UNIVERSIDAD ORT
Uruguay

Documentación Obligatorio Arquitectura de Sistemas

“Un sueño LluviaDeBits”

27/11/2025

Alumnos: Nicolás Faccio, Santiago Marque y Sebastián Quiroz
N° de estudiante: 302569, 286903 y 323189



Índice

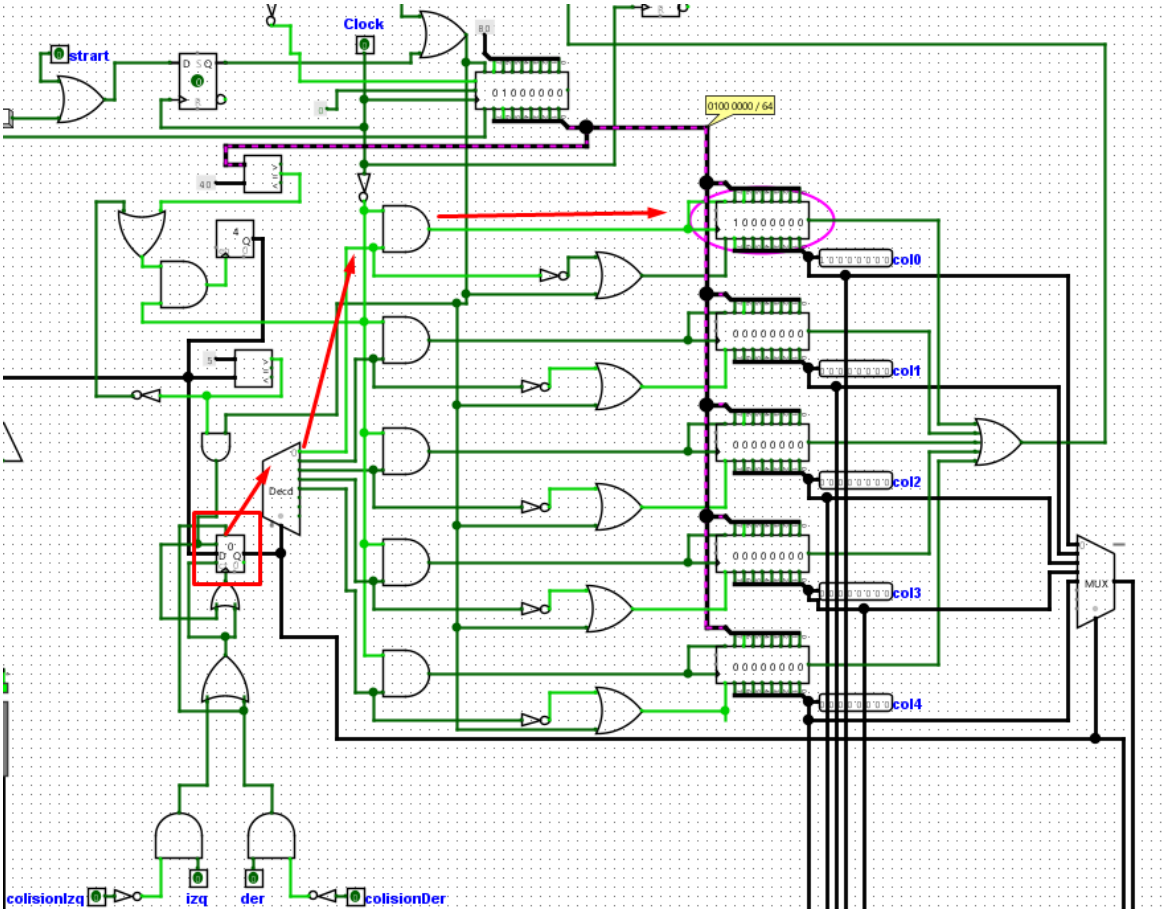
1. Introducción

Este documento presenta el diseño completo del videojuego LLUVIADEBITS, desarrollado para el obligatorio de Arquitectura de Sistemas. El proyecto consiste en un sistema secuencial y combinacional capaz de simular la caída de bits dentro de un panel 5×8, incorporando movimiento lateral, detección de colisiones, guardado de estado, control de puntaje, manejo de velocidades, detección de fin de juego, parpadeo y reinicio automático. El objetivo de esta documentación es describir detalladamente todos los módulos, arquitecturas, memorias, funciones lógicas y máquinas de estado que componen el sistema.

2. Especificación Funcional

2.1 Caída del bit único

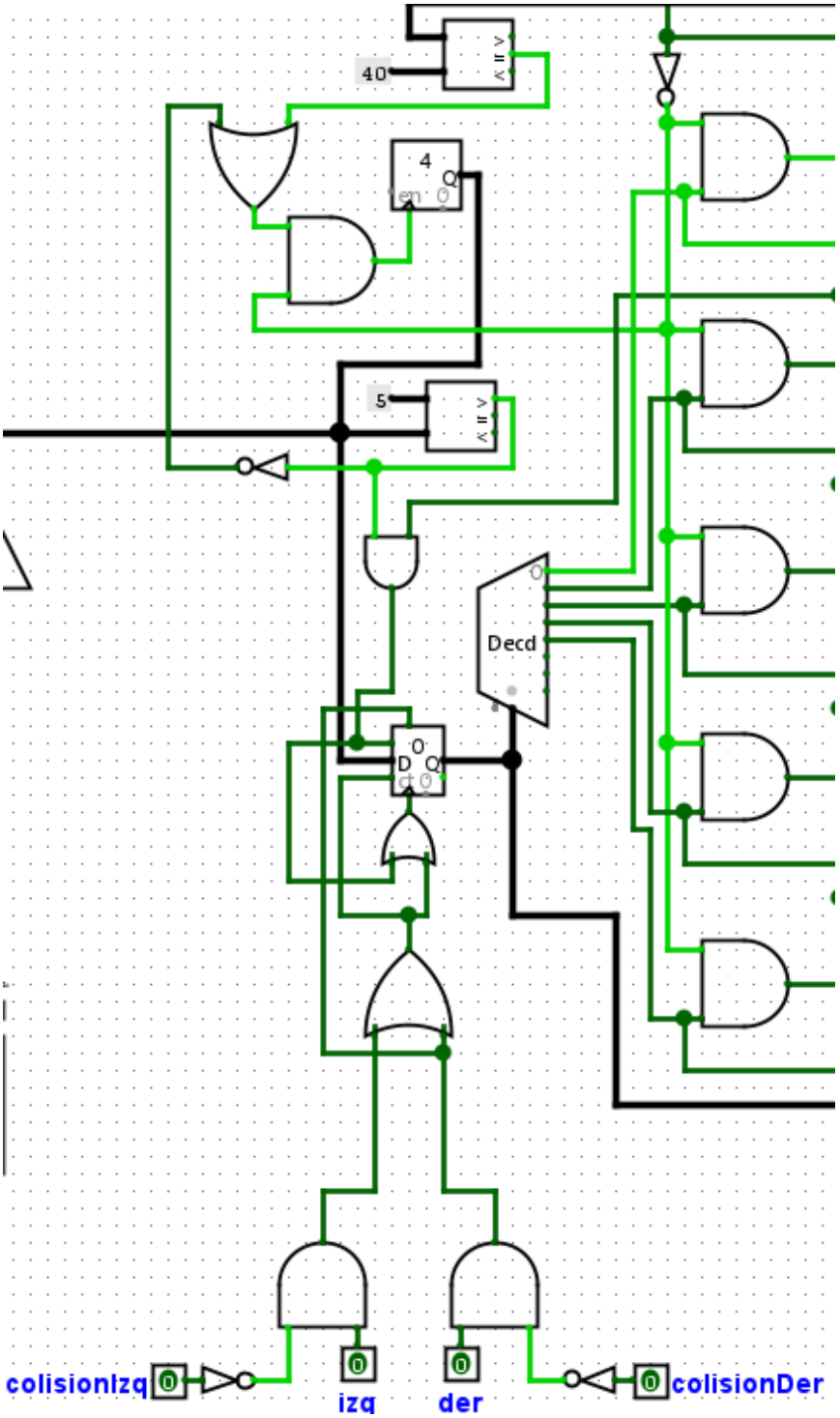
La caída del bit la decidimos implementar con un registro de desplazamiento principal, del cual se va pasando a otros cinco registro de desplazamiento, la posición actual en el eje vertical.



Dependiendo la posición del contador, se carga la información del registro de desplazamiento principal en el respectivo registro de desplazamiento que muestra la columna.

2.2 Movimiento lateral

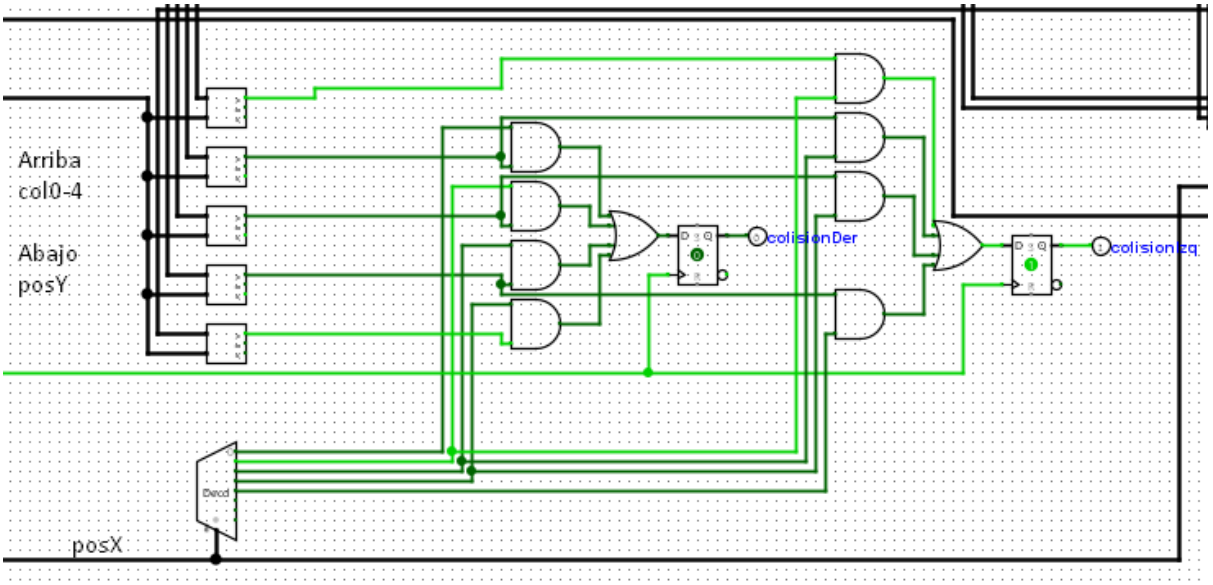
El jugador utiliza botones para desplazar el bit a izquierda o derecha varias veces dentro de un mismo ciclo de caída. El sistema bloquea movimientos fuera del panel.



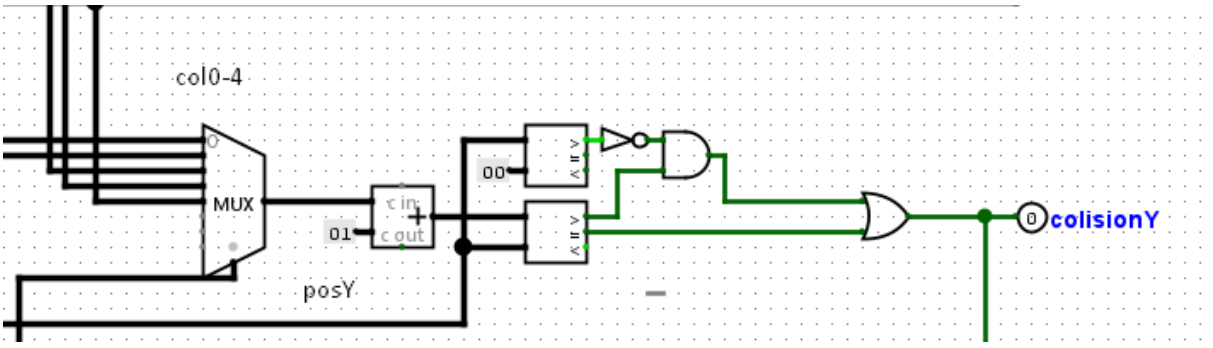
Con un contador entre cero y cuatro limitamos el movimiento entre esas cinco columnas. A su vez, cuando el número que hay en el registro de desplazamiento principal es igual a Ox40 (01000000), se manda una señal a un generador de número aleatorio, el cual va a generar un número de tres bit indefinidamente hasta que este número esté entre cero y cuatro. La función de este número es ser cargado en el contador de movimiento a la hora de lanzar nuestro siguiente bit en el registro de desplazamiento, sin embargo, no funciona siempre,a veces cuando se va a generar en la columna 0, se genera en la 4, y viceversa.

2.3 Detección de colisiones

Para la detección de colisiones implementamos soluciones desde el lado de la aritmética.



Nuestra implementación para la colisión horizontal fue esta, la posX se decodifica y permite revisar si se cumplen las condiciones de colisión para esa columna. Lo que buscamos verificar, es que las columnas de los costados en memoria sean mayores a la posY. Por ejemplo, si estamos en la posX es la columna tres (posX = 011), nuestra posY es 0000 1000, nuestra col4 en memoria es 0000 1111 y nuestra col2 en memoria 0000 0111, va a colisionar con la columna de la derecha, la cuatro, ya que 0000 1111 > 0000 1000, sin embargo, no va a colisionar con la columna de la izquierda ya que 0000 0111 < 0000 1000. Al haber una colisión, manda una señal en movimientoCuadrado que no permite hacer el movimiento lateral para ese lado, ya que manda un 0 al AND que lo controla.

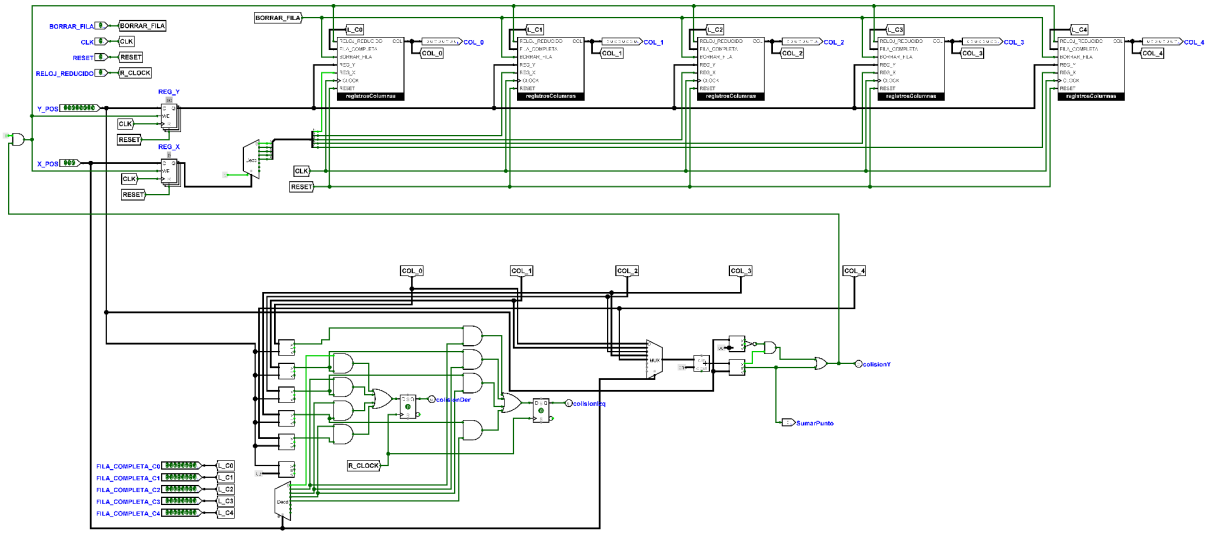


Para la colisionY, mandamos en un multiplexor todas las columnas desde la memoria, y dependiendo la posX, se selecciona cual vamos a comparar. Para la

comparación utilizamos la lógica de que si la columna en memoria + 1 es igual a la posY, por ejemplo, si posX está en la columna uno (posX = 001), posY es igual a 0001 0000 y col1 es igual a 0000 1111, si le sumamos 1 a col1, nos queda en 0001 0000, y al ser igual a posY, manda la señal de colisionY.
Al haber una colisión vertical, se manda una señal a movimientoCuadrado que permite lanzar el siguiente bit, guarda el nuevo estado y a su vez se manda una señal al puntaje para aumentar un punto.

2.4 Guardado del nuevo estado

Tras la colisión, el bit se almacena en la última posición válida y pasa a formar parte del estado fijo.



El circuito de Guardar Nuevo Estado es el encargado de tomar la pieza que está cayendo y transformarla en datos permanentes dentro del tablero. Todo el sistema se divide en tres partes que trabajan en conjunto: el manejo de las columnas, el control de la posición del bloque y la detección de colisiones.

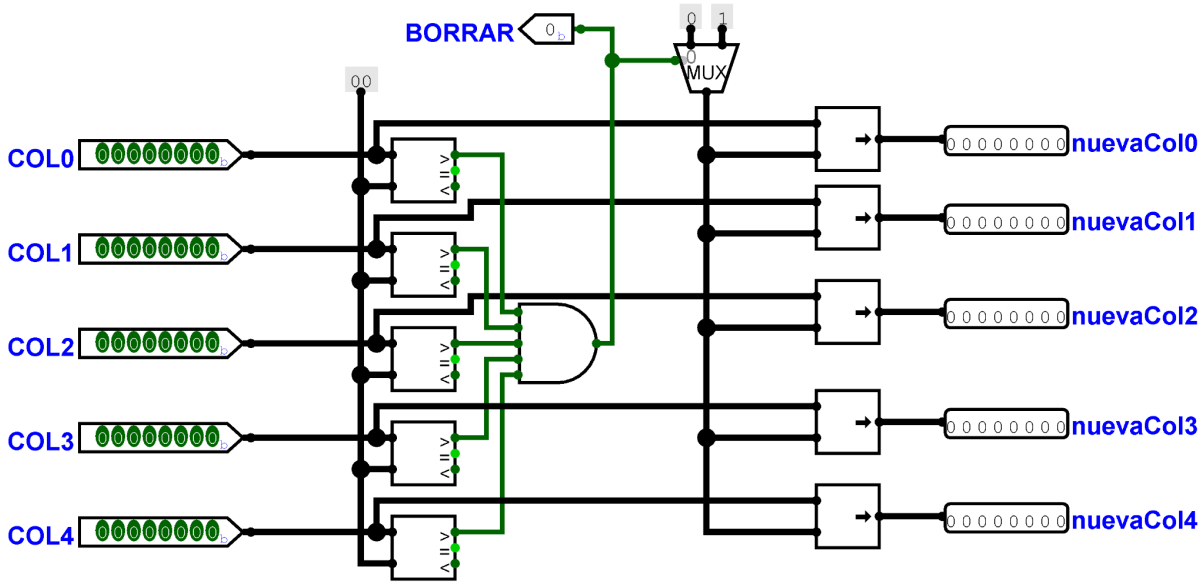
En la parte superior están los registros de columnas. Cada columna tiene su propio módulo donde se guarda su estado actual. Estos registros reciben la columna seleccionada, la altura donde está el bloque, la señal de fila completa y el reloj reducido que marca el ritmo de actualización. También reaccionan a señales como borrar fila o reset. El objetivo es que todas las columnas se comporten de la misma forma y que puedan mover sus bloques hacia abajo cuando se elimina una fila.

En el centro del circuito están los registros X e Y. REG_X indica en qué columna está el bloque y REG_Y marca la altura. Estas posiciones cambian según el movimiento del jugador y la caída automática marcada por el reloj. Estas dos señales viajan hacia las columnas y hacia la parte inferior del circuito.

La zona inferior se encarga de detectar colisiones. Ahí se toma la columna elegida y se revisa qué hay justo debajo del bloque. Si hay un uno en la memoria o si la pieza llegó al fondo del tablero, se activa la señal de colisión. Esa señal le indica al sistema que la pieza ya no puede seguir bajando y que debe guardarse en la columna correspondiente.Cuando eso ocurre, el registro de la columna escribe el bit en su posición final y la pieza pasa a formar parte del tablero fijo. El sistema queda listo para dejar caer la siguiente pieza.

2.5 Completado de líneas

Cuando una fila se completa, es eliminada y los bloques superiores descienden una posición. Se otorgan puntos adicionales.

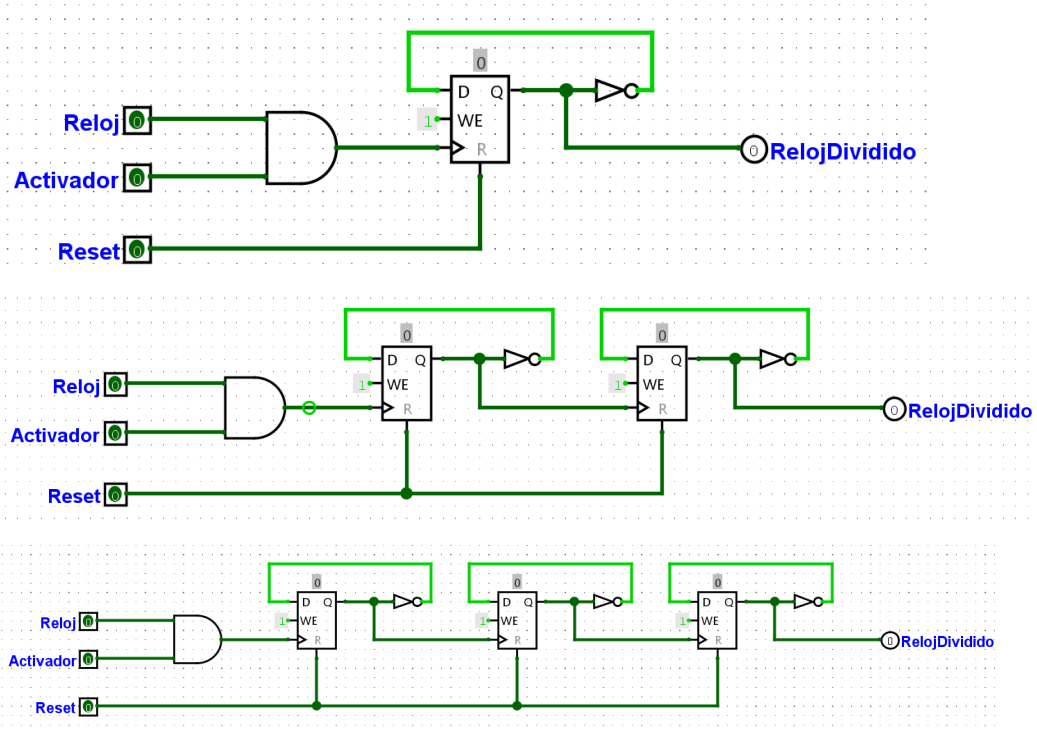


Este módulo entra en acción cuando una fila queda completamente ocupada. Al activarse BORRAR, la fila se elimina y todos los bloques que estaban arriba bajan una posición. Cada columna (COL0, COL1, etc) pasa por una lógica que detecta qué bits están por encima de la fila borrada y los desplaza hacia abajo. Si no hay que borrar nada, las columnas salen tal cual entran.

Cuando sí corresponde borrar, el MUX de cada columna elige la versión desplazada, y el resultado final queda almacenado como nuevaColX. De esta forma, el sistema actualiza el tablero automáticamente después de completar y eliminar una fila, dejando todo listo para que el juego continúe de forma fluida.

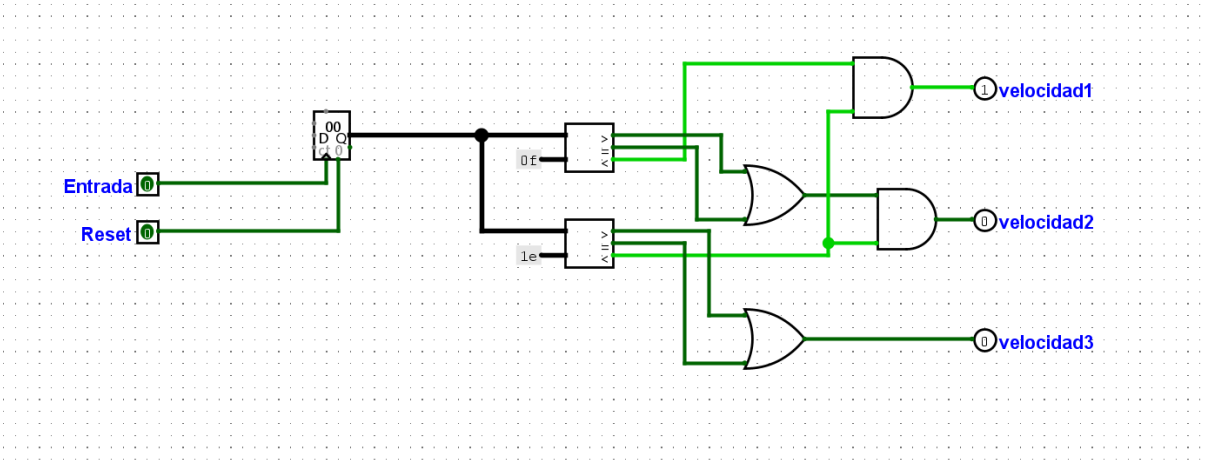
2.6 Velocidades del juego

El juego tiene tres velocidades determinadas por la densidad de bloques. Una señal visual indica la velocidad.

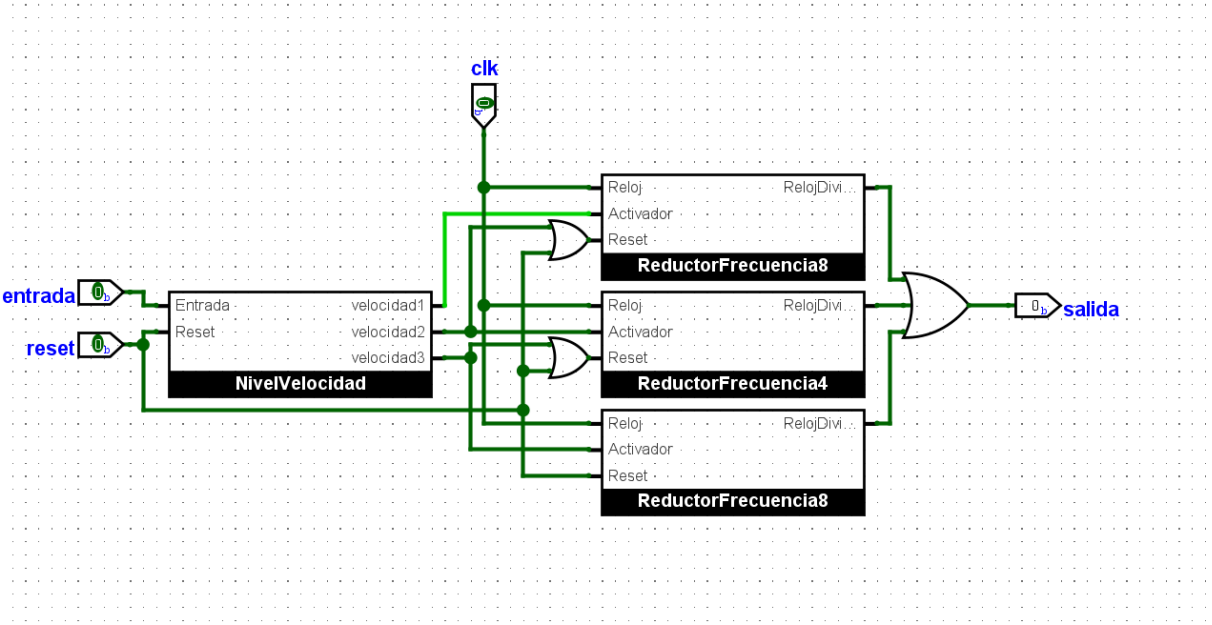


El sistema utiliza tres reductores de frecuencia basados en biestables D síncronos. Cada uno recibe como entrada un reloj de 16 Hz y divide dicha señal según su configuración interna, generando frecuencias de salida equivalentes a 16/2, 16/4 y 16/8. Esto se logra realimentando la salida Q hacia la entrada D, pero invertida, de modo que en cada flanco activo del reloj el biestable invierte su estado. Como cada biestable divide la frecuencia entre dos, los reductores adicionales se implementan encadenando dos y tres biestables respectivamente, obteniendo así las cuatro frecuencias de trabajo necesarias para los distintos modos de juego.

La entrada Activador permite habilitar o congelar el divisor: cuando está en 1, el biestable alterna su estado y la frecuencia se reduce, cuando está en 0, el biestable conserva su valor actual y la salida permanece fija. Por su parte, la entrada Reset fuerza la salida Q a 0, asegurando un estado inicial conocido en todos los reductores



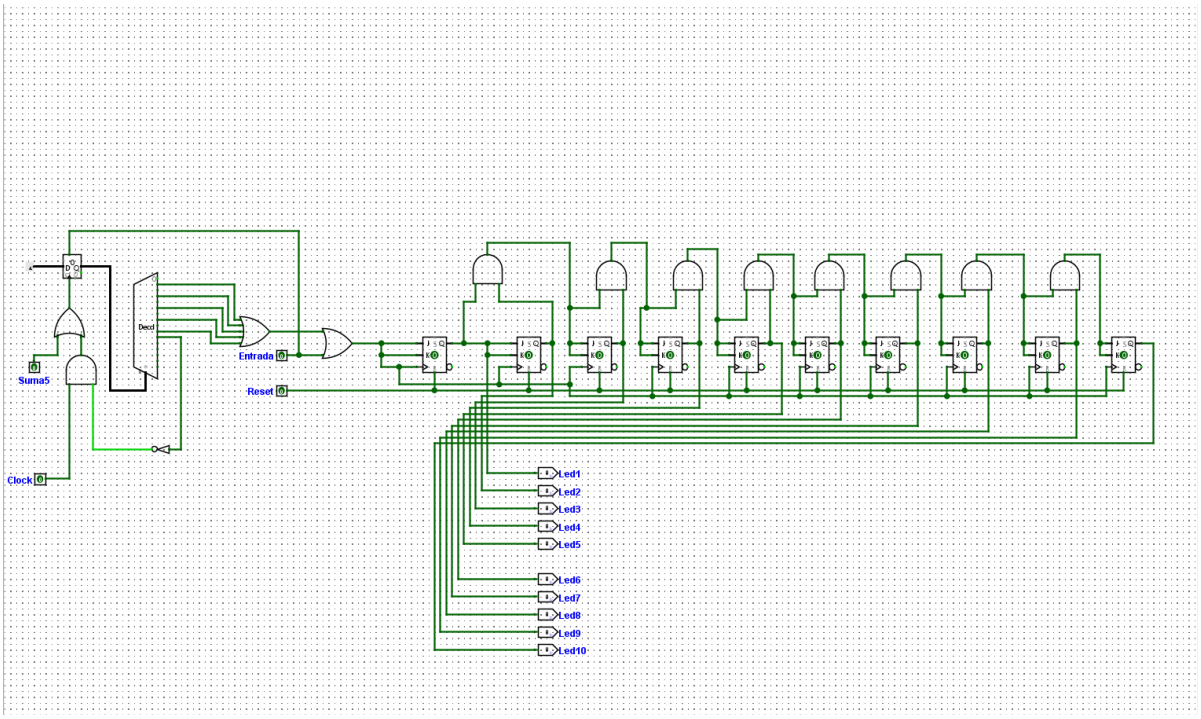
Por otro lado, el sistema de gestión de niveles de velocidades determina en qué nivel de dificultad está el juego en función de la cantidad de bloques colocados. Para ello, cada vez que un bloque colisiona se incrementa un contador interno. El valor de este contador se envía a un conjunto de comparadores, los cuales evalúan si se cumplen las condiciones establecidas para aumentar o disminuir la velocidad de caída, estas mismas salidas son las que posteriormente activan los reductores de frecuencia.



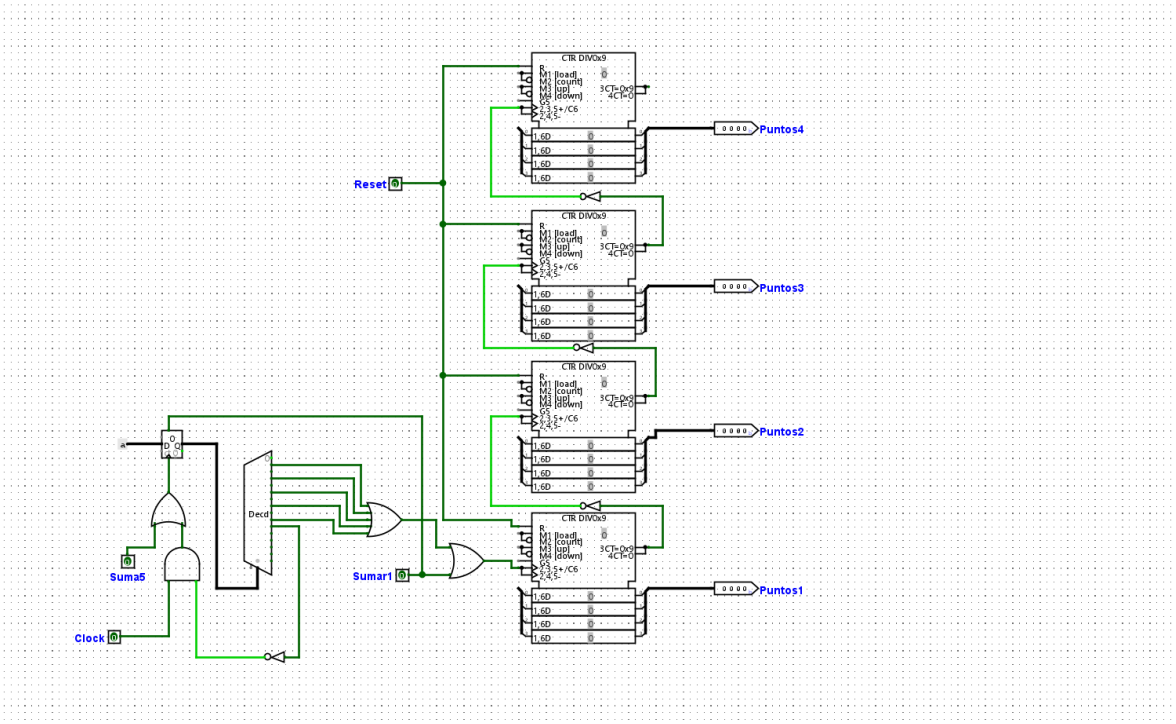
Finalmente, el sistema de manejo de velocidades utiliza los tres reductores de frecuencia junto con el módulo de gestión de nivel para determinar qué señal de reloj debe emplearse en cada momento. De esta forma, el sistema selecciona dinámicamente la frecuencia adecuada según el nivel alcanzado, conformando el bloque encargado de controlar la velocidad efectiva del juego.

2.7 Puntaje y Récord

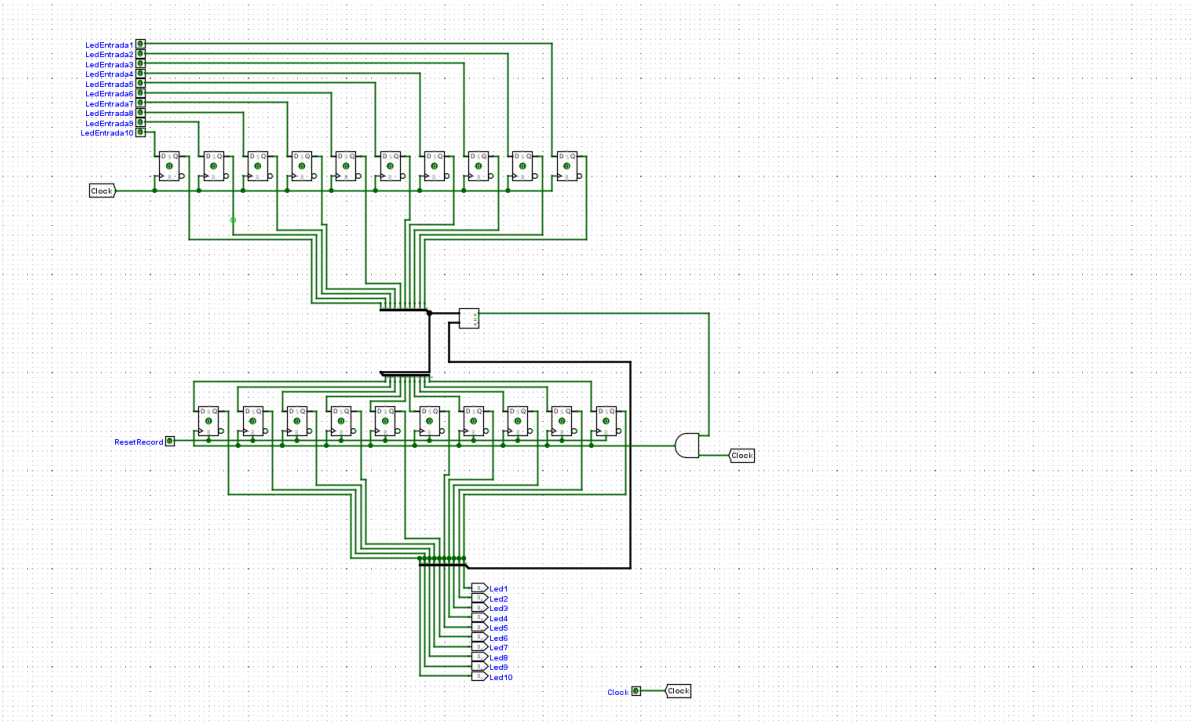
El puntaje suma: +1 por cada bloque colocado y +5 por cada fila eliminada. El récord se actualiza automáticamente cuando es superado.



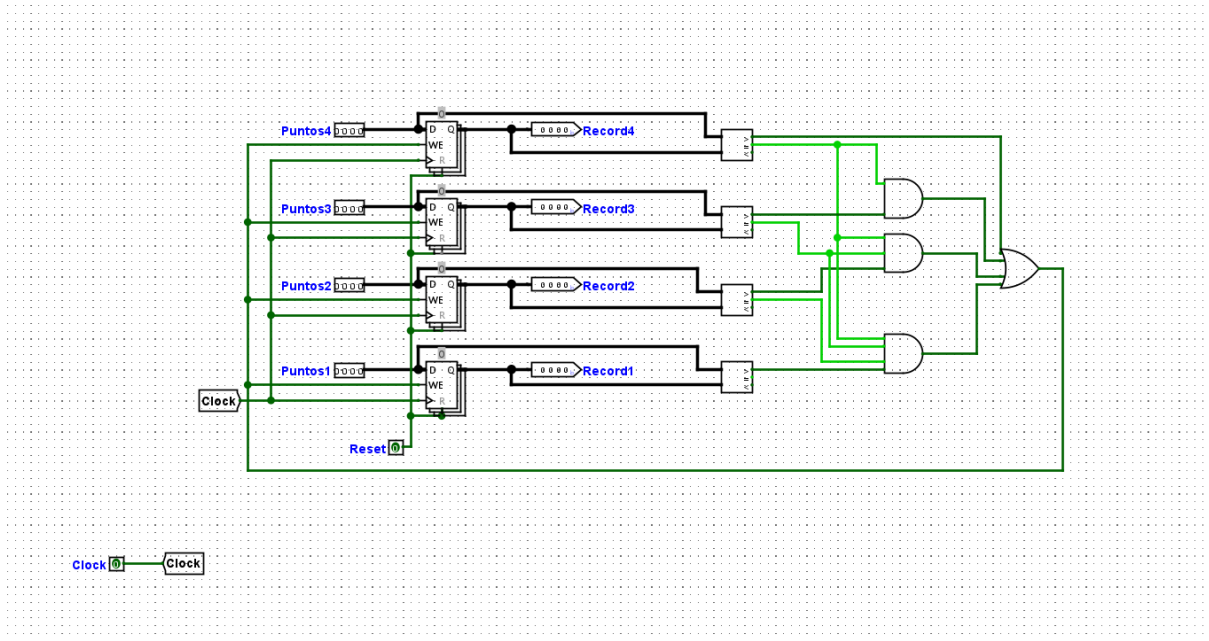
El circuito implementa un contador secuencial de diez estados utilizando una cadena de biestables JK configurados como biestables T, de modo que cada pulso recibido provoca la inversión del estado del biestable correspondiente y habilita al siguiente en la cadena. El pulso de avance se genera mediante lógica combinacional que combina la señal de reloj con condiciones adicionales, mientras que un decodificador determina, a partir del estado actual, cuándo debe producirse el siguiente avance o un reinicio automático. La señal de Reset permite inicializar todo el sistema forzando todos los biestables a cero. Finalmente, cada salida Q alimenta un LED, permitiendo visualizar de forma directa el estado actual del contador mediante una secuencia de diez luces que se activan una a la vez.



El circuito mostrado implementa el sistema de conteo de puntaje del juego utilizando cuatro contadores decimales (CTR DIV/09), cada uno encargado de almacenar un dígito del puntaje total. El pulso de incremento se genera mediante lógica combinacional que combina la señal de reloj con la condición “Suma5”, pasando luego por un descodificador que determina cuándo debe producirse el avance efectivo en el contador correspondiente. La señal “Sumar1” habilita el incremento del dígito de menor peso, y cada contador propaga el acarreo al siguiente cuando alcanza su valor máximo, permitiendo así la formación de un número de varios dígitos. Además, la señal de Reset se distribuye a todos los contadores, garantizando que el puntaje pueda ser reiniciado a cero en cualquier momento. Las salidas de cada contador se presentan en los bloques “Puntos1” a “Puntos4”, que representan visualmente el valor actual del puntaje.



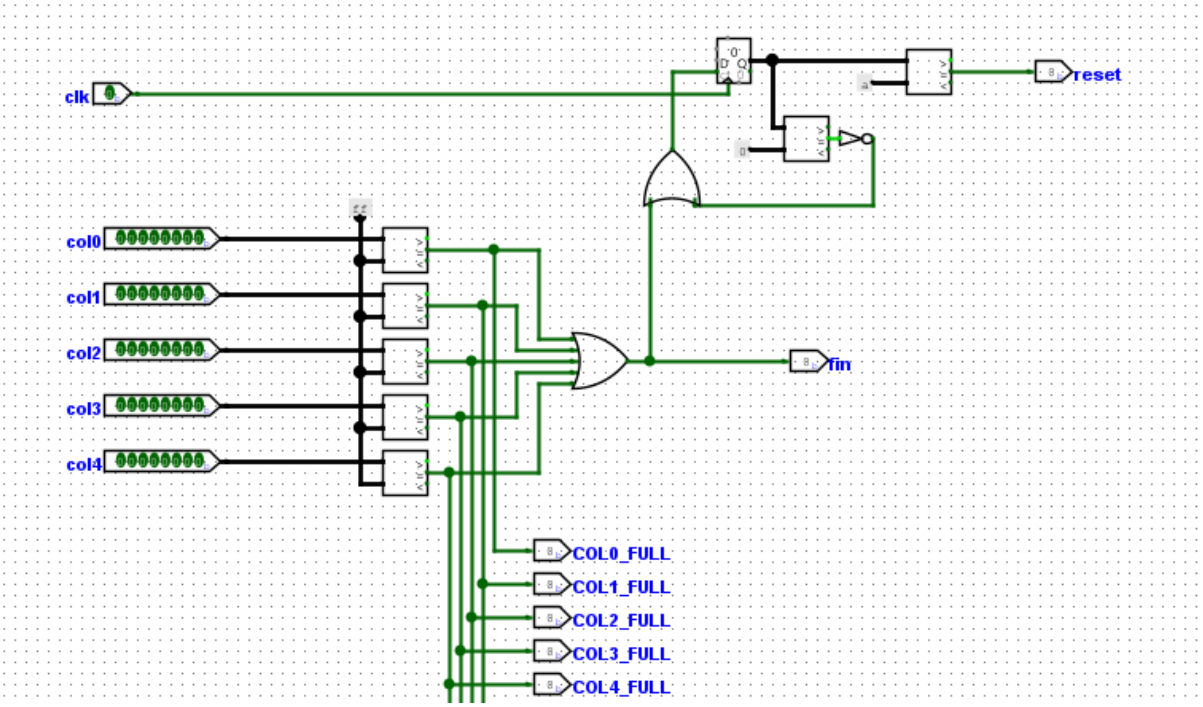
El circuito implementa el sistema de récord del juego comparando el puntaje actual con el valor almacenado previamente. Los diez biestables superiores reciben el estado de los LEDs del puntaje actual, y mediante lógica combinacional se determina si este valor supera al récord. Cuando eso ocurre, se habilitan los diez biestables inferiores, que copian y conservan el nuevo récord hasta que se activa la señal de “ResetRecord”. La salida final muestra en forma estable el valor más alto alcanzado durante la partida.



El circuito compara el puntaje actual con el récord guardado evaluando los cuatro dígitos de mayor a menor. Si el puntaje actual supera al récord en cualquiera de los niveles, la lógica habilita la escritura y los registros “Record1–Record4” actualizan el

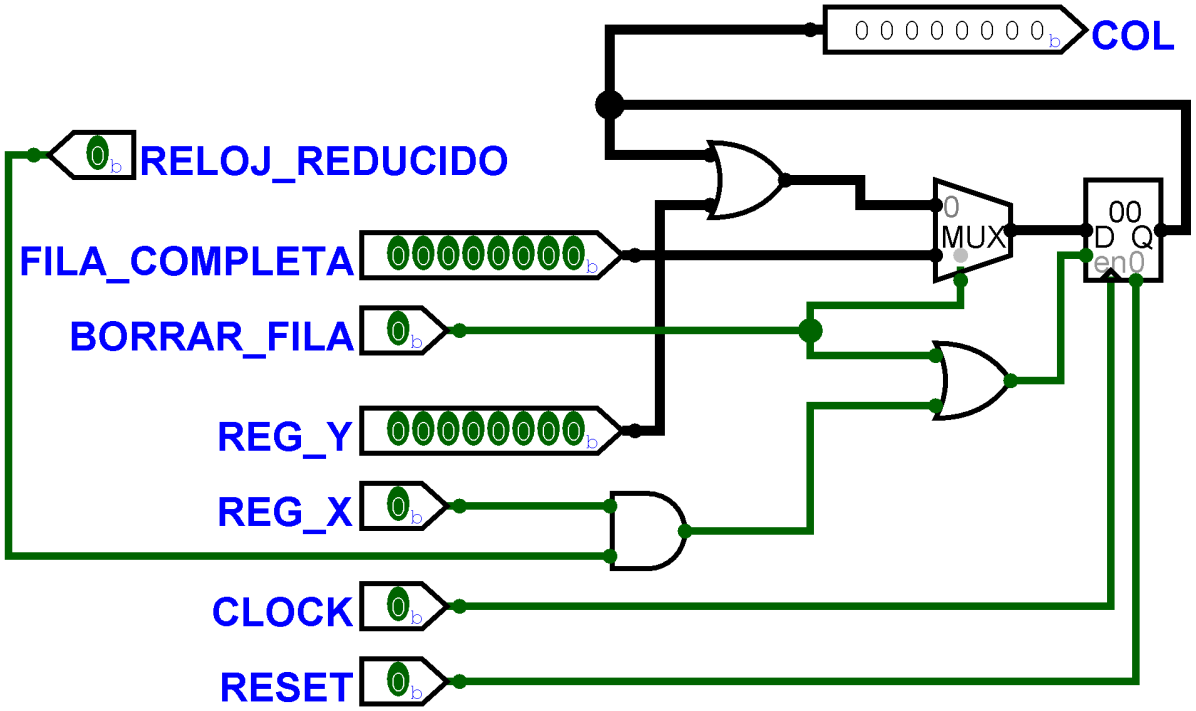
nuevo valor en el siguiente pulso de reloj. La señal Reset permite reiniciar el récord a cero.

2.9 Fin de juego



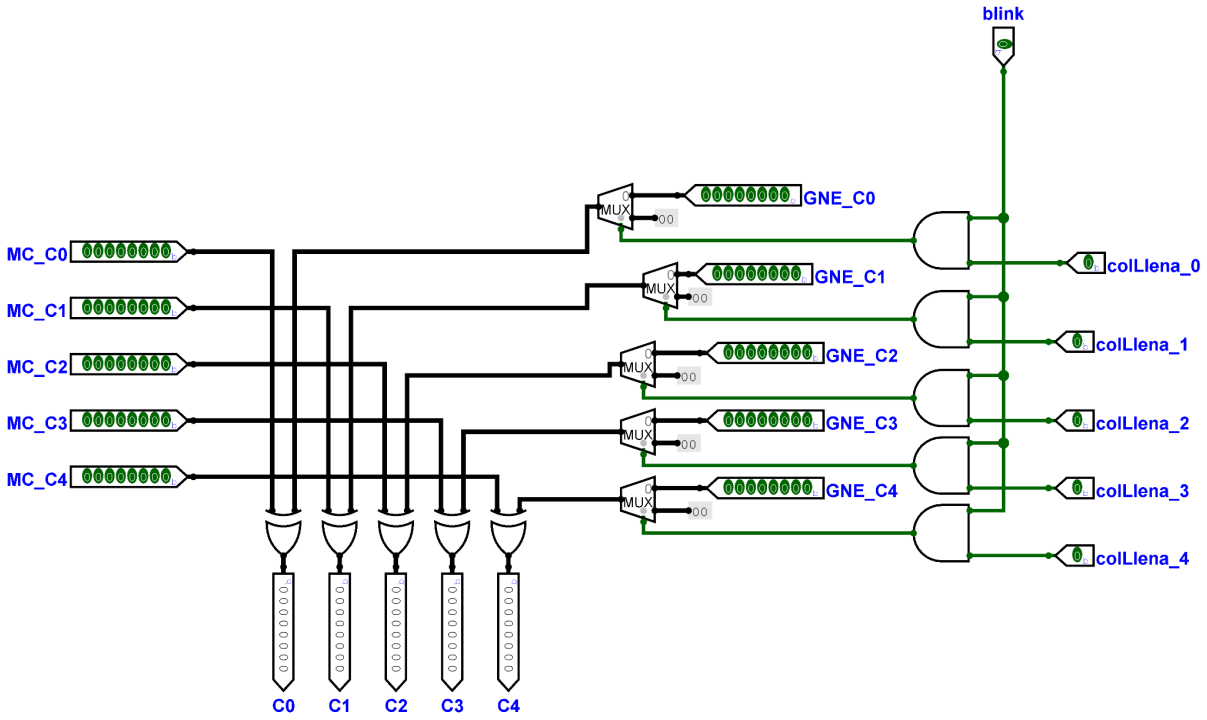
Para la finalización del juego lo que buscamos es que una columna este llena, o sea, al ser un número de 8bits, su valor sea igual a 0xff. Al pasar esto en alguna columna, se manda por una señal a mergeTableros columna fue la que se lleno y por un contador y un multiplexor, la columna llena va a estar destellando entre 0xff y 0x0 durante 15 pulsos de reloj, ya que cuando el reloj llegue a 0xa, este va a mandar la señal de reset lo cual va a resetear todas las columnas de la memoria.

2.10 Registro de columnas



A la hora de la creación del sistema para Guardar un Nuevo Estado, decidimos implementar una serie de registros personalizados para el correcto manejo de la información de cada columna del tablero.
Para formar cada columna, se deben usar una serie de datos (REG_X, REG_Y, FILA_COMPLETA) y señales (RELOJ_REDUCIDO, BORRAR_FILA, CLOCK, RESET) distintas, por eso, en este registro personalizado, nos aseguramos de que los datos mantienen una estructura igual para cada columna en un unico circuito centralizado.

2.11 Juntar Tableros



Este circuito se encarga de unir toda la información que se muestra en el tablero. Por un lado tiene MC_CX, que es el bit que va cayendo, y por otro lado GNE_CX, que contiene todos los bits fijos que ya quedaron guardados. Además recibe colLlena_X, que indica si una columna completó todos sus niveles.

Cuando una columna no está llena, simplemente combina el bit dinámico con los valores estáticos usando XOR, así el movimiento se ve correctamente sobre el tablero. En cambio, si la columna está llena y la señal Blink está activa, el MUX alterna entre mostrar el estado real o apagar esa columna por un momento, creando el efecto de parpadeo para avisar al jugador. El resultado final de todo eso sale como el bus unificado hacia el display.