Universidad del Istmo de Guatemala

Facultad de Ingeniería

Segundo Parcial

Arquitectura de Computadoras y Microcontroladores 1

Maximiliano González

15 de septiembre de 2025

Indice

Serie 0 - (5 pts) - Instrucciones: FSM en HDL

Implemente la maquina/proceso del parcial 1 en Verilog-SystemVerilog como una FSM

Video:

Serie 1 - (5 pts) - Instrucciones: Estudio de sumadores, simulación en Logisim

Utilizando Logisim evolution, realice la simulación de los siguientes 3 sumadores junto a su análisis de tiempo

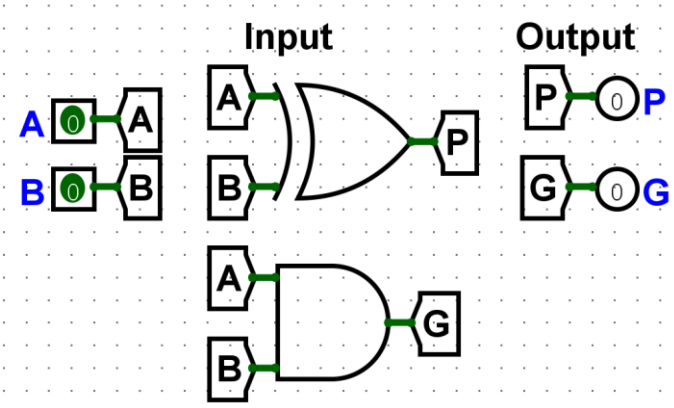
- **Ripple-Carry Adder**
- **Carry-Lookahead Adder**
- **Prefix Adder**

Entregable: Video explicando la operación y funcionamiento de cada arquitectura incluyendo las ventajas/desventajas de cada uno. Finalmente, conteste que sumador usaría con énfasis en cantidad de compuertas y velocidad para aplicaciones:

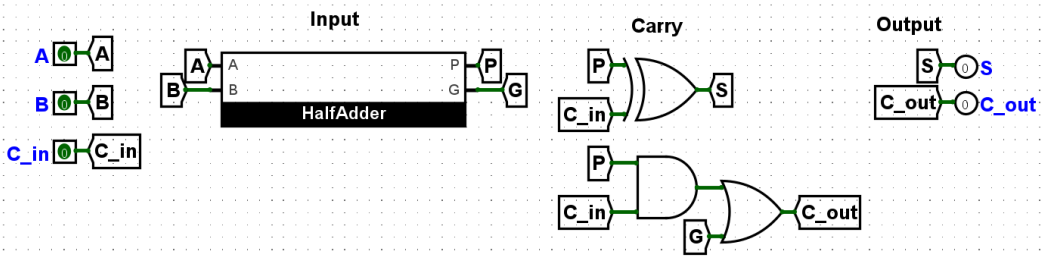
- **Lentas con restricción de espacio y presupuesto**
- **Rápidas sin restricción de espacio y presupuesto**
- **Rápidas con restricción de espacio y presupuesto**

Full Adder

- Half Adder

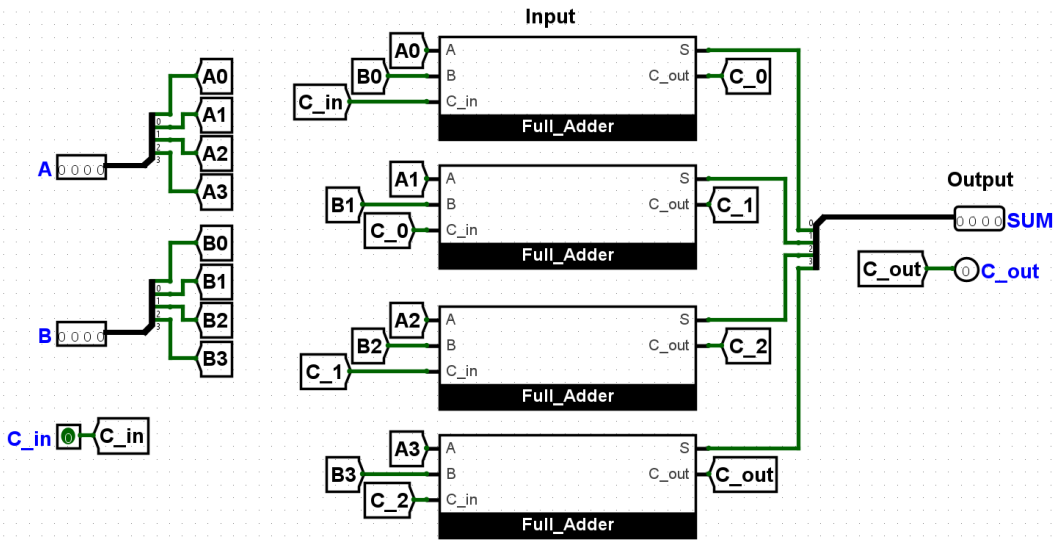


- Full Adder Final



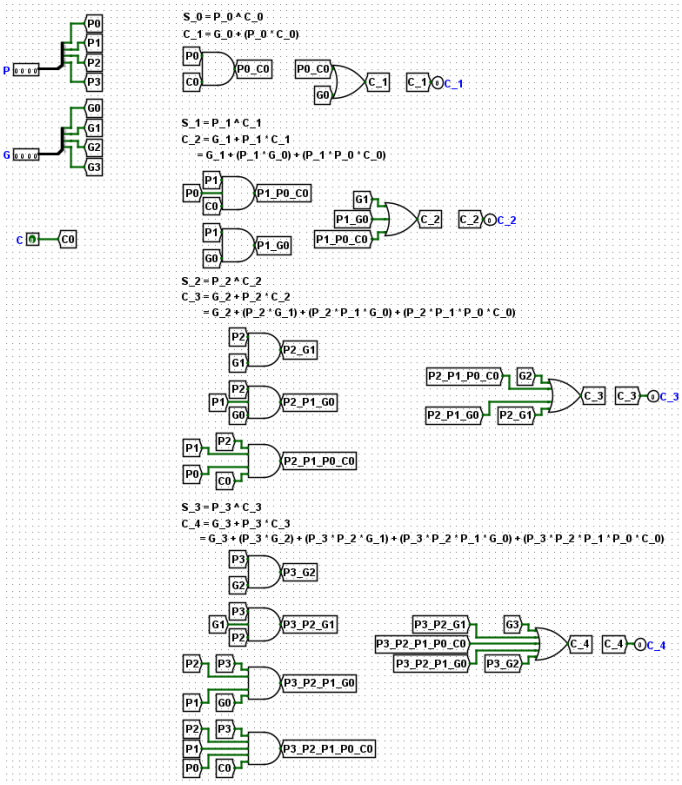
Ripple-Carry Adder

- Ripple-Carry Adder Final

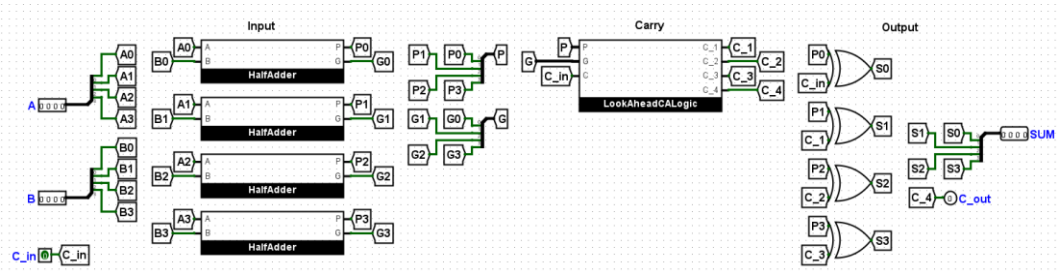


Carry-Lookahead Adder

- Lookahead Logic

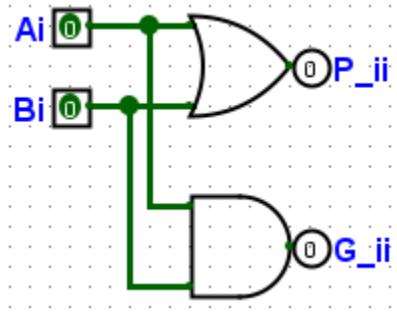


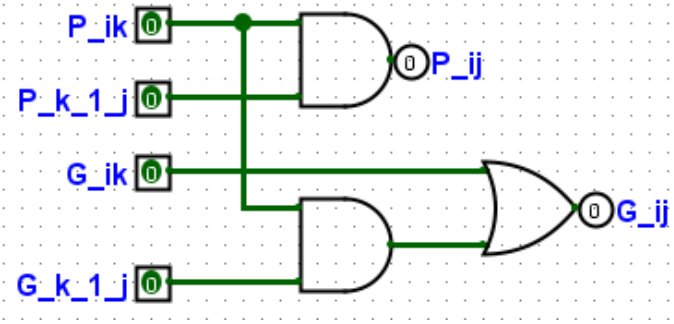
- Lookahead Adder Final

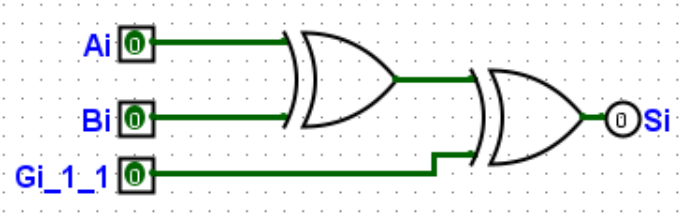


Prefix Adder

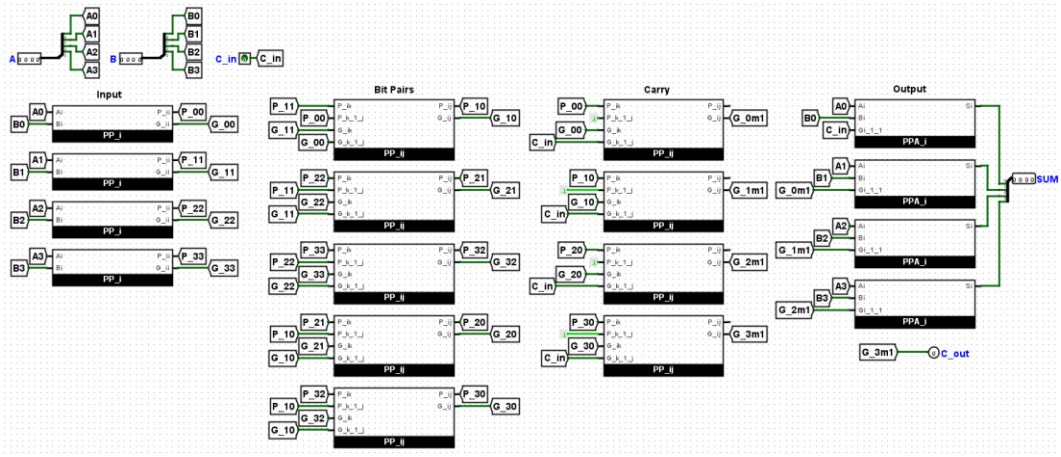
- Parallel Prefix Adder Logica



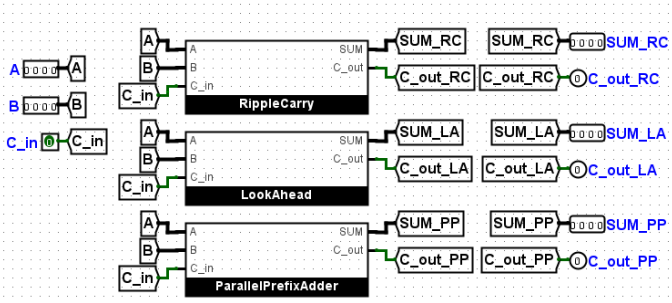


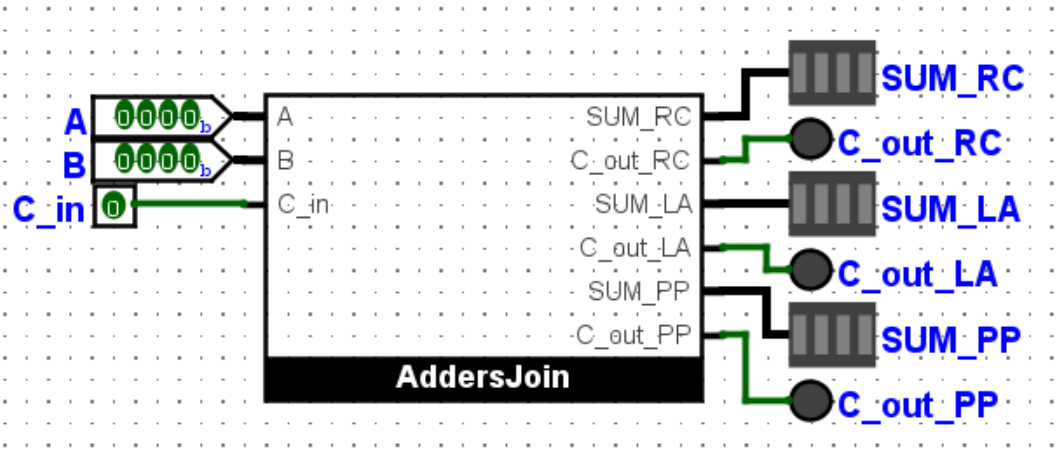


- Parallel Prefix Adder Final



Adders Circuito Final





- Video:

Cálculo de tiempos

- Gates Utilizadas
  - XOR - https://www.digikey.com/en/products/detail/texas-instruments/SN74AHC1G86DCKR/373814

5.6 Switching Characteristics, 3.3 V ± 0.3 V

over recommended operating free-air temperature range (unless otherwise noted) (see Load Circuit and Voltage Waveforms)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	LOAD CAPACITANCE	T_A = 25°C			–40°C to 85°C		–40°C to 125°C		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{PLH}	A or B	Y	C_L = 15 pF		7	11	1	13	1	14	ns
t_{PHL}					7	11	1	13	1	14	

Propagation Delay: $13ns$

Contamination Delay: $1ns$

- OR - https://www.digikey.com/en/products/detail/texas-instruments/SN74LVC1G32DBVR/381323

PARAMETER	FROM (INPUT)	TO (OUTPUT)	–40°C to 85°C								UNIT
			V_{CC} = 1.8 V ± 0.15 V		V_{CC} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A or B	Y	1.9	7.2	0.8	4.4	0.9	3.6	0.8	3.4	ns

Propagation Delay: $3.6ns$

Contamination Delay: $0.9ns$

- AND - https://www.digikey.com/en/products/detail/texas-instruments/SN74LVC1G08DCKR/385719

PARAMETER	FROM (INPUT)	TO (OUTPUT)	–40°C to 85°C								UNIT
			V_{CC} = 1.8 V ± 0.15 V		V_{CC} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A or B	Y	1.5	7.2	0.7	4.4	0.8	3.6	0.8	3.4	ns

Propagation Delay: $3.6ns$

Contamination Delay: $0.8ns$

- Ripple Carry Adder
  - $t_{ripple} = Nt_{FA}$

    $= 4 * (3.6ns + 3.6ns + 13ns)$

    $= 80.8ns$

- Look Ahead Adder
  - $t_{CLA} = t_{pg} + t_{pg\_block} + \left(\frac{N}{k-1}\right) t_{AND\_OR} + kt_{FA}$

    $= 13ns + (5 * 3.6ns) + \left(\frac{4}{4-1}\right)(2 * 3.6ns) + ((3.6ns * 2) + 13ns)$

    $= 31ns + 9.6ns + 20.2ns$

    $= 60.8ns$

- Parallel Prefix Adder
    - $t_{PA} = t_{pg} + log_2 N(t_{pg-prefix}) + t_{XOR}$
      $= 3.6ns + log_2 4(3.6ns * 2) + 13ns$
      $= 31ns$

Preguntas

- Lentas con restricción de espacio y presupuesto

    - Se utilizarían Ripple Carry o Full Adders porque, aunque no sean tan rápidas, como las Look Ahead y Parallel Prefix Adders, utilizan menos espacio y son más economicas de hacer por su nivel de complejidad a comparación de las Look Ahead y Parallel Prefix Adders.

- Rápidas sin restricción de espacio y presupuesto

    - Se utilizaría el Parallel Prefix Adder porque usa menos espacio que el Look Ahead Adder por la cantidad de compuertas que utiliza. Por esta misma razón, son más económicas que las Look Ahead Adders y más rápidas que las Ripple Carry Adders.

- Rápidas con restricción de espacio y presupuesto

    - Se utilizarían las Parallel Prefix Adders por las mismas razones mencionadas anteriormente.

Serie 2 – (5 pts) - Instrucciones: Simulación e Implementación de ALU.

Implemente una ALU (Arithmetic Logic Unit) personalizada (Numero de bits distinto a convencional) vista en el capitulo 5 de su literatura. La siguiente figura (Figure 5.17 N - bit ALU with output flags) implementa 4 operaciones en registros de N bits (Suma, resta AND y OR).

1. **Utilizando su último número de carne (1 si termina en 0) multiplicado x2 menos su penúltimo número de carne para la cantidad de bits N de los buses de datos (Tamaño de palabra, usualmente 8, 16, 32, 64... etc bits). Como por ejemplo si mi carne termina en 1998, construiré una arquitectura con buses del tamaño $8*2-9=7$ bits. [Que no sea mayor a los 10 bits]**

    a. **Extienda la funcionalidad de 4 a 6 operaciones, las operaciones a agregarse son:**

    (Ambas explicadas en la Figure 5.18 4-bit shifters: (a) shift left, (b) logical shift right, (c) arithmetic shift right, notar uso de multiplexores)

        i. **Shift right**
        ii. **Shift left**

    Considere su ALUControl debe ampliar el número de bits para poder soportar 6 operaciones y no 4. Utilice bloques jerárquicos para su simulación (De utilidad al implementarlo en HDL)

2. **Realice la implementación de esta ALU en Vivado utilizando SystemVerilog simulando su resultado mostrando su ALU personalizada se pudo implementar en hardware**

Entregable: Video explicando la funcionalidad e implementación de la simulación en Logism Evolution. Video explicando la funcionalidad y simulación en Vivado.

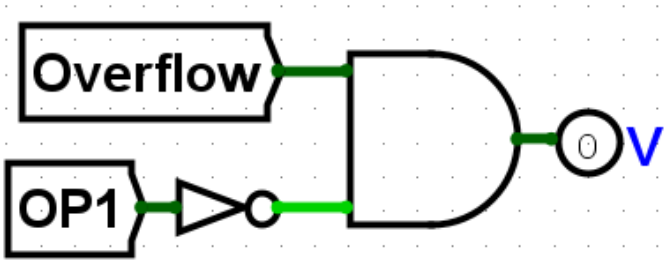
$Carne$: 15990

$|(1 * 2) - 9| = 7\ bits$

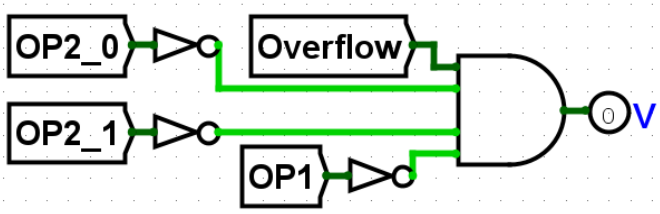
Cantidad de bits de los buses de datos: 7 bits

ALU Flags

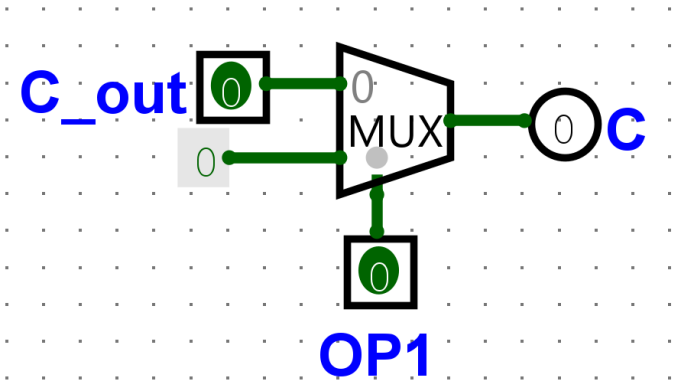
- Flag V



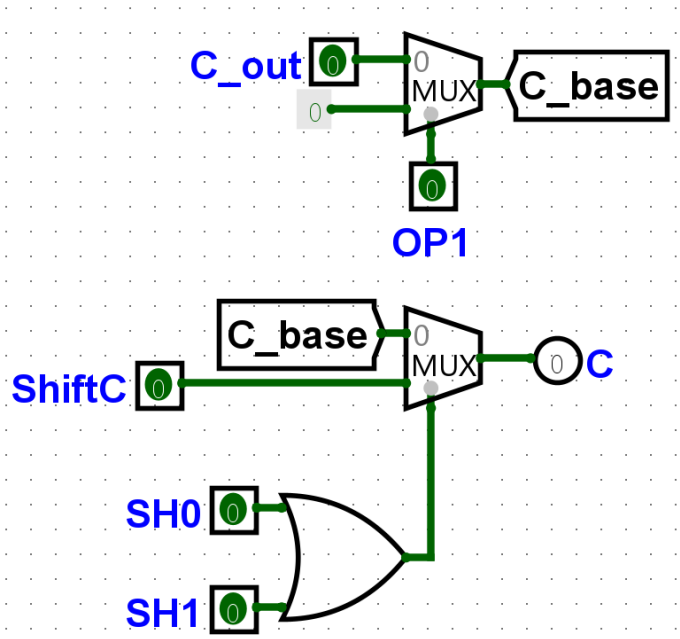
- Flag V Shifter



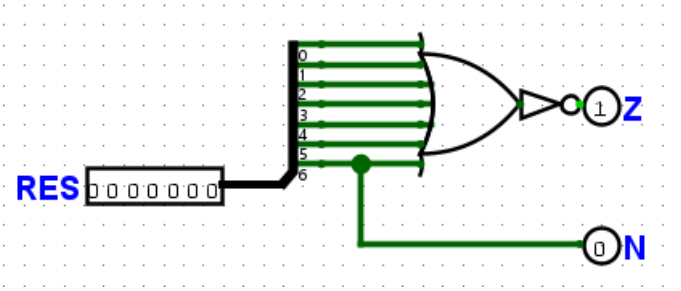
- Flag C



- Flag C ALU with Shifter

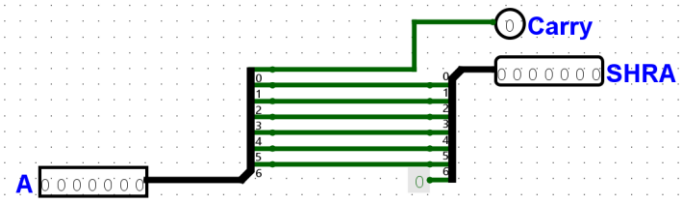


- Flag Z and N

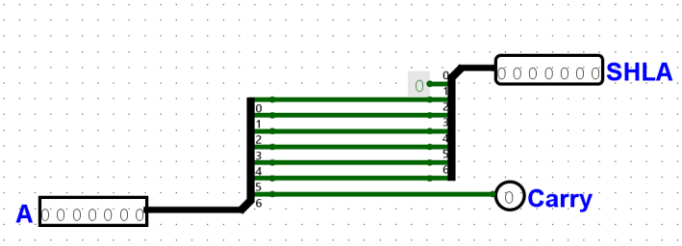


ALU Shifter

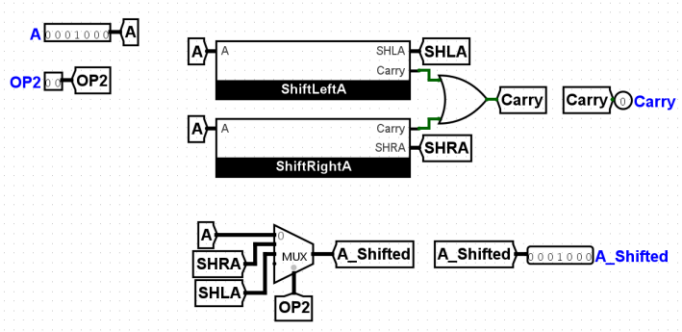
- Shift Right



- Shift Left

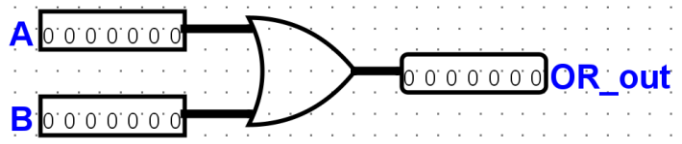


- Shift Circuit

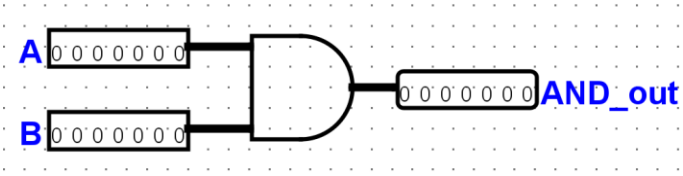


ALU Logic Unit

- OR Circuit

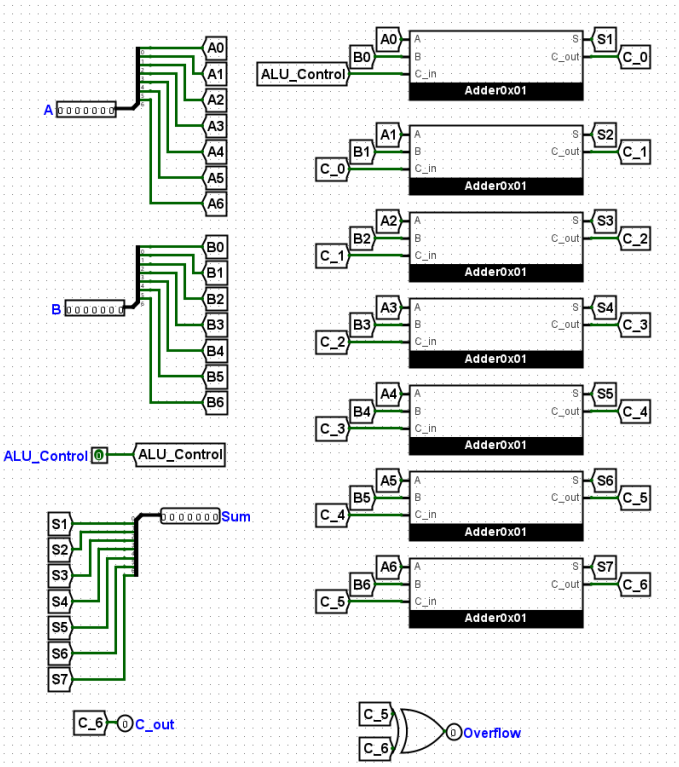


- AND Circuit



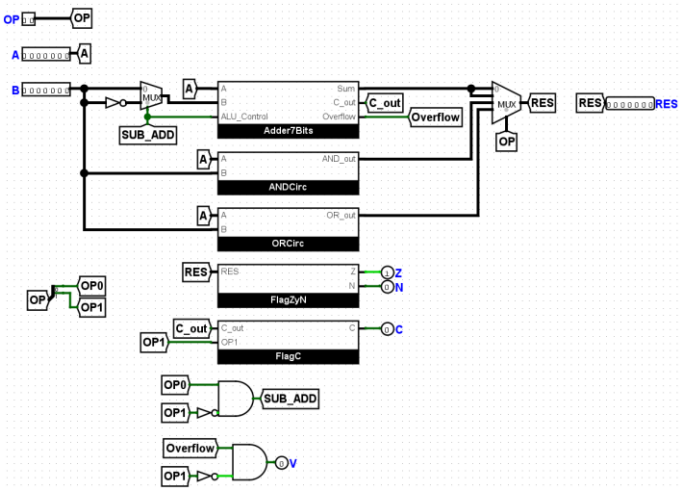
ALU Arithmetic Unit

- 7bit Adder

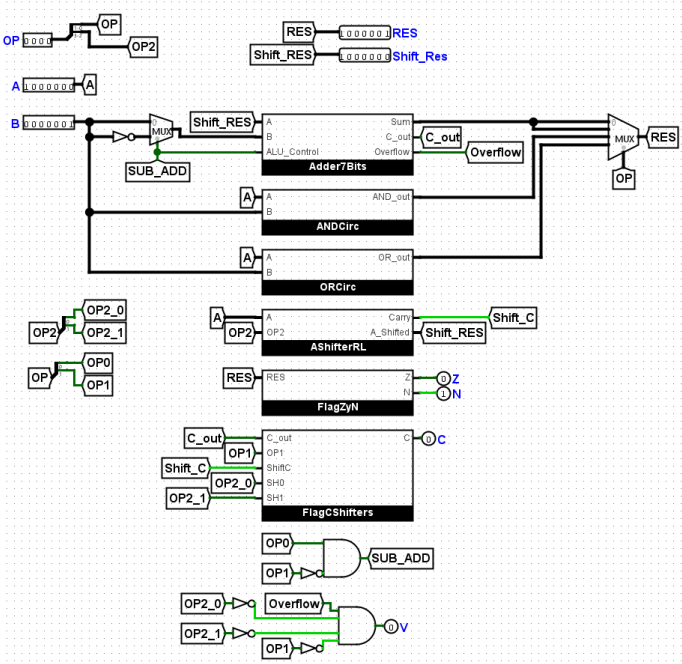


ALUs

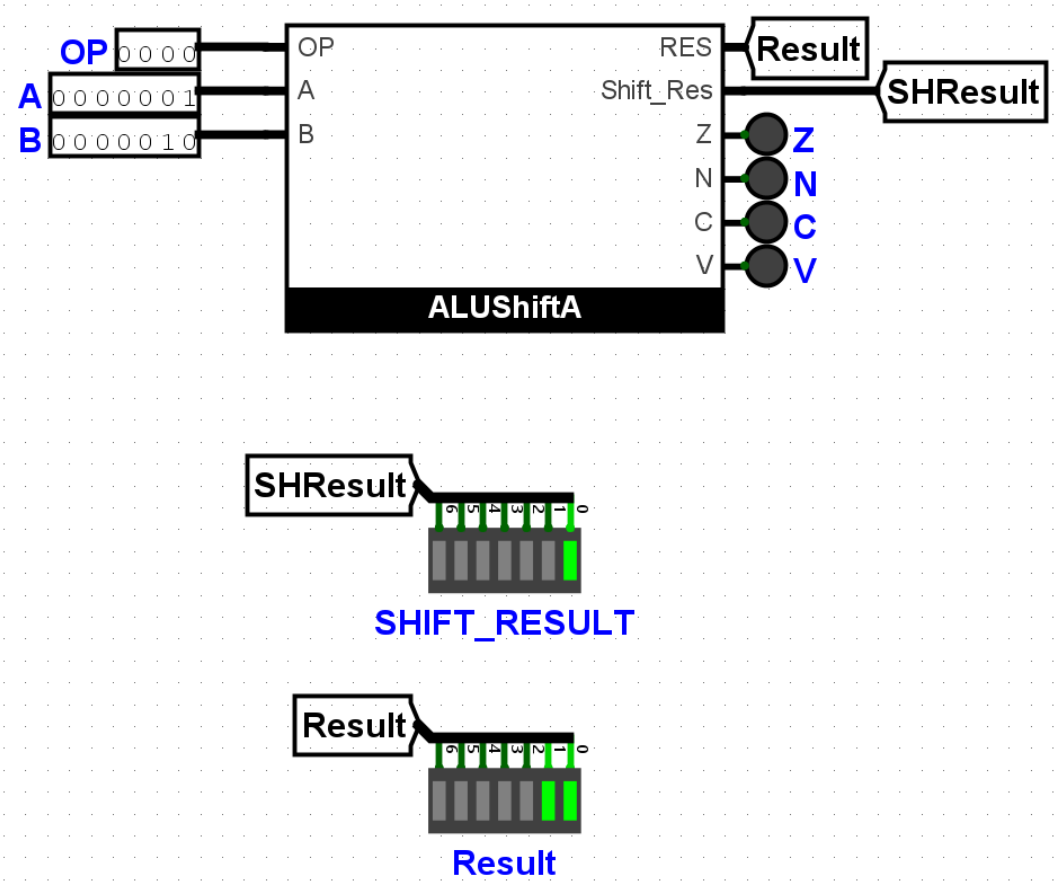
- **ALU without Shifter**



- **ALU with Shifter**



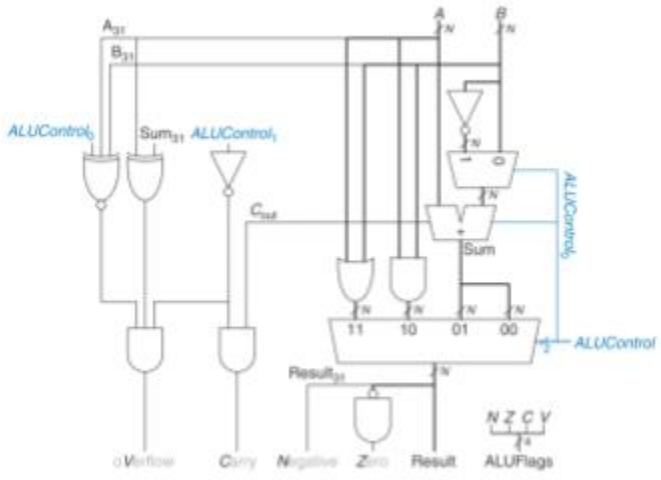
- **ALU Final**



- **Videos:**

Serie 3 – (5 pts) - Instrucciones: Implementación en Hardware real

1. **Utilizando una FPGA, implemente el código HDL creado en el inciso anterior en hardware real utilizando entradas y salidas reales de una placa de evaluación.**



2. **Utilizando una FPGA, implemente el código HDL de su maquina/proceso del parcial 1**

Serie 4 – (5 pts) - Instrucciones: Microcontroladores MMIO

Realice un reloj, utilizando displays de 7 segmentos, con opción a formato 12hrs y 24 hrs seleccionable con un botón. Programe una alarma, la cual realice una acción (buzzer?, LED parpadeante?) para indicar se llego a la hora de alarma.

Entregable de todo el parcial: Repositorio en github con estructura de repo formal. En el repo adjunte archivos fuente y documentación con discusión relevante. Incluya videos cortos de explicación en cada inciso y sub inciso. Se tomará el ultimo commit antes de la fecha de entrega del parcial