ARQUITECTURA DE COMPUTADORAS

Etapa 1

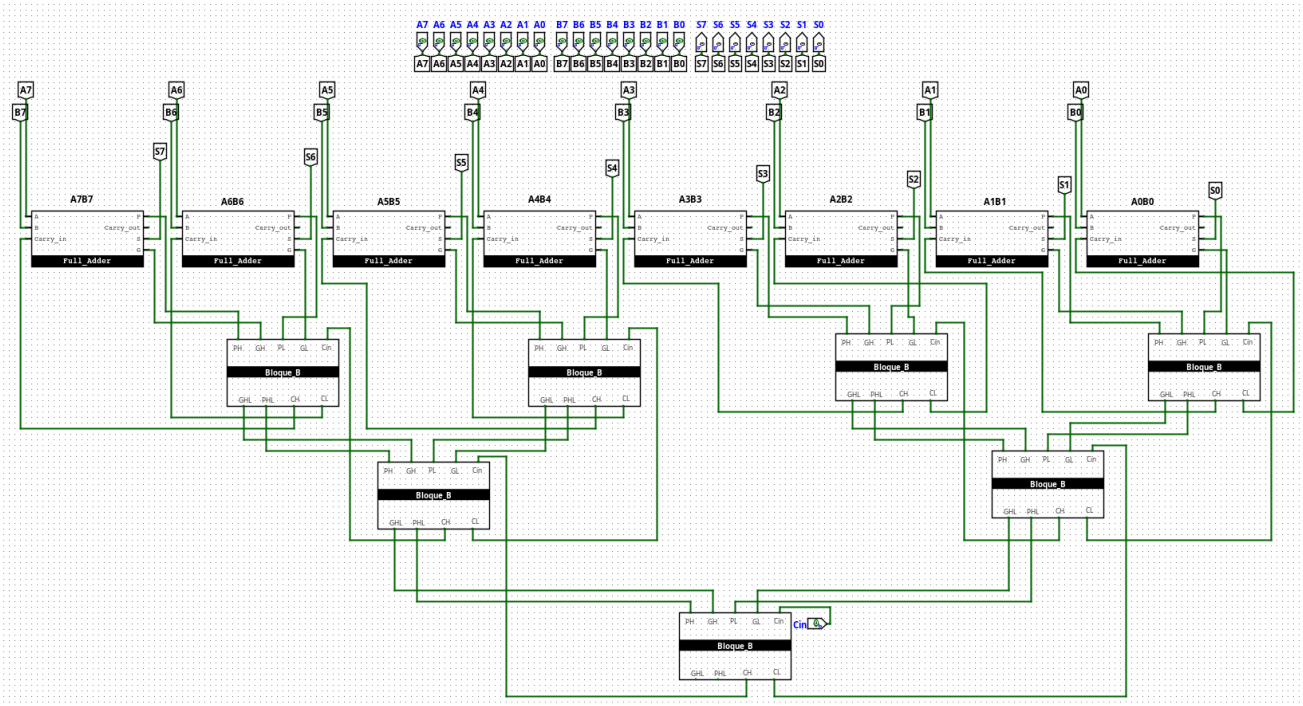
Comisión 25
Integrante 1: Luciana Vecchiotti – LU 146460
Integrante 2: Joaquín Gamonal – LU 146610
Integrante 3: Alvaro Martin – LU 145714
Integrante 4: Tomas Gadea – LU 146155

Fecha: 2025-04-22

Logisim-Evolution v3.9.0

INTRODUCCIÓN

En este primer cuatrimestre del año 2025, correspondiente a la asignatura “Arquitectura de Computadoras”, se nos ha presentado como proyecto el desafío de diseñar un circuito sumador **Lookahead Tree Adder**, que denominaremos **SUMLAT** (ver Anexo 1).



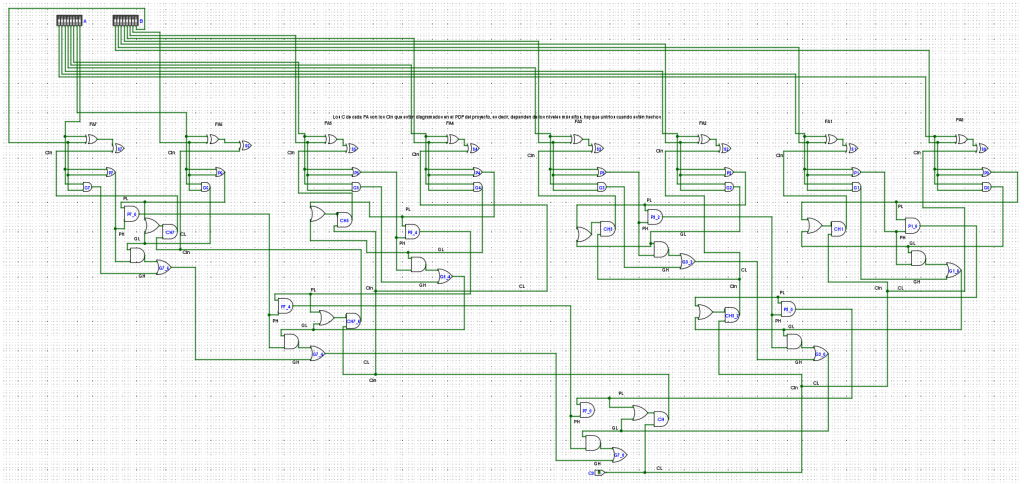
Anexo 1: Circuito final

Dicho circuito, diseñado con la herramienta de simulación lógica **Logisim-Evolution**, tiene como objetivo principal **acelerar la operación de suma binaria** en comparación con sumadores más simples como el **Ripple-Carry Adder**. Este enfoque mejora significativamente el rendimiento al evitar la propagación secuencial de acarreos, utilizando un árbol jerárquico para calcularlos anticipadamente.

DESARROLLO

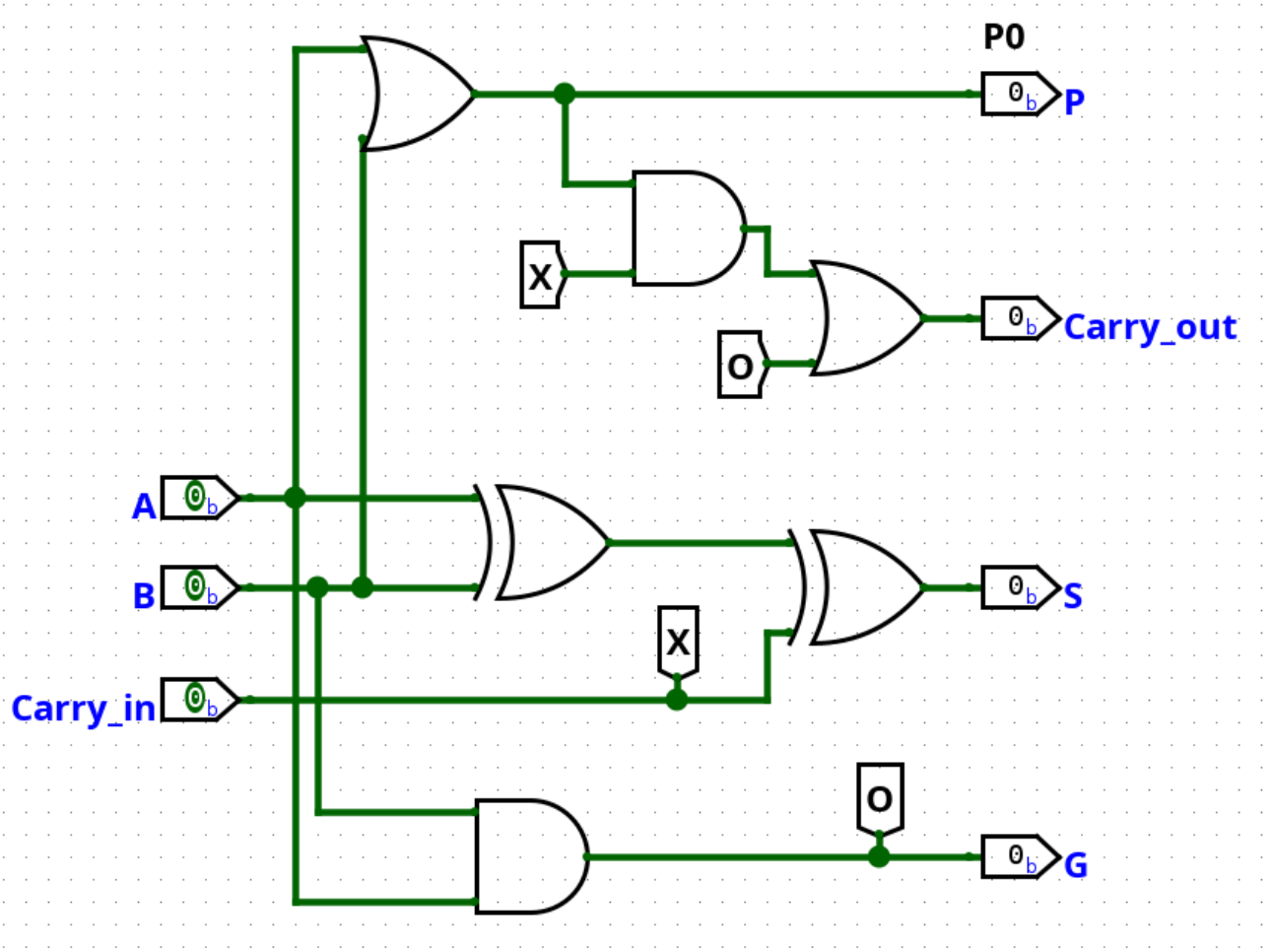
Desde un primer momento se nos brindó un esquema visual de cómo debería lucir el árbol sumador una vez finalizada su implementación en Logisim-Evolution, por lo que decidimos comenzar a trabajar de manera inmediata tras la presentación del enunciado.

Nuestra primera aproximación fue implementar todos los bloques necesarios en un único circuito, tanto los **Full Adder** como los **bloques de niveles superiores B**. Sin embargo, durante el proceso de creación y posterior análisis, observamos que el circuito resultante era extremadamente confuso: cables cruzados en todas direcciones, errores en las conexiones de compuertas, y una dificultad general para interpretar su funcionamiento (ver Anexo 2). Este intento inicial tuvo lugar el día martes 8/4, el mismo día en que se presentó el proyecto.

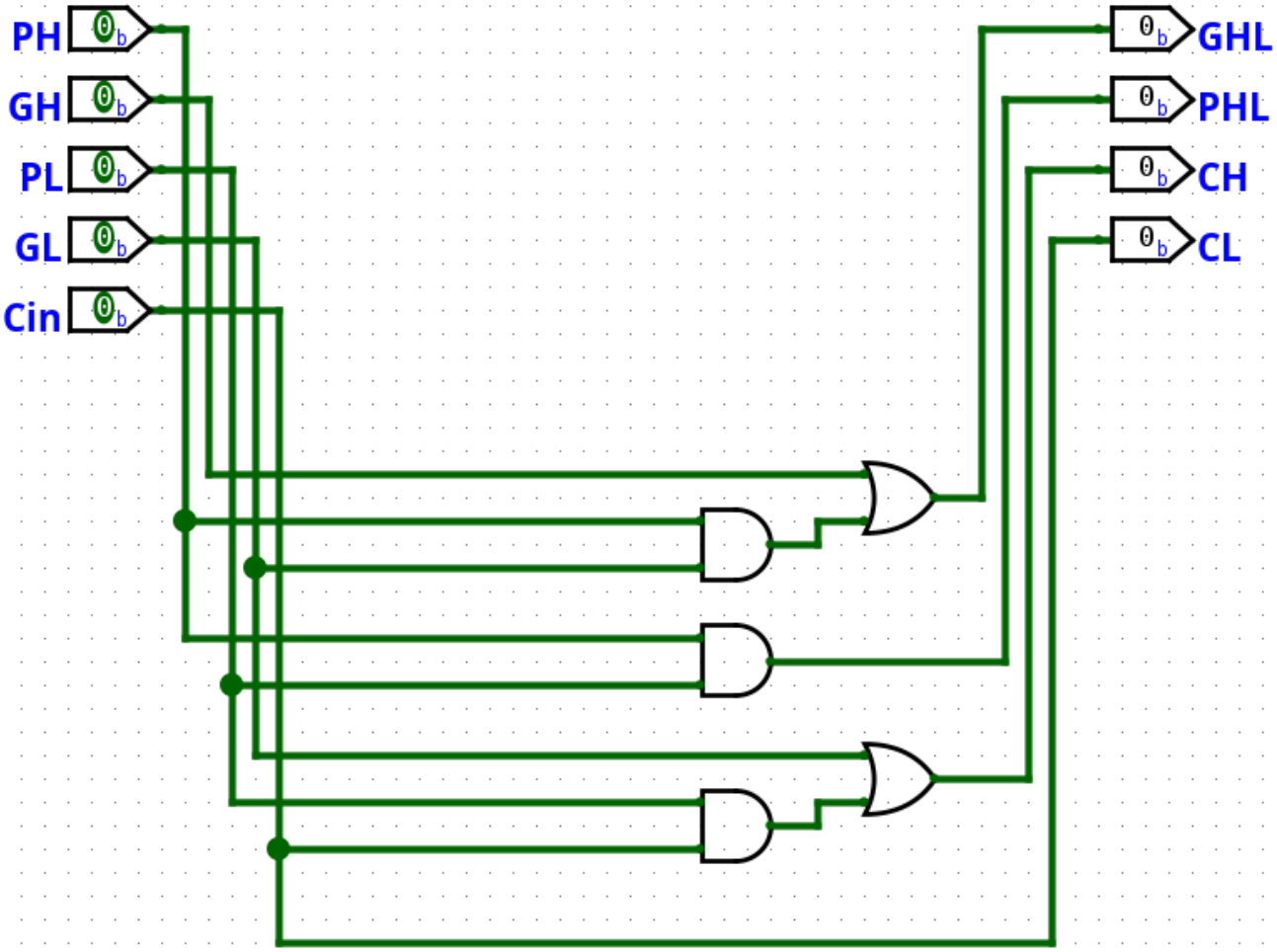


Anexo 2: Primer circuito desarrollado

El día 9/4 se nos facilitó un PDF desde la cátedra explicando el uso de **librerías**, lo cual nos permitió reorganizar el circuito de manera modular. A partir de este punto, decidimos construir dos subcircuitos separados: uno que representara el **Full Adder con lógica de generación y propagación (FAGP)** y otro que implementara el **bloque B** para niveles superiores. Ambos serían utilizados como **librerías internas** dentro del circuito principal SUMLAT (ver Anexos 3 y 4).



Anexo 3: Bloque FAGP



Anexo 4: Bloque B

Esta estructura modular permite una implementación más clara, ordenada y mantenible del circuito completo.

DESCRIPCIÓN DEL CIRCUITO

Bloque FAGP (Full Adder con lógica de generación y propagación)

Este bloque corresponde a un *Full Adder* tradicional, pero con lógica adicional para determinar si genera (**G**) y/o propaga (**P**) un acarreo. Recibe como entrada dos bits **A** y **B**, junto con un **acarreo de entrada** (**Cin**), y entrega como salida:

- **G (Genera):** vale 1 si A = 1 y B = 1, es decir, **G = A AND B**
- **P (Propaga):** vale 1 si al menos uno de los bits es 1, es decir, **P = A OR B**

- **S (Suma):** el resultado de **A⊕B⊕Cin** (A XOR B XOR carry_in)

- **Cout:** acarreo de salida

Este bloque será utilizado por los bloques superiores para calcular los acarreos sin necesidad de esperar a que se propaguen uno por uno.

Bloque B (bloque jerárquico de generación y propagación)

El **bloque B** representa un nivel jerárquico superior del sumador, combinando los resultados de dos bloques anteriores (que pueden ser FAGP o bloques B de nivel inferior), denominados:

- **H (High):** bloque correspondiente a los bits más significativos (de mayor peso posicional).

- **L (Low):** bloque correspondiente a los bits menos significativos (de menor peso posicional).

Recibe como entrada los valores de propagación y generación de ambos bloques:

- **PH, GH**, que corresponden al valor de propagación (**P**) y generación (**G**) del bloque **H** (más significativo)

- **PL, GL**, que corresponden al valor de propagación (**P**) y generación (**G**) del bloque **L** (menos significativo)

- **Cin**: acarreo de entrada

Y calcula:

- **PHL (Propagación total):** si ambos bloques propagan, entonces el bloque completo propaga ($PHL = PH \cdot PL$)

- **GHL (Generación total):** si L genera y H propaga, o si H genera, entonces el bloque completo genera ($GHL = GH + (PH \cdot GL)$)

Además, se calculan los acarreos de entrada para los bloques H y L:

- **CL = Cin**

- **CH = GL + (PL · Cin)**

Este mecanismo permite al bloque B anticipar los acarreos para sus bloques hijos, habilitando así el cálculo paralelo característico del Lookahead Tree Adder.

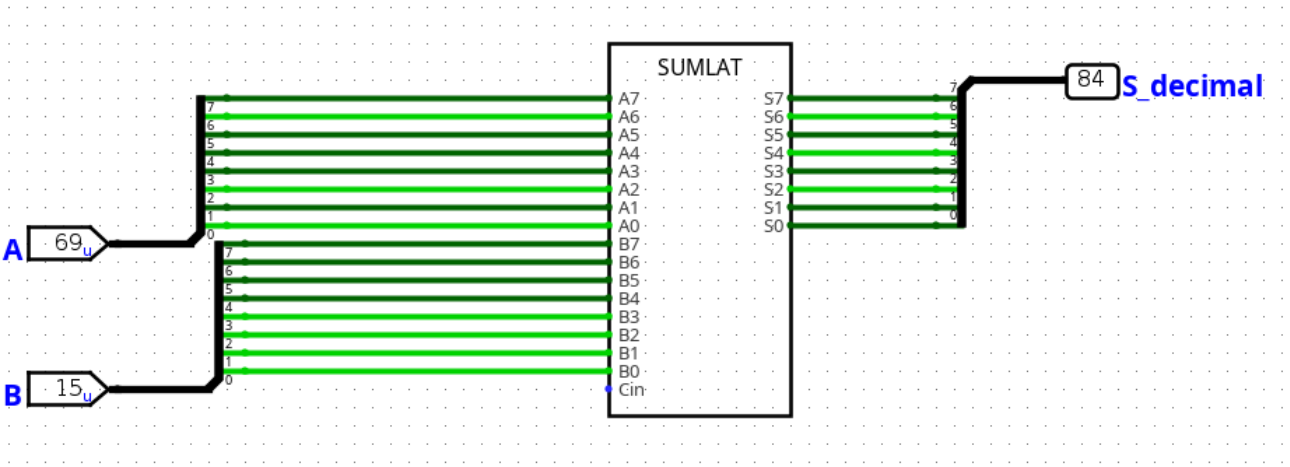
RESULTADOS

Se realizaron pruebas con diferentes combinaciones de números de 8 bits y se obtuvieron los resultados esperados. En los casos de **overflow** (cuando el resultado excede los 8 bits), este se descarta, tal como es de esperarse en un sumador de tamaño fijo.

Se verificó que la salida del sumador coincidiera con la suma binaria correcta en todos los casos válidos.

Asimismo, se observó que, a pesar de una mayor complejidad estructural, el **Lookahead Tree Adder** presentó una mejora significativa en velocidad en comparación con un **Ripple Carry Adder**, validando su eficiencia en términos de tiempo de propagación de acarreo.

Como detalle extra, hemos agregado un circuito auxiliar llamado **prueba_SUMLAT**, donde, insertando 2 valores “A” y “B” en formato decimal, obtendremos el resultado de la suma en decimal. Este circuito auxiliar permite una mayor comodidad para probar la correctitud del circuito (ver Anexo 5).



Anexo 5: bloque prueba_SUMLAT