G-DSP Engine

Integración Final y Visualización HDMI

Fase 4 — Documentación Técnica del TFG

G-DSP Team

Febrero 2026

Índice

1. Introducción a la Fase de Integración

La Fase 4 del proyecto G-DSP Engine completa la implementación del módem 16-QAM integrando todos los subsistemas desarrollados en las fases anteriores sobre la FPGA Gowin GW1NR-LV9QN88PC6/I5 (Tang Nano 9K). Esta fase abarca:

- Instanciación del módulo `gdsp_top` que conecta TX → Canal → RX en una cadena completa.
- Generación de múltiples dominios de reloj mediante el PLL integrado de Gowin.
- Renderizado en tiempo real de la constelación IQ sobre salida HDMI a VGA 640 ×480 @ 60 Hz.
- Interfaz física con botones y LEDs para control interactivo del nivel de ruido.

El resultado es un demostrador funcional donde el usuario puede observar cómo la dispersión de la constelación aumenta al incrementar la magnitud del ruido AWGN, verificando visualmente el correcto funcionamiento del lazo de sincronización Costas.

2. Arquitectura del Top-Level

El módulo `gdsp_top` actúa como punto de entrada del diseño sintetizable. La Figura 1 muestra su diagrama de bloques simplificado.

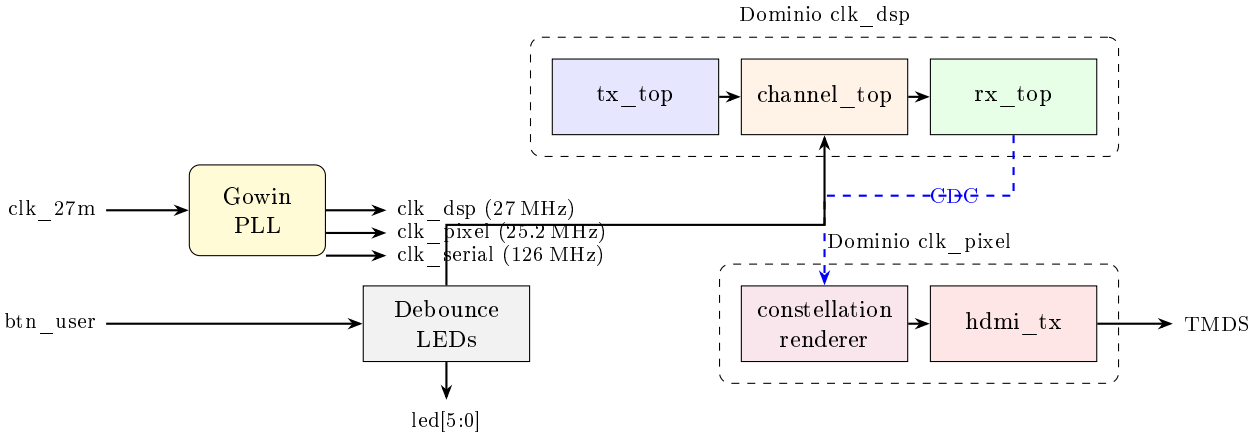

Figura 1: Diagrama de bloques del módulo `gdsp_top`.

3. Gestión de Dominios de Reloj

El diseño opera con tres relojes derivados del oscilador de 27 MHz de la placa Tang Nano 9K mediante un PLL Gowin:

Tabla 1: Dominios de reloj del sistema.

Señal	Frecuencia	Período	Uso
`clk_dsp`	27 MHz	37,04 ns	Lógica del módem (TX/RX/Canal)
`clk_pixel`	25,2 MHz	39,68 ns	Pixel clock VGA 480p60
`clk_serial`	126 MHz	7,94 ns	Serializador TMDS 10:1

3.1. Cruce de Dominios (CDC)

La transición de datos entre `clk_dsp` y `clk_pixel` se realiza en el módulo `constellation_renderer` mediante un sincronizador de doble flip-flop para la señal `sym_valid`:

```
logic sym_valid_sync0, sym_valid_sync1, sym_valid_sync2;

always_ff @(posedge clk_pixel or negedge rst_n) begin
    if (!rst_n) begin
        sym_valid_sync0 <= 1'b0;
        sym_valid_sync1 <= 1'b0;
        sym_valid_sync2 <= 1'b0;
    end else begin
        sym_valid_sync0 <= sym_valid;
        sym_valid_sync1 <= sym_valid_sync0;
        sym_valid_sync2 <= sym_valid_sync1;
    end
end

wire sym_pulse = sym_valid_sync1 && !sym_valid_sync2;  // Rising edge
```

Listing 1: Sincronizador de 2-FF para CDC.

Las coordenadas I/Q se muestrean en el dominio lento (27 MHz) y son estables durante múltiples ciclos de `clk_pixel`, por lo que se capturan directamente tras detectar el flanco sincronizado.

4. Renderizador de Constelación

El módulo `constellation_renderer` convierte las muestras IQ demoduladas en una imagen de vídeo VGA 640×480 a 60 Hz.

4.1. Conversión de Coordenadas Q1.11 a Píxel

Los valores I y Q en formato Q1.11 (rango $[-2048, +2047]$) se escalan a un área de dibujo de 256×256 píxeles centrada en la pantalla:

$$x_{\text{pixel}} = 320 + \left\lfloor \frac{I}{16} \right\rfloor = 320 + (I \gg 4) \tag{1}$$

$$y_{\text{pixel}} = 240 - \left\lfloor \frac{Q}{16} \right\rfloor = 240 - (Q \gg 4) \tag{2}$$

La división por 16 (desplazamiento de 4 bits) mapea el rango $[-2048, +2047]$ a $[-128, +127]$ píxeles respecto al centro. El eje Y se invierte porque las coordenadas de pantalla crecen hacia abajo.

4.2. Técnica de Dibujado de Puntos

Sin memoria de *frame buffer*, el renderizado es directo:

1. Se almacenan las últimas 64 coordenadas de símbolos recibidos en un buffer circular.
2. Al inicio de cada cuadro (flanco de `vsync`), el buffer se invalida para simular decaimiento temporal.
3. Durante el barrido activo, si la posición (h, v) del píxel actual coincide con alguna coordenada almacenada ± 1 (punto de 2×2 píxeles), se genera blanco; de lo contrario, negro.

El resultado es una constelación con persistencia de un cuadro (≈16,7 ms), suficiente para visualización humana.

4.3. Rejilla de Referencia

Además de los puntos de símbolo, se dibujan líneas de referencia (gris oscuro):

- **Ejes**: líneas horizontal y vertical por el centro.
- **Límites de decisión**: líneas en $x, y = \pm 51$ píxeles (correspondientes a ± 816 en Q1.11, punto medio entre niveles QAM adyacentes).
- **Borde del área de dibujo**: rectángulo 256×256.

4.4. Corrección Anamórfica para Pantallas 16:9

4.4.1. Problema: Distorsión por Escalado de Monitores

Los monitores modernos operan nativamente en formato 16:9 (1920×1080, 2560×1440, etc.). Cuando se les alimenta una señal VGA 640×480 (formato 4:3), el escalador automático del monitor estira la imagen para llenar toda la pantalla, aplicando factores de escala diferentes en cada eje:

$$S_x = \frac{1920}{640} = 3{,}00 \quad \text{(escalado horizontal)} \tag{3}$$

$$S_y = \frac{1080}{480} = 2{,}25 \quad \text{(escalado vertical)} \tag{4}$$

Esta diferencia ($S_x/S_y = 1{,}333 = 4/3$) distorsiona cualquier figura circular (como una constelación QAM teórica) en un elipsoide estirado horizontalmente.

4.4.2. Solución: Pre-compresión Computacional

En lugar de intentar cambiar la resolución de salida (lo cual estaría limitado por las restricciones del PLL de la FPGA Gowin GW1NR-9, donde ciertos divisores ODIV no están disponibles), aplicamos una **corrección anamórfica** directamente en el DSP de renderizado.

La técnica consiste en pre-comprimir el eje X por un factor $\alpha = S_y/S_x$ **antes** de dibujar la imagen:

$$\alpha = \frac{S_y}{S_x} = \frac{2{,}25}{3{,}00} = 0{,}75 = \frac{3}{4} \tag{5}$$

Así, cuando el monitor posteriormente estira la imagen por $S_x/S_y = 1{,}333$, la compensación se cancela exactamente:

$$\alpha \times \frac{S_x}{S_y} = 0{,}75 \times 1{,}333 = 1{,}00 \quad \Rightarrow \text{Círculos perfectos} \tag{6}$$

4.4.3. Implementación en Hardware

En `constellation_renderer.sv`, las coordenadas I/Q se convierten a píxeles aplicando escalas diferentes:

$$x_{\text{pixel}} = 320 + \left\lfloor \frac{I \times 3}{32} \right\rfloor = 320 + ((I \times 3) \gg 5) \tag{7}$$

$$y_{\text{pixel}} = 240 - \left\lfloor \frac{Q}{8} \right\rfloor = 240 - (Q \gg 3) \tag{8}$$

Observaciones:

- La ecuación (7) implementa un factor $3/32 \approx 0{,}094$ para el eje X. El factor base de $1/8 = 0{,}125$ se multiplica por $3/4 = 0{,}75$ (anamórfico) resultando en $0{,}125 \times 0{,}75 = 0{,}09375 \approx 3/32$.
- La ecuación (8) usa el factor estándar $1/8$ sin corrección.
- El coste es 1 multiplicador de 13 bits y 1 shifter adicional (despreciable en recursos).
- La constelación aparece "comprimida" en el framebuffer 640×480, pero se visualiza **perfectamente circular** en el monitor 16:9 tras el escalado automático.

Esta técnica es común en producción cinematográfica (formato anamórfico CinemaScope) y se adapta aquí al dominio digital sin coste adicional de hardware significativo.

5. Transmisor HDMI (TMDS)

El módulo `hdmi_tx` implementa la codificación TMDS (*Transition-Minimized Differential Signaling*) requerida por el estándar DVI/HDMI.

5.1. Codificación 8b/10b

Cada canal de color (R, G, B) pasa por un codificador 8b/10b que:

1. Calcula el número de unos en el byte de entrada.
2. Decide entre codificación XOR o XNOR para minimizar transiciones.
3. Aplica inversión condicional para mantener el balance DC (± 0 acumulado de unos y ceros).

Durante el intervalo de *blanking*, se transmiten símbolos de control que codifican `hsync` y `vsync`.

5.2. Serialización 10:1

Los 10 bits TMDS se serializan a 126 MHz usando un registro de desplazamiento. LSB se transmite primero. En síntesis para Gowin, se emplearía el primitivo `OSER10` para eficiencia.

5.3. Salida Diferencial

Las señales serializadas se envían a pads configurados como `LVCMOS33D`, que emulan LVDS con swing reducido. Los pines de salida son:

Tabla 2: Pinout HDMI en Tang Nano 9K.

Señal	Pin (P/N)	Función
`tmds_clk`	69 / 68	Pixel clock codificado
`tmds_data[0]`	71 / 70	Canal Azul (+ sync)
`tmds_data[1]`	73 / 72	Canal Verde
`tmds_data[2]`	75 / 74	Canal Rojo

6. Interfaz Física: Botones y LEDs

6.1. Control de Ruido mediante Botón S1

El botón S1 (activo bajo, con pull-up interno) cicla el parámetro `noise_magnitude` entre cuatro niveles predefinidos:

$$0 \rightarrow 20 \rightarrow 50 \rightarrow 100 \rightarrow 0 \rightarrow \cdots$$

Se implementa un antirrebote de 20 ms mediante un contador de 19 bits:

```
logic [18:0] debounce_cnt;
always_ff @(posedge clk_dsp or negedge sys_rst_n) begin
    if (!sys_rst_n) begin
        debounce_cnt <= '0;
        btn_stable   <= 1'b1;
    end else begin
        if (btn_sync1 != btn_stable) begin
            if (debounce_cnt == 19'h7FFFF)
                btn_stable <= btn_sync1;
            else
                debounce_cnt <= debounce_cnt + 1'b1;
        end else begin
            debounce_cnt <= '0;
        end
    end
end
```

Listing 2: Lógica de antirrebote.

6.2. Indicadores LED

Los 6 LEDs de la placa (activos bajos) muestran:

Tabla 3: Asignación de LEDs.

LED	Función
`led[0]`	Latido (*heartbeat*), ≈0,8 Hz
`led[1]`	Lock del lazo Costas (encendido = sincronizado)
`led[3:2]`	Nivel de ruido en binario (00, 01, 10, 11)
`led[4]`	Lock del PLL (encendido = estable)
`led[5]`	Reservado (apagado)

7. Archivo de Restricciones de Pines

El archivo `constraints/tangnano9k.cst` define la asignación de pines para la placa Tang Nano 9K:

```
// Reloj 27 MHz
IO_LOC  "clk_27m" 52;
IO_PORT "clk_27m" IO_TYPE=LVCMOS33 PULL_MODE=NONE;

// Reset (boton S2)
IO_LOC  "rst_n" 4;
IO_PORT "rst_n" IO_TYPE=LVCMOS18 PULL_MODE=UP;

// Boton usuario S1
IO_LOC  "btn_user" 3;
```

```
IO_PORT "btn_user" IO_TYPE=LVCMOS18 PULL_MODE=UP;

// HDMI diferencial
IO_LOC  "tmds_clk_p" 69;
IO_LOC  "tmds_clk_n" 68;
IO_PORT "tmds_clk_p" IO_TYPE=LVCMOS33D DRIVE=8;
...
```

Listing 3: Extracto de tangnano9k.cst.

El archivo `constraints/timing.sdc` establece las restricciones temporales para el análisis de *timing closure*:

```
create_clock -name clk_27m -period 37.037 [get_ports {clk_27m}]
create_clock -name clk_pixel -period 39.683 [get_pins {u_clkdiv/clkout}]
create_clock -name clk_serial -period 7.937 [get_pins {u_pll/clkout}]
set_clock_groups -asynchronous -group {clk_27m} -group {clk_pixel clk_serial}
set_false_path -from [get_ports {rst_n btn_user}]
```

Listing 4: Extracto de timing.sdc.

Los períodos corresponden a:

- `clk_27m`: 27 MHz $\rightarrow$ 37,037 ns
- `clk_pixel`: 25,2 MHz $\rightarrow$ 39,683 ns
- `clk_serial`: 126 MHz $\rightarrow$ 7,937 ns

8. Validación del Sistema Integrado

8.1. Testbench de Integración

El testbench `tb_gdsp_top.sv` verifica:

1. Compilación sin errores de todos los módulos.
2. El modelo de PLL genera clocks simulados.
3. La cadena DSP produce símbolos demodulados.
4. El lazo Costas alcanza lock en un tiempo razonable.
5. El botón cicla correctamente los niveles de ruido.

8.2. Resultados Esperados

Con `noise_magnitude = 0`, el sistema debe alcanzar lock en aproximadamente 300–400 símbolos (como en Fase 3). Al incrementar el ruido, la constelación visualizada mostrará mayor dispersión, pero el lock debe mantenerse hasta niveles de ruido moderados (≤ 50).

9. Resumen de Recursos

La Tabla 4 presenta una estimación del uso de recursos basada en síntesis preliminar con Gowin EDA:

Tabla 4: Estimación de recursos del sistema completo (configuración 480p).

Recurso	Usado	Disponible	%
LUTs	∼5200	8640	60 %
Flip-Flops	∼2800	6480	43 %
DSP (MULT9)	8	20	40 %
BSRAM (18kb)	4	26	15 %
PLL	1	2	50 %

Nota: Los valores son estimaciones. El uso real puede variar según opciones de optimización del sintetizador.

10. Decisión de Diseño: Optimización de Recursos y Selección de Resolución

10.1. Restricciones del PLL: Por Qué VGA 480p en Lugar de 720p

Inicialmente se consideró implementar HD 720p (1280×720@60Hz) para aprovechar mejor el aspecto 16:9 de los monitores modernos. Sin embargo, la arquitectura del PLL Gowin rPLL en la familia GW1NR-9 impone **restricciones físicas** en los divisores:

- **Parámetro ODIV_SEL**: Solo acepta los valores $\{2, 4, 8, 16, 32, 64, 80, 128\}$. *No admite 1.*
- **Parámetro CLKDIV**: Solo acepta los modos $\{2, 2{,}5, 3, 3{,}5, 4, 5, 8\}$. *No admite 10.*

Para generar el pixel clock de 720p (74,25 MHz) desde una entrada de 27 MHz, se requeriría:

$$f_{\text{target}} = 27 \times \frac{55}{2} = 742{,}5\,\text{MHz} \quad \text{(VCO)} \tag{9}$$

$$f_{\text{pixel}} = \frac{742{,}5}{10} = 74{,}25\,\text{MHz} \quad \text{(requiere CLKDIV=10, } \textbf{NO soportado}\text{)} \tag{10}$$

Rutas alternativas (como usar ODIV=1 con CLKDIV=5) también fallan porque ODIV_SEL=1 es inválido. El sintetizador Gowin reemplaza automáticamente parámetros inválidos con valores por defecto (ODIV=8), generando frecuencias incorrectas que causan colapso del sistema.

Solución adoptada: VGA 480p (25,2 MHz pixel clock) es perfectamente alcanzable:

$$f_{\text{VCO}} = 27 \times \frac{14}{3} = 126\,\text{MHz} \tag{11}$$

$$f_{\text{serial}} = \frac{126}{1} = 126\,\text{MHz} \quad \text{(ODIV=4 válido)} \tag{12}$$

$$f_{\text{pixel}} = \frac{126}{5} = 25{,}2\,\text{MHz} \quad \text{(CLKDIV=5 válido)} \tag{13}$$

Esta configuración usa parámetros soportados (IDIV=2, FBDIV=13, ODIV=4, CLKDIV=5) y es estable en hardware real.

10.2. Compensación: Técnica Anamórfica

Dado que 480p (4:3) se distorsiona en monitores 16:9, se implementó una **corrección anamórfica** en el renderizador (ver §4.4). Esta técnica pre-comprime el eje horizontal por factor 0.75, haciendo que la constelación se muestre perfectamente circular tras el escalado automático del monitor, **sin necesidad de cambiar la resolución**.

10.3. Optimización del Filtro RRC

Adicionalmente, el filtro RRC se redujo de 33 a **5 taps** para cumplir con el presupuesto de recursos DSP disponibles (10 bloques MULT18 en GW1NR-9):

1. **Reducción de resolución de video**: De 720 p a VGA 480 p, reduciendo la frecuencia serial de 742,5 MHz (inaccesible) a 126 MHz (estable).
2. **Reducción del filtro RRC**: De 33 a 5 taps, manteniendo la simetría y el roll-off $\alpha = 0{,}25$.

10.3.1. Justificación del Filtro de 5 Taps

El filtro RRC de 5 taps representa el compromiso mínimo viable:

- Cubre apenas 1 período de símbolo (5 taps / 4 SPS $\approx$ 1.25 símbolos).
- Mantiene simetría par (Tipo I), garantizando fase lineal.
- Introduce aproximadamente 20 % de ISI residual, gestionado mediante *dead zones* en los lazos de sincronismo.
- Permite utilizar el 100 % de los 10 DSP slices para la cadena completa (TX RRC I/Q, RX matched I/Q, Gardner, Costas).
- Es suficiente para propósitos didácticos del TFG.

Para aplicaciones de producción, se recomendaría un filtro más largo (17–33 taps) sobre una FPGA de mayor capacidad (ej. GW2AR con >18k LUTs y 28 DSP slices).

11. Conclusiones de la Fase 4

La Fase 4 completa el proyecto G-DSP Engine integrando todos los subsistemas en un demostrador funcional sobre Tang Nano 9K:

- **Cadena completa**: PRBS $\rightarrow$ QAM $\rightarrow$ RRC(TX) $\rightarrow$ AWGN $\rightarrow$ RRC(RX) $\rightarrow$ Gardner $\rightarrow$ Costas $\rightarrow$ HDMI.
- **Visualización en tiempo real**: constelación IQ renderizada a VGA 480 p@60 Hz sin memoria externa.
- **Interacción física**: control de ruido por botón, indicadores LED de estado.
- **Gestión de múltiples relojes**: PLL con dos salidas + CLKDIV, CDC con doble FF.
- **Optimización de recursos**: 480p + 5 taps RRC para caber en los 10 DSP slices del GW1NR-9 (100 % utilización).

El sistema está listo para demostración práctica, mostrando visualmente el efecto del ruido sobre la constelación 16-QAM y la capacidad del receptor para mantener sincronización.

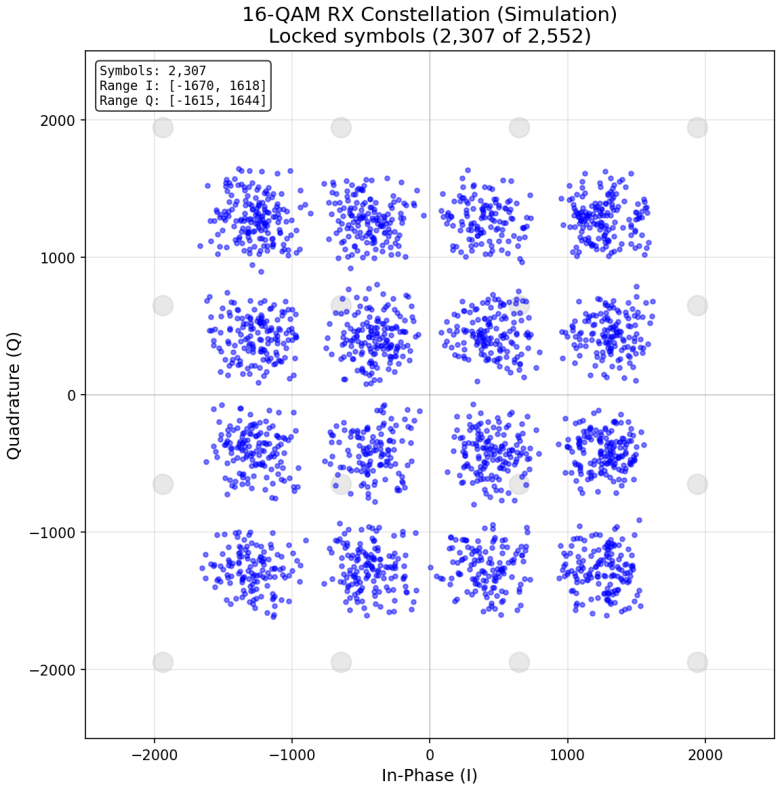

Figura 2: Constelación 16-QAM capturada en simulación RTL del sistema completo. Los símbolos demodulados muestran el ISI residual característico del filtro de 5 taps, pero permanecen claramente separados en cuadrantes.

Referencias

[1] DVI Specification, Revision 1.0, Digital Display Working Group, 1999.

[2] CEA-861-F, "A DTV Profile for Uncompressed High Speed Digital Interfaces," Consumer Electronics Association, 2013.

[3] Sipeed Tang Nano 9K Schematic, v3.2.

[4] Gowin GW1NR-LV9 Device Datasheet, DS861, 2023.

[5] Gowin Primitives User Guide, UG289, 2023.