



A continuación encontrarás un plan detallado para cada grupo (Entrada, Señal y Salida) con las variables clave, su tamaño en bits y las tareas de investigación/diseño que deberán realizar en un plazo de dos semanas. Se incluyen referencias a documentación existente sobre sintetizadores en FPGA que justifican el uso de ciertas resoluciones y bit-widths.

### Variables y tamaño de bits

Estas variables se plantean como sugerencia inicial. Ajustadas según las necesidades de recursos de vuestra FPGA y la calidad de audio deseada. Las citas muestran cómo proyectos previos usan resoluciones similares.

Grupo	Variable	Tamaño (bits)	Descripción breve

<b>Entrada</b>	uart_data_byte	8	Byte recibido por UART (MIDI). El protocolo UART añade 1 bit de inicio y 2 de parada, totalizando 11 bits por byte transmitido.
	midi_status	4	Tipo de mensaje (Note On/Off, CC, etc.).
	midi_channel	4	Canal MIDI (0–15).
	note_number	7	Nota MIDI (0–127), se almacena como 8 bits para comodidad.
	velocity	7	Velocidad (0–127).
	control_number	7	Número de controlador (CC).
	control_value	7	Valor del controlador (0–127).
	tempo	16	Tempo (BPM), detectado o fijado (0–65 535).
<b>Señal</b>	wavetable_sample	24	Valor de la tabla de ondas. Los proyectos de síntesis digital usan wavetables de 24 bits y 4096 muestras para alta fidelidad; estas tablas se pueden truncar a 8 bits para LFOs.

	phase_accumulator	32	Acumulador de fase DDS para obtener resolución fina.
	step_size	16	Incremento de fase (control de frecuencia).
	phase_offset	8	Desfase de cada forma de onda.
	mod_index	8	Índice de modulación ( $\beta$ ) para FM o AM.
	amplitude	16	Amplitud/ganancia.
	env_attack	16	Duración del ataque (número de muestras).
	env_decay	16	Duración del decay.
	env_sustain	16	Nivel de sostenido.
	env_release	16	Tiempo de liberación de la envolvente.
	lfo_value	8	Salida del LFO. El WPI reduce un LFO de 24 bits a 8 bits (1 bit de signo + 7 de datos).

Salida	pcm_sample	24	Muestra PCM combinada antes de la conversión. Los codecs de audio suelen trabajar a 24 bits.
	dsd_output	1	Bitstream de modulación delta-sigma (DSD).
	gain	8–16	Factor de ganancia antes de la modulación.
	mix_sum	26	Suma intermedia de canales (4 canales sumados con margen de saturación).

## Objetivos y tareas por grupo

### 1. Grupo Entrada

**Objetivo:** Capturar y decodificar los mensajes MIDI provenientes del controlador (Traktor F1) a través de UART.

- **Estudiar el protocolo MIDI y UART**
  - Comprobar que cada byte MIDI se envía a 31 250 baudios y se compone de 1 bit de inicio, 8 bits de datos y 2 bits de parada.
  - Identificar los tipos de mensaje: Note On, Note Off, Control Change, Pitch Bend, etc., y cómo se componen (estatus + canal + datos).
- **Diseñar el módulo UART→MIDI**
  - Implementar un receptor UART en VHDL/Verilog que detecte start/stop bits y entregue el byte de datos (8 bits).
  - Generar un parser que separe midi\_status (4 bits de MSB), midi\_channel (4 bits de LSB), note\_number y velocity.
  - Llevar un buffer FIFO para almacenar bytes en caso de ráfagas rápidas.
  - Proveer un mecanismo para leer Control Change y mapearlo a parámetros internos (control\_number, control\_value).
- **Implementar control de tempo / BPM**
  - Investigar técnicas para detectar pulsos o Beat a partir de mensajes MIDI (ej. clock MIDI o tapping manual).
  - Representar el tempo con un contador (tempo de 16 bits) para cubrir el rango 0–300 BPM.
- **Pruebas**
  - Simular el receptor con bytes MIDI conocidos, midiendo el retardo (≈286 μs por nota a 115 200 baudios).

- Validar la decodificación de notas y controles comparando con software (por ejemplo, MIDI Monitor).

## 2. Grupo Señal

**Objetivo:** Generar y procesar la señal de audio a partir de los parámetros de entrada mediante osciladores y moduladores.

- **Elegir arquitectura de síntesis**
  - Utilizar Direct Digital Synthesis (DDS) con tablas de ondas de 24 bits y 4096 muestras o reducir a 1024/2048 si el FPGA lo requiere.
  - Investigar la relación entre tamaño de tabla (`wavetable_sample`), resolución de acumulador (`phase_accumulator`) y exactitud de frecuencia.
- **Diseñar osciladores básicos**
  - Implementar osciladores de seno, cuadrada, rampa y diente de sierra (lookup table + acumulador).
  - Ajustar `step_size` (16 bits) según la frecuencia deseada:  $\text{step\_size} = (\text{frecuencia} \times \text{tamaño\_tabla}) / f_s$ .
  - Permitir un `phase_offset` de 8 bits para combinar formas y crear timbres complejos.
- **Incluir modulación AM/FM**
  - Definir un índice de modulación `mod_index` (8 bits) y aplicar modulación de fase o frecuencia.
  - Estudiar el uso de LFOs: según el proyecto WPI, se pueden usar LFOs de 24 bits reducidos a 8 bits para controlar parámetros, y mapear entradas MIDI de 8 bits a escalas de 16 bits.
- **Implementar envolvente (ADSR)**
  - Crear un generador de envolventes exponenciales con tiempos de ataque (`env_attack`), decay (`env_decay`), sostenido (`env_sustain`) y release (`env_release`).
  - Ajustar los tiempos en función de la frecuencia de muestreo (p.ej., 96 kHz).
- **Gestión de polifonía y mezcla**
  - Soportar al menos 4 canales simultáneos.
  - Sumar los canales en un bus de mayor ancho (`mix_sum` de 26 bits) para evitar saturaciones y normalizar antes de enviar a la salida.
- **Pruebas**
  - Simular las formas de onda generadas, comprobar la linealidad de frecuencia y ausencia de aliasing.
  - Verificar la correcta aplicación de la envolvente y la modulación.

### Expresión matemática de cada bloque

1. Portadora (onda base):

$$c(t) = \sin(2\pi f_c t)$$

2. Moduladora (trino):

$$m(t) = \sin(2\pi f_m t)$$

3. Oscilador FM (mezcla):

$$s(t) = \sin(2\pi f_c t + \beta \cdot m(t))$$

donde  $\beta$  es el índice de modulación.

4. Envolvente exponencial:

$$e(t) = e^{-\alpha t}$$

5. Salida (canto de pájaro sintético):

$$y(t) = s(t) \cdot e(t) = \sin(2\pi f_c t + \beta \sin(2\pi f_m t)) \cdot e^{-\alpha t}$$

## 3. Grupo Salida

**Objetivo:** Convertir la señal PCM interna en un flujo DSD de 1 bit y prepararla para amplificación clase D.

- **Estudiar modulación delta–sigma**
  - Elegir el orden del modulador (1º o 2º) y el factor de sobremuestreo (OSR). Por ejemplo, a 96 kHz con un OSR de 64 se obtienen 6,144 MHz.
  - Implementar integradores y cuantizador de 1 bit (`dsd_output`). Seleccionar la longitud de palabra de los integradores (16–24 bits) para minimizar el quantization noise.
- **Implementar ganancia y mezcla final**
  - Diseñar un control de `gain` (8–16 bits) antes del modulador para ajustar el volumen general.
  - Comprobar que la suma de los canales no sature el modulador; usar un normalizador (`pcm_sample` de 24 bits).
- **Preparar la salida de audio y filtros**
  - Investigar el filtrado pasabajo necesario tras el modulador delta–sigma para eliminar ruido de alta frecuencia y adecuar la señal a los amplificadores clase D.
  - Diseñar un filtro LC o pasivo acorde con la frecuencia de muestreo DSD y las impedancias de carga.
- **Validar el sistema de salida**
  - Simular la modulación delta–sigma con señales de prueba (seno de 1 kHz) y comprobar la distorsión (SNDR).
  - Ensayar la conexión a un módulo clase D y medir la respuesta de frecuencia.

## Plan de investigación y entregables (2 semanas)

Semana	Grupo Entrada	Grupo Señal	Grupo Salida
<b>Semana 1</b>	<ul style="list-style-type: none"> <li>– Estudiar protocolo MIDI y UART, diseñar el receptor y parser.</li> <li>– Determinar el mapeo de mensajes a variables internas y delimitar tiempo.</li> <li>– Preparar esquemas de temporización y buffers.</li> </ul>	<ul style="list-style-type: none"> <li>– Decidir tamaño de tablas y resoluciones.</li> <li>– Diseñar osciladores (sin, cuadrada, rampa, diente de sierra) y calcular step_size.</li> <li>– Investigar modulación AM/FM y definir índices.</li> <li>– Prototipar generador ADSR y LFO.</li> </ul>	<ul style="list-style-type: none"> <li>– Investigar moduladores delta-sigma y elegir OSR y orden.</li> <li>– Definir filtros de salida y dimensionar integradores.</li> <li>– Diseñar lógica de mezcla y control de ganancia.</li> </ul>
<b>Semana 2</b>	<ul style="list-style-type: none"> <li>– Implementar el receptor UART/MIDI en Verilog/VHDL y simularlo.</li> <li>– Implementar mapeo de mensajes a parámetros.</li> <li>– Probar latencia y fiabilidad a 31 250 baudios.</li> </ul>	<ul style="list-style-type: none"> <li>– Codificar los osciladores y moduladores en Verilog/VHDL.</li> <li>– Integrar ADSR, LFO y modulaciones.</li> <li>– Simular la polifonía, mezcla y evitar saturación.</li> </ul>	<ul style="list-style-type: none"> <li>– Codificar modulador delta-sigma y aplicar ganancia.</li> <li>– Integrar filtros de salida y validar con señales de prueba.</li> <li>– Probar el sistema de mezcla (dos canales) y evaluar la salida DSD.</li> </ul>

**Reuniones:** se recomienda una reunión de seguimiento al final de cada semana para revisar avances, resolver bloqueos y sincronizar la integración.



### Recursos sugeridos

- Documentación de síntesis en FPGA

- El proyecto de WPI utiliza wavetables de 24 bits y 4096 muestras y demuestra cómo escalar una forma de onda a 8 bits para uso en moduladores.
- **Interfaz de audio**
  - En sistemas con codecs I2S, la señal serie de audio puede convertirse a un bus paralelo de 24 bits dentro del FPGA, y los puertos del procesador se ajustan a ese mismo ancho.
- **Latencia y comunicación MIDI**
  - La transmisión MIDI típica incluye 1 bit de inicio, 8 de datos y 2 de parada; tres bytes por nota implican un retardo de  $\sim 286 \mu\text{s}$  a 115 200 baudios.
- **Multiplicadores en FPGA**
  - Algunos FPGAs integran multiplicadores de 18 bits; conviene tener en cuenta este límite para fijar el ancho de palabra de las operaciones DSP.

Esta hoja de ruta os permitirá abordar el proyecto de manera organizada y con una clara división de responsabilidades. Cada equipo debe documentar sus decisiones (especialmente ajustes de bit-width y tamaños de tablas) y justificar cambios respecto a estas recomendaciones.