Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2016

Dra. Dana K. Urribarri
DCIC - UNS

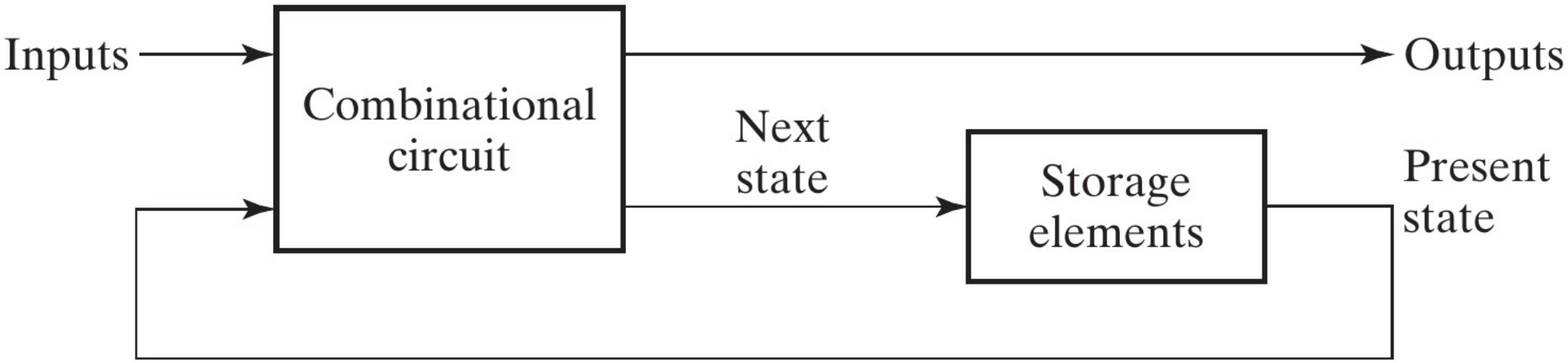
Circuitos Secuenciales

Circuitos secuenciales

- La respuesta de un circuito combinacional depende de las entradas externas.
- La respuesta de un circuito secuencial depende:
  - De las entradas (externas)
  - De la historia pasada (estado interno)
- Lo último se logra a través de elementos de memoria.

Circuitos secuenciales

- En general los sistemas digitales incluyen una parte combinacional y elementos de memoria.
- La parte combinacional del circuito:
  - Recibe las entradas externas y el estado actual.
  - Determina las salidas y el próximo estado del sistema.



Circuitos secuenciales

Los elementos de memoria almacenan el estado del sistema y hacen que el circuito se describa como *secuencial.*

Tienen dos propiedades:

- Tiempo de transición:

  Retardo entre la variación de la entrada y el cambio en la salida

- Capacidad de retención (hold)

  Capacidad de mantener el estado aún luego de que las señales que indujeron el cambio hayan desaparecido

Circuitos secuenciales

Tipos de circuitos secuenciales:

- Por nivel (o asincrónicos):

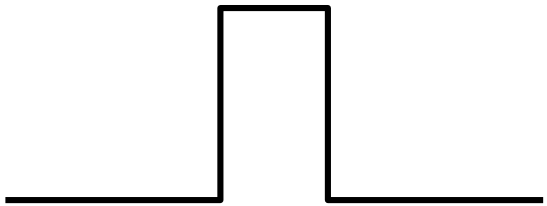
  Opera en función de los niveles (valores: 0–1, V_H–V_L) de las entradas.

- Por pulso:

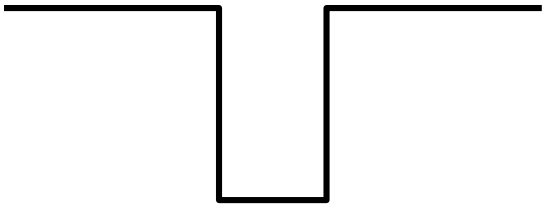
  Opera en función de los cambios de niveles de las entradas.

Pulso

Cambio de un nivel de voltaje a otro, seguido del retorno al nivel de voltaje inicial. El tiempo de desvío del voltaje inicial es relativamente corto en comparación con el tiempo entre los cambios.



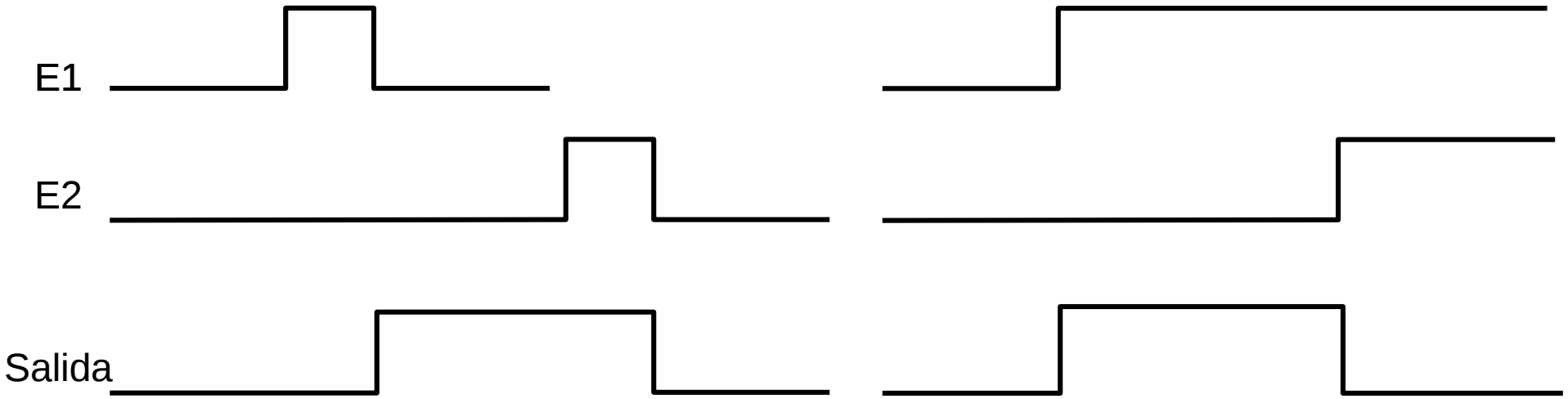
Pulso positivo



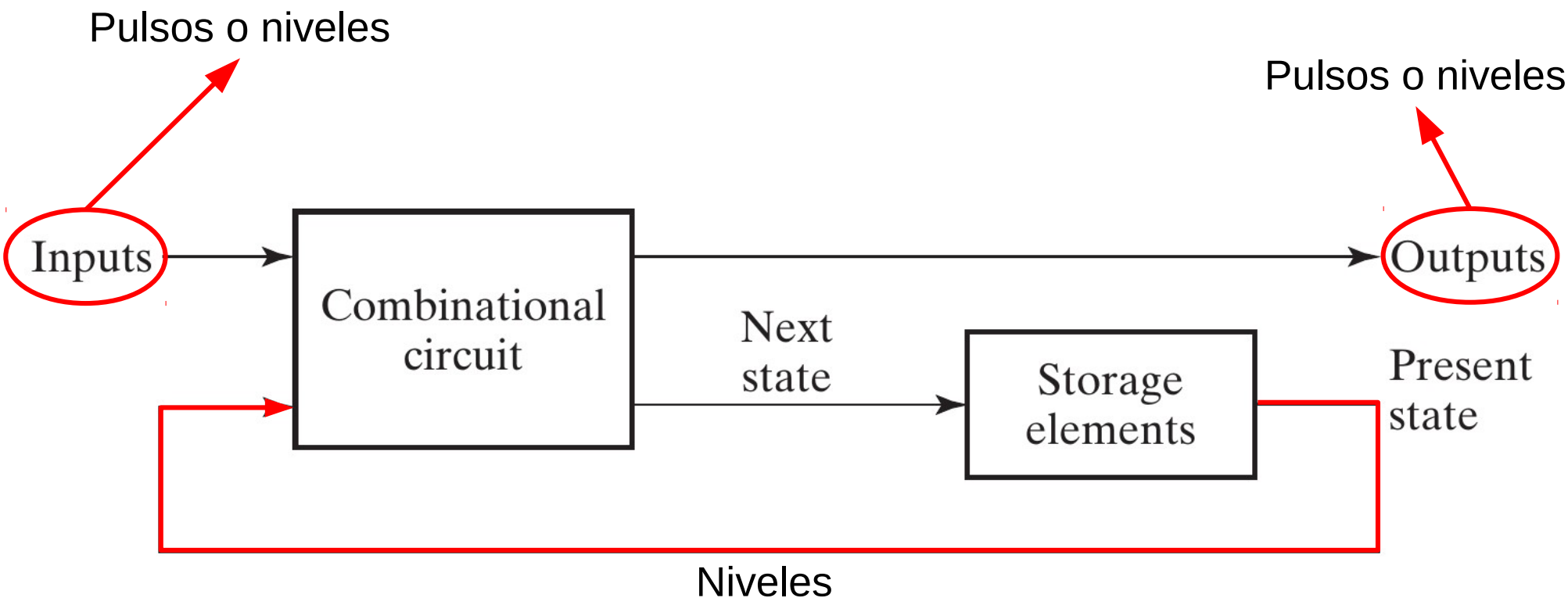
Pulso negativo

Circuitos secuenciales

- Por pulso
- Por nivel



Circuitos secuenciales

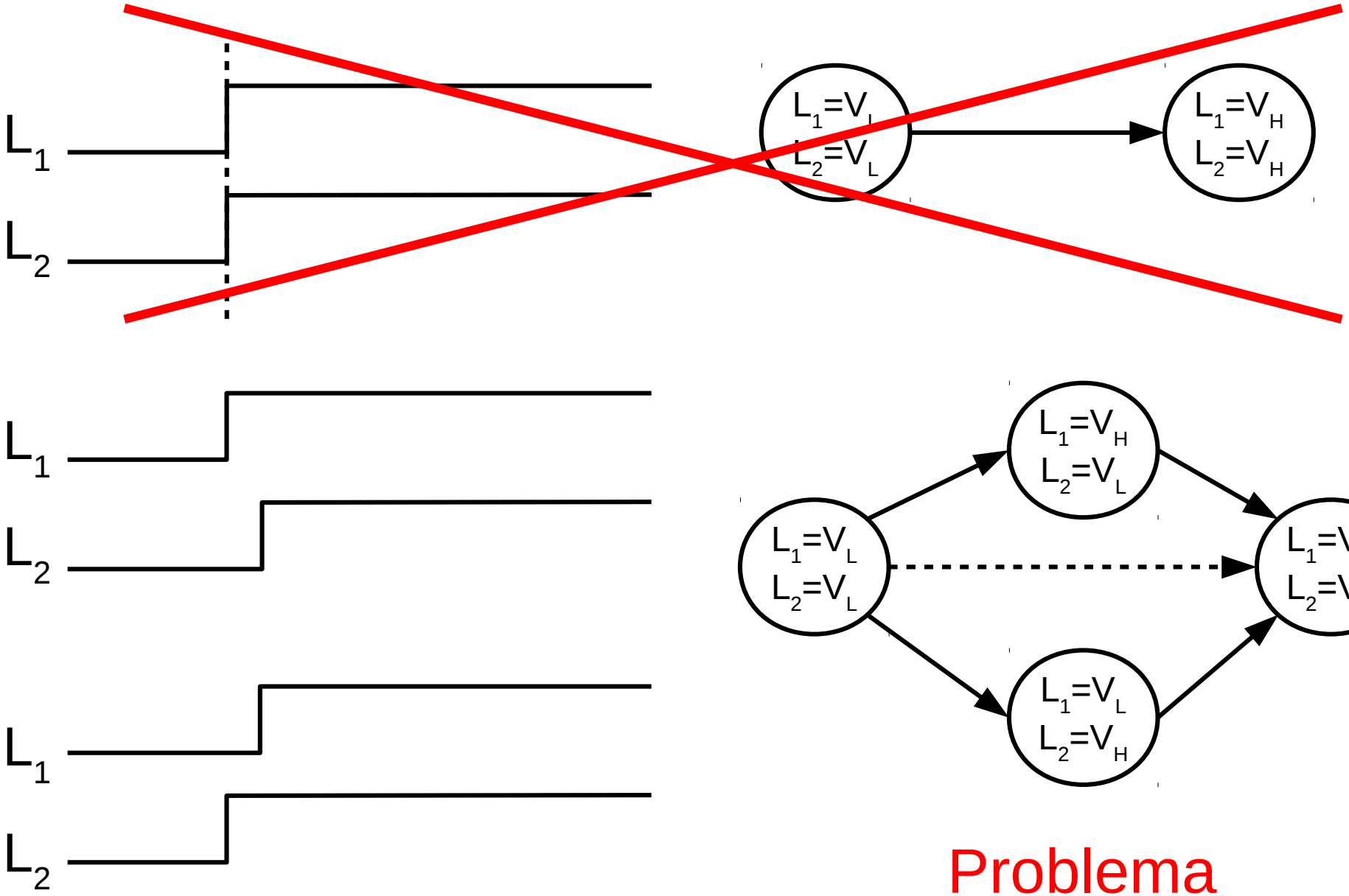


Circuitos secuenciales por nivel

- Los circuitos por nivel puros (asincrónicos) son difíciles de diseñar porque son muy dependientes del orden en el que cambian las señales de entrada.

- No vamos a considerarlos.

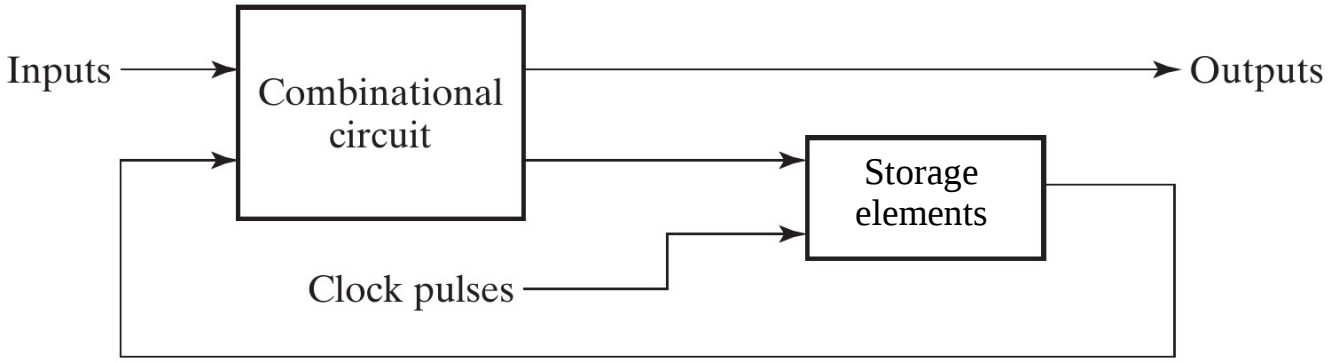
Circuitos secuenciales por nivel



Problema

Circuitos secuenciales por pulso

- Al menos una de las entradas debe ser un pulso.
- Típicamente hay una única señal pulso que es el reloj del circuito (sincrónico) que es la que dispara el cambio en las salidas.
- Si hay más de una entrada pulso, se asume que no ocurren en simultaneo.



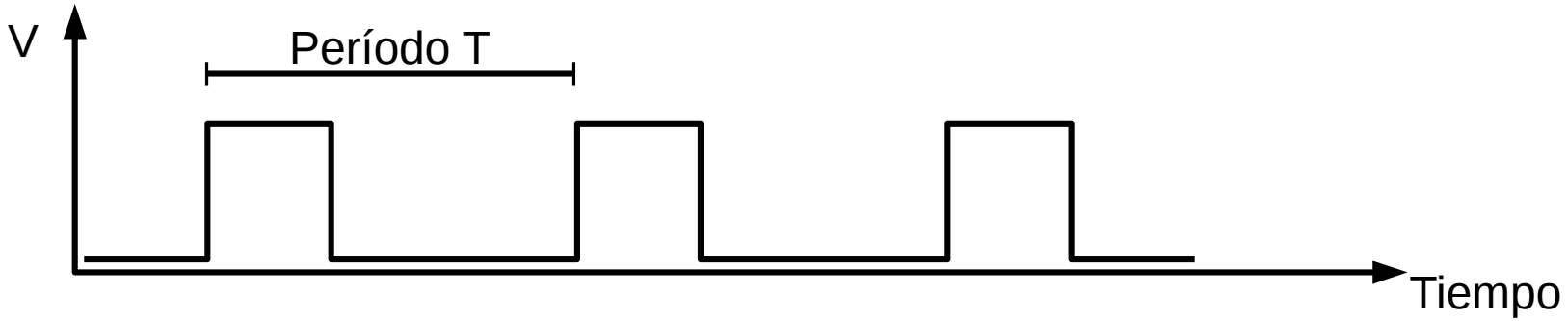
(a) Block diagram

(b) Timing diagram of clock pulses

Reloj

Tren de pulsos correspondiente a una señal periódica.

- T : período (segundos)
- F : frecuencia (ciclos por segundo ó Hertz)
- F = 1/T

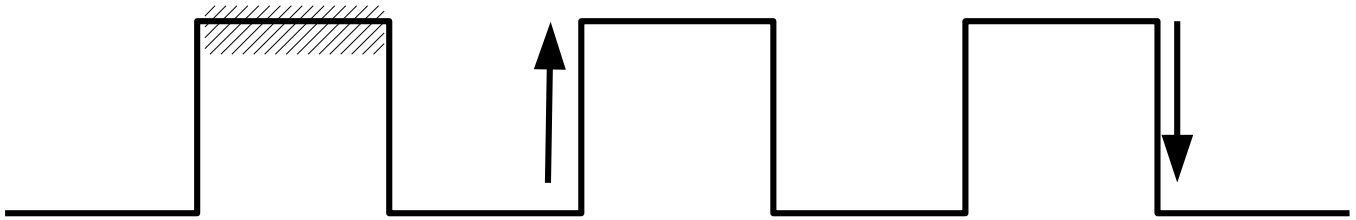


- Las compuertas trabajan en el orden de los nseg.
- T = 1nseg
- F = 1/(1 nseg) = 1/(10^{-9} seg) = 10^{9} Hz = 1 GHz

Reloj

El reloj puede dispara las acciones:

- Durante el pulso
- En el flanco ascendente
- En el flanco descendente



Elementos de memoria

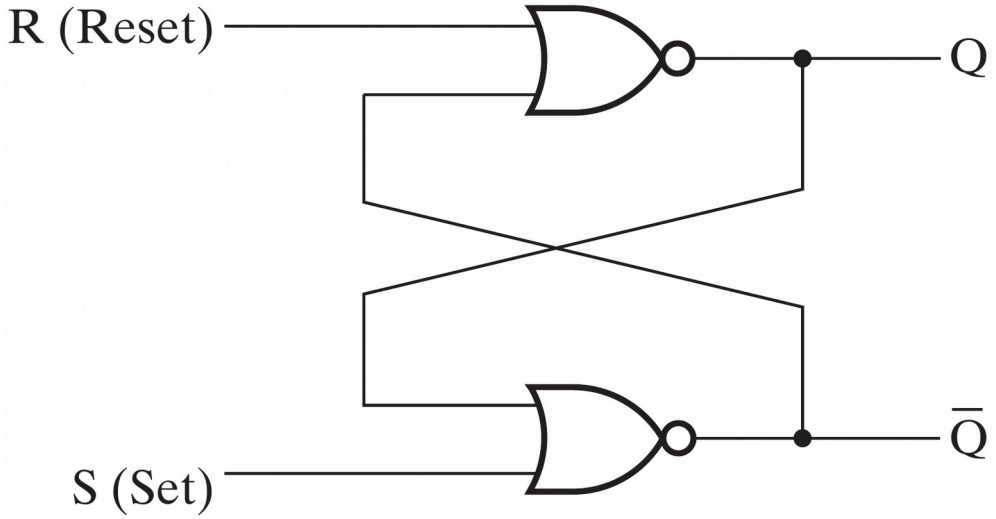
- Latches
- Flip flops

Latches

- Circuito asincrónico.
- Cambios en la entrada afectan inmediatamente la salida.
- Los latches son los bloques constructores de los Flip-flops (FF).

Latch SR y $\overline{S}$ $\overline{R}$

- El latch SR se construye a partir de dos NOR realimentadas.
- Tienen dos entradas: S (set) y R (reset)
- Tiene dos salidas: Q y Q'

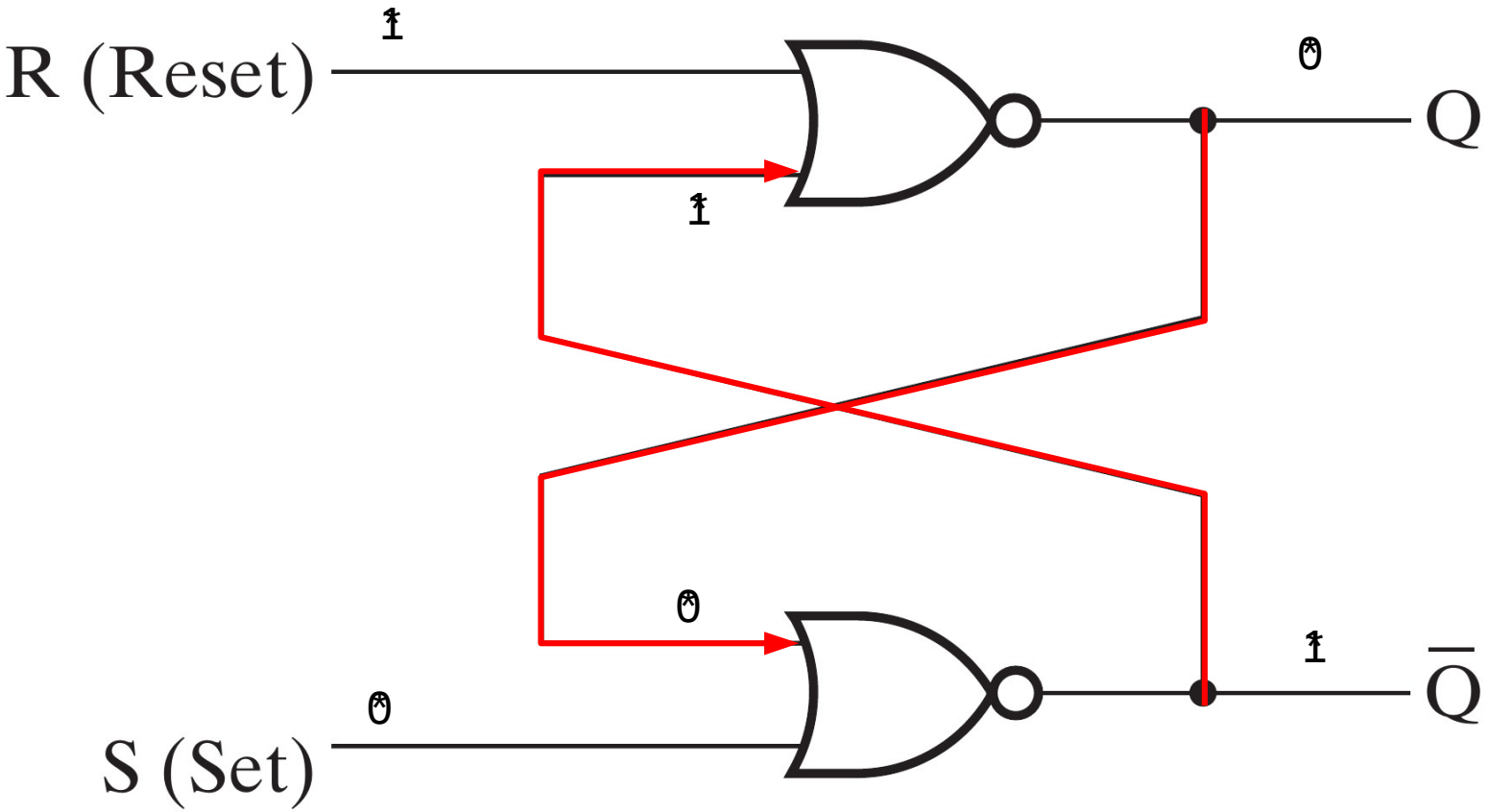


Latch SR y $\overline{S}$ $\overline{R}$

- Las salidas pueden tener dos estados:
  - Q=1 y Q'=0
  - Q=0 y Q'=1
- Cambian en función de 3 posibles combinaciones de entradas:
  - Set: Q → 1, Q' → 0
  - Reset: Q → 0, Q' → 1
  - Hold: Q → Q, Q' → Q'

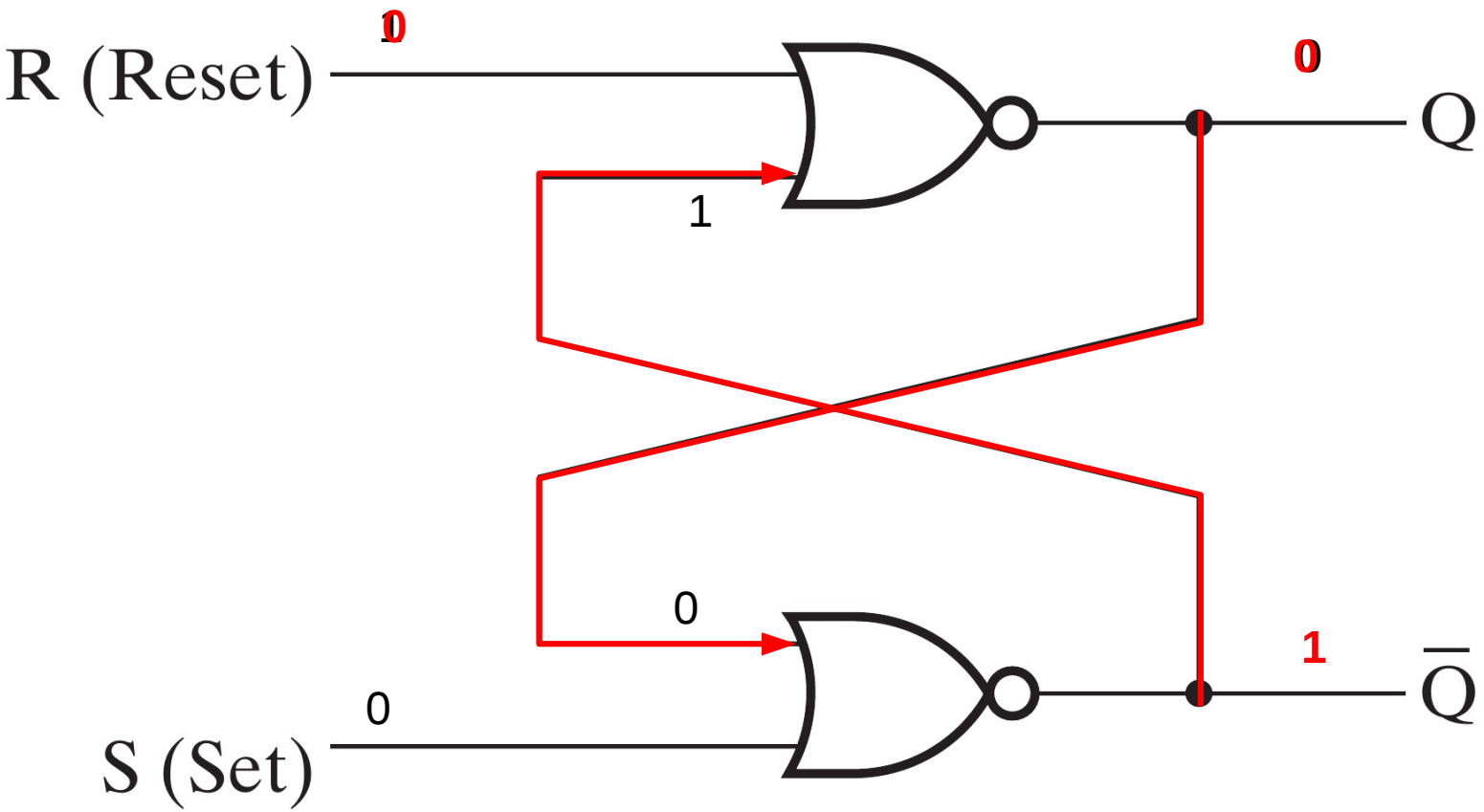
S	R	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	Indefinido	

Latch SR y $\overline{S}$ $\overline{R}$



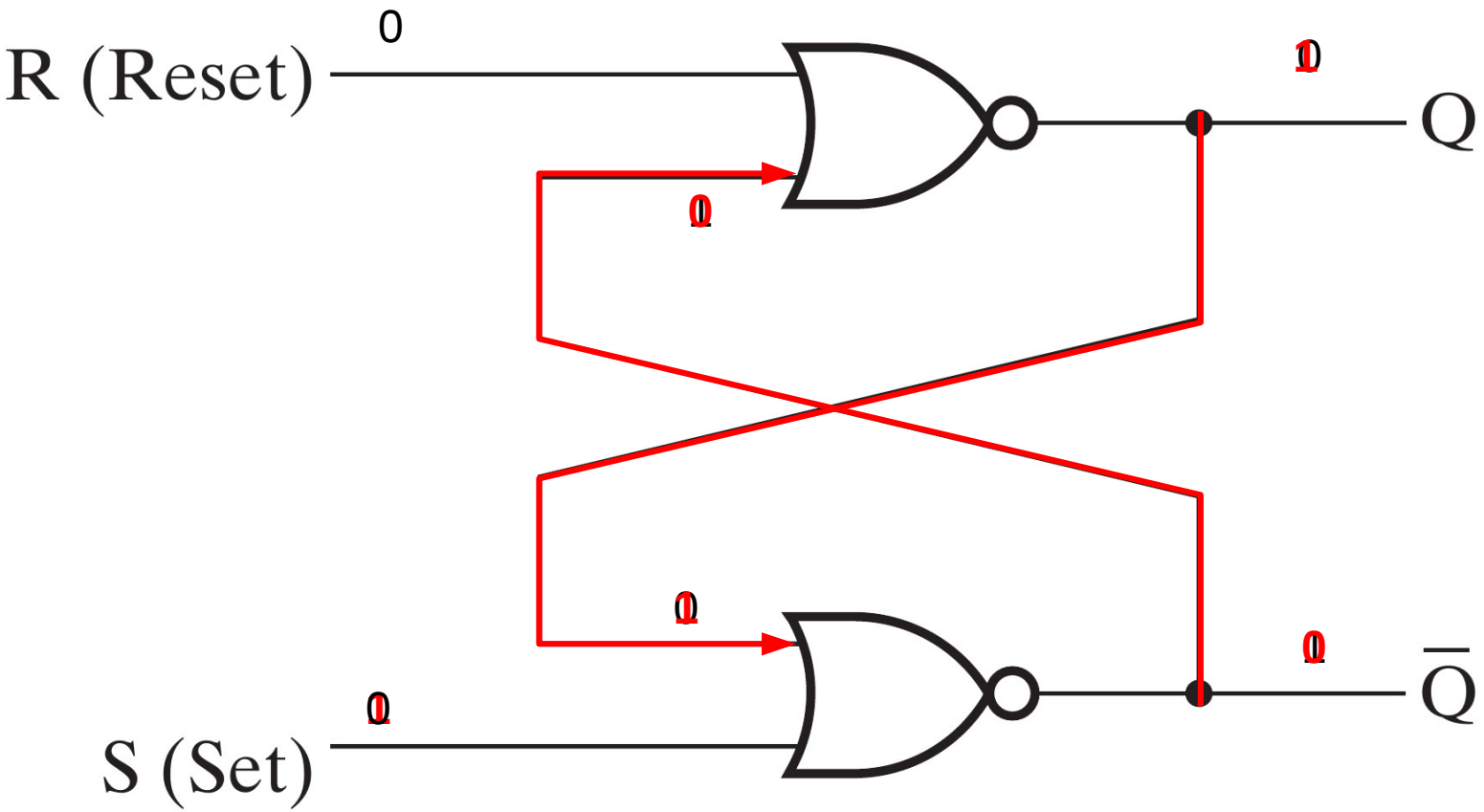
Reset

Latch SR y $\overline{S}$ $\overline{R}$



Hold

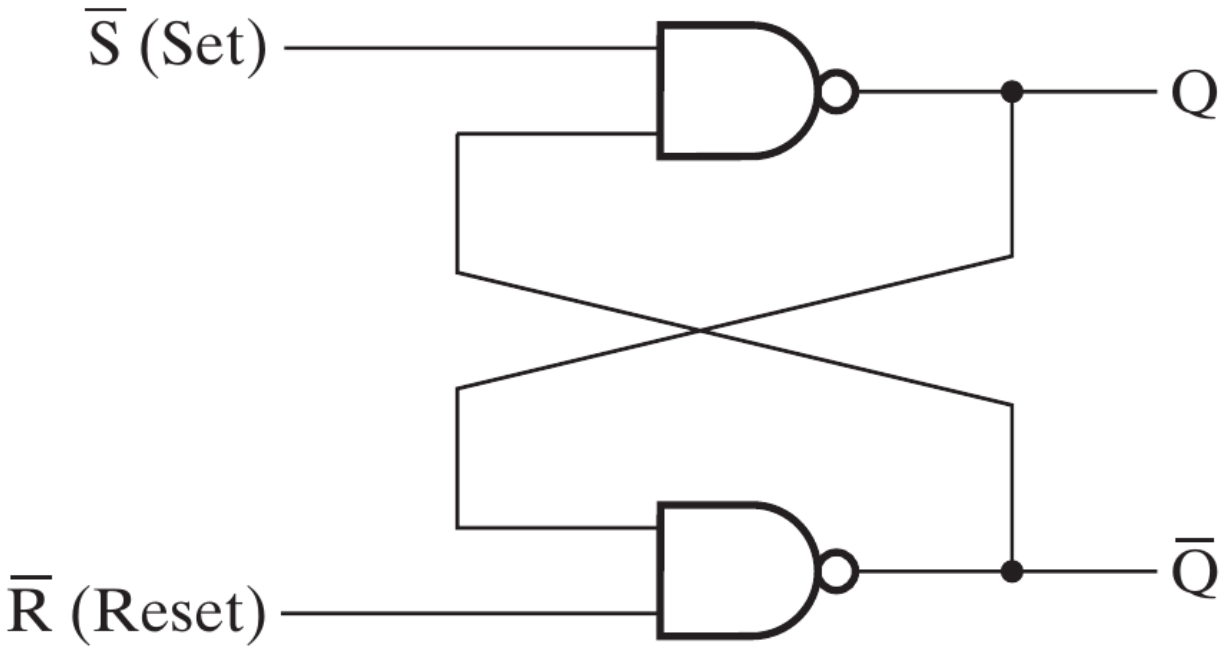
Latch SR y $\overline{S}$ $\overline{R}$



Set

Latch SR y $\overline{S}$ $\overline{R}$

- Un latch equivalente puede implementarse con dos NANDs y las entradas negadas.



S'	R'	Q	Q'
1	1	Q	Q'
1	0	0	1
0	1	1	0
0	0	Indefinido	

Flip flops

- Flip flop (biestable): Tienen dos estados estables
- Pueden realizar hasta tres operaciones:
  - Poner la salida en 1 (set)
  - Poner la salida en 0 (reset)
  - Complementar la salida (toggle)
- Hay 4 tipos de flip flops
  - SR: dos entradas. Realiza set y reset.
  - D: una entrada. Realiza set y reset.
  - JK: dos entradas. Realiza las tres operaciones.
  - T: una entradas. Realiza toggle.

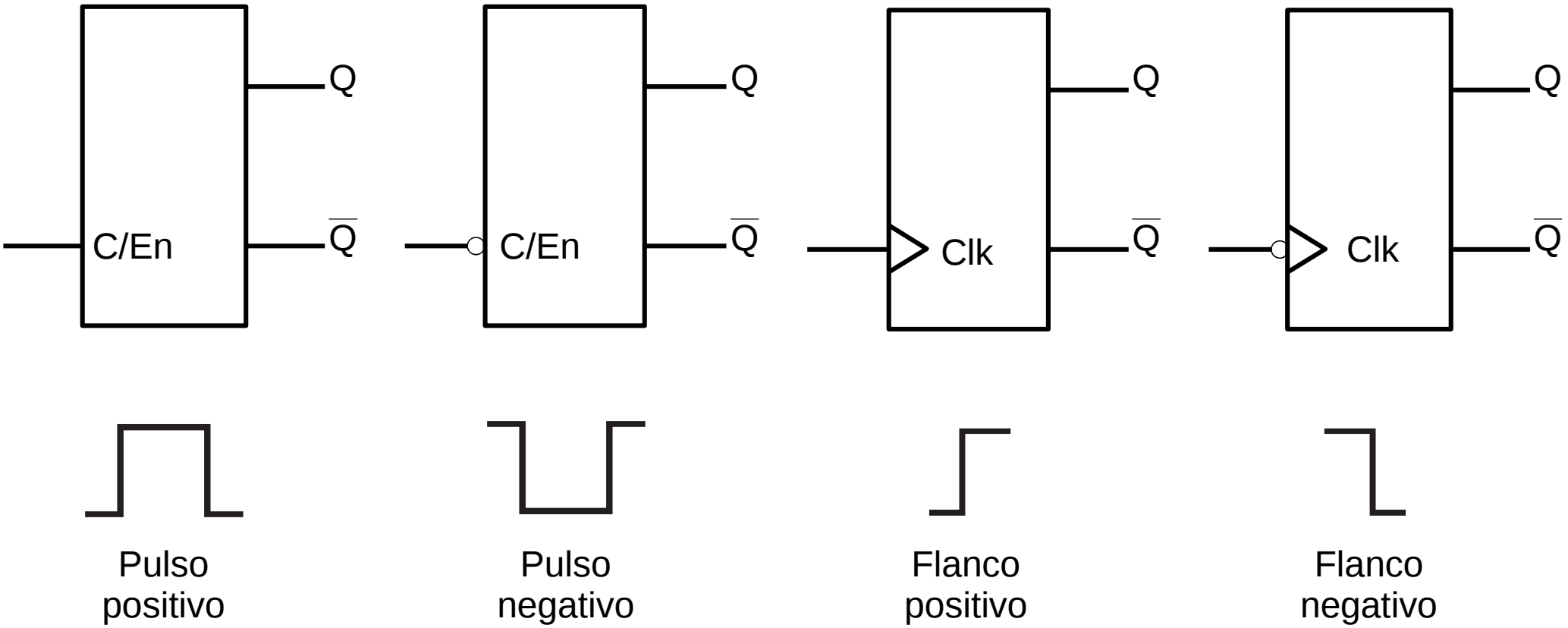
Flip flops

- A diferencia del latch tienen una entrada de control (Control, Enable, Clock)
- Mientras la entrada de control esté deshabilitada, la salida no cambia (independientemente de lo que ocurra con las demás entradas).

Flip flops

Habilitados por pulso

Habilitados por flanco



Pulso positivo

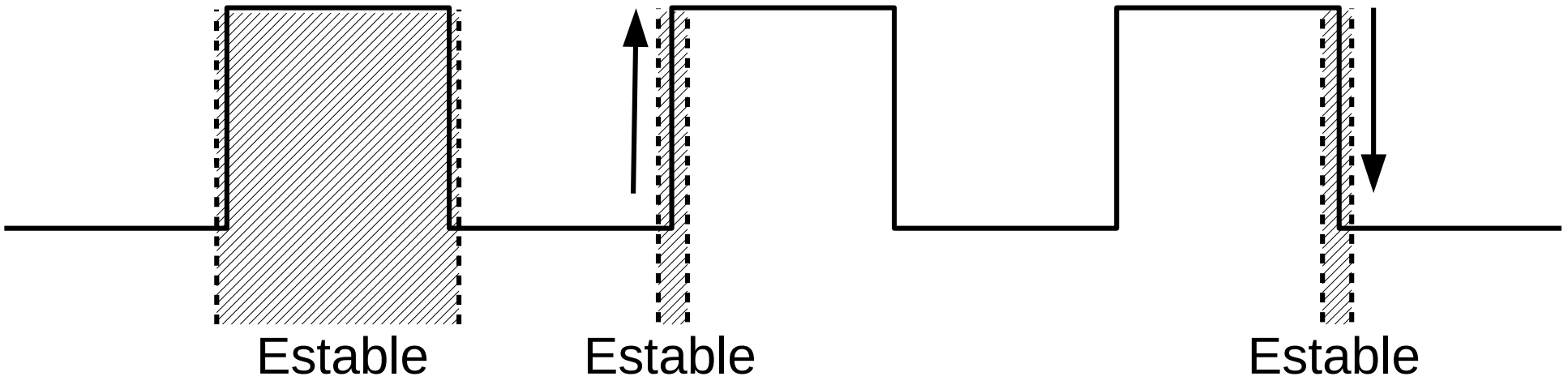
Pulso negativo

Flanco positivo

Flanco negativo

Flip flops

- Las entradas deben estar estables:
  - Un poco antes de que la señal de control de habilite.
  - Durante el período completo en que esté habilitada.
  - Un poco después de que se deshabilite.



Flip flops

✗ Habilitación por pulso:

Mientras que el pulso se mantenga en el nivel alto, cualquier cambio en las entradas se refleja en las salidas.

El pulso no puede ser arbitrariamente angosto.

✔ Habilitación por flanco ascendente o descendente:

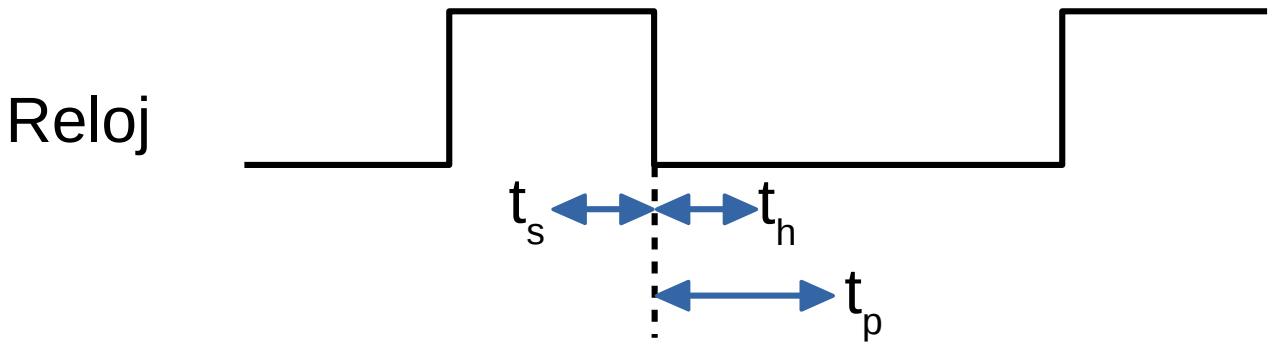
Disparar las acciones sólo durante la transición entre niveles.

Se logra mayor frecuencia y mejores resultados.

Flip flops

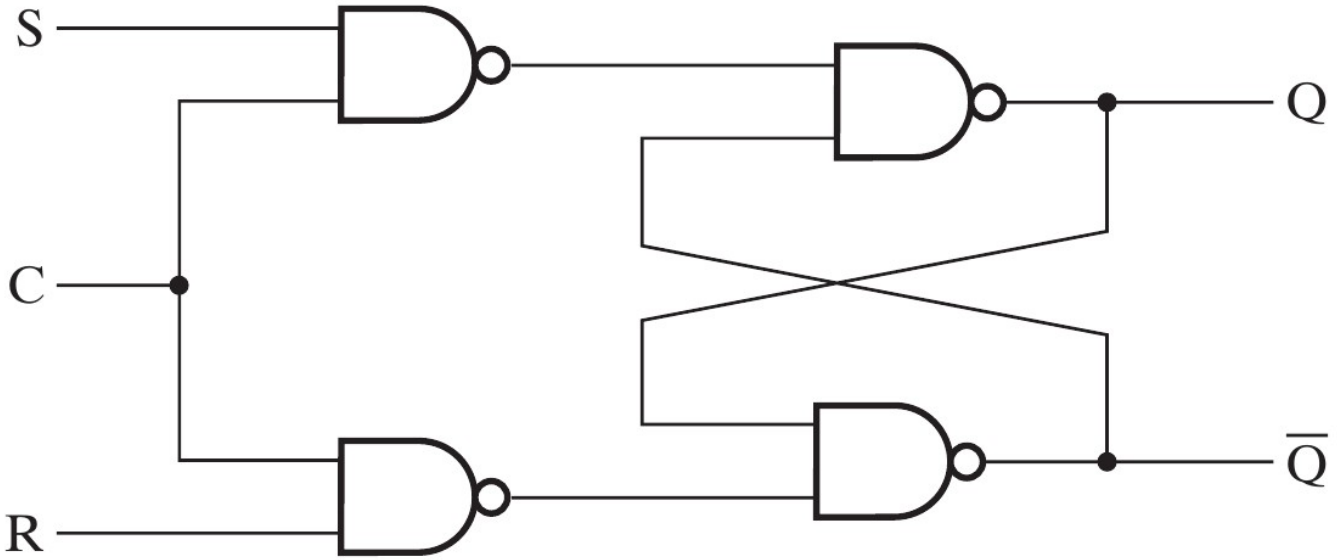
Temporizado:

- Tiempo de set-up: t_s
  - Tiempo que las entradas deben estar estables *antes* del disparo.
- Tiempo de hold: t_h
  - Tiempo que las entradas deben estar estables *después* del disparo.
- Tiempo de propagación: t_p
  - Retardo entre el disparo y el cambio en las salidas.



Flip flop SR

- A la latch $\overline{S}$ $\overline{R}$ se le puede agregar una entrada de control para indicarle cuándo cambiar las salidas.
- Mientras la entrada C = 0, las entradas del latch $\overline{S}$ $\overline{R}$ = 11 y no cambia de estado.



C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

Flip flop SR

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	?
1	1	1	?

S	R	$Q \rightarrow Q^+$
0	0	0 → 0
0	0	1 → 1
0	1	0 → 0
0	1	1 → 0
1	0	0 → 1
1	0	1 → 1

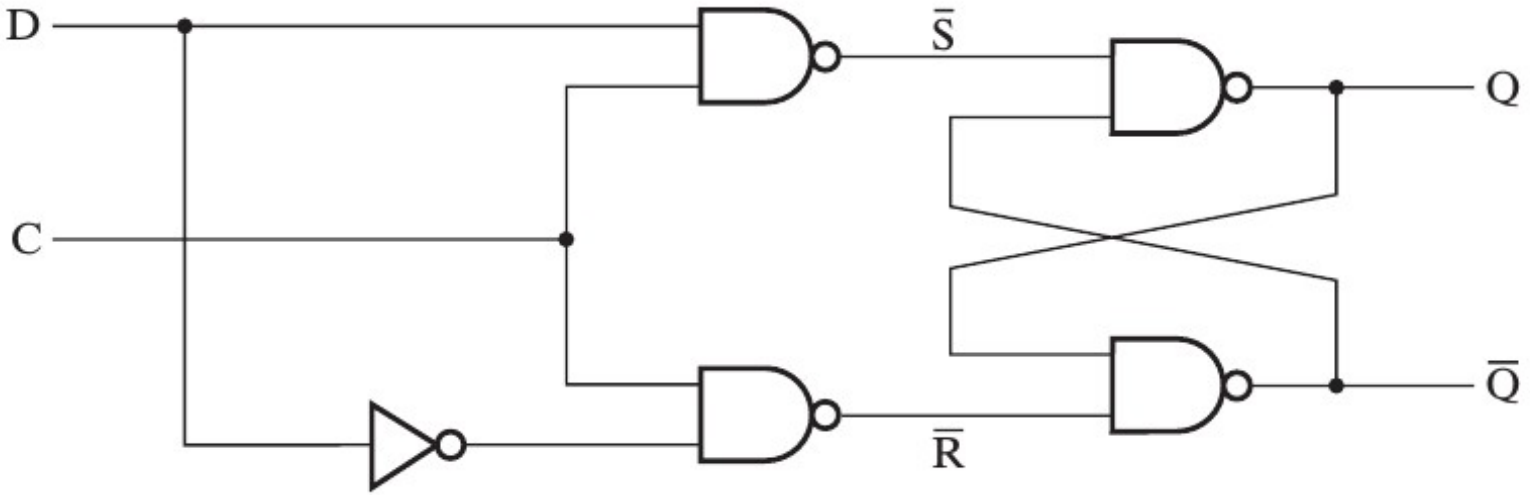
$Q \rightarrow Q^+$	S	R
0 → 0	0	*
0 → 1	1	0
1 → 0	0	1
1 → 1	*	0

Flip flop D

Para eliminar el estado indefinido: asegurarse que las entradas $\overline{S}$ y $\overline{R}$ nunca sean iguales.

Este FF almacena el dato D.

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state



(a) Logic diagram

Flip flop D

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

D	$Q \rightarrow Q^+$
0	$0 \rightarrow 0$
0	$1 \rightarrow 0$
1	$0 \rightarrow 1$
1	$1 \rightarrow 1$

$Q \rightarrow Q^+$	D
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	0
$1 \rightarrow 1$	1

Flip flop JK

- Corrige el estado indefinido del FF SR
  - La entrada J pone la salida en 1.
  - La entrada K pone la salida en 0.
  - Ambas entradas habilitadas complementan la salida.
- Cuando la entrada de control está habilitada:

JK Flip-Flop

J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

Flip flop JK

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J	K	$Q \rightarrow Q^+$
0	0	$0 \rightarrow 0$
0	0	$1 \rightarrow 1$
0	1	$0 \rightarrow 0$
0	1	$1 \rightarrow 0$
1	0	$0 \rightarrow 1$
1	0	$1 \rightarrow 1$
1	1	$0 \rightarrow 1$
1	1	$1 \rightarrow 0$

$Q \rightarrow Q^+$	J	K
$0 \rightarrow 0$	0	*
$0 \rightarrow 1$	1	*
$1 \rightarrow 0$	*	1
$1 \rightarrow 1$	*	0

Flip flop T

- Este FF hace toggle del estado.
- Cuando la entrada de control está habilitada:

T Flip-Flop

T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Flip flop T

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

T	$Q \rightarrow Q^+$
0	$0 \rightarrow 0$
0	$1 \rightarrow 1$
1	$0 \rightarrow 1$
1	$1 \rightarrow 0$

$Q \rightarrow Q^+$	T
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	1
$1 \rightarrow 1$	0

Etapa de diseño

1) *Formulación*: A partir de la especificación del problema, obtener una tabla de estados (por pulso) o tabla de flujo (por nivel).
2) *Eliminación de redundancia*: Reducir la tabla de estados.
3) *Asignación de estados*: Asignar a cada estado de la tabla un código binario.
4) *Expresiones del próximo estado*: Seleccionar los tipos de FF a usar. Derivar las ecuaciones de entradas a los FF a partir del próximo estado codificado en la tabla.
5) *Expresiones de salida*: Derivar las ecuaciones de salida a partir de la tabla de estados.
6) *Implementación*: Implementar los circuitos combinacionales derivados en 4 y 5.

Formulación

- *Entrada*: dos pulsos x_1 y x_2.
- *Salida*: un pulso coincidente con la secuencia x_1 x_2 x_2.

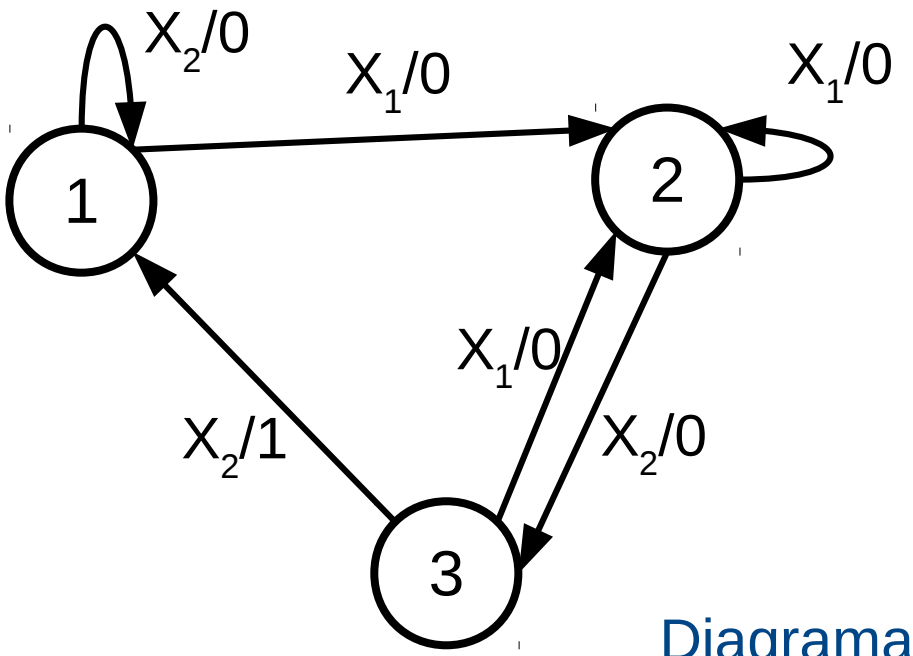


Diagrama de Mealy

	x_1	x_2
1	2/0	1/0
2	2/0	3/0
3	2/0	1/1

Formulación

- *Entrada*: dos pulsos x_1 y x_2.
- *Salida*: un nivel. Se pone en 1 con un pulso x_2 y se pone en 0 con la secuencia x_2 x_1 x_1.

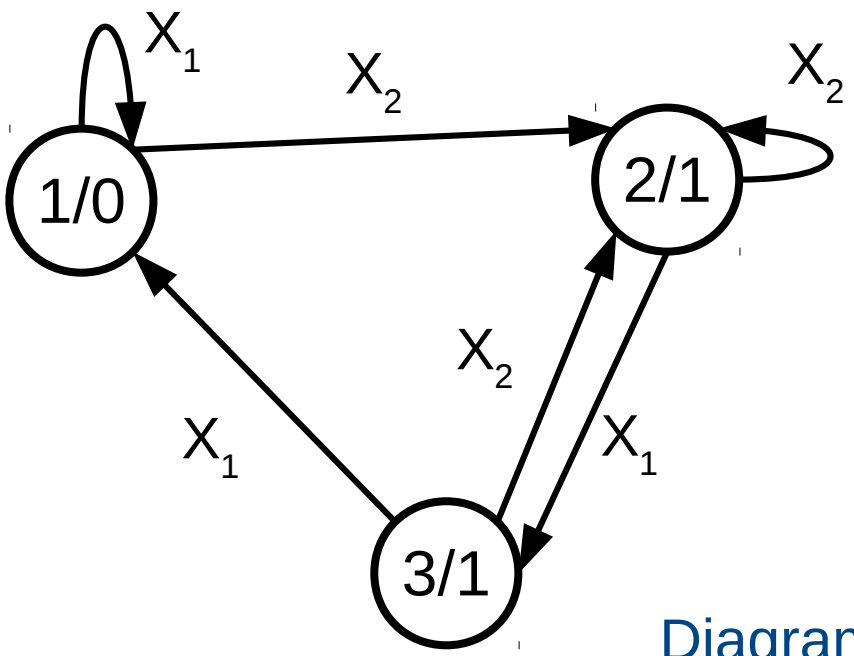


Diagrama de Moore

	x_1	x_2	Z
1	1	2	0
2	3	2	1
3	1	2	1

Formulación

- *Entradas*: dos niveles x_1 y x_2 y una señal de reloj.
- *Salidas*: un nivel z que se activa sólo cuando ambos niveles se activan simultáneamente luego de haber estado simultáneamente desactivados.

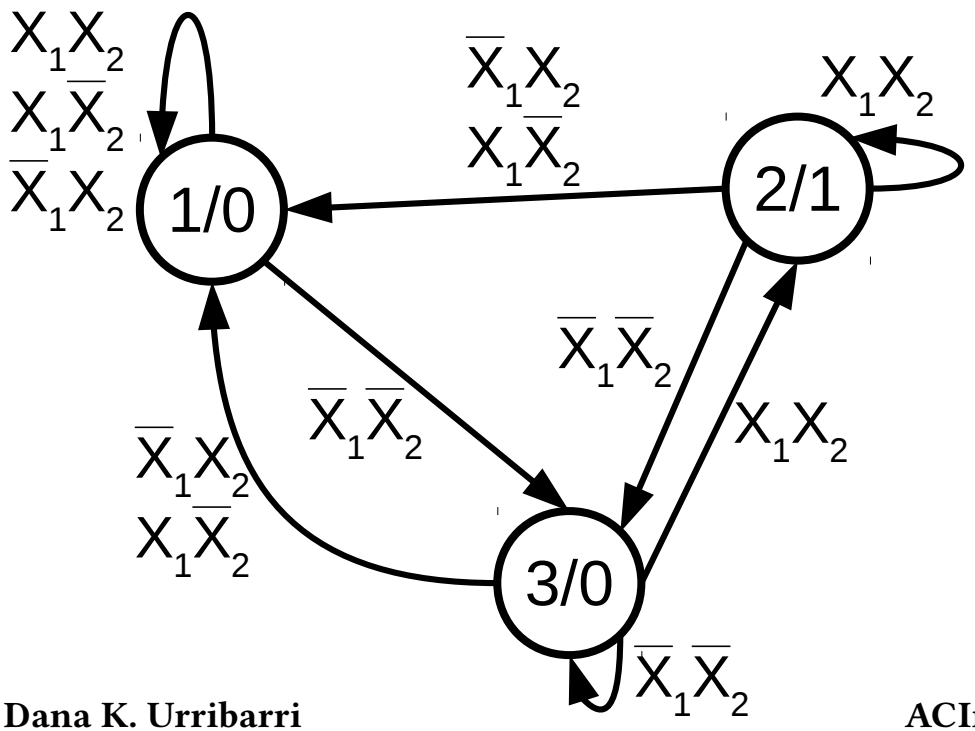
N señales nivel dan lugar a 2^N posibles transiciones distintas en función de qué señales están activas.

Las dos señales dan 4 posibles transiciones.

En el diagrama y en la tabla se omite la señal de reloj.

Formulación

- *Entradas*: dos niveles x_1 y x_2 y una señal de reloj.
- *Salidas*: un nivel z que se activa sólo cuando ambos niveles se activan simultáneamente luego de haber estado simultáneamente desactivados.



	x_1x_2	$x_1\bar{x}_2$	$\bar{x}_1x_2$	$\bar{x}_1\bar{x}_2$	z
1	1	1	1	3	0
2	2	1	1	3	1
3	2	1	1	3	0

Eliminación de redundancia

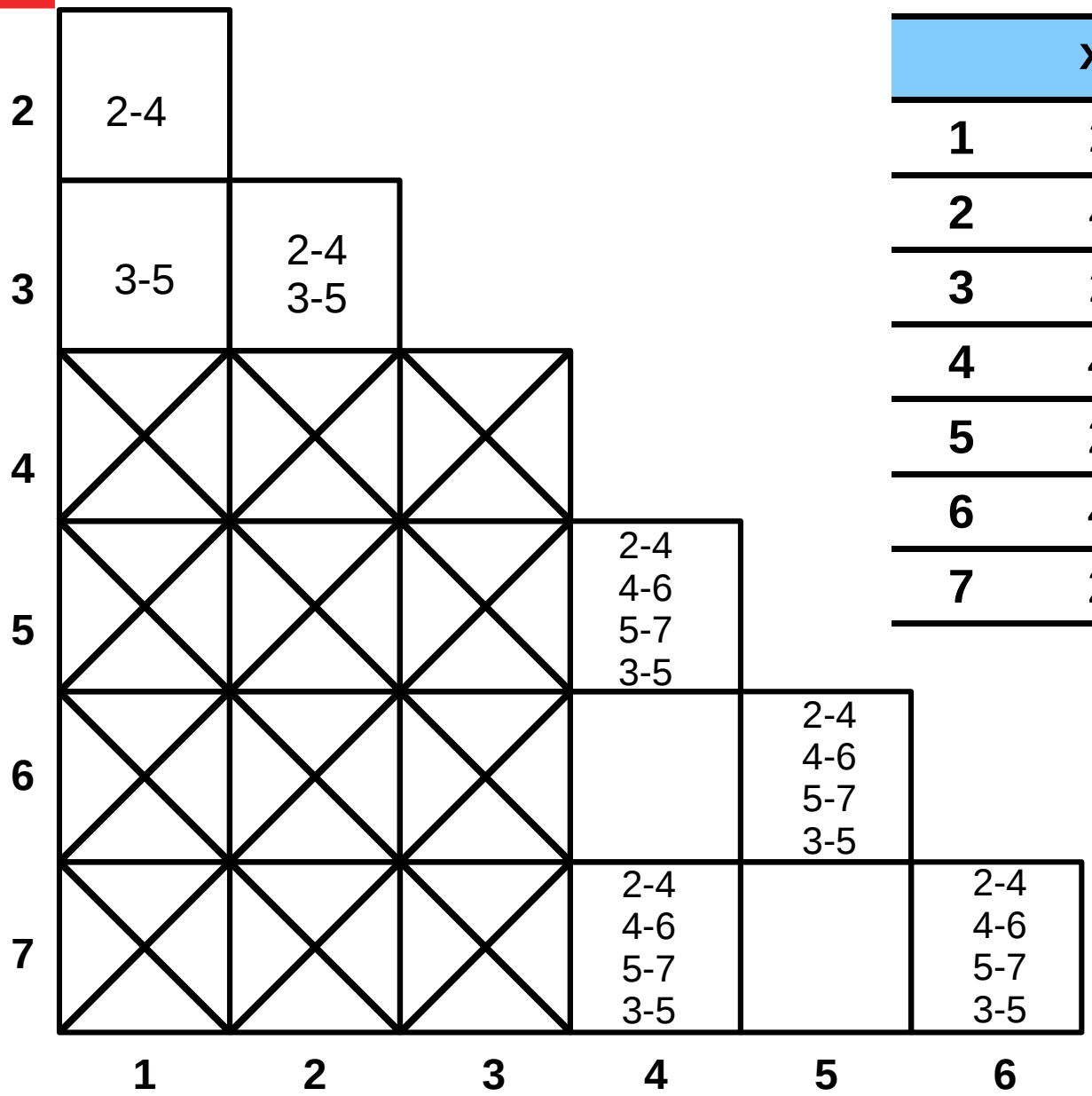
- Para reducir el número de estados hay dos opciones (por pulso)
  - Encontrar estados *equivalentes* para eliminar redundancia.
  - Si hay opcionales (*don't care*) en el próximo estados y/o en las salidas, es posible encontrar estados *compatibles*.

Eliminación de redundancia

- Dos estados S_1 y S_2 son equivalentes si:
  - S_1 y S_2 producen la misma salida. En un diagrama de Mealy debe ser válido para todas las combinaciones de entrada.
  - Para cada combinación de entradas, S_1 y S_2 deben tener el mismo próximo estado o equivalentes.

- Vamos a generar una tabla de pares implicado.

Eliminación de redundancia

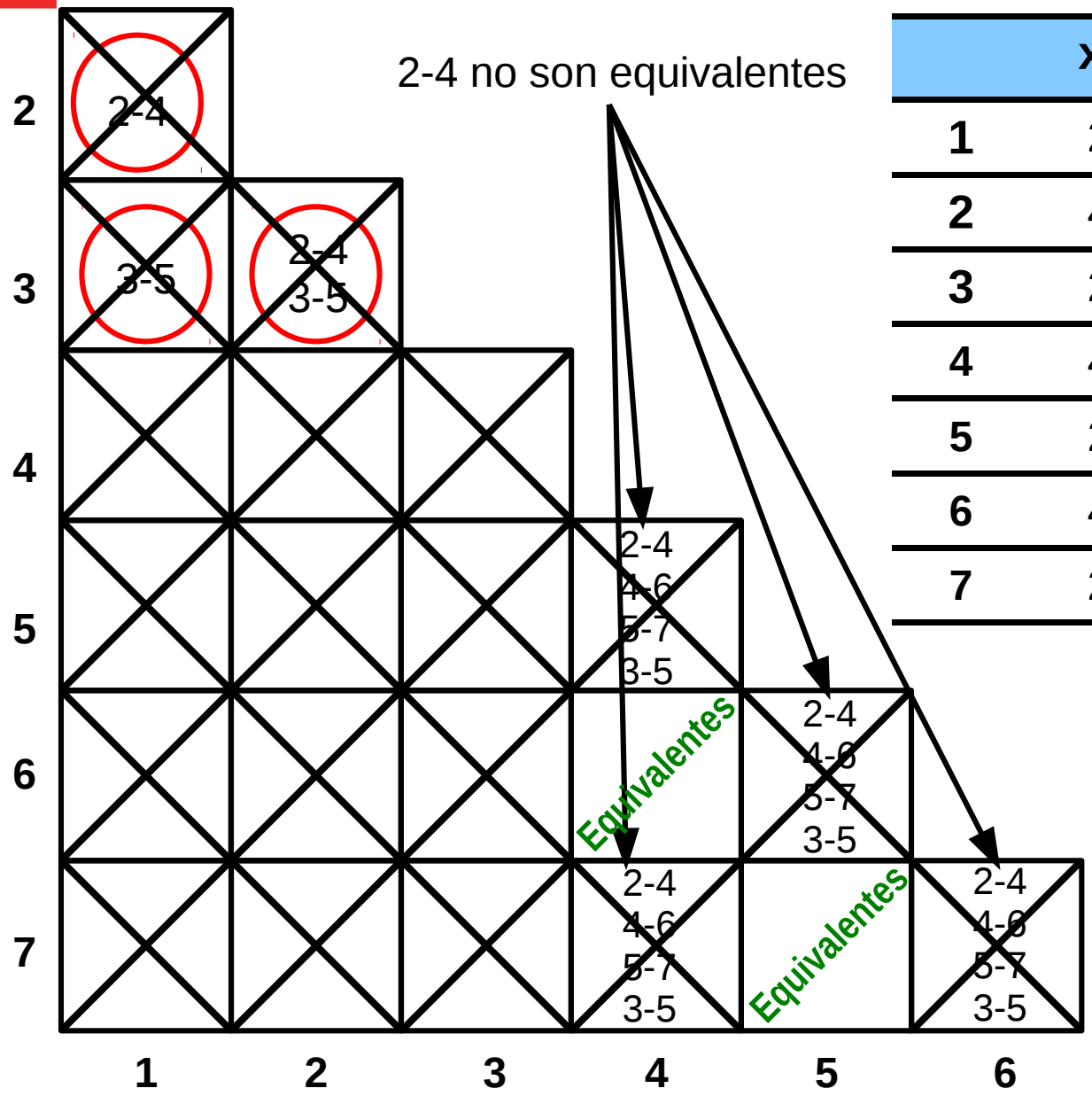


	x_1	x_2	x_3	x_4	Z
1	2	2	3	3	0
2	4	4	3	3	0
3	2	2	5	5	0
4	4	4	7	3	1
5	2	6	5	5	1
6	4	4	7	3	1
7	2	6	5	5	1

En función de las salidas se marcan los estados que se saben no equivalentes.

Se comparan todos lo pares de posibles estados equivalentes y se anotan los pares implicados (pares que deben ser equivalentes para que el par inicial lo sea).

Eliminación de redundancia



	x_1	x_2	x_3	x_4	Z
1	2	2	3	3	0
2	4	4	3	3	0
3	2	2	5	5	0
4	4	4	7	3	1
5	2	6	5	5	1
6	4	4	7	3	1
7	2	6	5	5	1

Vemos cada casilla no tachada y verificamos qué ocurre con los pares implicados.
Si alguno no es equivalente, entonces los pares iniciales no son equivalentes.

Las casillas no tachadas son los estados equivalentes.

Eliminación de redundancia

- En tablas no completamente especificadas, dos estados S_1 y S_2 son compatibles si:
  - Un estado produce una salida especificada y el otro una salida opcional. En un diagrama de Mealy debe ser válido para la misma combinación de entrada.
  - Para cada combinación de entradas, los próximos estados de S_1 y S_2 deben ser el mismo estado, estados compatibles o uno estar establecido y el otro opcional.

- Si S_1 y S_2 son compatibles, los opcionales deben adoptar el valor especificado.

Eliminación de redundancia

	x_1	x_2	x_3	x_4	Z
5	8	–	10	6	0
6	–	7	10	5	*

	x_1	x_2	x_3	x_4	Z
11	8	7	10	11	0

	x_1	x_2	x_3	x_4	Z
5	8	–	4	6	0
6	–	7	10	5	*

Sólo si 4 y 10 compatibles.

	x_1	x_2	x_3	x_4	Z
11	8	7	10	11	0

Eliminación de redundancia

- Diferencia entre estados equivalentes y compatibles
  - Si un estado es equivalente a otros *n* estados, entonces los *n*+1 estados son equivalentes.
  - Para que un conjunto de *n* estados pueda ser cubierto por un único estado, se requiere que todos los estados sean compatibles de a pares.

	x_1	x_2	x_3	x_4	Z
1	1	4	–	7	0
2	2	5	6	–	0
3	3	–	6	7	0
4	1	4	6	–	1
5	2	5	6	–	0

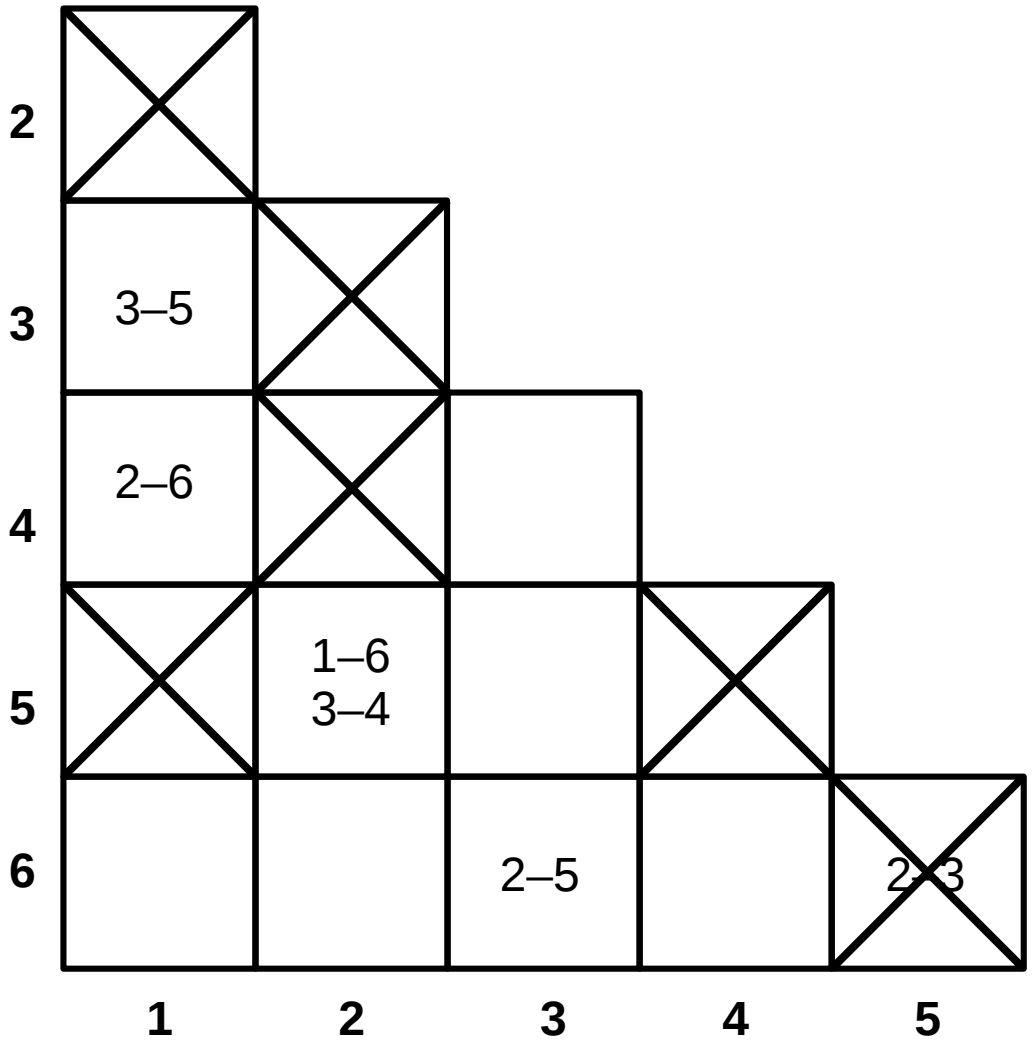
3 es compatible con 1 y con 2
Pero 1 y 2 no son compatibles entre si.

Una posibilidad es generar dos estados:
1–3 un estado A
2–3 un estado B

Eliminación de redundancia

- Una vez que se encontraron todos lo pares compatibles, hay que encontrar el mínimo conjunto cerrado que cubre a todos los estados.
- No existe un algoritmo para resolver esto.
- Se propone comenzar a partir de los máximos compatibles, que se obtiene:
  - Se obtiene el producto de la suma de los pares incompatibles.
  - Se desarrolla obteniendo una suma de productos no redundante.
  - El conjunto de estados faltantes en cada uno de los productos son los máximos compatibles.
  - Los máximos compatibles y sus subconjuntos son los posibles compatibles.

Eliminación de redundancia



	x_1	x_2	x_3	x_4	z_1	z_2
1	2	–	6	3	0	0
2	1	–	4	5	1	1
3	–	5	–	5	–	0
4	6	–	–	–	0	–
5	6	3	3	–	1	–
6	–	2	–	–	–	–

En función de las salidas marcamos los estados que no van a ser compatibles.

Marcamos en la tabla los pares de estados implicados que deberían ser compatibles.

Vemos qué nuevos estados incompatibles surgen.

Eliminación de redundancia

(1+2)(1+5)(2+3)(2+4)(4+5)(5+6) =

1246 + 1345 + 1346 + 25

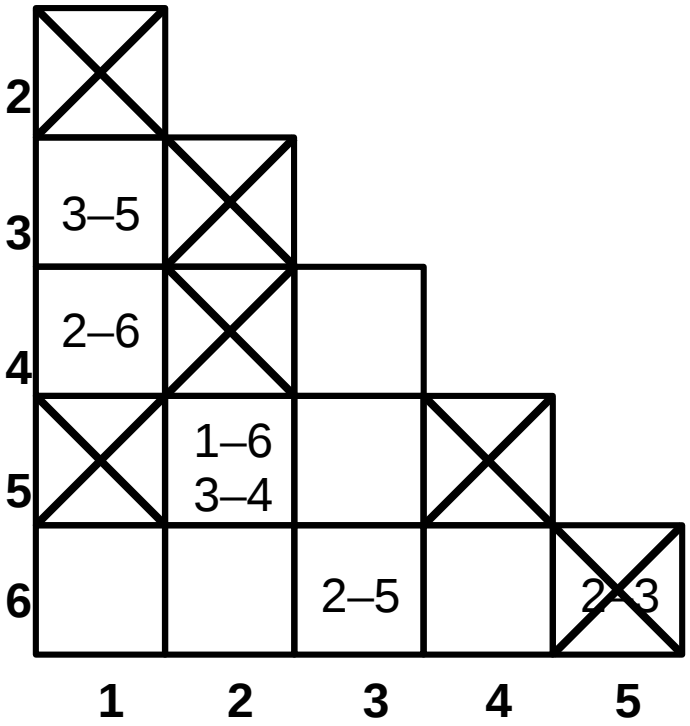
35 26 25 1346

Los máximos compatibles serán:

35, 26, 25 y 1346

1346 implica los otros tres máximos compatibles.

Cota superior para la solución → 4 estados.

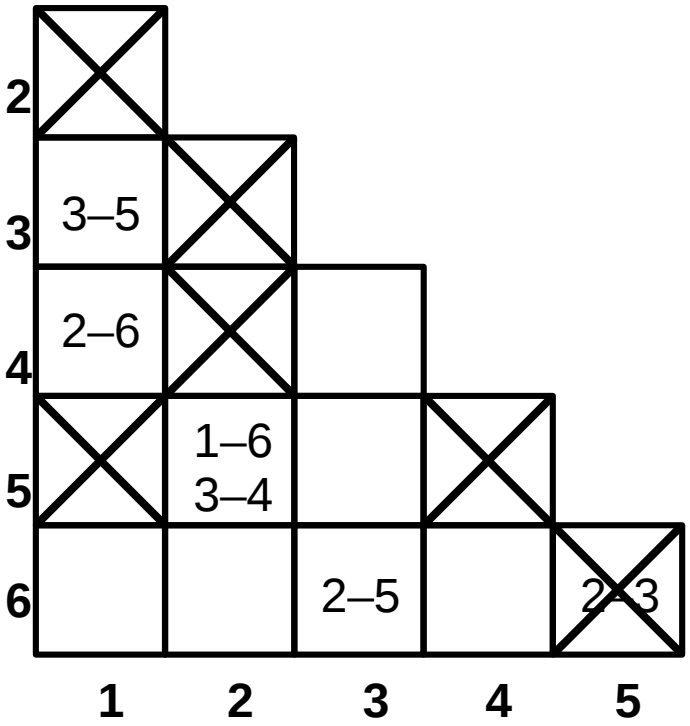


Eliminación de redundancia

Los estados 1 y 4 aparecen sólo en el máximo compatible 1346.

Se prueba dividiéndolo de diferentes formas conservando el 1 y el 4:

134; 146; 14; 13 y 46; 16 y 34



Eliminación de redundancia

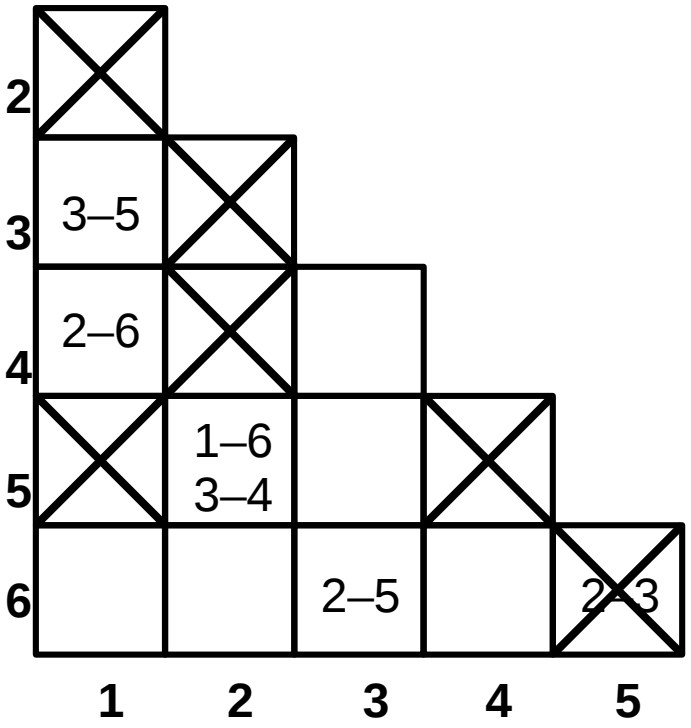
1) 134 implica 26 y 35. Solución

2) 146 implica 26. Hay que agregar 35

3) 14 implica 26. Hay que agregar 35

4) 13 y 14 implican 35.
   Hay que agregar el 2

5) 16 y 34 no implican nada.
   Hay que agregar 25.



3) y 5) son soluciones de 3 estados y sin redundancia.

Eliminación de redundancia

Solución 3

- A: 14
- B: 26
- C: 35

	x_1	x_2	x_3	x_4	z_1	z_2
A	B	–	B	C	0	0
B	A	B	A	C	1	1
C	B	C	C	C	1	0

Solución 5

- A: 16
- B: 34
- C: 25

	x_1	x_2	x_3	x_4	z_1	z_2
A	C	C	A	B	0	0
B	A	C	–	C	0	0
C	A	B	B	C	1	1