_{10}\left(\frac{3 \times 2^{20}}{2\,M^2}\right) \approx 59{,}0 - 20 \log_{10}(M)} \tag{11}$$

La Tabla 3 presenta valores representativos.

Tabla 3: Mapeo de `noise_magnitude` (M) a SNR.

M	$\sigma_{w,\text{norm}}$	SNR_{dB}	**Observación**
0	0	∞	Sin ruido (bypass)
4	0,0011	46,9	Prácticamente limpio
16	0,0045	34,9	Ruido imperceptible
32	0,0090	28,8	Leve dispersión
64	0,0181	22,8	Nube visible
128	0,0362	16,8	Constelación degradada
200	0,0565	12,9	Límite del módem
255	0,0720	10,8	Ruido máximo

4. Detalles de Implementación RTL

4.1. Banco de 16 LFSRs con polinomios primitivos verificados

La calidad de la aproximación CLT depende críticamente de que las 16 fuentes de ruido uniforme sean **estadísticamente independientes**. Esto se garantiza mediante tres mecanismos:

1. Polinomios primitivos distintos. Cada LFSR utiliza un polinomio trinomial $x^n + x^k + 1$ cuya primitividad ha sido verificada a partir de las tablas de referencia de Xilinx (XAPP052, P. Alfke, 1996). Se emplean 11 anchuras distintas (de 15 a 31 bits) y 5 polinomios recíprocos adicionales ($x^n + x^{n-k} + 1$). La Tabla 4 recoge el conjunto completo.

Tabla 4: Polinomios primitivos de los 16 LFSRs de ruido.

#	**Anchura** n	**Polinomio**	**Período**	**Nota**
0	15	$x^{15} + x^{14} + 1$	32 767	
1	17	$x^{17} + x^{14} + 1$	131 071	
2	18	$x^{18} + x^{11} + 1$	262 143	
3	20	$x^{20} + x^{17} + 1$	1 048 575	
4	21	$x^{21} + x^{19} + 1$	2 097 151	
5	22	$x^{22} + x^{21} + 1$	4 194 303	
6	23	$x^{23} + x^{18} + 1$	8 388 607	ITU-T O.151
7	25	$x^{25} + x^{22} + 1$	33 554 431	
8	28	$x^{28} + x^{25} + 1$	268 435 455	
9	29	$x^{29} + x^{27} + 1$	536 870 911	
10	31	$x^{31} + x^{28} + 1$	2 147 483 647	
11	15	$x^{15} + x^{1} + 1$	32 767	Recíproco de #0
12	17	$x^{17} + x^{3} + 1$	131 071	Recíproco de #1
13	20	$x^{20} + x^{3} + 1$	1 048 575	Recíproco de #3
14	23	$x^{23} + x^{5} + 1$	8 388 607	Recíproco de #6
15	25	$x^{25} + x^{3} + 1$	33 554 431	Recíproco de #7

Nota sobre anchuras sin trinomio primitivo. Las longitudes $n \in \{16, 19, 24, 26, 27, 30\}$ carecen de trinomios primitivos de la forma $x^n + x^k + 1$ en GF(2) [1]. En la versión inicial del diseño se incluían estas longitudes; se detectaron durante la revisión técnica y se sustituyeron por pares recíprocos de longitudes válidas. Dos LFSRs de la *misma* anchura pero con polinomios recíprocos generan secuencias distintas (una es la inversión temporal de la otra), por lo que mantienen la independencia.

2. Semillas (seeds) únicas por LFSR y por instancia. Cada LFSR recibe una semilla calculada combinando:

- El índice $g \in [0, 15]$ del LFSR dentro del generador.
- Un parámetro `INSTANCE_ID` que distingue entre la instancia del canal I (`ID=0`) y la del canal Q (`ID=1`).

La expresión de cálculo de la semilla es:

$$\text{seed}(g, \text{ID}) = \left[(\texttt{0xDEAD\_BEE0} + g \cdot \texttt{0x1357\_9BDF}) \oplus (\texttt{0xA5A5\_A5A5} \gg g) \oplus (\text{ID} \cdot \texttt{0x5A5A\_DEAD})\right] \mid 1 \tag{12}$$

El OR final con 1 impide el estado de bloqueo ($s = 0$) inherente a todo LFSR estándar. El término dependiente de `INSTANCE_ID` garantiza que los 16 LFSRs de la instancia I produzcan secuencias completamente distintas a los 16 LFSRs de la instancia Q, previniendo la correlación cruzada entre canales.

3. Bug corregido: correlación I/Q. En la versión inicial del módulo `channel_top`, ambas instancias del generador de ruido (`u_noise_I` y `u_noise_Q`) se instanciaban sin parámetro diferenciador. Al compartir `INSTANCE_ID = 0`, las 32 semillas resultantes eran idénticas, produciendo ruidos $w_I[n] = w_Q[n]$ perfectamente correlacionados. Esto **invalida** la hipótesis de canal AWGN, que exige $E[w_I \cdot w_Q] = 0$.

La corrección consiste en parametrizar la instanciación:

```
awgn_generator #(.INSTANCE_ID(0)) u_noise_I ( ... );
awgn_generator #(.INSTANCE_ID(1)) u_noise_Q ( ... );
```

Listing 1: Instanciación corregida con `INSTANCE_ID` distinto.

4.2. Árbol de sumadores

Las 16 salidas uniformes de 12 bits se suman mediante un árbol de sumadores combinacional de 4 niveles:

$$\underbrace{16 \to 8}_{\text{Nivel 0}} \to \underbrace{8 \to 4}_{\text{Nivel 1}} \to \underbrace{4 \to 2}_{\text{Nivel 2}} \to \underbrace{2 \to 1}_{\text{Nivel 3}} \tag{13}$$

Cada sumador opera con operandos de DATA_WIDTH $+ 4 = 16$ bits (extensión con ceros para capturar el acarreo). El resultado final es un valor sin signo de 16 bits que se centra y registra en un flip-flop (primera etapa del pipeline).

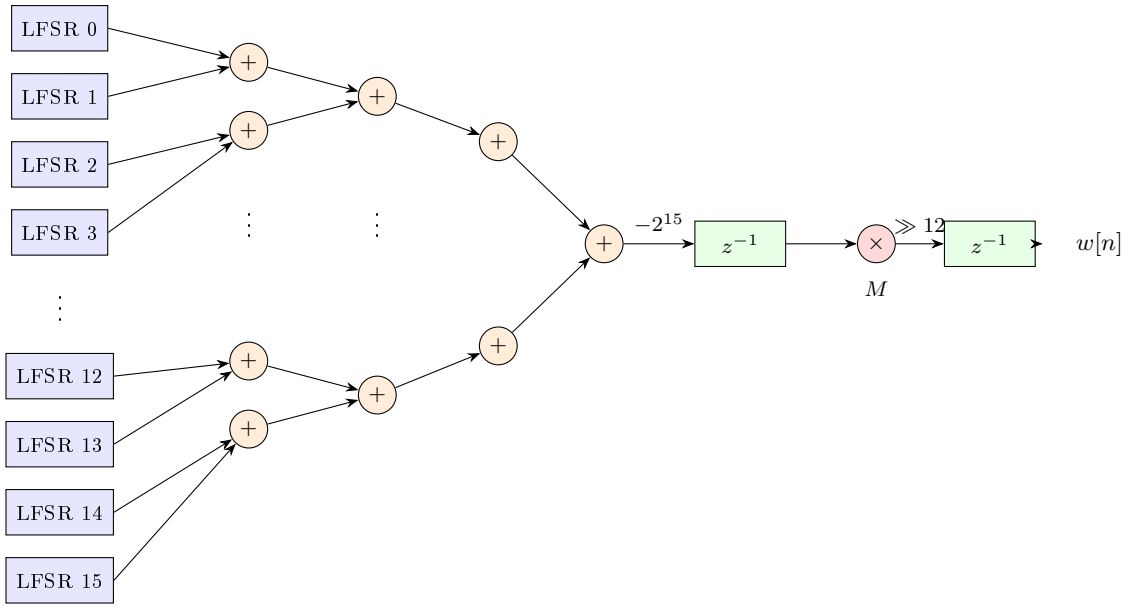

Figura 1: Pipeline del generador AWGN: 16 LFSRs, árbol de sumadores de 4 niveles, centrado, escalado por M y truncación.

4.3. Saturación aritmética en la suma señal + ruido

La operación central del canal es la adición:

$$r[n] = s[n] + w[n] \tag{14}$$

donde tanto s como w son valores Q1.11 de 12 bits firmados (rango $[-2048, +2047]$). La suma puede producir valores fuera de este rango:

$$r \in [-2048 + (-2048),\ +2047 + 2047] = [-4096,\ +4094] \tag{15}$$

lo que requiere 13 bits con signo para representarse sin pérdida.

Por qué no truncar. Si se toma simplemente $r[11:0]$ (los 12 bits inferiores), un valor como $+2100$ (= `0x0834`) se interpretaría correctamente; pero $+2048$ (= `0x0800`) tiene el bit 11 a 1 y se interpretaría como -2048: una **inversión catastrófica de signo** que inyecta un error de 4096 unidades (la distancia máxima posible). Esto distorsiona severamente la constelación y falsea los resultados de BER.

Solución: saturación. Se implementa aritmética de saturación (*clipping*):

$$r_{\text{sat}}[n] = \begin{cases} +2047 & \text{si } r[n] > +2047 \\ -2048 & \text{si } r[n] < -2048 \\ r[n] & \text{en otro caso} \end{cases} \tag{16}$$

En SystemVerilog, la extensión de signo y la comparación se realizan en 13 bits:

```
logic signed [DATA_WIDTH:0] sum_I;  // 13-bit signed
assign sum_I = {tx_I_d[2][DATA_WIDTH-1], tx_I_d[2]}  // sign-extend TX
             + {noise_I[DATA_WIDTH-1],    noise_I};     // sign-extend noise

always_comb begin
    if (sum_I > 13'sd2047)
        rx_I_sat = 12'sd2047;         // SAT_POS
```

```
    else if (sum_I < -13'sd2048)
        rx_I_sat = -12'sd2048;        // SAT_NEG
    else
        rx_I_sat = sum_I[11:0];       // Dentro de rango
end
```

Listing 2: Saturación aritmética en `channel_top.sv`.

Probabilidad de saturación. Para $M = 128$ (SNR $\approx 17\,\text{dB}$), los niveles extremos de la constelación son ± 1943 (margen de $2047 - 1943 = 104$ unidades). La probabilidad de que el ruido supere este margen es:

$$P_{\text{sat}} = 2\,Q\left(\frac{104}{\sigma_w}\right) \tag{17}$$

donde $Q(\cdot)$ es la función Q gaussiana. Para $M = 128$, $\sigma_w \approx 74$, de donde $P_{\text{sat}} \approx 2\,Q(1{,}4) \approx 16\,\%$: la saturación ya es frecuente, pero introduce una compresión suave análoga al recorte de un ADC real. Para $M \leq 64$ (SNR $\geq 23\,\text{dB}$), $P_{\text{sat}} < 0{,}1\,\%$.

4.4. Alineamiento de latencia y pipeline

El generador de ruido introduce 3 ciclos de latencia (LFSRs $\rightarrow$ suma registrada $\rightarrow$ escalado registrado). Para que la adición $s[n] + w[n]$ sea coherente en el tiempo, la señal TX se retarda mediante una línea de registros de 3 etapas:

$$s_{\text{alineada}}[n] = s[n-3] \tag{18}$$

La suma se registra en una cuarta etapa, dando una latencia total del bloque de canal de **4 ciclos de reloj**.

Tabla 5: Latencia del subsistema de canal AWGN.

Etapa	**Ciclos**	**Descripción**
LFSRs y árbol de sumas	1	Comb. + registro
Centrado registrado	1	Pipeline stage
Escalado ($\times M$) + truncación	1	Registro de salida
Suma saturante + registro	1	Salida final
Total	**4**	

5. Resultados de Validación

La validación del canal AWGN se realiza mediante el testbench `tb_channel.sv`, que efectúa un barrido eparamétrico de `noise_magnitude` ($M \in \{0, 16, 64, 128, 255\}$) midiendo:

- Estadísticas de las muestras de salida (media, mínimo, máximo) por nivel de ruido.
- Conteo de eventos de saturación.
- Exportación de las muestras I/Q a fichero CSV para representación gráfica de la constelación.

5.1. Histograma de la distribución de ruido

La Figura 2 muestra el histograma normalizado de 10 000 muestras de ruido generadas con $M = 128$, superpuesto con la PDF gaussiana teórica de varianza equivalente. La concordancia visual confirma la validez de la aproximación CLT con $N = 16$.

Figura 2: Histograma del ruido generado ($M = 128$, $N_{\text{muestras}} = 10\,000$) frente a la PDF gaussiana teórica $\mathcal{N}(0, \sigma_w^2)$.

5.2. Dispersión de la constelación

La Figura 3 ilustra cómo la nube de puntos I/Q se expande al incrementar M. Para $M = 0$ se observa la constelación ideal de 16 puntos; para $M = 255$ los clústeres se solapan, indicando una elevada probabilidad de error de símbolo.

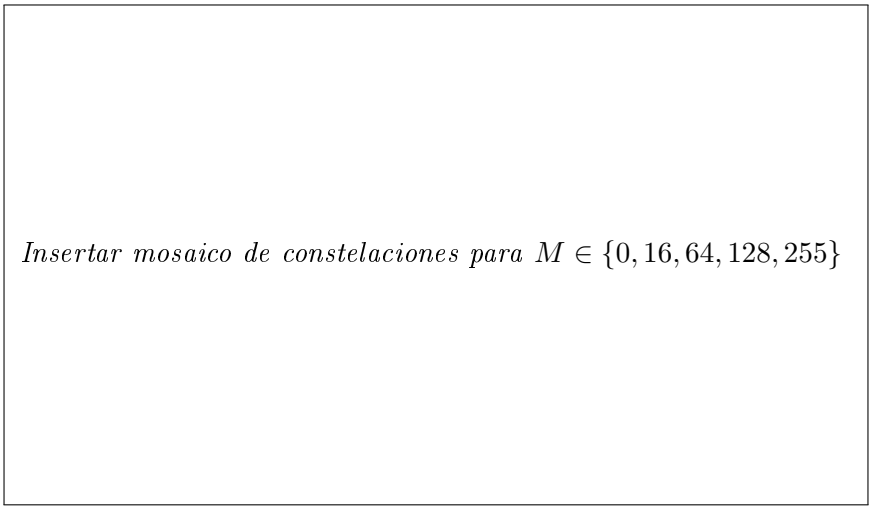

Figura 3: Constelación 16-QAM recibida para distintos valores de `noise_magnitude`.

6. Síntesis de Decisiones de Diseño

Tabla 6: Resumen de decisiones de diseño de la Fase 2 (Canal AWGN).

Aspecto	**Decisión**	**Justificación**
Generación gaussiana	CLT ($N = 16$)	Coste mínimo (∼400 LUT, 0 BSRAM); latencia fija (3 ciclos); curtosis en exceso $< 0{,}08$. Descartados Box-Muller y Zigurat por alto consumo de recursos/latencia variable.
Independencia de fuentes	16 polinomios primitivos verificados	11 anchuras únicas + 5 recíprocos; eliminadas las anchuras sin trinomio primitivo ($n \in \{16, 19, 24, 26, 27, 30\}$). Referencia: XAPP052.
Independencia I/Q	`INSTANCE_ID`	Parámetro que modifica las 16 semillas de cada instancia; corrige el bug de correlación I/Q detectado en revisión.
Control de SNR	Registro `noise_mag` [7:0]	Escalado lineal; SNR $\approx 59 - 20\log_{10}(M)$ dB; rango ∼11–∞ dB.
Gestión de overflow	Saturación aritmética	Suma en 13 bits; recorte a $[-2048, +2047]$; evita inversión de signo catastrófica de la truncación simple.
Alineamiento temporal	Línea de retardo 3 FF	Compensa la latencia del generador para suma coherente.

Referencias

[1] R. W. Watson, "Table of primitive polynomials over GF(2) of degree up to 100," *Mathematics of Computation*, vol. 16, pp. 368–369, 1962.

[2] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," Xilinx Application Note XAPP052, v1.1, 1996.

[3] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.