Laboratorio de Arquitectura de computadores GITT Entrega 4

Fecha de entrega: Viernes 16/11/2012 Grupo: Viernes

Autores: Manuel Montoya Catalá

**Funcionamiento del CSA**

# Para implementar CSA dividimos nuestro sumador de M bits en etapas de sumadores

mas pequeños de "N" bits cada uno.

- Todas las etapas son iguales, son sumadores CSA, esto se hace para mejorar rendimiento como ahora veremos.

# Cada etapa CSA de N bits está formada por:

- 2 sumadores RCA de N bits: -> 1º con carry siempre a 0

-> 2º con carry siempre a 1

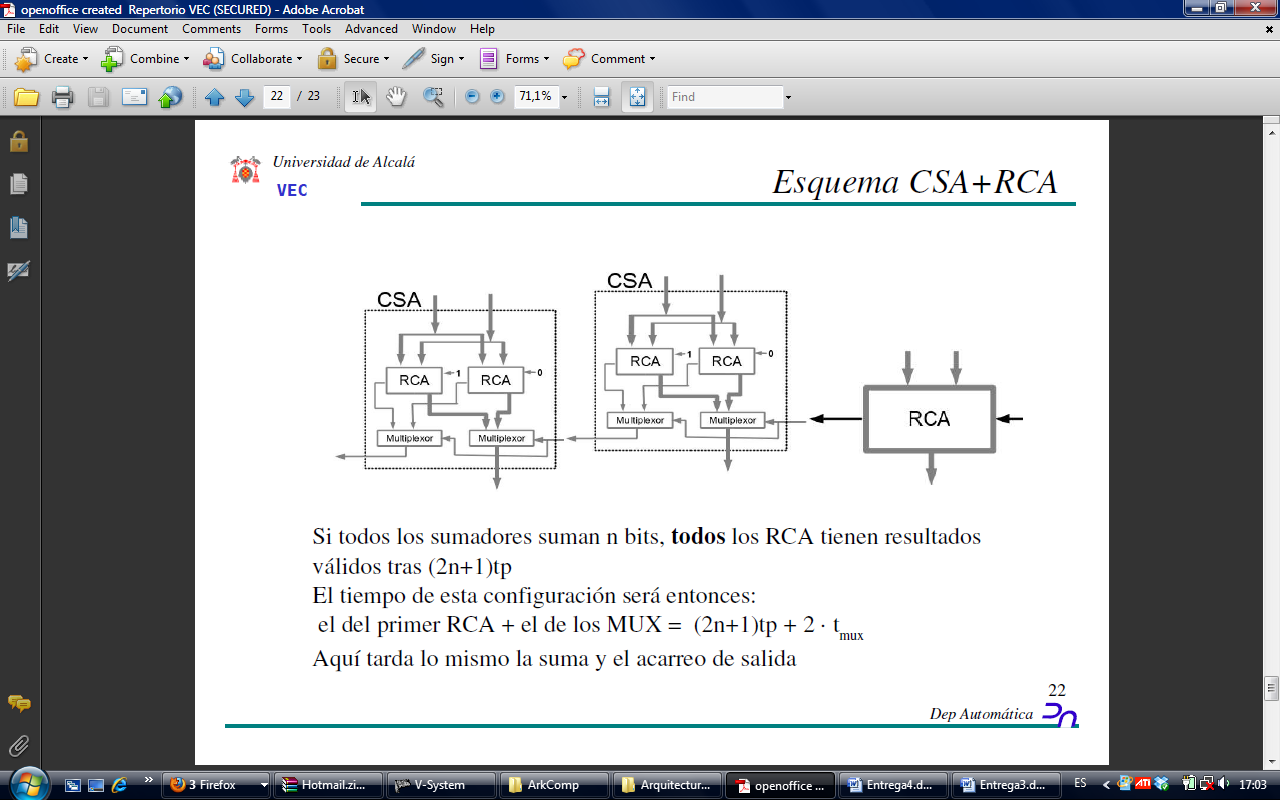
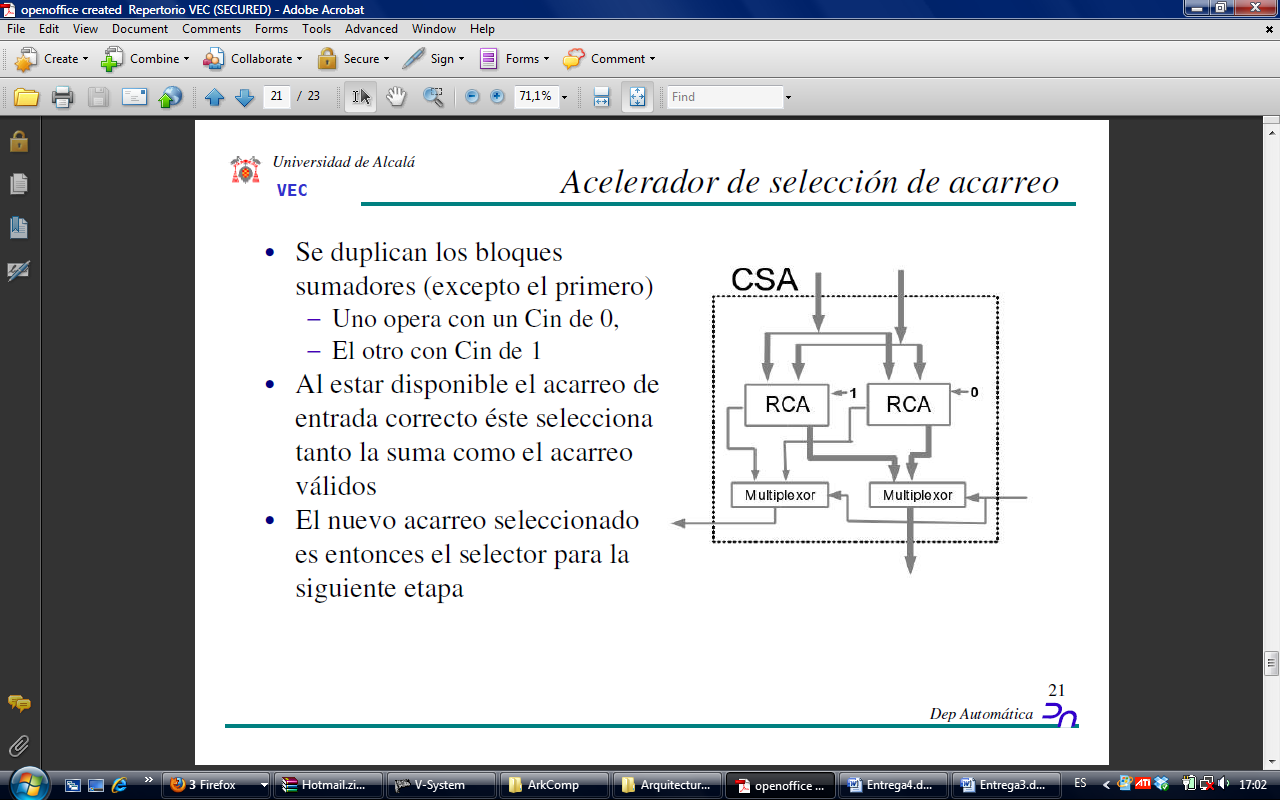
De esta forma independientemente de cómo sea el carry de entrada a la etapa, siempre podemos empezar a calcular salidas y acarreo de salida, cuando por fin nos llegue el verdadero carry de entrada a la etapa, sabremos cual de los 2 sumadores CRA tiene los resultador correctos.

- 2 multiplexores: -> 1º multiplexa las salidas de los 2 sumadores RCA, a través del

carry verdadero de entrada

-> 2º multiplexa los carris de salida de los 2 sumadores RCA, a

través del carry verdadero de entrada

  
# Así pues, de una forma cualitativa:

- Cada bloque CSA puede calcular sus resultados (suma y carry out) desde el momento en que sus sumandos (a y b) son estables, pero no los poenen a la salida hasta que no les llegue su verdadero carry de entrada, momento en el cual saben que grupo de resultados son correctos.

# Tiempos:

- Si tenemos un sumador CSA de "M" bits implementado en etapas CSA de "N" bits:

· Los sumadores RCA de cada etapa tardan **Retardo\_RCA\_Carry = 2T\*N + T** en estabilizarse

· Siendo su salida correcta y estable cuando lo sea el carry de entrada (y despues de pasar por el multiplexor) se tarda **t\_mux** en estabilizar tando el carry como la salida de cada etapa.

· Teniendo un sumador CSA de **k = M/N etapas,** se tarda **t\_mux** en propagar los carris por lo que el tiempo en el peor de los casos es:

**Retardo\_CSA\_Carry = Retardo\_CSA\_Suma = (2N + 1)T + k\*t\_mux**

**Funcionamiento del CLA**

# Un sumador CLA de "N" bits se basa en que podemos saber el acarreo de entrada de cada bit de la suma Ci directamente a través de las entradas a y b y el acarreo de entrada del sumador.

- El carry de entrada de uno de los N bits del sumador es:

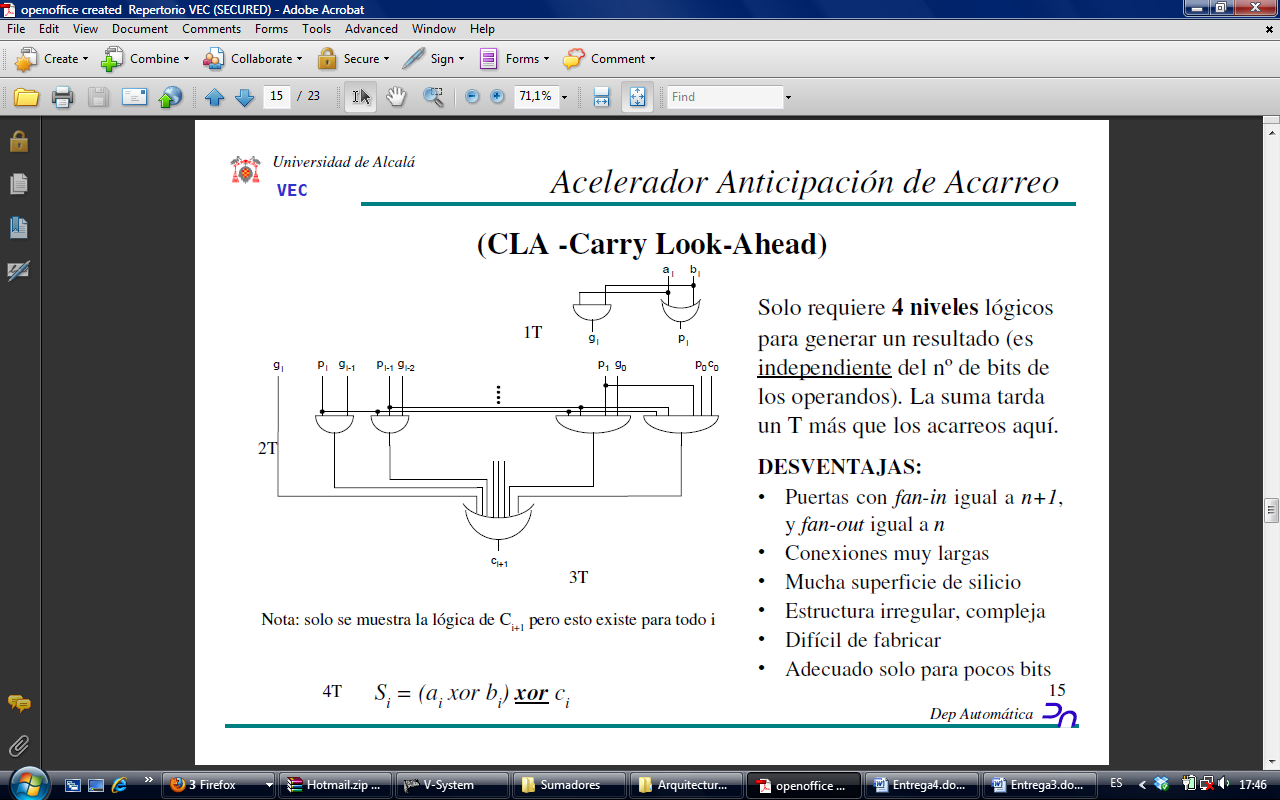
**Ci+1 = aibi + Ci (ai+ bi) = gi + Ci pi** Desarrollando:

**Ci+1 =**  + **+ + ···· +**

- La salida de cada uno de los N bits del sumador es:

**Si = (ai xor bi) xor ci**

- Los tiempos que tarda en estabilizarse la salida son:

 - Las **gi y pi** se generan con una sola puerta cada una, y todas en paralelo.

- Se tarda 3T en generarse el acarreo (Debido a las operaciones de **Ci+1**)

- La salida tardará un T más debido a la puerta "xor" entre **ci** y **(ai xor bi)**

(La salida de la puerta xor entre

ai y bi se genera en paralelo con ci)

- Podemos señalar que:

Un sumador CLA tarda un tiempo de puerta T en generar **gi y pi** desde que tenemos las entradas al sumador **a y b** estables.

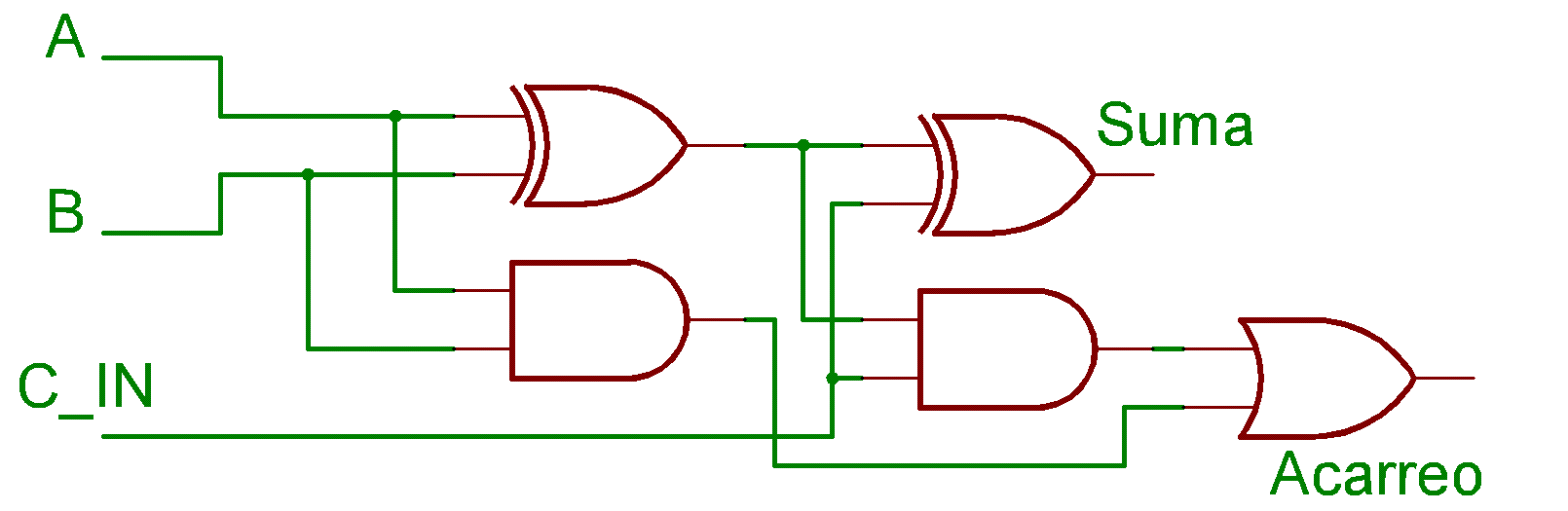
- En la situación en la que el acarreo de entrada se estabilice después, se tarda:

· 2T en generarse cada Ci (exepto **)**

· 3T en generarse la salida Si (exepto )

**Resumen contadores:**

# Dado que la mayoría usan el sumador elemental, aquí lo ponemos:



**\* Sumador RCA:**

- Retardo de propagación del Carry: **2T**

- Retardo de carry cuando cambia una entrada (A ó B): **3T**

- Retardo de suma debido a cambio en carry: **1T**

- Retardo de acarreo debido a cambio en carry: **2T** (Retardo de propagación)

- En el peor caso: **Retardo\_Acarreo = 2T\*(N-1) + 3T = 2T\*N + T**

**Retardo Suma = = 2T\*(N-2) + 3T + T = 2T\*N**

**-** Si lo que desencadena la suma es un cambio en el carry de entrada del bloque y no una entrada "a ó b", se tarda un T menos en generar todo.

**\* Sumador CSK:**

- Usa sumadores RCA -> Mismos retardos internos (sumas y carris internos)

- No puede empezar a calcular sus sumas y carris internos hasta que no tenga el carry de entrada ya generado y estable

- Genera el Carry de salida del bloque en **2T**si ya se estabilizaron los acarreos internos

- Debido a un cambio en el Carry de entrada la suma interna tarda:

**Retardo Suma = 2T\*(N-1) +T = (2N -1)T**

y el acarreo no adelantado tarda:

**Retardo Acarreo= 2T\*(N-1) +2T = 2N\*T**

**\* Sumador CSA:**

- Usa sumadores RCA -> Mismos retardos internos (sumas y carris internos)

- Genera sus sumas y carris desde el momento que las entradas (a y b) son estables.

- Cuando su carri de entrada es generado, si ya ha sean estabilizado los RCA que contiene, tarda **2T\_mux** en generar la suma y el acarreo (a la vez)

**\* Sumador CLA:**

- No usa sumadores RCA

- Genera las funciones gi y pi desde el momento en que las entradas a y b son estables

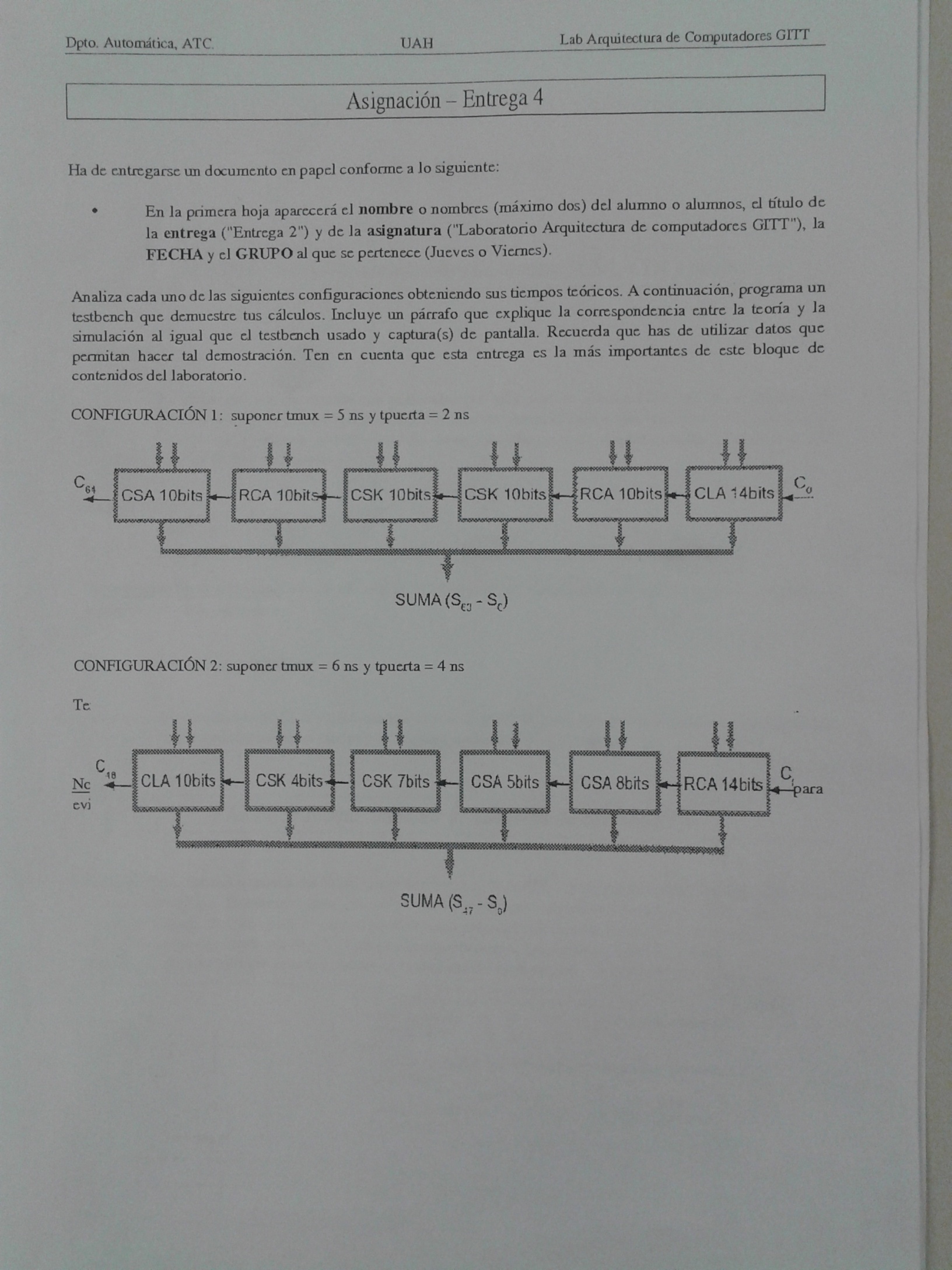
- Cuando el carry de entrada al bloque se hace estable tarda: **2T** para el acarreo

**3T** para la suma

- Retardos independientes del numero de bits del sumador

**Configuracion 1:**

# La configuración del primer sumador loco es:



# Cuando se estabilizan las entradas a y b:

- Los RCA calculan sus "ai xor bi"

- Los CSA empiezan a calcular las salidas de sus RCA

- Los CLA calculan sus funciones gi y pi

- Los CSK calculan su función P

# Propagación del acarreo:

- El bloque CLA tardará **3\*t\_puerta** en generar el acarreo

- El RCA lo propaga por cada sumador elemental hasta la salida: **2\*10\*t\_puerta**

- En este momento el primer CSK tarda **2\*t**\_**puerta** en generar su carry de salida y tarda

(2\*10 -1)\*t\_puertaen generar los valores de sus sumas

- Al llegar el carry de salida del primer CSK (tras 2\*t\_puerta), el segundo CSK hace el mismo trabajo, tardando lo mismo, tarda **2\*t**\_**puerta** en generar su carry de salida.

- Al llegar al RCA, este tarda  **2\*10\*t\_puerta** en generar el carry de salida y

**(2\*10 – 1)\*t\_puerta**  en generar las sumas internas.

- Cuando se estabiliza el carry de entrada del CSA sólo se tarda **t\_mux** en generar tanto el carry como la salida de la suma ya que sus RCA internos ya están calculados.

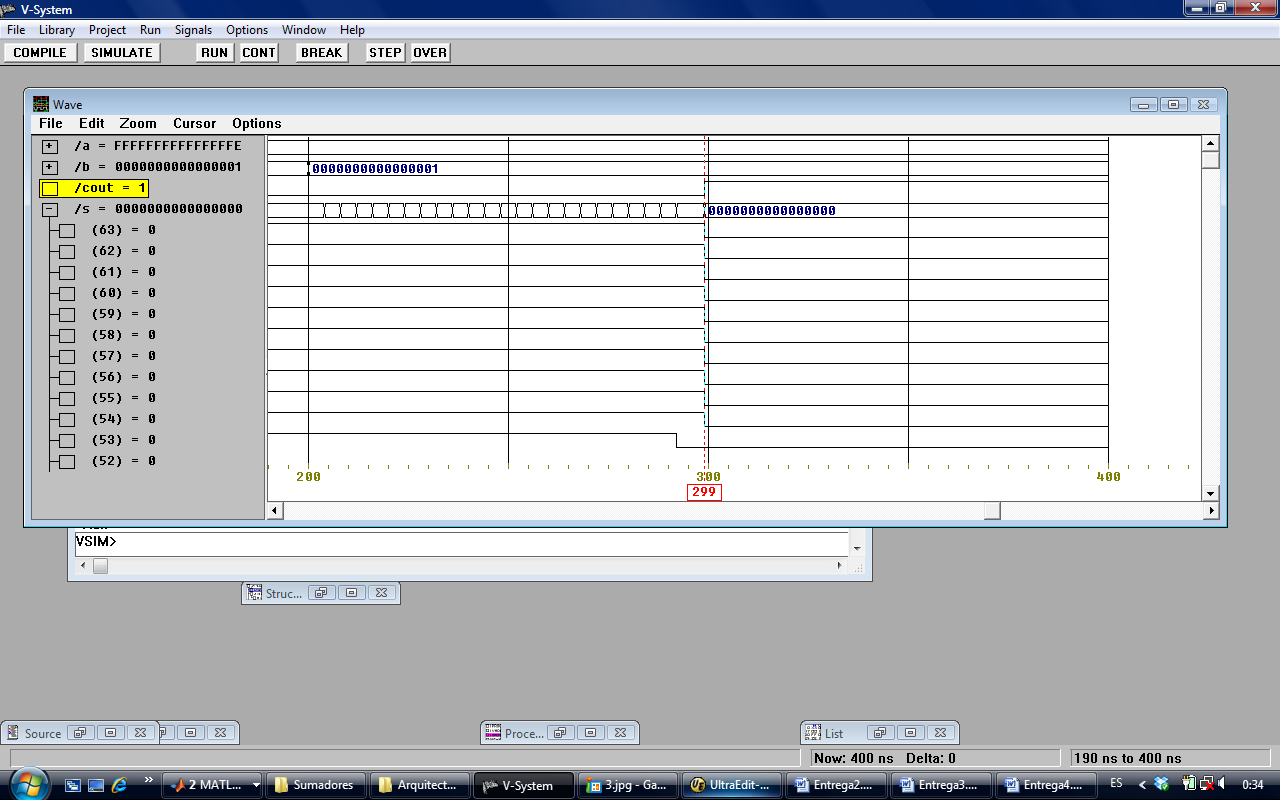
# Así pues quien más tarda en estabilizarse es tanto la suma como el acarreo del sumador CSA:

- Siendo t\_puerta = 2 ns y t\_mux = 5ns

**Retardo = (3 + 20 + 2 + 2 + 20)\*t\_puerta + t\_mux = 99 ns**

- Gráficamente tenemos que, teniendo el peor caso en los 200 ns:

Se puede ver como experimentalmente el retardo también es 99 ns



**#Test bench cuestión 1**

entity sumador\_completo\_64bits\_tb is

end;

architecture tst of sumador\_completo\_64bits\_tb is

signal a,b,s: bit\_vector(63 downto 0);

signal cin, cout : bit := '0';

signal ci:bit\_vector (5 downto 0) := "000000";

begin

ci(0) <= cin;

sumador1: entity work.sumador\_cla\_Nbits generic map (14,2 ns)

port map (a(13 downto 0),b(13 downto 0),ci(0),s(13 downto 0),ci(1));

sumador2: entity work.sumador\_rca\_Nbits generic map (10,2 ns)

port map (a(23 downto 14),b(23 downto 14),ci(1),s(23 downto 14),ci(2));

sumador3: entity work.sumador\_csk\_Nbits generic map (10,2 ns)

port map (a(33 downto 24),b(33 downto 24),ci(2),s(33 downto 24),ci(3));

sumador4: entity work.sumador\_csk\_Nbits generic map (10,2 ns)

port map (a(43 downto 34),b(43 downto 34),ci(3),s(43 downto 34),ci(4));

sumador5: entity work.sumador\_rca\_Nbits generic map (10,2 ns)

port map (a(53 downto 44),b(53 downto 44),ci(4),s(53 downto 44),ci(5));

sumador6: entity work.sumador\_csa\_Nbits generic map (10,2 ns,5 ns)

port map (a(63 downto 54),b(63 downto 54),ci(5),s(63 downto 54),cout);

process

begin

a<=x"FFFFFFFFFFFFFFFE"; b<=x"0000000000000000"; cin <='1';

wait for 200 ns;

b<=x"0000000000000001";

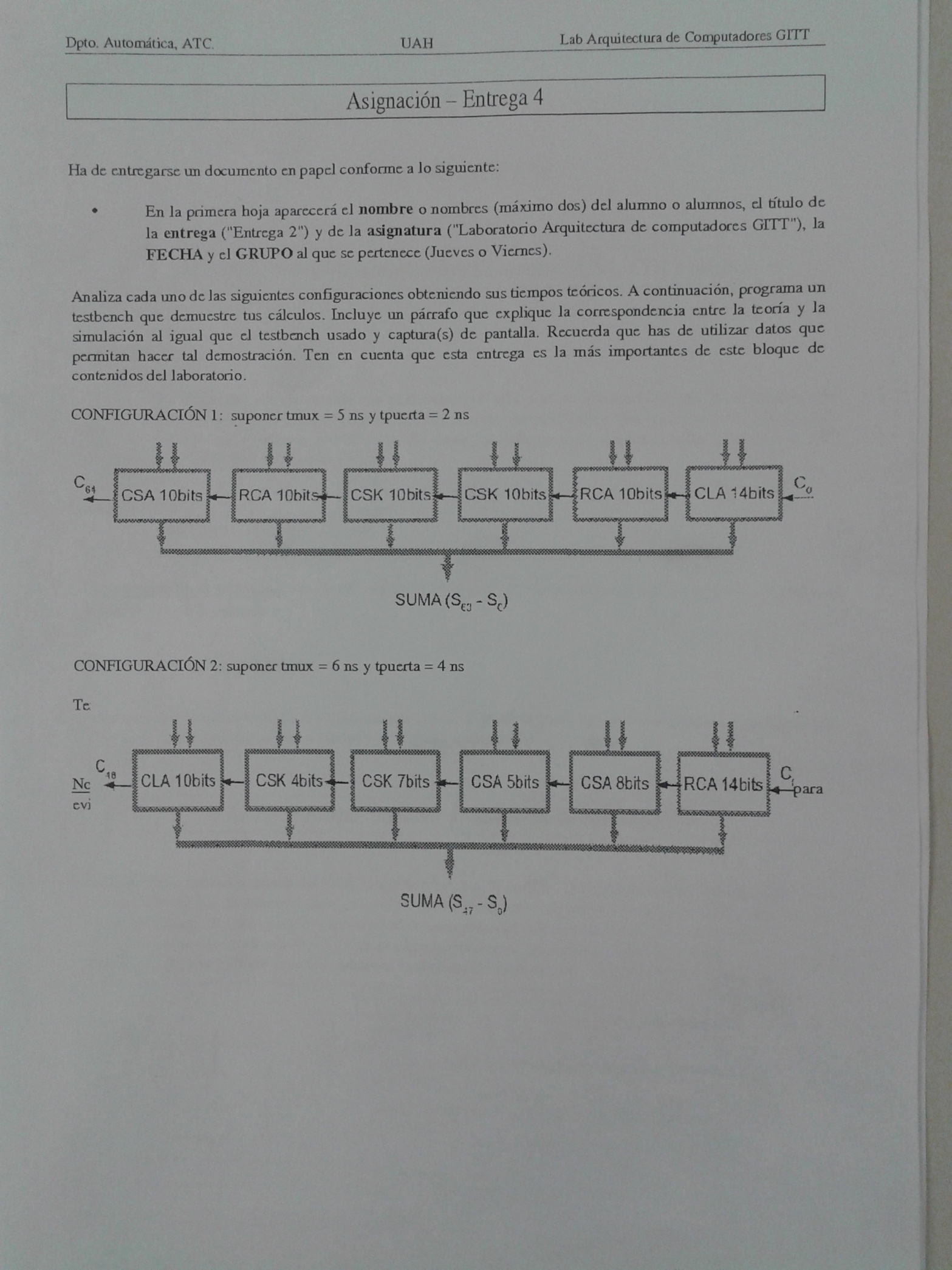
wait;

end process;

end;

**Configuración 2:**

# La configuración del primer sumador loco es:



# Cuando se estabilizan las entradas a y b:

- Los RCA calculan sus "ai xor bi"

- Los CSA empiezan a calcular las salidas de sus RCA

- Los CLA calculan sus funciones gi y pi

- Los CSK calculan su función P

# Propagación del acarreo:

- El bloque RCA en el peor de los casos tardará **(2\*14 + 1)\*t\_puerta**  en generar el acarreo acarreo de salida del bloque

- El primer CSA ya habrá calculado las salidas de sus RCA internos por lo que solo tarda **t\_mux** en propagar el acarreo y generar sus salidas

- El segundo CSA también habrá calculado las salidas de sus RCA internos por lo que solo tarda **t\_mux** en propagar el acarreo y generar sus salidas

- Al llegarle el acarreo al primer CSK, tarda 2\*t\_puertaen generar su carry de salida y tarda **(2\*7 -1)\*t\_puerta** en generar los valores de sus sumas

- Al llegar el carry de salida del primer CSK (tras 2\*t\_puerta), el segundo CSK hace el mismo trabajo, tardando lo mismo, tarda 2\*t\_puertaen generar su carry de salida.

- Por último al llegarle el carry al CLA, este tarda solo 3\*t\_puerta en generar sus salidas

# Así pues quien más tarda en estabilizarse es el último bit de suma (33) del primer CSK:

- Siendo t\_puerta = 4 ns y t\_mux = 6ns

**Retardo\_Primer \_CSK= (29 + 13)\*t\_puerta + (1 +1)\*t\_mux = 180 ns**

**-** Caculemos los tiempos del segundo CSK y el CLA para asegurarnos:

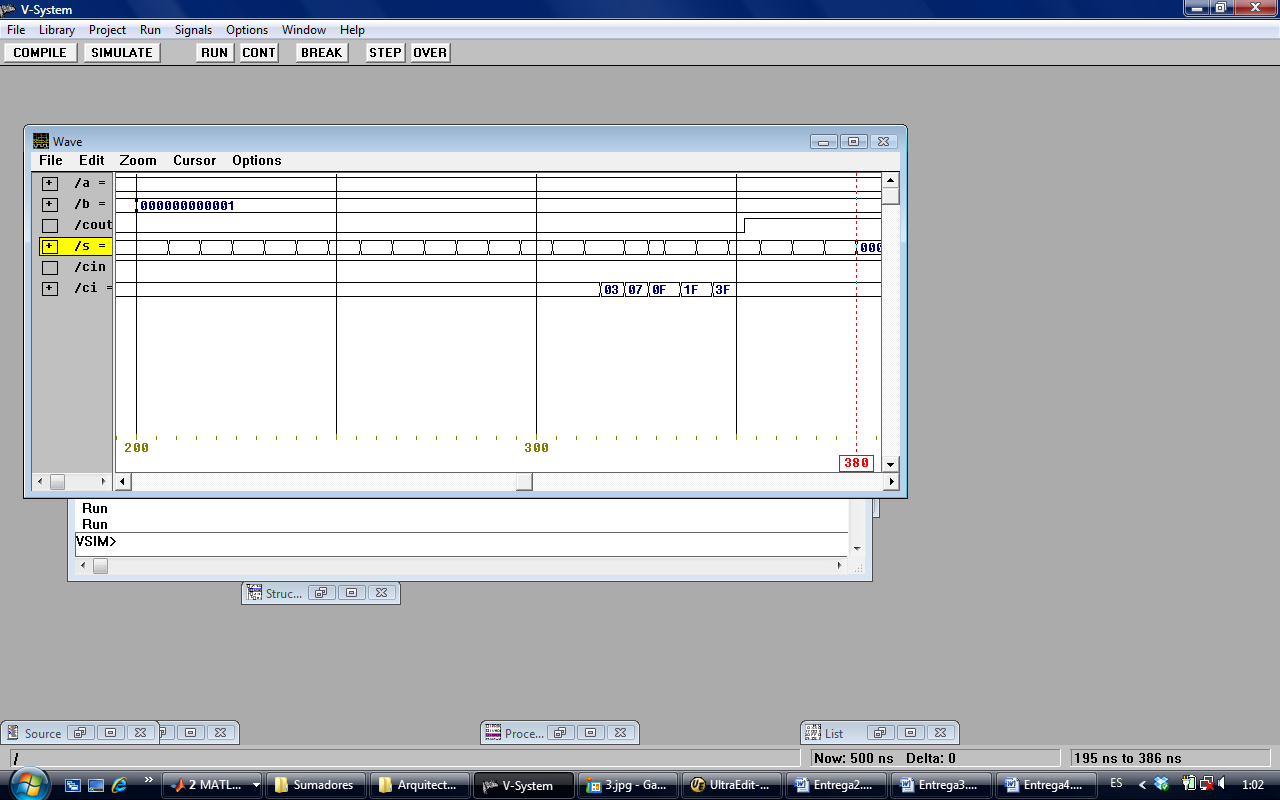
**Retardo\_Segundo\_CSK= (29 + 2 + 7)\*t\_puerta + (1 +1)\*t\_mux = 164 ns**

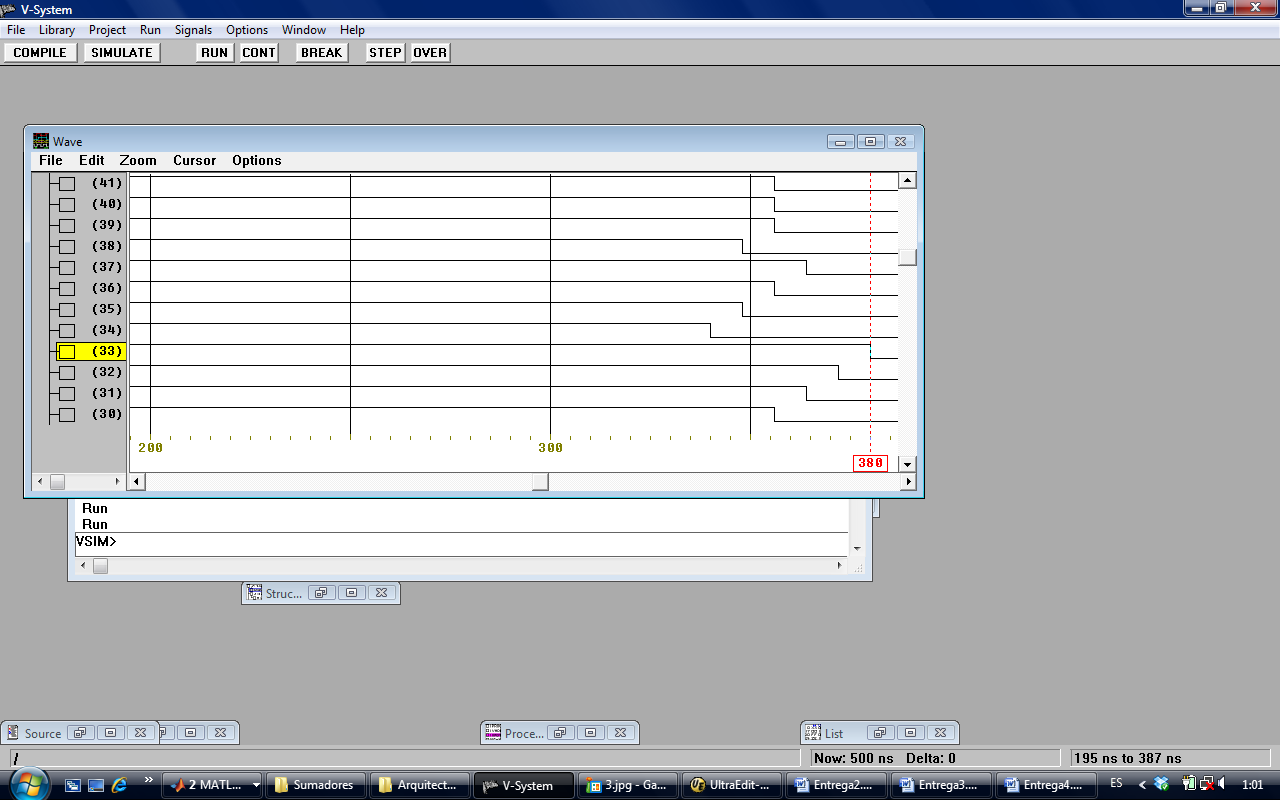
**Retardo\_CLA = (29 + 2 + 2 + 3)\*t\_puerta + (1 +1)\*t\_mux = 156 ns**

**-**Vemos que efectivamente, el peor de los casos es el bit 33 del primer CSK

- Gráficamente tenemos que, teniendo el peor caso en los 200 ns:

Se puede ver como experimentalmente el retardo también es 180 ns





**#Test bench cuestión 2**

entity sumador\_completo\_48bits\_tb is

end;

architecture tst of sumador\_completo\_48bits\_tb is

signal a,b,s: bit\_vector(47 downto 0);

signal cin, cout : bit := '0';

signal ci:bit\_vector (5 downto 0) := "000000";

begin

ci(0) <= cin;

sumador1: entity work.sumador\_rca\_Nbits generic map (14,4 ns)

port map (a(13 downto 0),b(13 downto 0),ci(0),s(13 downto 0),ci(1));

sumador2: entity work.sumador\_csa\_Nbits generic map (8,4 ns,6 ns)

port map (a(21 downto 14),b(21 downto 14),ci(1),s(21 downto 14),ci(2));

sumador3: entity work.sumador\_csa\_Nbits generic map (5,4 ns,6 ns)

port map (a(26 downto 22),b(26 downto 22),ci(2),s(26 downto 22),ci(3));

sumador4: entity work.sumador\_csk\_Nbits generic map (7,4 ns)

port map (a(33 downto 27),b(33 downto 27),ci(3),s(33 downto 27),ci(4));

sumador5: entity work.sumador\_csk\_Nbits generic map (4,4 ns)

port map (a(37 downto 34),b(37 downto 34),ci(4),s(37 downto 34),ci(5));

sumador6: entity work.sumador\_cla\_Nbits generic map (10,4 ns)

port map (a(47 downto 38),b(47 downto 38),ci(5),s(47 downto 38),cout);

process

begin

a<=x"FFFFFFFFFFFE"; b<=x"000000000000"; cin <='1';

wait for 200 ns;

b<=x"000000000001";

wait;

end process;

end;