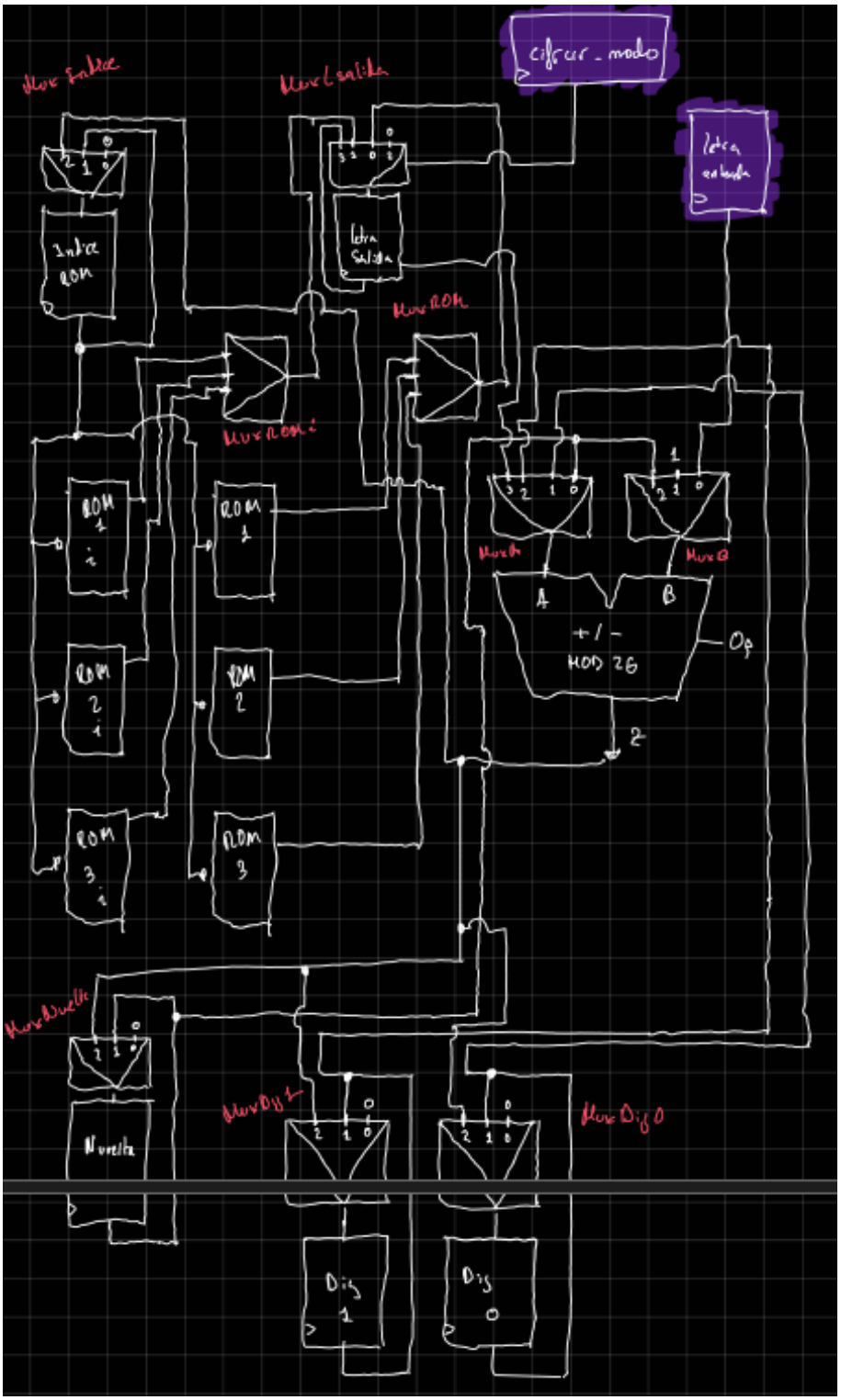
Implementación Hardware de Máquina Enigma

Proyecto de Ingeniería de Computadores - Diseño de Sistemas Digitales

Desarrollado por:
Juan Pastrana García
Omar Ouahri Vigil



Especificaciones Técnicas:
Plataforma: `Xilinx Artix-7 (Basys 3)`
Lenguaje de Descripción: `VHDL-93`
Entorno de Síntesis: `Vivado 2023.x`

Índice

1. Introducción y Justificación **2**

2. Contexto Histórico: El Legado de la Enigma **2**

2.1. Orígenes y Mecánica (1918-1930) 2

2.2. La Fortaleza Matemática 2

2.3. Bletchley Park y el Nacimiento de la Computación 2

3. Arquitectura del Sistema **3**

3.1. Diseño Conceptual (Datapath) 3

3.2. Unidad de Control (FSM) 5

4. Detalles de Implementación RTL **6**

4.1. Jerarquía Top-Level 6

4.2. La ALU Modular 7

4.3. Integridad de Señal (Debouncer) 7

5. Manual de Operación **8**

5.1. Mapa de Puertos Físicos 8

5.2. Procedimiento de Verificación 8

6. Conclusiones y Futuro **9**

7. Bibliografía y Recursos **10**

7.1. Fuentes Históricas y Criptografía 10

7.2. Recursos Técnicos (VHDL y FPGA) 10

1 Introducción y Justificación

El presente proyecto tiene como objetivo fundamental la recreación funcional de la **Máquina Enigma** utilizando tecnologías de diseño digital moderno. A diferencia de los simuladores por software, este proyecto implementa la lógica de cifrado directamente en hardware reconfigurable (FPGA), lo que permite un procesamiento paralelo y una latencia de respuesta mínima.

La elección de la FPGA **Artix-7** permite trasladar los componentes físicos originales (rotores, trinquetes mecánicos y cableado) a bloques lógicos configurables (CLBs), sustituyendo la corriente eléctrica por señales digitales y los contactos mecánicos por multiplexores y unidades aritméticas, intentando ser lo mas filedignos a la máquina orginal.

2 Contexto Histórico: El Legado de la Enigma

La Máquina Enigma no fue solo un dispositivo de cifrado; fue el eje central de la guerra de información durante la Segunda Guerra Mundial. Su historia abarca desde la ingeniería alemana hasta el nacimiento de la informática moderna.

2.1 Orígenes y Mecánica (1918-1930)

Patentada originalmente por el ingeniero alemán **Arthur Scherbius** en 1918, la Enigma nació como un dispositivo comercial para proteger secretos bancarios. Sin embargo, la *Reichsmarine* (Marina Alemana) y posteriormente la *Wehrmacht* (Ejército) adoptaron versiones militarizadas en los años 20, añadiendo el **Panel de Conexiones (Plugboard)** frontal, que aumentaba exponencialmente la seguridad.

Físicamente, era una máquina electro-mecánica portátil. Al pulsar una tecla, se cerraba un circuito eléctrico que viajaba a través de tres (o cuatro) **rotores giratorios**, rebotaba en un **reflector** y volvía a cruzar los rotores por un camino diferente para iluminar una bombilla con la letra cifrada.

2.2 La Fortaleza Matemática

La genialidad de la Enigma residía en su comportamiento **polialfabético**. A diferencia de los cifrados simples (como el César), aquí la regla de sustitución cambia con cada pulsación.

- La mecánica de trinquete forzaba el giro de los rotores tras cada letra.
- Esto implica que pulsar ’A’ tres veces podría generar la secuencia cifrada ’G’, ’P’, ’Z’.
- Con la configuración de rotores, anillos y cables, la máquina ofrecía un espacio de claves de aproximadamente **158 trillones (**$1{,}5 \times 10^{20}$**)** de combinaciones.

2.3 Bletchley Park y el Nacimiento de la Computación

El descifrado de la Enigma es considerado uno de los mayores hitos de la inteligencia militar. Inicialmente atacada por matemáticos polacos (Marian Rejewski), la tarea fue asumida por el Reino Unido en **Bletchley Park** al inicio de la guerra.

El matemático **Alan Turing** diseñó la *Bombe*, una máquina electromecánica masiva diseñada para buscar configuraciones de Enigma descartando contradicciones lógicas a gran velocidad. El éxito en la ruptura del código (inteligencia conocida como Ültra") permitió a los Aliados conocer la posición de los submarinos U-boot en el Atlántico. Los historiadores estiman que el trabajo realizado sobre la Enigma **acortó la guerra en dos años y salvó millones de vidas**, sentando además las bases teóricas de los ordenadores modernos.

Nuestro proyecto rinde homenaje a esa ingeniería inversa, reconstruyendo la lógica que Turing tuvo que desmantelar.

3 Arquitectura del Sistema

Para mantener el diseño limpio y depurable, hemos aplicado una estricta separación de responsabilidades: la ruta de datos (*Datapath*) se encarga de transformar la información, mientras que la Unidad de Control como su propio nombre indica gestiona las señales de control según su estado los cuales también gestiona.

3.1 Diseño Conceptual (Datapath)

Antes de abordar la codificación en VHDL, se diseñó el esquema de flujo de datos mostrado en la Figura 1. Este diagrama es crítico para entender cómo se transforma un carácter.

La señal de entrada (letras A-Z) se digitaliza en un bus de 5 bits. La señal atraviesa secuencialmente las siguientes etapas:

1. **Etapa de Entrada:** Suma del offset de los rotores (simulando el giro físico).
2. **Reflector:** Sustitución fija que devuelve la señal por el circuito.
3. **Etapa de Salida (Inversa):** Resta del offset para recuperar la posición relativa original.

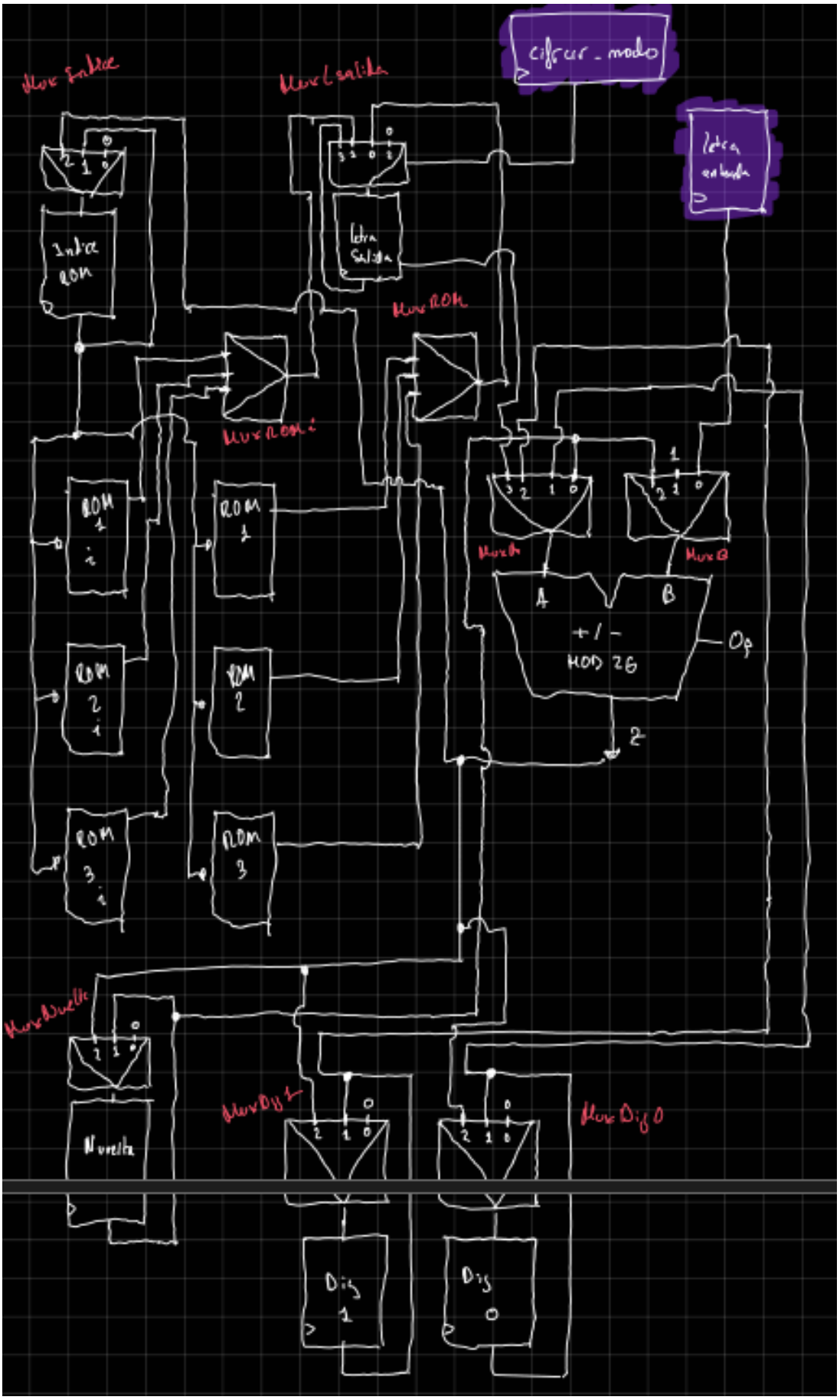


Figura 1: Diagrama esquemático manual del Datapath. Representa la traducción de la mecánica de rotores a bloques de suma/resta y multiplexación.

3.2 Unidad de Control (FSM)

El cerebro del sistema es una Máquina de Estados Finitos (Moore). Su diseño fue iterativo; partimos de un boceto manual para definir las transiciones lógicas necesarias para gestionar el rebote de los botones y el ciclo de cifrado.

Como se muestra en el boceto de diseño (Figura 2), la máquina evoluciona desde un estado de reposo (`S0`, `S1`) hacia estados de cálculo (`S2`, `S3`) y finalmente actualiza la posición mecánica de los rotores (`S4`, `S5`).

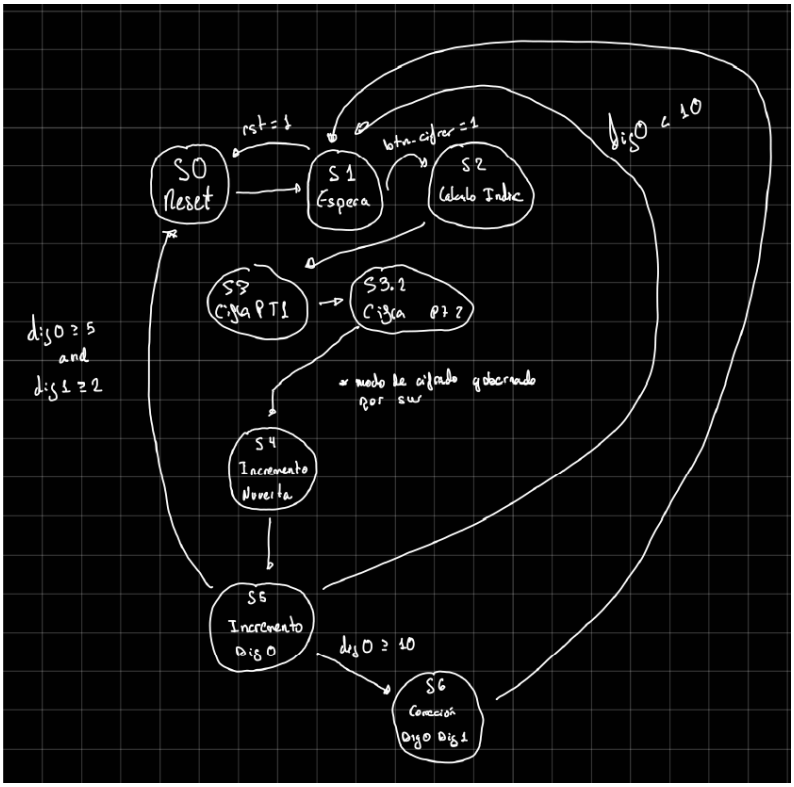


Figura 2: Boceto preliminar del diagrama de estados. Este esquema manual sirvió como base para la codificación de la FSM en VHDL.

Posteriormente, este diseño lógico se tradujo a hardware sintetizable, resultando en el esquema final implementado en Vivado (Figura 3).

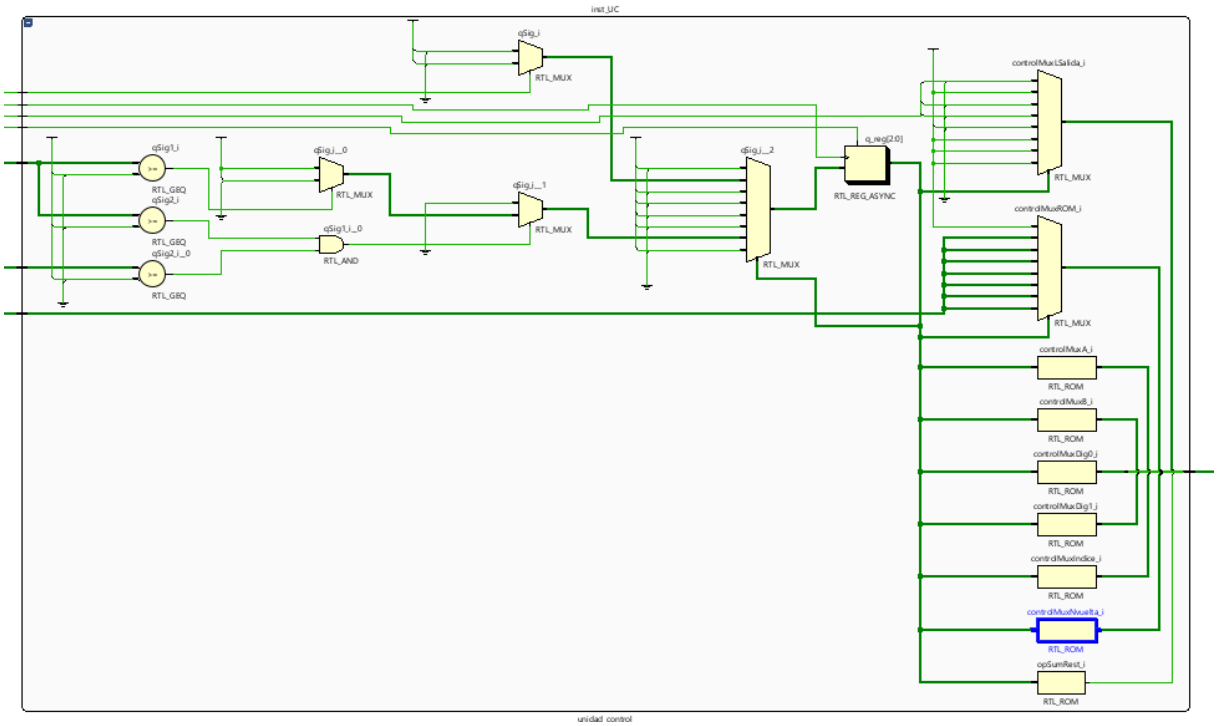


Figura 3: Implementación final RTL de la Unidad de Control tras la síntesis generado por Vivado.

4 Detalles de Implementación RTL

La implementación sigue una estructura modular jerárquica para facilitar la validación independiente de cada componente.

4.1 Jerarquía Top-Level

El módulo `Top_Enigma` actúa como la placa base del sistema, interconectando los periféricos de entrada/salida con el núcleo criptográfico (Figura 4).

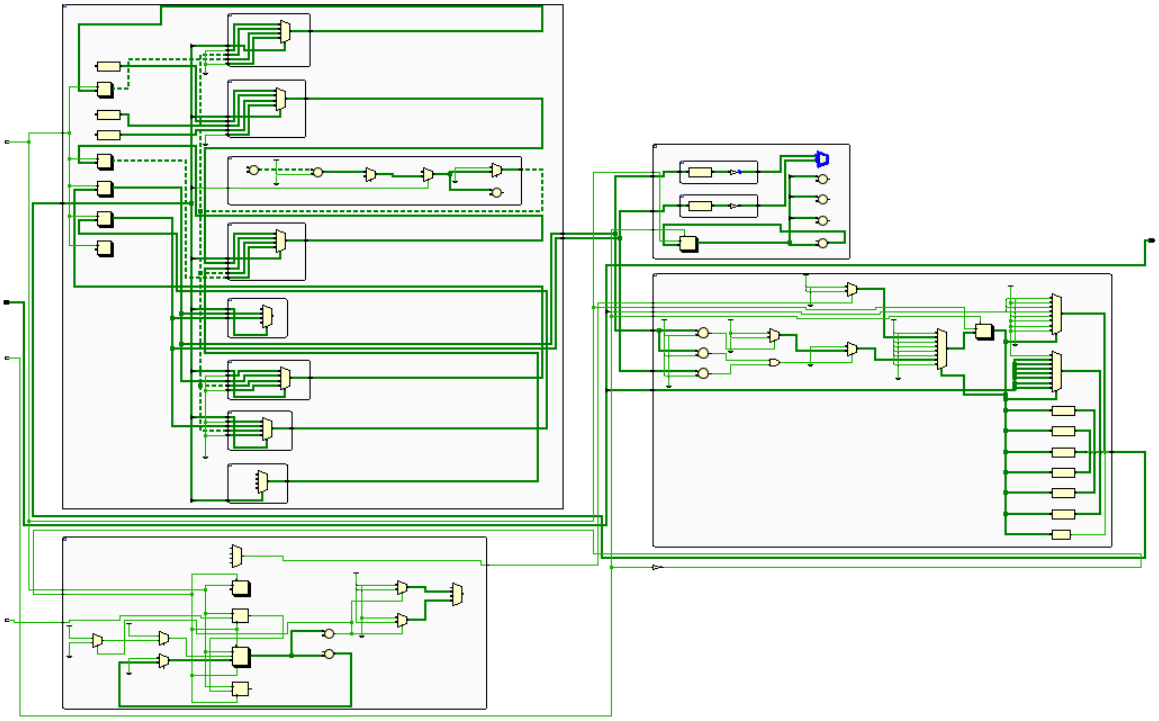


Figura 4: Vista de Jerarquía Top-Level. Los buses verdes destacan el flujo de datos de 5 bits a través del sistema.

4.2 La ALU Modular

Resolver la aritmética modular en hardware presentó un desafío, el manejo de números negativos en VHDL. Diseñamos una **ALU personalizada** (Figura 5) que aplica un offset de seguridad (+26) antes de la operación módulo, garantizando que el resultado siempre se mantenga dentro del rango positivo [0-25].

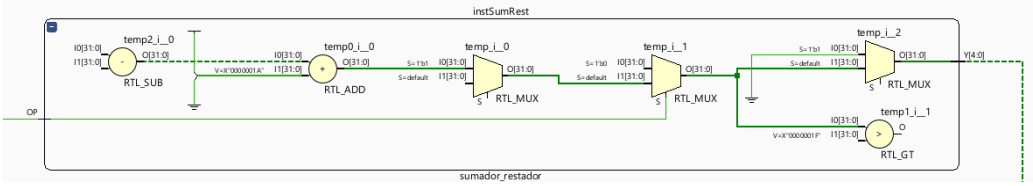


Figura 5: Esquema RTL de la ALU Modular.

4.3 Integridad de Señal (Debouncer)

Dado que trabajamos con pulsadores mecánicos, el *ruido* de rebote es inevitable. Para evitar falsos positivos, implementamos un filtro digital (**Debouncer**) que valida la señal solo si se mantiene estable durante 50ms, asegurando que cada pulsación física corresponda exactamente a un carácter cifrado.

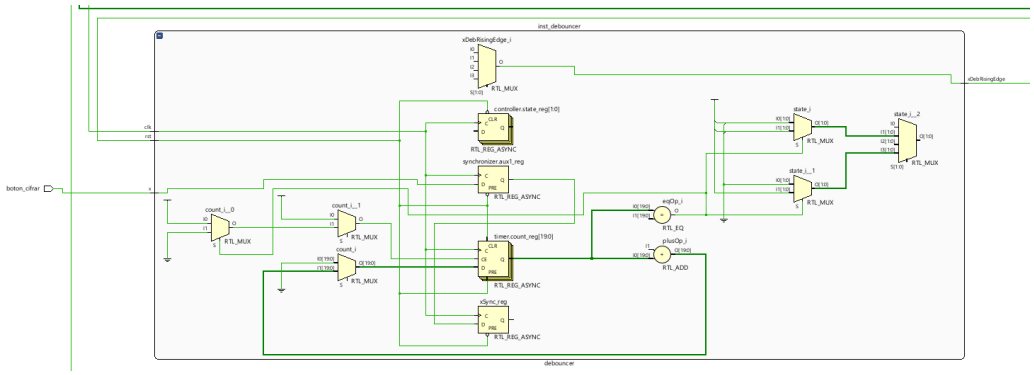


Figura 6: Lógica interna del Debouncer.

5 Manual de Operación

Para operar la máquina correctamente en la placa Basys 3, siga el siguiente mapa de interfaz.

5.1 Mapa de Puertos Físicos

La interacción se realiza exclusivamente a través de los periféricos integrados en la placa. Consulte la Tabla 1 para la asignación de pines.

Componente	Puerto	Función
RESET	BTN Center	**Inicio.** Obligatorio al encender para limpiar registros.
EJECUTAR	BTN Right	**Acción.** Cifra la letra actual y mueve los rotores.
Entrada	SW [4:0]	Selector de letra binaria (A=00000 ... Z=11001).
Config.	SW [14:13]	Configuración de cableado interno (Rotores).
Modo	SW [15]	`Down`: Cifrar / `Up`: Descifrar.

Cuadro 1: Asignación de pines y funcionalidad.

5.2 Procedimiento de Verificación

Para confirmar que el sistema funciona correctamente:

1. **Carga:** Programar el bitstream en la FPGA.
2. **Reset:** Pulsar `BTN Center`. Los displays de estado deben ir a `00`.
3. **Prueba de Cifrado:**
   - Configurar 'A' (`00000`) en los switches.
   - Pulsar `BTN Right`. Anotar el resultado del display derecho.
4. **Prueba de Reciprocidad (Descifrado):**

- Activar el modo descifrado (`SW15` arriba).
- Pulsar Reset (para sincronizar la posición inicial de rotores).
- Introducir la letra cifrada que obtuvimos antes.
- Al pulsar `BTN Right`, el sistema debe devolver una 'A'.

6 Conclusiones y Futuro

Este proyecto valida la capacidad de las FPGAs para emular sistemas electromecánicos complejos. Hemos logrado una implementación eficiente y funcionalmente idéntica a la Enigma original.

De cara a futuras iteraciones, planteamos incorporar una interfaz UART para la transmisión de textos completos desde PC y la carga dinámica de configuraciones de rotores vía memoria RAM.

7 Bibliografía y Recursos

Referencias utilizadas para el modelado teórico y la implementación técnica.

7.1 Fuentes Históricas y Criptografía

- **Crypto Museum:** Especificaciones técnicas detalladas y fotografías de los modelos originales.
- **Bletchley Park Trust:** Archivo oficial de la sede de descifrado aliada.
- **101 Computing - Enigma Emulator:** Simulador interactivo utilizado para validar los vectores de prueba.
- **Wikipedia - Máquina Enigma:** Descripción general del funcionamiento mecánico.

7.2 Recursos Técnicos (VHDL y FPGA)

- **Xilinx UG470:** Guía de configuración para FPGAs Serie 7.
- **Digilent Basys 3 Reference Manual:** Documentación oficial de la placa de desarrollo.
- **Nandland:** Tutoriales y ejemplos de buenas prácticas en VHDL y FSMs.