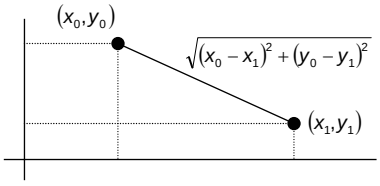
introducción de conceptos

Síntesis Arquitectónica y de Alto Nivel

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

especificación del sistema

- Diseñar un circuito que calcule la **raíz cuadrada aproximada** de un número que sea la suma del cuadrado de dos enteros (útil para calcular la distancia entre dos puntos dadas sus coordenadas cartesianas).



$$out = \sqrt{a^2+b^2}$$

$$out \approx max\left(0.875\cdot max\left(|a|,|b|\right)+0.5\cdot min\left(|a|,|b|\right), max\left(|a|,|b|\right)\right)$$

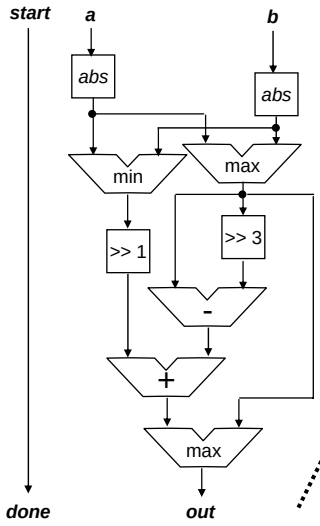
$0.5\cdot x = x >> 1$

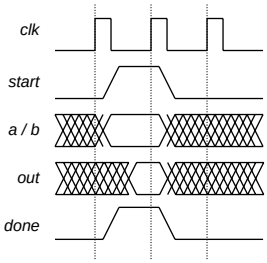
$0.875\cdot x = x - 0.125\cdot x = x - (x >> 3)$

$$out \approx max\left(\left(max\left(|a|,|b|\right)-\left(max\left(|a|,|b|\right)>> 3\right)\right)+min\left(|a|,|b|\right)>> 1, max\left(|a|,|b|\right)\right)$$

implementación monociclo

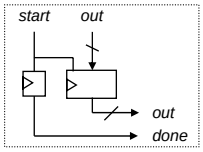
$$out \approx max\left(\left(max\left(|a|, |b| \right) - \left(max\left(|a|, |b| \right) >> 3 \right) \right) + min\left(|a|, |b| \right) >> 1, max\left(|a|, |b| \right) \right)$$

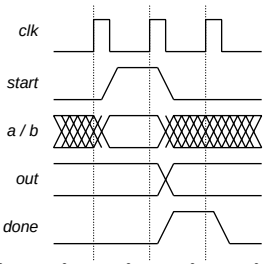




max, min = 45 ns;
+, -, abs = 55 ns; >> = 2 ns

t_{CLK} = 257 ns, f_{CLK} = 4 MHz
3.9 Mop/s

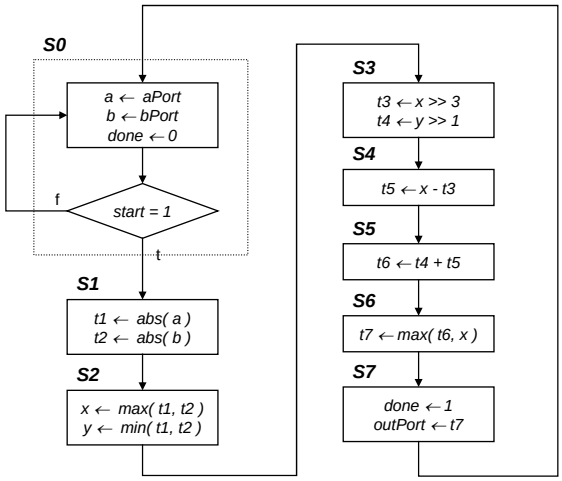


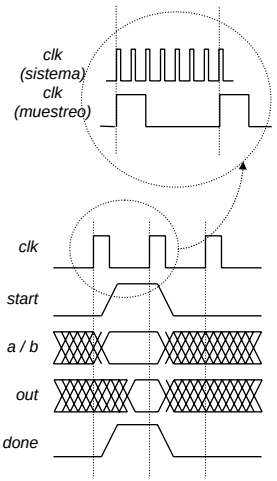


síntesis arquitectónica y de alto nivel

implementación multiciclo

$$out \approx max\left(\left(max\left(|a|, |b| \right) - \left(max\left(|a|, |b| \right) >> 3 \right) \right) + min\left(|a|, |b| \right) >> 1, max\left(|a|, |b| \right) \right)$$





síntesis arquitectónica y de alto nivel

reuso de recursos funcionales (i)

Primera fase: **Selección de unidades funcionales**

⌦ La **tabla de tipos de operación frente a estados** permite obtener el número de unidades funcionales mínimo necesario para implementar un diagrama ASM

- Suponiendo que operaciones de diferente tipo se realizan (en un único ciclo) en unidades funcionales de diferente tipo:

	S0	S1	S2	S3	S4	S5	S6	S7	
abs		2							2
max			1				1		1
min			1						1
>>				2					2
–					1				1
+						1			1
	0	2	2	2	1	1	1	0	8

número máx. de operaciones del mismo tipo que se realizan simultáneamente

si se usan 8 u.f., nunca está activo más del 25% del HW.

número mínimo de unidades funcionales mono-operacionales necesarias

número max. de operaciones que se realizan por ciclo

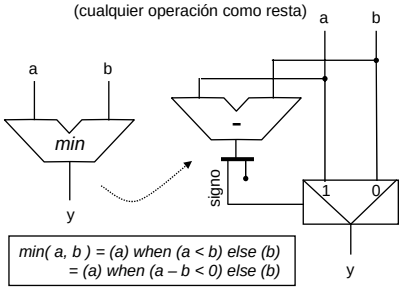
número máx. de operaciones que se realizan simultáneamente

- Sin embargo, esto no tiene por qué ser así, ya que:
  - ⇨ Unas operaciones pueden transformarse en otras.
  - ⇨ Pueden utilizarse unidades funcionales multi-operacionales con un coste menor que la suma de los costes de las unidades mono-operacionales correspondientes.

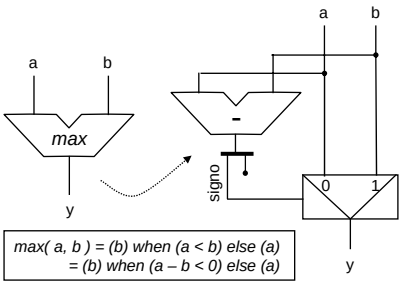
reuso de recursos funcionales (ii)

Transformación de operaciones

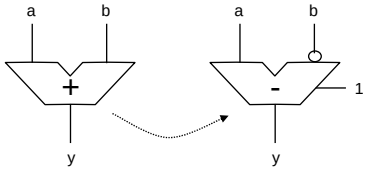
(cualquier operación como resta)



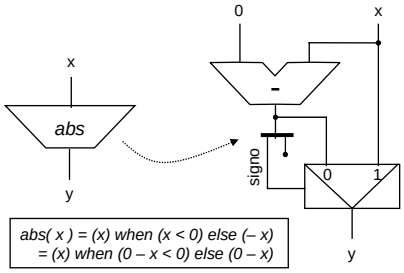
min(a, b) = (a) when (a < b) else (b)
= (a) when (a – b < 0) else (b)



max(a, b) = (b) when (a < b) else (a)
= (b) when (a – b < 0) else (a)



$a + b = a - (-b) =_{TC} a - not(b) - 1$



abs(x) = (x) when (x < 0) else (– x)
= (x) when (0 – x < 0) else (0 – x)

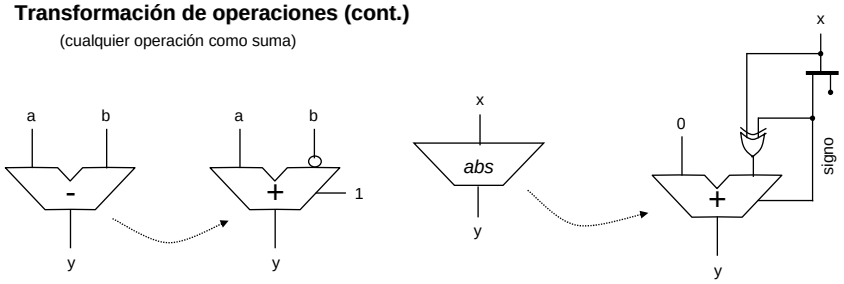
© J.M. Mendías, 2004

síntesis arquitectónica y de alto nivel

reuso de recursos funcionales (iii)

Transformación de operaciones (cont.)

(cualquier operación como suma)



$a - b = a + (-b) =_{TC} a + not(b) + 1$

abs(x) = (x) when (x < 0) else (– x)
= (0+x) when (x < 0) else (0+(– x))
$=_{TC}$ (0+x+0) when (x < 0) else (0+not(x)+1)
= (0+xor(x,0)+0) when (x < 0) else (0+xor(x,1)+1)
= 0 + xor(x, 0 when x<0 else 1) + (0 when x<0 else 1)
= 0 + xor(x, signo(x)) + signo(x)

⌦ **Conclusión**: las operaciones de este ejemplo o no requieren HW (desplazamientos) o el núcleo HW operativo que requieren es el mismo (un restador o un sumador), por lo que pueden considerarse equivalentes.

reuso de recursos funcionales (iv)

Primera fase: **Selección de unidades funcionales**

	S0	S1	S2	S3	S4	S5	S6	S7	
abs		2							2
max			1				1		1
min			1						1
>>				2					2
–					1				1
+						1			1
	0	2	2	2	1	1	1	0	8

ahora, aparte de cierta lógica, sólo se requieren 4 u.f.: 2 restadores y 2 desplazadores

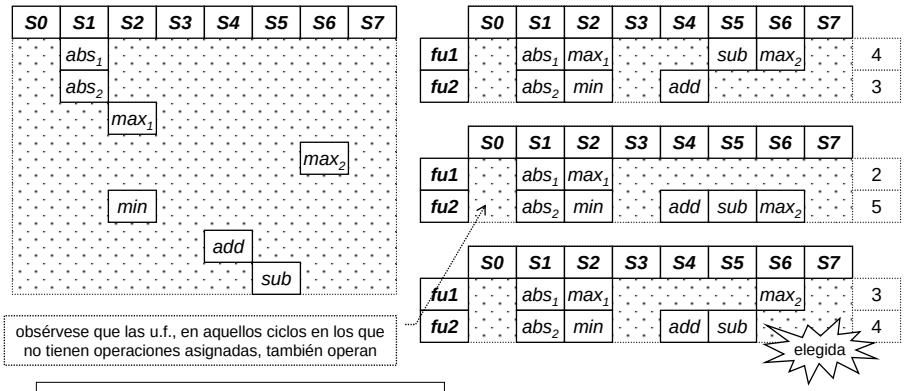
transformando funcionalidades

	S0	S1	S2	S3	S4	S5	S6	S7	
>>				2					2
–		2	2		1	1	1		2
	0	2	2	2	1	1	1	0	4

reuso de recursos funcionales (v)

Segunda fase: **Asignación de operaciones**

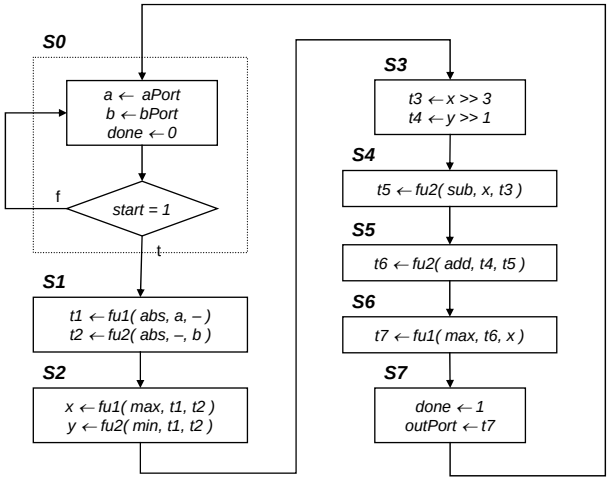
- Una vez decidido el tipo y el número mínimo de unidades funcionales, es necesario proyectar cada una de las operaciones sobre una instancia particular.
  - Para ello **para cada tipo de unidad funcional** se construye una **tabla de operaciones frente a ciclos**, y se intentan agrupar filas compatibles hasta que se alcance el número mínimo calculado anteriormente. Diferentes agrupamientos tendrán un coste diferente.



obsérvese que las u.f., en aquellos ciclos en los que no tienen operaciones asignadas, también operan

reuso de recursos funcionales (vi)

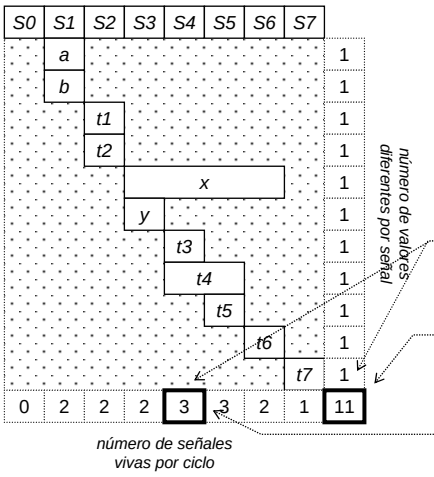
Tercera fase: **Reescritura del diagrama ASM**



reuso de recursos de almacenamiento (i)

Primera fase: **Selección de unidades de almacenamiento**

⌦ Las **tablas de vida de las señales** permiten obtener el número de unidades de almacenamiento mínimo necesario para implementar un diagrama ASM



Se dice que **una señal está viva**, desde el ciclo siguiente al ciclo en que se carga, hasta el ciclo en que por última vez es utilizada como operando.

La **vida de una señal**, se corresponde con el intervalo de tiempo durante el cual debe ser almacenada en un registro (y por tanto también puede ser leída desde él)

si se usan 11 reg., nunca está activo más del 27% del HW

número mínimo de unidades de almacenamiento requeridas según el diagrama ASM original

número máx de valores que deben permanecer almacenados simultáneamente

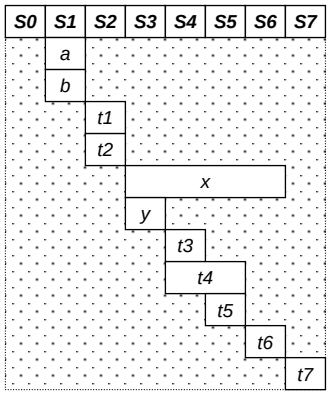
síntesis arquitectónica y de alto nivel

reuso de recursos de almacenamiento (ii)

Segunda fase: **Asignación de registros**

⌦ Una vez decidido el número mínimo de unidades de almacenamiento, es necesario proyectar cada una de las señales sobre una instancia particular.

- Para ello se construye una **tabla de señales frente a ciclos**, y se intentan agrupar filas compatibles hasta que se alcance el número mínimo calculado anteriormente. Diferentes agrupamientos tendrán un coste diferente.



obsérvese que en aquellos ciclos en los que los registros no tienen señales asignadas, siguen almacenando valores

	S0	S1	S2	S3	S4	S5	S6	S7	
r1		a	t1		t4		t6	t7	5
r2		b	t2	y	t3	t5			5
r3				x					1

	S0	S1	S2	S3	S4	S5	S6	S7	
r1		a	t1	x				t7	4
r2		b	t2	y	t3	t5	t6		6
r3					t4				1

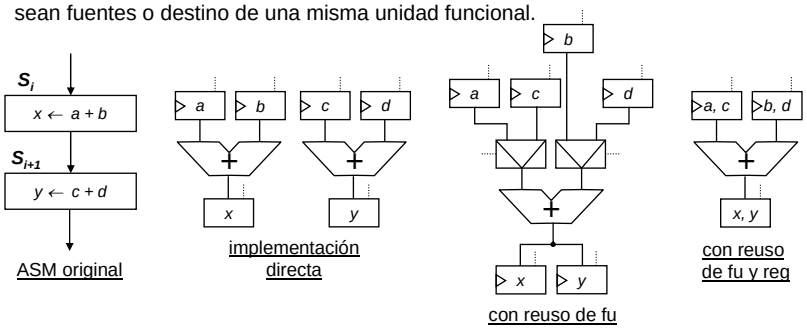
elegida

síntesis arquitectónica y de alto nivel

reuso de recursos de almacenamiento (iii)

Criterios de asignación

- A la hora de asignar registros es un buen criterio tratar de combinar señales que sean fuentes o destino de una misma unidad funcional.

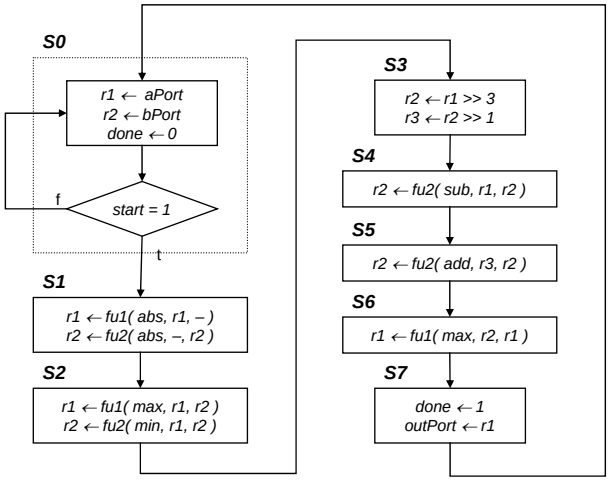


- Otros criterios son: equilibrar las señales almacenadas por registro, equilibrar el número de ciclos en los que los registros tienen valores válidos, etc.

- Criterios análogos son también aplicables durante la asignación de unidades funcionales. De hecho, las 2 tareas de asignación están íntimamente relacionadas.

reuso de recursos de almacenamiento (iv)

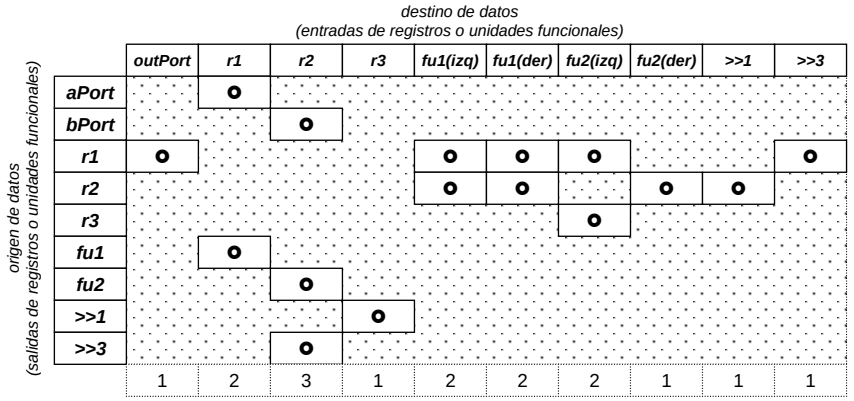
Tercera fase: **Reescritura del diagrama ASM**



diseño del interconexionado (i)

Primera fase: **Selección de interconexiones**

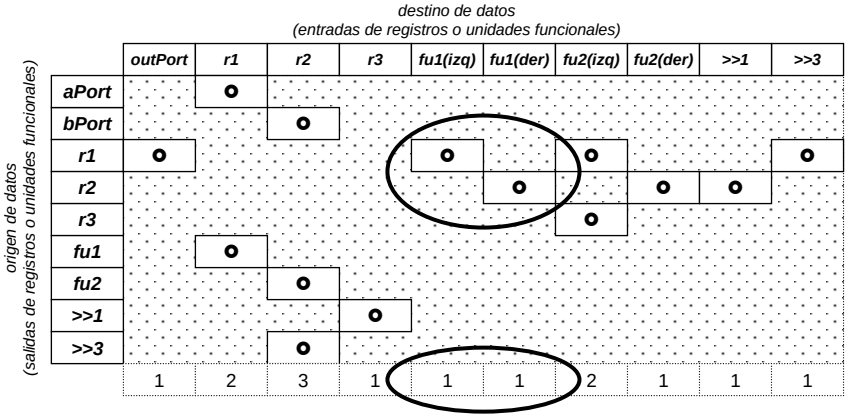
- Como paso inicial debe construirse una **tabla de interconexiones** (*origen de datos* frente a *destino de datos*) entre las distintas clases de recursos.
  - Cuando un *destino de datos* tenga más de un origen asociado será necesario utilizar algún mecanismo de multiplexación (o un multiplexor o lógica triestado).



diseño del interconexionado (ii)

Primera fase: **Selección de interconexiones (cont.)**

- Para reducir el número global de interconexiones, se pueden reorganizar los argumentos de las u.f. aplicando la conmutatividad cuando sea posible:
  - *r1 ← fu1(max, r2, r1)* pasa a ser *r1 ← fu1(max, r1, r2)*



diseño del interconexionado (iii)

Primera fase: **Selección de interconexiones (cont.)**

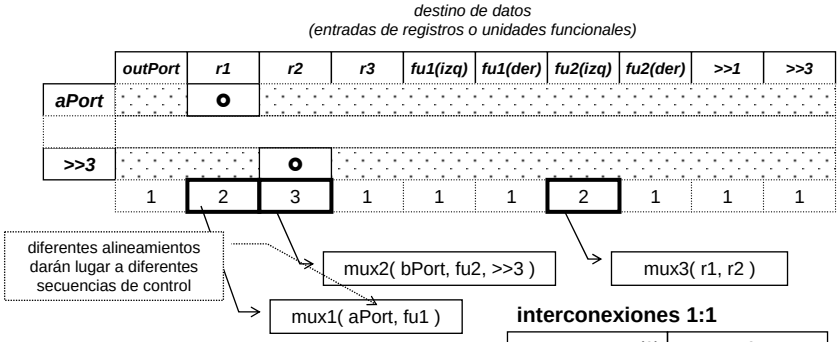
- En un sistema podemos encontrar 4 tipos de interconexiones 1:1, 1:*n*, *m*:1 y *m*:*n*.
- Las interconexiones punto a punto (1:1) y multipunto (1:*n*), no requieren un tratamiento especial. Sin embargo, según se traten las de tipo *m*:1 y *m*:*n*, podemos diseñar circuitos según 3 modelos de interconexionado diferentes:
  - **Modelo basado en multiplexores**:
    - Las interconexiones de tipo *m*:1 se implementan conectando un multiplexor *m* a 1 a la entrada del destino.
    - Las interconexiones de tipo *m*:*n* se descomponen en n interconexiones de tipo *m*:1.
  - **Modelo basado en buses**:
    - Tanto las interconexiones de tipo *m*:1, como las *m*:*n* se implementan mediante un bus que conecta todos los destinos y todas las fuentes. Los destinos se conectan al bus directamente y las fuentes a través de un buffer triestado.
  - **Modelo mixto**:
    - Algunas interconexiones se implementan con multiplexores y otras como buses.

- Según el modelo que se elija para el interconexionado el proceso de diseño es diferente.

interconexionado con multiplexores (i)

Segunda fase: **Asignación de multiplexores**

- De la tabla de interconexiones se extrae el número y tipo de mux. necesarios.



interconexiones 1:n

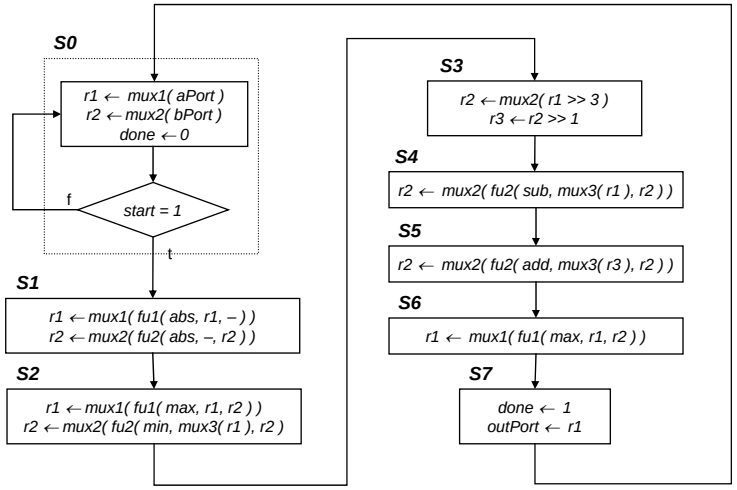
r1 → { outPort, fu1(izq), mux3(0), >>3 }
r2 → { fu1(der), fu2(der), >>1 }

interconexiones 1:1

aPort → mux1(0)	*>>1 → r3*
bPort → mux2(0)	*>>3 → mux2(2)*
r3 → mux3(1)	*mux1 → r1*
fu1 → r1	*mux2 → r2*
fu2 → mux2(1)	*mux3 → fu2(izq)*

interconexionado con multiplexores (ii)

Tercera fase: **Reescritura del diagrama ASM**



interconexionado con buses (i)

Segunda fase: **Selección de buses**

- Las **tablas de transferencias por ciclos** permiten obtener el número de buses mínimo necesario para implementar un diagrama ASM.
  - Recuérdese que un bus es una conexión *n* a *m* cuyo uso se multiplexa en el tiempo de manera que en todo ciclo se comporte como una conexión 1 a *m*.

S0	S1	S2	S3	S4	S5	S6	S7
aPort → r1							
bPort → r2							
	r1 → fu1(izq)	r1 → fu1(izq) r1 → fu2(izq)	r1 → >>3	r1 → fu2(izq)		r1 → fu1(izq)	r1 → outPort
	r2 → fu2(der)	r2 → fu1(der) r2 → fu2(der)	r2 → >>1	r2 → fu2(der)	r2 → fu2(der)	r2 → fu1(der)	
					r3 → fu2(izq)		
	fu1 → r1	fu1 → r1				fu1 → r1	
	fu2 → r2	fu2 → r2		fu2 → r2	fu2 → r2		
			>>1 → r3				
			>>3 → r2				
2	4	4	4	3	3	3	1

número máx de transferencias simultáneas de diferentes fuentes

número max. de transferencias que se realizan por ciclo procedentes de fuentes diferentes

lo más común es agrupar transferencias que proceden de la misma fuente, las interconexiones 1:*n* no añaden coste

interconexionado con buses (ii)

Tercera fase: **Asignación de buses**

- Una vez decidido el número mínimo de buses, es necesario proyectar cada una de las transferencias sobre una instancia particular.
  - Posibles criterios: homogeneizar tipo de recursos conectados, maximizar buses tipo 1:*n*

	S0	S1	S2	S3	S4	S5	S6	S7
bus1		r1 → fu1(izq)	r1 → fu1(izq) r1 → fu2(izq)	r1 → >>3	r1 → fu2(izq)		r1 → fu1(izq)	r1 → outPort
bus2		r2 → fu2(der)	r2 → fu1(der) r2 → fu2(der)	r2 → >>1	r2 → fu2(der)	r2 → fu2(der)	r2 → fu1(der)	
bus3	aPort → r1	fu1 → r1	fu1 → r1	>>1 → r3		r3 → fu2(izq)	fu1 → r1	
bus4	bPort → r2	fu2 → r2	fu2 → r2	>>3 → r2	fu2 → r2	fu2 → r2		

elegida

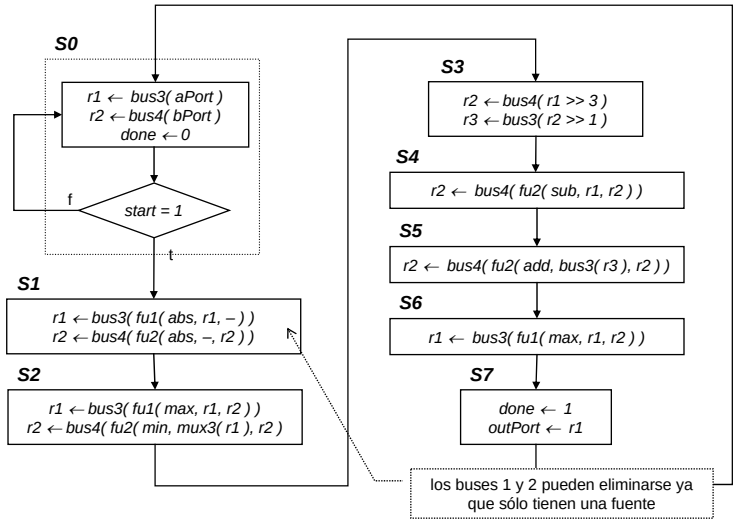
De esta tabla es fácil extraer los elementos interconectados por el bus (para construir el datapath) , y cúal de las fuentes vuelca información sobre él en cada ciclo (para construir el controlador)

		S0	S1	S2	S3	S4	S5	S6	S7
r1 → { outPort, fu1(izq), fu2(izq), >>3 }	**bus1 (1:n)**		r1					r1	
r2 → { fu1(der), fu2(der), >>1 }	**bus2 (1:n)**		r2						
{ aPort, r3, fu1, >>1 } → { r1, r3 }	**bus3 (m:n)**	aPort	fu1		>>1		r3	fu1	
{ bPort, fu2, >>3 } → r2	**bus4 (m:1)**	bPort	fu2		>>3	fu2			

síntesis arquitectónica y de alto nivel

interconexionado con buses (iii)

Cuarta fase: **Reescritura del diagrama ASM**



los buses 1 y 2 pueden eliminarse ya que sólo tienen una fuente

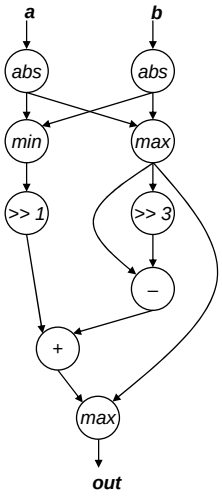
síntesis arquitectónica y de alto nivel

otras alternativas de optimización

- Hasta el momento hemos optimizado diagramas ASM realizando algunas **suposiciones** que han simplificado en exceso el problema:
  - **Que el ciclo de ejecución de las operaciones era inamovible**.
    - ⇨ Planificación de operaciones.
  - **Que toda operación tarda un ciclo en ejecutarse**.
    - ⇨ Encadenamiento.
    - ⇨ Multiciclo.
  - **Que se deben usar únicamente registros para almacenar señales**.
    - ⇨ Síntesis con bancos de registros.
    - ⇨ Síntesis con memorias RAM.
    - ⇨ Otros elementos de almacenamiento: LIFOs, FIFOs, contadores, etc.
  - **Que las unidades funcionales son combinacionales y simples**.
    - ⇨ Síntesis con unidades segmentadas.
    - ⇨ Otras unidades funcionales: multiplicadores-acumuladores, multiplicadores-sumadores, u.f. vectoriales, contadores, etc.

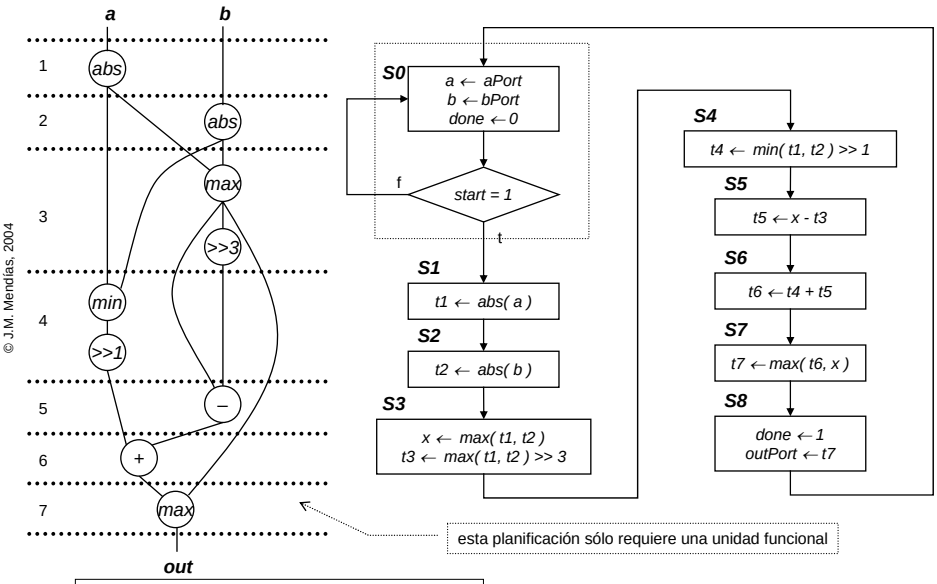
planificación de operaciones (i)

- Hasta el momento hemos sintetizado partiendo de diagramas ASM en donde las operaciones tenían un ciclo de ejecución asignado.
  - El ciclo en que se realiza una operación influencia enormemente los reusos posteriores.
  - Sin embargo, dicha planificación fue arbitraria y pueden existir otras mejores.
- **Planificación**: decide el ciclo en que se ejecuta cada una de las operaciones que componen una computación.
  - Toda planificación debe **respetar las dependencias de datos**, es decir, toda operación debe planificarse en un ciclo igual o posterior al ciclo en que se planifican las operaciones que calculan sus argumentos.
  - Las tareas fundamentales para realizar una planificación son:
    - ⇨ Construir un **diagrama de flujo** de la computación.
    - ⇨ Decidir el **número de ciclos** en que se ejecutará la computación.
    - ⇨ Asignar a cada operación un **ciclo de ejecución** según algún criterio de coste.



$$out \approx max\left(\left(max\left(|a|,|b|\right)-\left(max\left(|a|,|b|\right)>>3\right)\right)+min\left(|a|,|b|\right)>>1, max\left(|a|,|b|\right)\right)$$

planificación de operaciones (ii)



esta planificación sólo requiere una unidad funcional

unidades funcionales encadenadas y multiciclo (i)

- Hasta el momento hemos considerado que **toda operación tarda un ciclo** en realizarse, por lo que el ciclo de reloj debe ser mayor que el retardo de la transferencia entre registros más lenta.
- Calculemos el ciclo de reloj y el rendimiento del circuito, considerando:
  - Unidades funcionales: fu1 = 45 ns, fu2 = 55 ns, >> = 2 ns
  - Registros: t_{hold}, t_{setup}, t_{CLKQ} = 0 ns
  - Multiplexores: mux1, mux3 = 7 ns, mux2 = 10 ns
  - Retardo del interconexionado: 0 ns

t_{CLK} = 72 ns, f_{CLK} = 14 MHz
72 × 8 = 576 ns, 1.7 Mop/s

transferencias entre registros	*retardo de la transferencia*	*tiempo desperdiciado*
S0: *r1 ← mux1(aPort)*	**7 ns**	72 -10 = **62 ns**
r2 ← mux2(bPort)	**10 ns**	
done ← 1	**0 ns**	
S1: *r1 ← mux1(fu1(abs, r1, –))*	45 + 7 = **52 ns**	72 - 65 = **7 ns**
r2 ← mux2(fu2(abs, –, r2))	55 + 10 = **65 ns**	
S2: *r1 ← mux1(fu1(max, r1, r2))*	45 + 7 = **52 ns**	72 - 72 = **0 ns**
r2 ← mux2(fu2(min, mux3(r1), r2)	55 + 10 + 7 = **72 ns**	
S3: *r2 ← mux2(r1 >> 3)*	2 + 10 = **12 ns**	72 - 12 = **60 ns**
r3 ← r2 >> 1	**2 ns**	
S4: *r2 ← mux2(fu2(sub, mux3(r1), r2))*	55 + 10 + 7 = **72 ns**	72 - 72 = **0 ns**
S5: *r2 ← mux2(fu2(add, mux3(r3), r2))*	55 + 10 + 7 = **72 ns**	72 - 72 = **0 ns**
S6: *r1 ← mux1(fu1(max, r2, r1))*	45 + 7 = **52 ns**	72 - 52 = **20 ns**
S7: *done ← 1*	**0 ns**	72 - 0 = **72 ns**
outPort ← r1	**0 ns**	
	max = 72 ns	**total = 221 ns** = 38 %

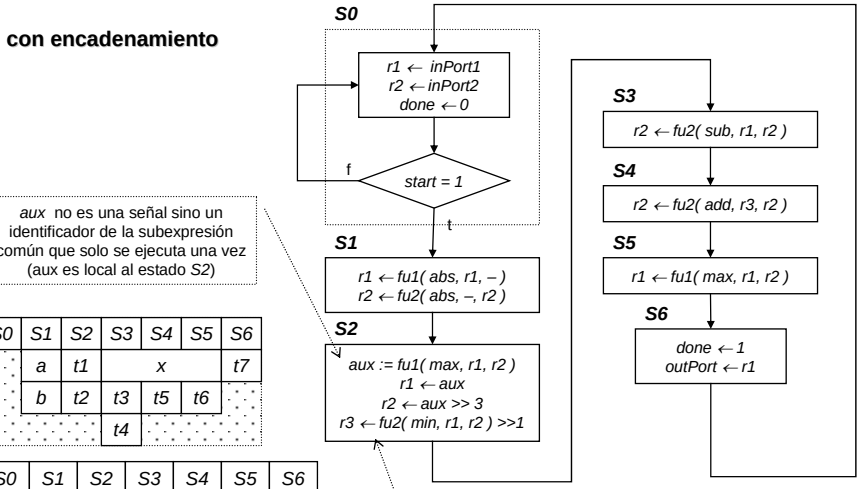
el desequilibrio entre tiempo de ciclo y el retardo de las transferencias provoca desperdicio de tiempo

unidades funcionales encadenadas y multiciclo (ii)

- **Objetivo**: reducir el tiempo desperdiciado en cada ciclo, ajustando el tiempo de ciclo y el retardo medio de las transferencias entre registros.
- **Encadenamiento**: consiste en agrupar la ejecución de varias operaciones con dependencias de datos en un mismo ciclo para aprovechar al máximo el tiempo disponible. Al encadenar operaciones:
  - El número de ciclos requeridos para ejecutar la computación puede disminuir.
  - Pueden desaparecen algunos registros intermedios.
  - Al concentrarse mayor número de operaciones por ciclo, la reusabilidad del HW puede disminuir.
- **Multiciclo**: consiste en permitir que las operaciones puedan tardar en ejecutarse más de un ciclo, por lo que el tiempo de ciclo no está limitado por la transferencia entre registros más lenta. Al permitir multiciclo:
  - El número de ciclo requeridos para ejecutar la computación puede aumentar.
  - La vida de las variables puede aumentar (durante la ejecución de una operación multiciclo los registros fuente deben permanecer estables).
  - Al dilatarse por varios ciclos la ejecución de una operación, la reusabilidad del HW puede disminuir.

unidades funcionales encadenadas y multiciclo (iii)

con encadenamiento



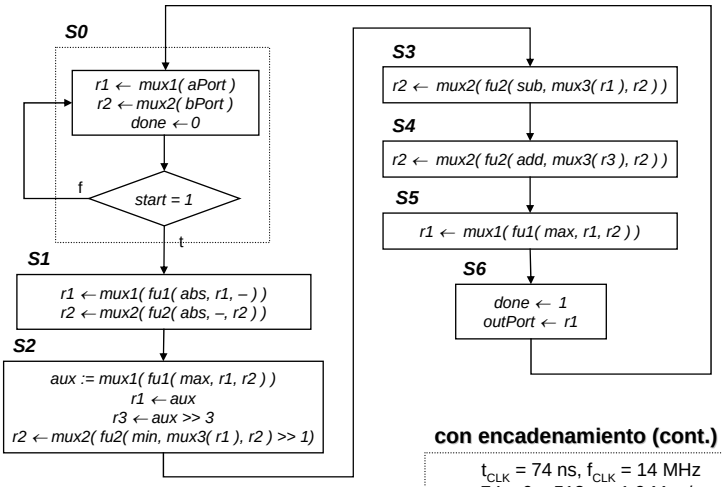
aux no es una señal sino un identificador de la subexpresión común que solo se ejecuta una vez (aux es local al estado *S2*)

	S0	S1	S2	S3	S4	S5	S6
r1		a	t1	x			t7
r2		b	t2	t3	t5	t6	
r3				t4			

	S0	S1	S2	S3	S4	S5	S6
fu1		abs_1	max_1				max_2
fu2		abs_2	min	add	sub		

la operación de desplazamiento tiene un tiempo de cálculo muy pequeño, por lo que puede encadenarse sin aumentar excesivamente el tiempo de ciclo

unidades funcionales encadenadas y multiciclo (iv)



con encadenamiento (cont.)

t_{CLK} = 74 ns, f_{CLK} = 14 MHz
74 × 6 = 518 ns, 1.9 Mop/s
tiempo desperdiciado = 173 ns = 33%

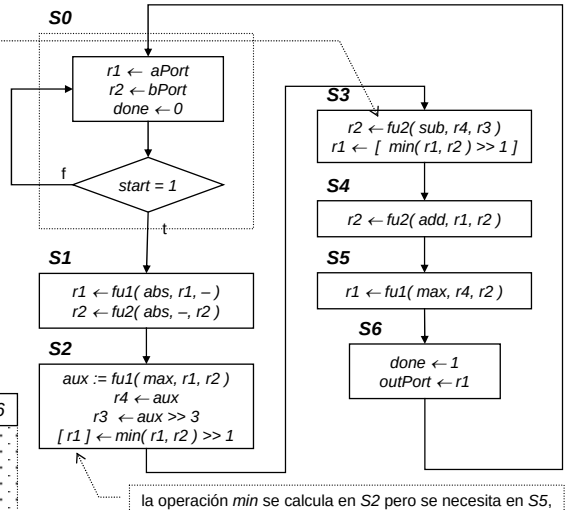
unidades funcionales encadenadas y multiciclo (v)

con multiciclo (cont.)

mediante corchetes se indica una transferencia que se inicia en un ciclo y finaliza en otro: la unidad funcional *min* está ocupada durante los ciclos *S2* y *S3*, los registros *t1* y *t2* deben estar estables durante los ciclos *S2* y *S3*; *t4*, en cambio, se asigna en *S3* y no en *S2*.

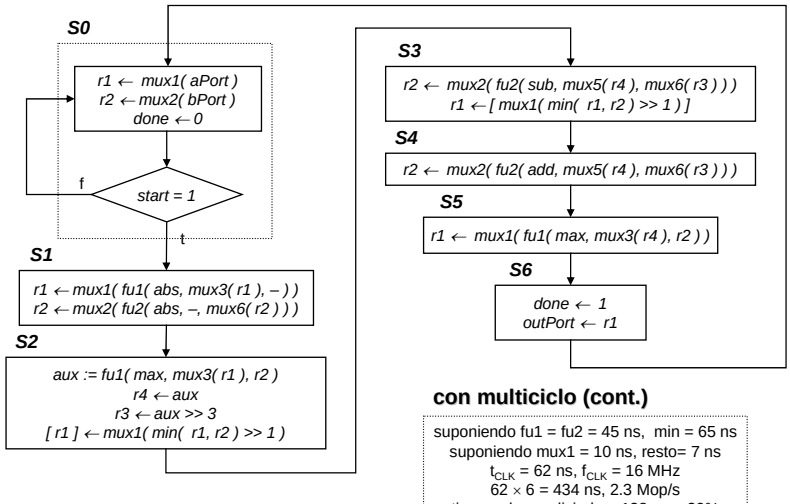
	S0	S1	S2	S3	S4	S5	S6
r1		a	t1		t4		t7
r2		b	t2		t5	t6	
r3				t3			
r3				x			

	S0	S1	S2	S3	S4	S5	S6
fu1		abs_1	max_1			max_2	
fu2		abs_2		add	sub		
min			min				



la operación *min* se calcula en *S2* pero se necesita en *S5*, luego es posible hacerla en una unidad funcional dedicada más lenta simplificando a su vez el diseño de *fu2*

unidades funcionales encadenadas y multiciclo (vi)



con multiciclo (cont.)

suponiendo fu1 = fu2 = 45 ns, min = 65 ns
suponiendo mux1 = 10 ns, resto= 7 ns
t_{CLK} = 62 ns, f_{CLK} = 16 MHz
62 × 6 = 434 ns, 2.3 Mop/s
tiempo desperdiciado = 128 ns = 29%

diseño con bancos de registros

- Es posible agrupar los registros en bancos de manera que compartan sus puertos con el exterior y se reduzca el interconexionado global del circuito.
- **Primera fase**: Selección de bancos mediante el análisis de la **tabla de accesos**

Son necesarios 4 accesos simultáneos: 2 de lectura y 2 de escritura. Solución 1 banco de 4 puertos o 2 bancos de 2 puertos.

	S0	S1	S2	S3	S4	S5	S6	S7	
r1	W	R/W	R/W	R	R		R/W	R	
r2	W	R/W	R/W	R/W	R/W	R/W	R		
r3				W		R			
	0	2	2	2	2	2	2	1	lecturas
	2	2	2	2	1	1	1	0	escrituras

- **Segunda fase**: Asignación de bancos
  - Si las señales con vidas no superpuestas pueden compartir registro, los registros con accesos no superpuestos pueden compartir el mismo banco.
  - Conociendo el número de puertos de R/W que tengan los bancos, los registros se agrupan de manera que en todo ciclo no se supere el número de lecturas o escrituras establecido.

para un banco con 1 puerto de lectura y 1 de escritura

	S0	S1	S2	S3	S4	S5	S6	S7
{ r1, r3 }	W	R/W	R/W	R/W	R	R	R/W	R
r2	W	R/W	R/W	R/W	R/W	R/W	R	

diseño con memorias RAM (i)

- Cuando existen **muchas señales vivas simultáneamente**, es conveniente utilizar módulos de memoria en lugar de registros.
- Las diferencias son:
  - Mientras los registros tienen un puerto de entrada y otro de salida (por lo que se puede leer y escribir en el mismo ciclo), las memorias tienen típicamente uno o varios puertos bidireccionales (por lo que las lecturas y escrituras deben de hacerse en ciclos diferentes si se hacen a través del mismo puerto).
  - Mientras los registros almacenan un único valor, las memorias almacenan varios.
  - Otras de índole tecnológico (señales de control, temporización, etc).
- Cuando se diseña con módulos de memoria se necesita concretar:
  - El **número de módulos de memoria** requeridos (función de las características de la memoria: velocidad, número de puertos, etc. y de los accesos a señales).
  - El **número de celdas de almacenamiento** requeridas por módulo (función del número máximo de señales vivas simultáneamente).
  - La asignación de **señales a módulos**.
  - La asignación de señales a celdas de almacenamiento, o lo que es lo mismo, la dirección que ocupará cada señal (para reducir la complejidad de la decodificación).

diseño con memorias RAM (ii)

- La implementación con RAM no es adecuada en este caso.

	S0	S1	S2	S3	S4	S5	S6	S7
a	W	R						
b	W	R						
t1		W	R					
t2		W	R					
x			W	R	R	X	R	
y			W	R				
t3				W	R			
t4				W	X	R		
t5					W	R		
t6						W	R	
t7							W	R
	0	2	2	2	3	3	2	1
	2	2	2	2	1	1	1	0
	0	2	2	2	2	2	2	1

vivas
escrituras
lecturas

Son necesarios 4 accesos simultáneos. Soluciones desde 4 RAM monopuerto hasta 1 RAM con 4 puertos.

Sin embargo bastan 3 celdas de almacenamiento

diseño con u.f. segmentadas (i)

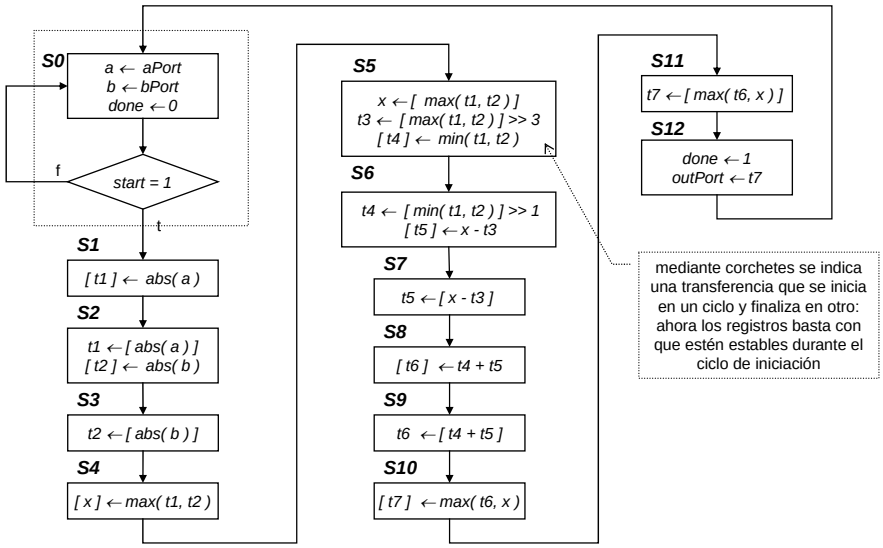
- Un método para mejorar el rendimiento de un circuito es el uso de u.f segmentadas. Para ello es necesario:
  - Construir la **tabla de reserva** de la u.f. segmentada
  - **Planificar el ASM** de manera que se inicie la ejecución de operaciones sobre la unidad segmentada en concordancia con la tabla de reserva.
- El caso más sencillo (y más común) es segmentar linealmente aquellas u.f. más lentas en un número de etapas que se ajuste a la frecuencia de reloj deseada.

- Supongamos que diseñamos una u.f. capaz de ejecutar todas las operaciones del SRA con un tiempo de cálculo de 64 ns.
- Si decidimos segmentarla linealmente en 2 etapas, una posible planificación sería:

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
etapa1		abs_1	abs_2		max_1	min	sub		add		max_2		
etapa2			abs_1	abs_2		max_1	min	sub		add		max_2	

obsérvese que las paradas se deben a dependencias de datos

diseño con u.f. segmentadas (ii)



definición de SAAN

Síntesis de alto nivel, algorítmica o arquitectónica

- **Entrada**: una descripción conductual de nivel algorítmico de una computación y una colección de ligaduras.
  - **Ligaduras típicas**: latencia, tiempo de ciclo, número y tipo de recursos, cadencia de admisión de datos.
- **Salida**: una ruta de datos (netlist de nivel RT) y una descripción conductual de nivel RT del controlador (secuencia de transferencias entre registros).
- **Tareas**:
  - **Planificación de operaciones (scheduling)**: fija el ciclo en que se ejecuta cada una de las operaciones que forman la computación.
  - **Selección de recursos (resource binding)**: determina el número y tipo de recursos funcionales (FUs, regs, mux/buses) requeridos para realizar la computación
  - **Asignación de recursos (resource allocation)**: fija el recurso en el que se ejecuta, almacena o transmite cada una de las operaciones y objetos de datos que la forman.
  - **Otras tareas**: transformación de operaciones, transformación de bucles, segmentación
- Cuando se parte de una de una descripción conductual explícitamente planificada, o de una secuencia de transferencias entre registros sin referencia a recursos, se conoce como **síntesis RT**.