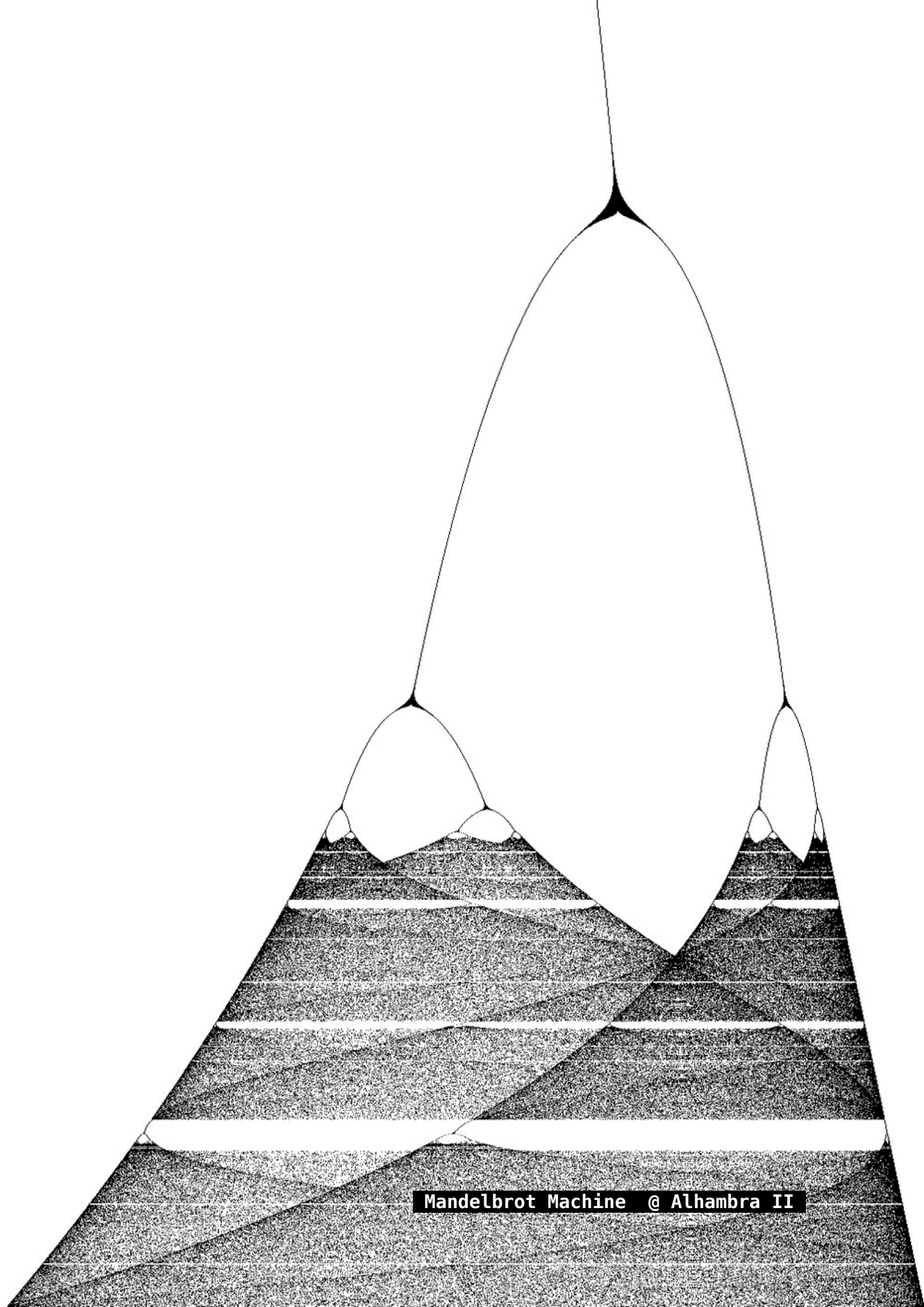
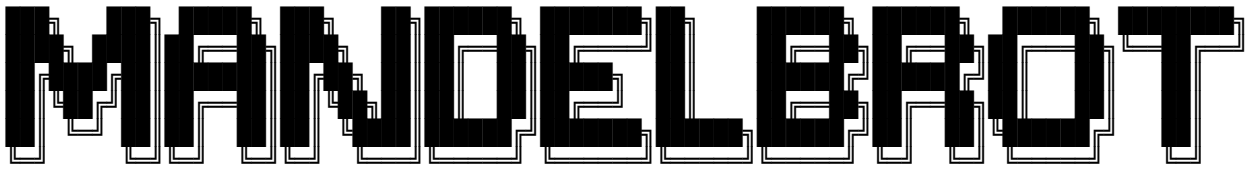

Mandelbrot Machine @ Alhambra II



M A C H I N E @ A L H A M B R A I I

Un ejercicio de migración entre tarjetas FPGAs de un diseño complejo

Carlos Venegas (X: @cavearr / Github: @cavearr) - 2025

Basado en el trabajo original de Jesús Arias (2022-2025)
https://www.ele.uva.es/~jesus/SIMRETRO/mandelmachine.pdf

0. Introducción y motivación

El presente trabajo aborda un problema recurrente en el diseño con FPGAs: la migración de proyectos entre plataformas con recursos dispares. Tomando como caso de estudio la excepcional Mandelbrot Machine de Jesús Arias, originalmente diseñada para la tarjeta SIMRETRO, se ha desarrollado un ambiente completo que permite su ejecución en la Alhambra II sin alterar una sola línea del código original.

Esta aproximación, más allá de resolver un problema técnico específico, ilustra una metodología de adaptación que preserva la integridad del diseño original mientras se construyen capas de abstracción que compensan las diferencias de hardware. El resultado no es simplemente una "versión" del proyecto original, sino un nuevo ambiente donde el núcleo computacional puede operar bajo condiciones radicalmente diferentes.

Al igual que en la imagen de portada, “el diagrama de bifurcación del conjunto de Mandelbrot”, cada decisión de diseño que tomé abrió nuevos caminos de exploración computacional, caminos apasionantes que nos traerán futuras evoluciones de este trabajo.

La Mandelbrot Machine original demuestra cómo una FPGA operando a 25 MHz puede superar en órdenes de magnitud a procesadores de propósito general en tareas específicas. Este proyecto extiende esa demostración al mostrar cómo las limitaciones de recursos pueden transformarse en oportunidades para explorar nuevas técnicas de procesamiento de señal y visualización.

El proyecto original está diseñado para la tarjeta SIMRETRO diseañada por Jesús Arias, que cuenta con una fpga ice40hx4k y una memoria SRAM externa, así como conector para un pad de consola NES con la que poder navegar por el fractal.

La placa objetivo del proyecto actual es la Alhambra-II una tarjeta que cuenta con la misma fpga pero no tiene memoria externa lo que supone una gran limitación ya que no hay memoria suficiente en las BRAMS internas de la FPGA para poder gestionar el framebuffer de vídeo lo que inicialmente hace irreproducible el proyecto.

Como he indicado anteriormente , el objetivo no era “adaptar” o “migrar” el proyecto original a la Alhambra-II modificando el código original, sino ver la posibilidad de crear una envoltura que permitiera integrar el código original como si fuera un “mandelbrot softcore” y hubiera otros módulos que en tiempo real pudieran suplir las necesidades de dicho softcore. Como ejercicio en busca de un tipo de metodología de reutilización me pareció muy interesante.

Las limitaciones se han convertido en fortalezas con un poco de imaginación. El resultado final se podría decir que es una especie de "vuelo a gran altura sobre el fractal" con la posibilidad de tener el zoom de una región concreta a la misma precisión que el proyecto original.

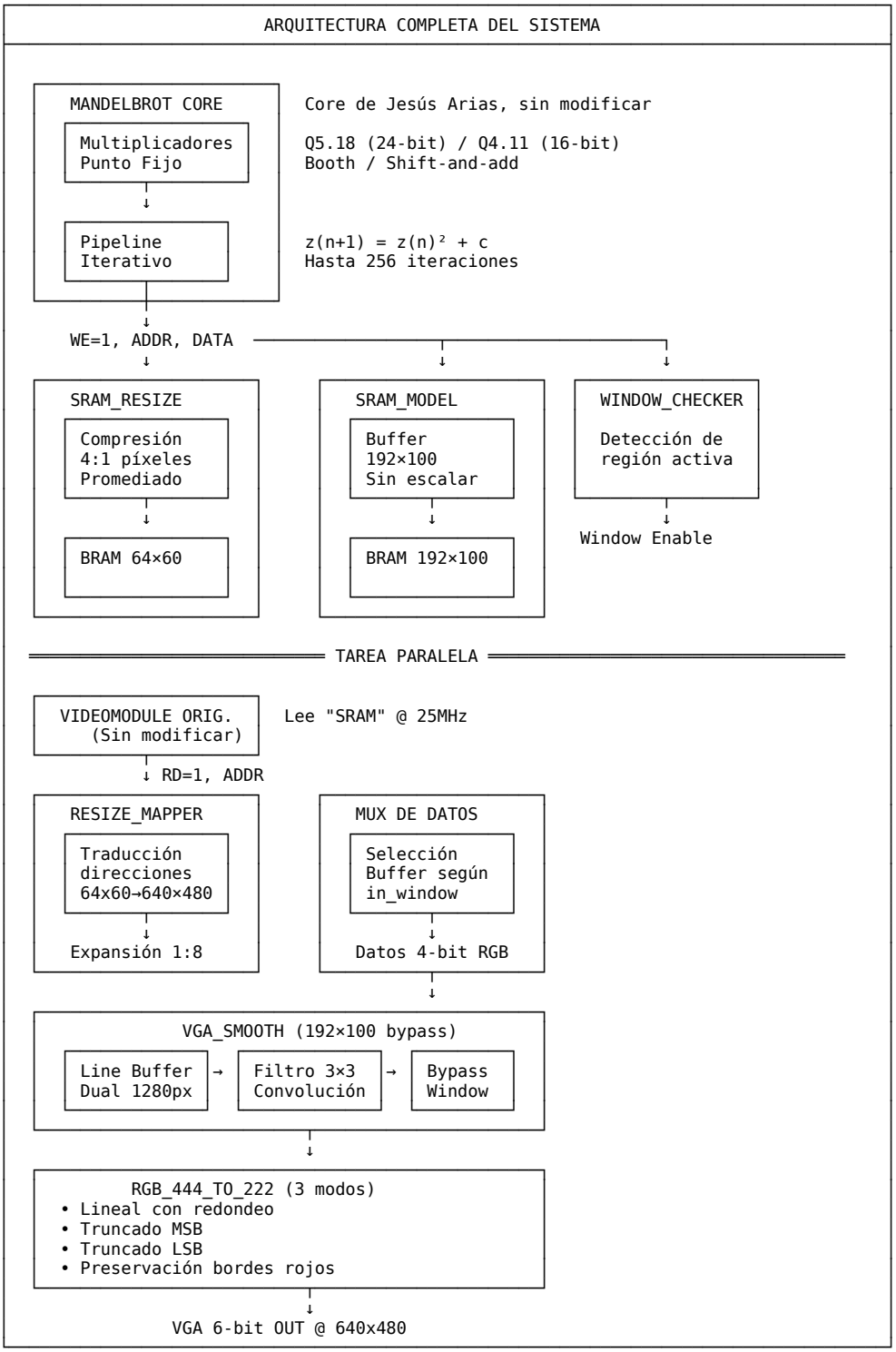
Además de que todo el diseño quedara autocontenido en la FPGA, otro objetivo para maximizar su difusión era utilizar para su control elementos comunes que el lector tenga o le sean fáciles y económicos de conseguir. La elección final fue joystick analógico típico en los kits de arduino que podría ser sustituido incluso por 2 potenciómetros, uno por eje.

El navegante podrá moverse por el fractal utilizando el joystick, y con los dos botones de la Alhambra-II podrá cambiar a modo zoom o modo “lupa”. El modo lupa permitirá en paralelo ver ventana de detalle a la máxima resolución del fractal (como en el proyecto original).

Abróchense los cinturones, comienza el viaje.

1. ARQUITECTURA DEL SISTEMA

El diseño se estructura en capas funcionales que operan de forma concurrente, aprovechando el paralelismo inherente de las FPGAs. A continuación se presenta el flujo de datos completo, no se ha incluido el gestor del joystick que se desarrolló a posteriori del documento, en futuras versiones será actualizado:



2. ESTRATEGIA DE EMULACIÓN DE MEMORIA

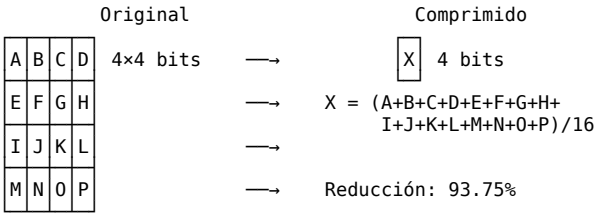
La ausencia de SRAM externa en la Alhambra II representa el principal desafío. La solución implementada opera en múltiples niveles:

2.1 Emulación Temporal

El módulo SRAM original espera ciclos de lectura/escritura asíncronos. Nuestra emulación aprovecha que el acceso es secuencial y predecible:

- Durante escritura: Compresión en tiempo real 4:1
- Durante lectura: Expansión y preparación del siguiente dato
- Latencia oculta mediante prefetch del siguiente píxel

2.2 Compresión Espacial



2.3 Ventana de Alta Resolución

Paralelamente se mantiene una región de 192×192 píxeles sin comprimir que actúa como "microscopio" sobre el fractal. Esta ventana:
- Se posiciona dinámicamente con el potenciómetro y encoder
- Preserva el detalle completo del cálculo original
- Se superpone al fondo escalado creando un efecto de "lupa mágica"

3. PIPELINE DE PROCESAMIENTO DE VIDEO

El procesamiento de video opera en tres etapas concurrentes:

ETAPA 1: Suavizado Espacial (vga_smooth.v)

Implementa un filtro de convolución 3×3 con características especiales:
- Buffer de línea dual para acceso a 3 líneas simultáneas
- Bypass selectivo para la ventana de 192×100 píxeles, se suaviza todo menos el área a alta resolución.
- Borde rojo de 1 píxel para delimitar la ventana de alta resolución.
- Latencia: 3 ciclos de reloj

ETAPA 2: Reducción de Profundidad de Color (rgb_444_to_222.v)

El proyecto original cuenta con una paleta de 64 colores tipo CGA, contando con un DAC de 4 bits por pixel. En la Alhambra-ii no hay pines de salida suficientes para gestionar un DAC de esa resolución. De hecho el DAC comercializado para la Alhambra-II por Alhambrabits y que es open source (en las referencias podrás encontrar el enlace al proyecto), es un DAC de 2 bits por color.

Esto implica tener que mapear de algún modo la paleta de color original a la nueva paleta, teniendo que realizar una reducción. Existen múltiples posibilidades, para este proyecto he optado por implementar tres métodos, de modo que el lector pueda intercambiarlos y observar los resultados.

Los tres algoritmos seleccionables son:

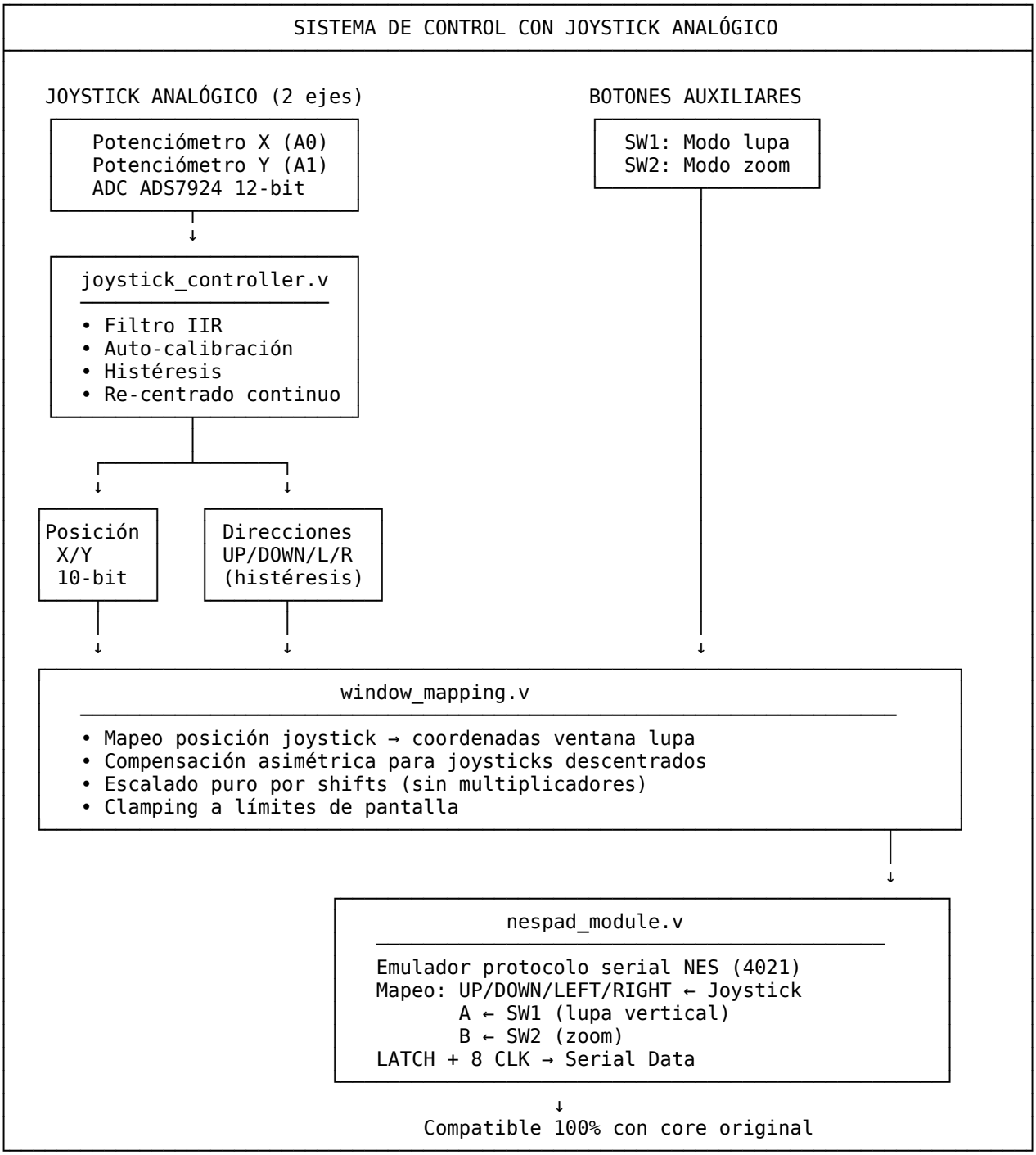
```
Modo 0 - Lineal con redondeo:
  out = (in + 2) >> 2
  Mejor preservación de gradientes

Modo 1 - Truncado MSB:
  out = in[3:2]
  Mínima lógica requerida

Modo 2 - Truncado LSB:
  out = in[1:0]
  Preserva variaciones sutiles
```

ETAPA 3: Sincronización y Salida

- Mantiene timing VGA estándar 640×480@60Hz
- Área activa: 512×480 (compatible con original)
- Sincronización preservada del módulo original

4. INTERFAZ DE USUARIO Y CONTROL

La adaptación del gamepad NES original se realiza mediante una capa de emulación que traduce las señales de entrada del joystick analógico a las esperadas por el core:



4.1 Arquitectura del Sistema de Entrada

El sistema de control se compone de tres módulos principales que trabajan en cascada:

joystick_controller.v

Controlador de joystick analógico con procesamiento avanzado de señal:

FILTRADO IIR (Infinite Impulse Response)

Suaviza el ruido del ADC mediante:

```
x_filt = x_filt + (x_adc - x_filt) >> FILTER_SHIFT
```

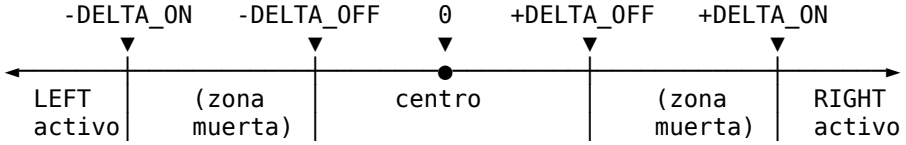
FILTER_SHIFT	α	Comportamiento
1	0.500	Respuesta rápida
2 (defecto)	0.250	Balanceado
3	0.125	Más suave, más latencia
4	0.0625	Muy suave, alta latencia

AUTO-CALIBRACIÓN

- Al inicio: promedia 32 muestras (2^CALIB_SHIFT) para encontrar el centro real
- Continuo: re-centra cuando el joystick está en reposo (RECENTER_BAND)
- Corrige deriva por temperatura o desgaste mecánico

HISTÉRESIS PARA DIRECCIONES

Previene el parpadeo en los umbrales usando dos niveles:



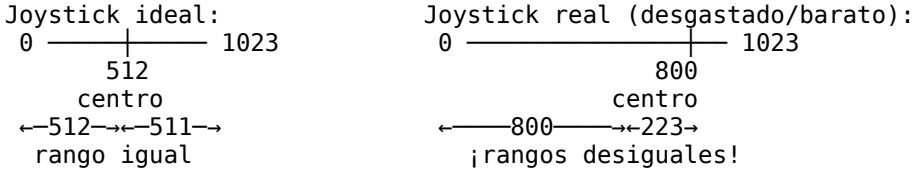
- DELTA_ON = 48 → Activa dirección al ~5% del centro
- DELTA_OFF = 32 → Desactiva al ~3% del centro
- Banda de histéresis = 16 cuentas (evita flickering)

window_mapping.v

Traduce la posición del joystick a coordenadas de la ventana (lupa) en pantalla:

PROBLEMA: JOYSTICKS ASIMÉTRICOS

Los joysticks baratos (como KY-023 de Arduino) tienen centro desplazado:



SOLUCIÓN: ESCALADO POR LADO

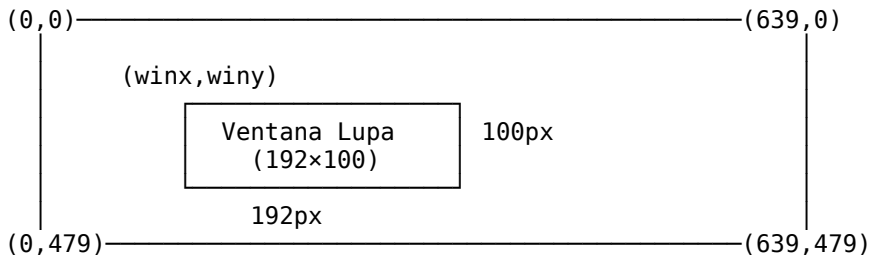
Calcula factor de escala diferente para cada lado de cada eje:

Lado	Rango Joy	Rango Win	Factor	Modo	Shift
Izq.	800	224	0.28	Reduce	>> 2
Der.	224	224	1.0	Reduce	>> 0

- Usa solo shifts (sin multiplicadores) para mínimo uso de LCs
- Clamping robusto: la ventana nunca sale de los límites de pantalla
- ~100-150 LCs en ice40

SISTEMA DE COORDENADAS

Pantalla 640×480, Ventana 192×100:



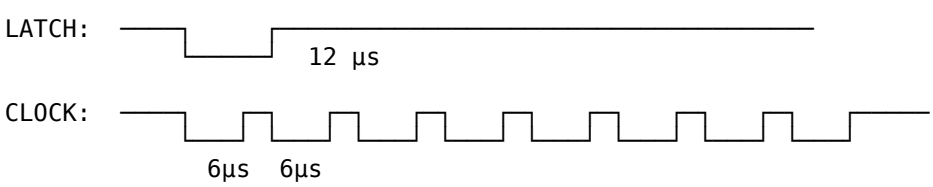
- winx: 0 a 448 (640-192)
- winy: 0 a 380 (480-100)
- Centro: winx=224, winy=190

nespad_module.v

Emula el protocolo serial del mando NES para compatibilidad con el core original:

PROTOCOLO NES (chip 4021)

3 señales: LATCH (carga), CLOCK (desplaza), DATA (salida serial)



MAPEO DE BOTONES

buttons[7:0] = {A, B, SELECT, START, UP, DOWN, LEFT, RIGHT}

Bit	Función en este proyecto
7 (A)	SW1 - Activa modo lupa vertical
6 (B)	SW2 - Activa modo zoom
5 (SEL)	No usado
4 (STA)	No usado
3 (UP)	Joystick arriba (histéresis)
2 (DWN)	Joystick abajo (histéresis)
1 (LFT)	Joystick izquierda (histéresis)
0 (RGT)	Joystick derecha (histéresis)

- Lógica activa-baja: 0 = pulsado, 1 = no pulsado
- Compatible 100% con sistemas que esperan mando NES real

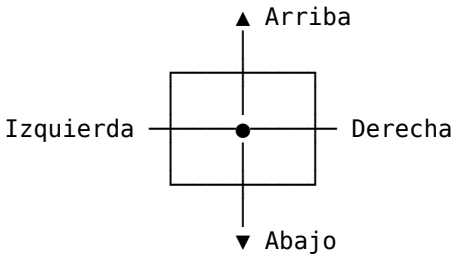
4.2 Usabilidad

El usuario al iniciar la FPGA verá el fractal de Mandelbrot en la pantalla, a muy poca resolución y una ventana encuadrada en rojo con el detalle a máxima resolución.

El joystick analógico permite controlar la posición de la lupa de forma intuitiva en dos ejes simultáneamente. Los botones auxiliares SW1 y SW2 permiten cambiar el modo de operación.

CONTROLES DEL JOYSTICK

Movimiento Joystick



Botones

SW1 (btn A)	Modo lupa vertical
SW2 (btn B)	Modo zoom

Los modos de operación se controlan mediante los botones:

Botones	Modo	Descripción
Ninguno	Movimiento lupa (horizontal)	El joystick mueve la ventana lupa en las dos direcciones. El eje X del joystick controla el movimiento horizontal de la lupa.
SW1	Movimiento lupa (vertical)	Mantener SW1 pulsado activa el movimiento libre de de la lupa con el joystick sin desplazar el fractal.
SW2	Zoom	Mantener SW2 pulsado y mover el joystick en el eje vertical permite hacer zoom + (arriba) o zoom - (abajo) sobre el fractal.

4.3 Parámetros de configuración

Los módulos tienen parámetros configurables para ajustar el comportamiento:

PARÁMETROS joystick_controller.v

Parámetro	Defecto	Descripción
FILTER_SHIFT	2	Fuerza del filtro IIR (2^N)
OUTPUT_SHIFT	2	Bits a descartar (12→10 bits)
DELTA_ON	48	Umbral para activar dirección
DELTA_OFF	32	Umbral para desactivar dirección
CALIB_SHIFT	5	Muestras calibración (2^N = 32)
RECENTER_BAND	200	Ventana de detección de reposo
RECENTER_SHIFT	6	Muestras re-centrado (2^N = 64)
STABLE_THRESH	3	Cambio máximo para "estable"
AXIS_MARGIN	0	Margen filtro diagonal

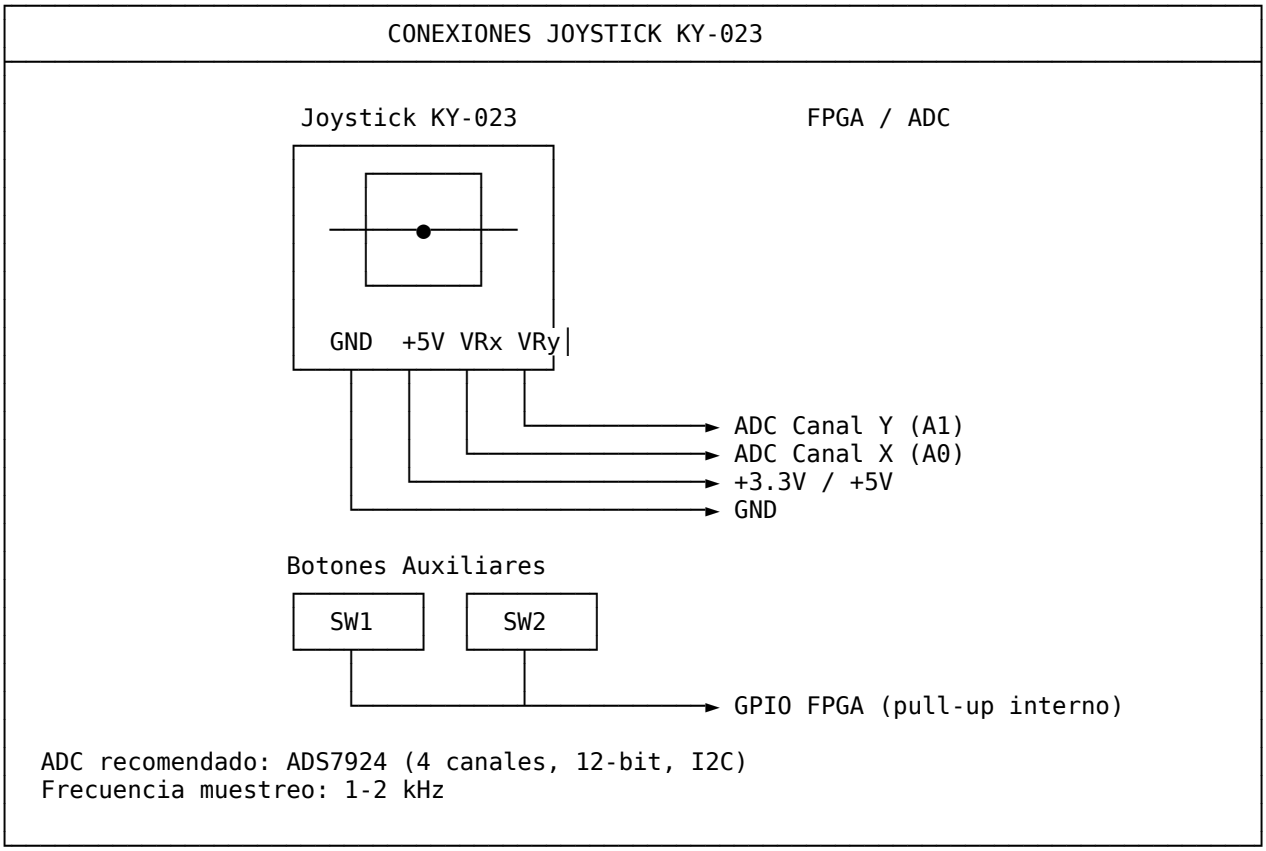
PARÁMETROS window_mapping.v

Parámetro	Defecto	Descripción
POS_WIDTH	10	Ancho de bits posición joystick
POS_CENTER_X	800	Centro X medido del joystick
POS_CENTER_Y	400	Centro Y medido del joystick
WIDTH	192	Ancho ventana lupa (píxeles)
HEIGHT	100	Alto ventana lupa (píxeles)
MAX_W	640	Ancho pantalla
MAX_H	480	Alto pantalla

NOTA: POS_CENTER_X y POS_CENTER_Y deben medirse con el joystick real.
Poner el joystick en reposo y leer los valores ADC para X e Y.

4.4 Conexión Hardware

Diagrama de conexiones físicas:



4.6 Notas de implementación

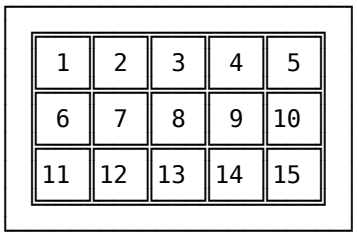
- El módulo joystick_controller procesa las señales en cada flanco de mclk (reloj de muestreo del ADC). Frecuencia recomendada: 1-2 kHz.
- La calibración automática requiere que el joystick esté en reposo durante el arranque (primeras 32 muestras).
- El re-centrado continuo corrige la deriva térmica y mecánica, actualizando el centro cuando el joystick permanece estable en la banda de reposo durante 64 muestras consecutivas.
- Los valores de histéresis (DELTA_ON/DELTA_OFF) están optimizados para el rango de 10 bits (0-1023). Si se usa otro rango, ajustar proporcionalmente.
- window_mapping.v no usa multiplicadores, solo shifts. Esto lo hace ideal para FPGAs pequeñas como ice40HX4K donde los recursos son muy limitados.
- El módulo nespad_module es completamente transparente: cualquier sistema que espere un mando NES real funcionará sin modificaciones.

5. CONEXIONADO FÍSICO

CONEXIONES DETALLADAS

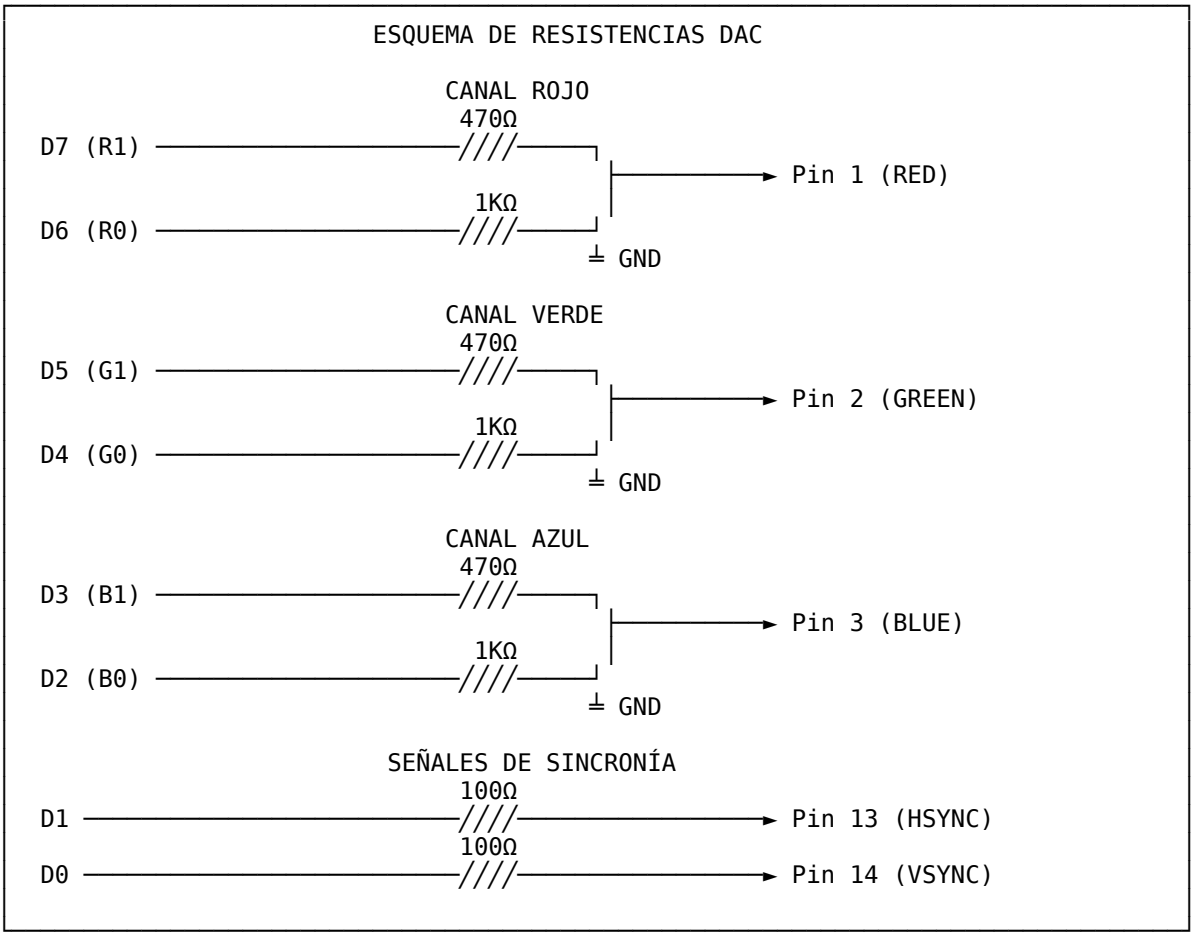
SALIDA VGA (Conector DB15 hembra)

Señal	Pin	FPGA
RED MSB	1	D7
RED LSB	1	D6
GRN MSB	2	D5
GRN LSB	2	D4
BLU MSB	3	D3
BLU LSB	3	D2
HSYNC	13	D1
VSYNC	14	D0
GND	5,10	GND



Vista frontal
conector DB15

ADAPTADOR VGA RGB 6-BIT



ESPECIFICACIONES TÉCNICAS DEL DAC

- Configuración: DAC de escalera resistiva de pesos binarios
  - Arquitectura: 2 bits por canal RGB (64 colores totales)
  - Principio: División resistiva ponderada

- Componentes pasivos:
  - R1[A,B,C]: 470Ω ±5% 1/16W (MSB - bit más significativo)
  - R2[A,B,C]: 1KΩ ±5% 1/16W (LSB - bit menos significativo)
  - R3, R4: 100Ω ±1% 1/10W (adaptación impedancia sincronismo)

- Análisis teórico (sin resistencias parásitas):

B1	B0	R equivalente	V salida (RL=75Ω)
0	0	∞	0.00V
0	1	1000Ω	0.33V
1	0	470Ω	0.66V
1	1	320Ω	1.00V

- Análisis real (con 200Ω serie en pines Alhambra II):

B1	B0	R total	V salida (RL=75Ω)
0	0	∞	0.00V
0	1	1200Ω	0.19V
1	0	670Ω	0.37V
1	1	520Ω||75Ω	0.56V

- Caracterización eléctrica:
  - Impedancia de salida efectiva: ~150-200Ω por canal
  - Ancho de banda (-3dB): >35 MHz (limitado por Cpará)
  - Tiempo de establecimiento: <10ns (dominado por RC)
  - Potencia disipada máxima: 23mW por canal @3.3V

- Limitaciones y consideraciones de diseño:
  - Voltaje máximo: 0.56V (80% del estándar VGA 0.7V)
  - Pérdida de saturación: ~44% respecto al diseño teórico
  - Compensación recomendada: Ajuste gamma en monitor
  - Compatibilidad: Funcional con >95% monitores modernos

- Notas de implementación:
  - Rutado: Mantener trazas RGB equidistantes (<5mm diferencia)
  - Montaje: Resistencias próximas al conector (<20mm)
  - GND: Plano de masa continuo bajo señales de video
  - Bypass: Condensador 100nF cerámico en alimentación FPGA

Notas de instalación:

- El adaptador VGA implementa un DAC R-2R de 6 bits siguiendo el diseño estándar de la comunidad FPGAwars. Proyecto completo disponible en: https://github.com/Alhambra-bits/AP-VGA

- La FPGA ICE40 tiene resistencias internas de 200Ω en serie con cada salida, lo que limita la tensión máxima alcanzable a ~0.41V en lugar de los 0.7V del estándar VGA. Esto resulta en una imagen funcional pero con brillo reducido (aproximadamente 60% del máximo)

- Las resistencias de sincronismo (R3, R4) de 100Ω proporcionan niveles TTL compatibles con la especificación VGA estándar

- El joystick de los kits de arduino puede tener muy mala calidad, en el código de Icestudio del proyecto se produce este caso y el offset de configuración de cada eje es alto, el usuario deberá calibrar con un polímetro y medir el valor en el centro de su joystick para ajustar estos valores.

- Longitud máxima de cables recomendada:
  - VGA: hasta 2m con cable de calidad
  - Encoder/Potenciómetro: 20cm para evitar ruido

- Para construcción casera del adaptador:
  - Usar resistencias de película metálica para mejor estabilidad
  - Montar sobre PCB o protoboard de calidad
  - Añadir condensadores de desacoplo (100nF) en alimentación si se observa ruido en la imagen

- Alternativas comerciales:
  - Adaptador VGA para la Alhambra-II de Alhambrabits
  - Pmod VGA de Digilent (compatible pero más caro)
  - Adaptadores genéricos FPGA-VGA de 6/8/12 bits

6. SÍNTESIS Y COMPATIBILIDAD

El proyecto está escrito en Verilog estándar y es totalmente compatible con herramientas open source:
  - Síntesis: Yosys 0.9+
  - Place & Route: nextpnr-ice40
  - Bitstream: icepack/iceprog
  - IDE: Icestudio 0.12 (incluye toda la toolchain)
Para adaptación a otras FPGAs se requiere:
  1. Ajustar el PLL para generar 25.175 MHz
  2. Remapear pines de E/S según la placa destino
  3. Verificar disponibilidad de BRAM (mínimo 29 bloques 512Bytes)

El diseño es escalable: en función de los recursos disponibles en la FPGA se podría reducir o ampliar las ventanas de memoria, la profundidad de color... gran parte de los límites son configurables por parámetros en el código.

8. VALOR EDUCATIVO Y POSIBILIDADES DE EXPLORACIÓN

Al implementar la solución, quedé sorprendido de la cantidad de conceptos que había utilizado, bajo mi punto de vista, se ha convertido en un laboratorio completo en el que podemos encontrar:

8.1 Arquitecturas de Cómputo Especializado

- Comparación de multiplicadores: Booth vs shift-and-add
- Aritmética de punto fijo vs punto flotante
- Pipeline y paralelismo en hardware
- Trade-offs área/velocidad/precisión

8.2 Procesamiento Digital de Señales

- Filtros FIR de media móvil
- Convolución 2D en tiempo real
- Técnicas de sobremuestreo y diezmado
- Reducción de ruido en señales analógicas

8.3 Algoritmos de Procesamiento de Imagen

- Escalado con diferentes kernels de interpolación
- Cuantización de color y dithering
- Gestión de memoria con acceso no lineal
- Sincronización de múltiples flujos de datos

8.4 Teoría de Fractales y Sistemas Dinámicos

- Convergencia y divergencia en sistemas iterativos
- Sensibilidad a condiciones iniciales
- Autosimilitud y dimensión fractal
- Visualización de conjuntos complejos

9. CÓDIGO FUENTE

El código está estructurado para facilitar la experimentación:

- Multiplicadores intercambiables (fmul.v, fmul24.v, fix_mpy.v, fmulbooth.v)
- Modos de reducción de color seleccionables
- Parámetros de filtrado ajustables
- Resoluciones de ventana configurables
- Diseño Top en Icestudio para facilitar la visión global de la arquitectura y facilitar la exploración inicial por parte del lector.

Todos los módulos importantes están implementados en verilog estándar y en general parametrizados de modo que se puedan reutilizar en otros proyectos.

Aunque este documento está planteado como un resumen general de la implementación, cada módulo cuenta con una documentación en sí mismo detallada que debería permitir utilizarlo en cualquier otro proyecto o ayudar a su comprensión.

10. CONCLUSIONES, TRABAJO FUTURO Y AGRADECIMIENTOS

Esta adaptación demuestra que las limitaciones de recursos pueden catalizar la innovación. La construcción de un ambiente alternativo que preserva la funcionalidad del diseño original mientras explora nuevas técnicas de visualización ilustra principios fundamentales del diseño con FPGAs:

- La importancia de entender las interfaces más que las implementaciones
- El valor del procesamiento concurrente y los pipelines
- La reutilización mediante abstracción en lugar de modificación
- La transformación de restricciones en características

Líneas de trabajo futuro podrían ser:
- Exploración de otros fractales (Julia, Burning Ship, etc.)
- Interfaz serie para control desde PC
- Nuevas interfaces (teclado, josticks...)
- OSD para mostrar temporalmente el modo de control en pantalla
- Punto flotante...¿por qué no?
- Todo lo que el lector pueda proyectar ;)

El código fuente completo, documentación adicional y actualizaciones están disponibles en el repositorio del proyecto.

10.1. AGRADECIMIENTOS

El proyecto queda abierto para la colaboración, aportaciones o retroalimentación positiva por parte de los lectores, espero que el proyecto te haya motivado y disfrutéis de la misma manera que lo he hecho yo.

Agradecimiento especial a Jesús Arias por compartir su excepcional trabajo y documentación, que han hecho posible esta adaptación educativa.

Agradecimiento a @Demócrito por su adaptación a bloque de Icestudio del código para gestionar el ADC por i2c de la Alhambra-ii de Jesús Arias y el bloque del encoder rotativo que fue una
base estupenda sobre la que trabajar, además de ser un lazo de retroalimentación imprescindible ;)

Agradecimiento a la comunidad FPGAWars por el interés mostrado en todos estos proyectos no euclidianos que proponemos, es una fuente de motivación.

11. REFERENCIAS

[1] Arias, J. (2022-2025). "The Mandelbrot Machine". Universidad de Valladolid.
https://www.ele.uva.es/~jesus/SIMRETRO/mandelmachine.pdf

[2] Mandelbrot, B. (1982). "The Fractal Geometry of Nature". W.H. Freeman.

[3] Icestudio - https://icestudio.io/

[4] FPGAwars Community - https://groups.google.com/g/fpga-wars-explorando-el-lado-libre/

[5] FPGAwars Github - https://github.com/fpgawars

[6] Alhambra II board - https://github.com/FPGAwars/Alhambra-II-FPGA

[6] Alhambra II VGA-AP board - https://github.com/Alhambra-bits/AP-VGA

FIN - Year 5 AC (After COVID)

(en homenaje a la ocurrente dedicatoria original)