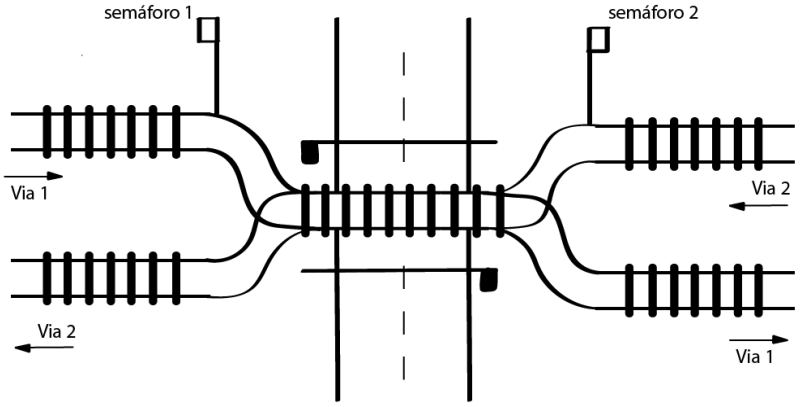
- Mínimo para pasar al oral: un problema
- Cada hoja debe tener Nombre y CI.
- Utilice solo un lado de las hojas
- Deben estar numeradas y la primer hoja debe decir el total de hojas
- Incluya un solo problema por hoja
- Sea prolijo

Problema 1

Se desea diseñar un circuito modo reloj capaz de controlar la intersección de dos vías férreas. Ambas vías son unidireccionales (los trenes no pueden ir marcha atrás) y tienen un camino en común como se observa en la figura.



Se cuenta con dos detectores (D1 y D2), uno en cada vía, que indican con un 0 en su salida cuando un tren se encuentra llegando al cruce y vuelve a su estado de reposo en 1 cuando el tren terminó de pasar por el cruce.

Para controlar el cruce de los trenes se dispone de dos semáforos (S1 y S2) ubicados uno en cada vía. Los mismos se encuentran en rojo cuando en su entrada se coloca un 1 lógico y se encuentran en verde cuando se coloca un 0. Estos semáforos deberán estar en rojo, mientras no se esté dando paso a un tren. El pasaje de un semáforo de rojo a verde debe realizarse de manera síncrona con el reloj del sistema mientras que el pasaje de verde a rojo se debe realizar de manera asíncrona.

Adicionalmente, la vía común es cruzada por una avenida que cuenta con barreras. Al detectar la presencia de un tren por cualquiera de las dos vías se deberá bajar inmediatamente la barrera para cortar el paso colocando un 0 lógico en B (ambas barreras se manejan con la misma señal B). Para garantizar que la barrera esté completamente baja, esta necesitará al menos un periodo de reloj completo en bajo antes de poder dar luz verde para el paso del tren. Una vez que no se detecte la presencia de trenes, se debe subir la barrera en forma síncrona poniendo B=1.

Al detectar un tren por alguna de las vías se deberá bajar la barrera, esperar el tiempo requerido por la barrera, y luego prender la luz verde del semáforo correspondiente. Al detectar que el tren finalizó su pasaje se debe subir la barrera y volver el semáforo a rojo. Si durante el pasaje de un tren se detecta la llegada de un tren en la otra vía, se deberá esperar a que finalice de pasar el tren en curso, luego cambiar su semáforo a rojo para posteriormente dar paso al otro tren colocando su semáforo en verde; todo esto sin levantar la barrera. Si ocurriera que se detecta la llegada de trenes por ambas vías antes de darle paso a ninguno, se le deberá dar prioridad al tren de la vía 1.

Se garantiza que los trenes demoran varios ciclos de reloj en atravesar el cruce.

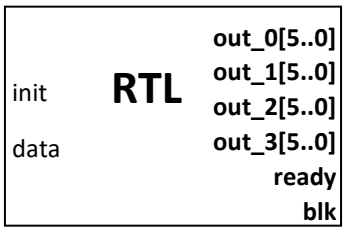
Se pide implementar el circuito modo reloj completo en forma mínima.

FACULTAD DE INGENIERIA
UNIVERSIDAD DE LA REPUBLICA

DISEÑO LÓGICO
Febrero 2021

Problema 2

Se desea diseñar un circuito RTL para recepcionar y decodificar una trasmisión de datos serial. Cada palabra trasmitida será de 9 bits y al decodificarla se obtendrá una palabra de 8 bits, el inicio de cada trasmisión se señaliza con un pulso de 1 Tclk por la señal ***init*** y los datos se reciben por la señal ***data*.**



La codificación es tal que para la palabra trasmitida **t[8..0]** y su correspondiente decodificación **d[7..0]** se cumple que:

$$d[n] = t[n+1] \text{ xor } t[n] \quad n = 0,1,..,7$$

De cada palabra decodificada los 2 bits más significativos (**d[7..6]**) indican qué salida se deberá actualizar con los 6 bits menos significativos (**d[5..0]**).

out_0[5..0] = d[5..0] si d[7..6] = 00
out_1[5..0] = d[5..0] si d[7..6] = 01
out_2[5..0] = d[5..0] si d[7..6] = 10
out_3[5..0] = d[5..0] si d[7..6] = 11

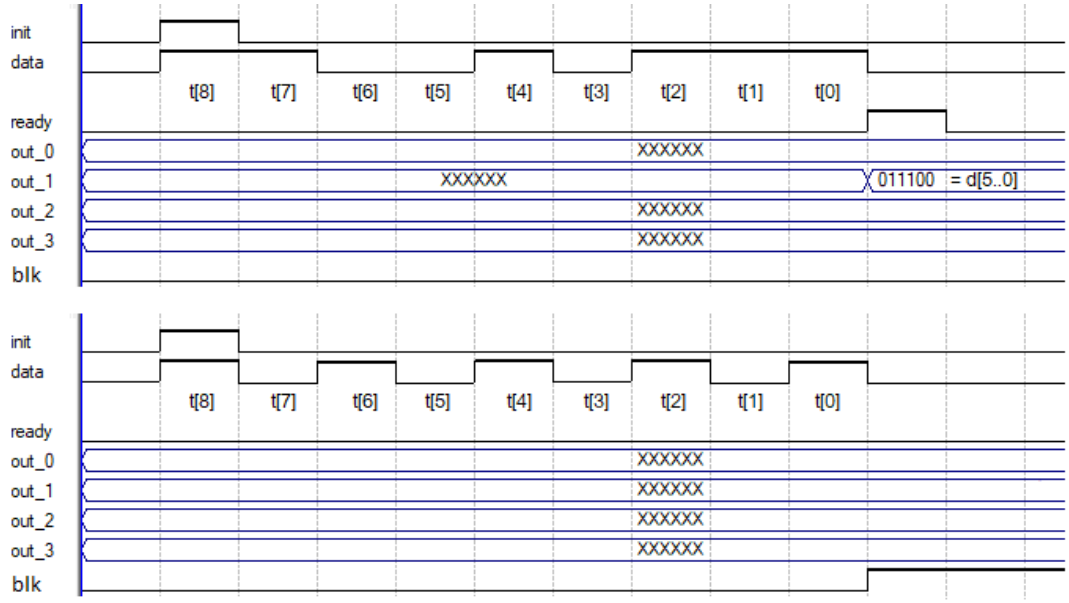
Luego de un reset las salidas deben tener los valores: ***ready***=0, ***blk***=0 y ***out_i[]***= 000000.

La salida ***out_i[]*** se debe actualizar en el período de reloj siguiente a la recepción del último bit (**t[0]**) y el dato en cada salida se debe mantener hasta que se actualice dicha salida con un nuevo dato.

Cada vez que se actualiza una salida ***out_i[]*** se debe dar un pulso de 1 Tclk por la salida ***ready*** (ver diagrama de tiempos).

Si el dato decodificado d[5..0] es 111111, NO se debe actualizar ninguna salida ***out_i[]*** a partir de ese momento. Además, se debe subir la señal ***blk*** en el siguiente período de reloj de recibido dicho dato. El circuito debe permanecer así hasta la ocurrencia de un reset.

Se debe considerar que las trasmisiones comienzan por el bit más significativo (**t[8]**) y que son posibles las trasmisiones consecutivas (luego de t[0] viene un nuevo t[8]).



Se pide descripción RTL, bloque de datos y bloque de control.

SOLUCIÓN

PROBLEMA 1

Diagrama

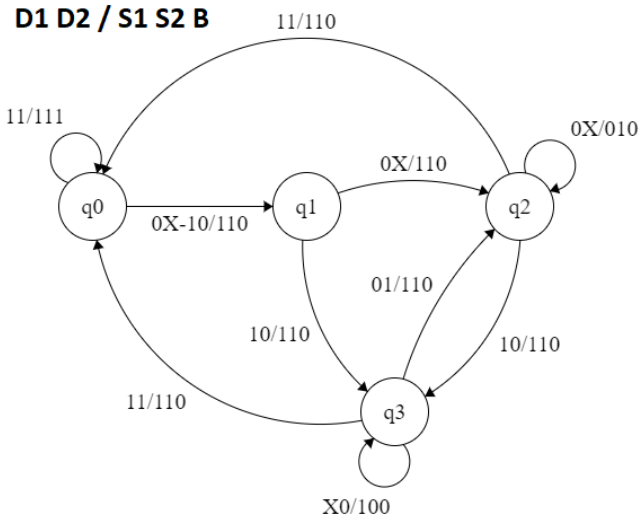


Tabla de Estados

	qt+1				S1-S2-B			
qt \ D1D2	00	01	11	10	00	01	11	10
q0	q1	q1	q0	q1	110	110	111	110
q1	q2	q2	X	q3	110	110	X	110
q2	q2	q2	q0	q3	010	010	110	110
q3	q3	q2	q0	q3	100	110	110	100

Minimización

q1	~~1-2~~ ~~1-3~~		
q2	X	X	
q3	X	X	X
	q0	q1	q2

Es mínimo

Codificación de estados

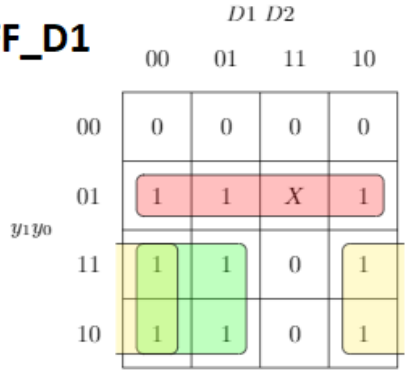
SOLUCIÓN

	qt+1				S1-S2-B			
y1y0	00	01	11	10	00	01	11	10
00 (q0)	01	01	00	01	110	110	111	110
01 (q1)	10	10	X	11	110	110	X	110
10 (q2)	10	10	00	11	010	010	110	110
11 (q3)	11	10	00	11	100	110	110	100

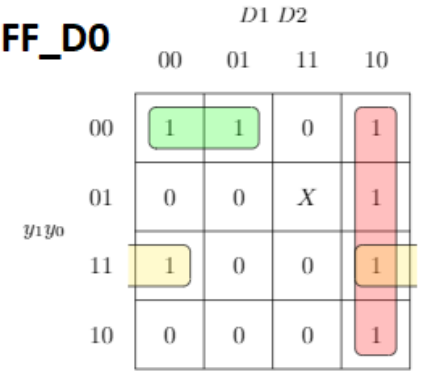
Elijo FF tipo D

Mapas K

FF_D1



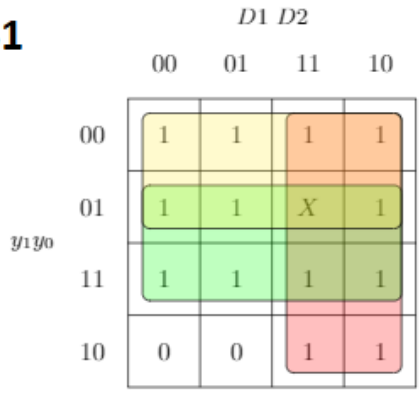
FF_D0



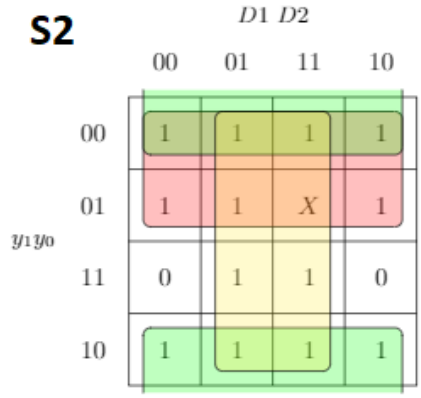
FF_D1 = !y1y0 + !D1y1 + !D2y1

FF_D0 = D1!D2 + !D1!y1!y0 + !D2y1y0

S1



S2



S1 = D1 + !y1 + y0

S2 = D2 + !y0 + !y1

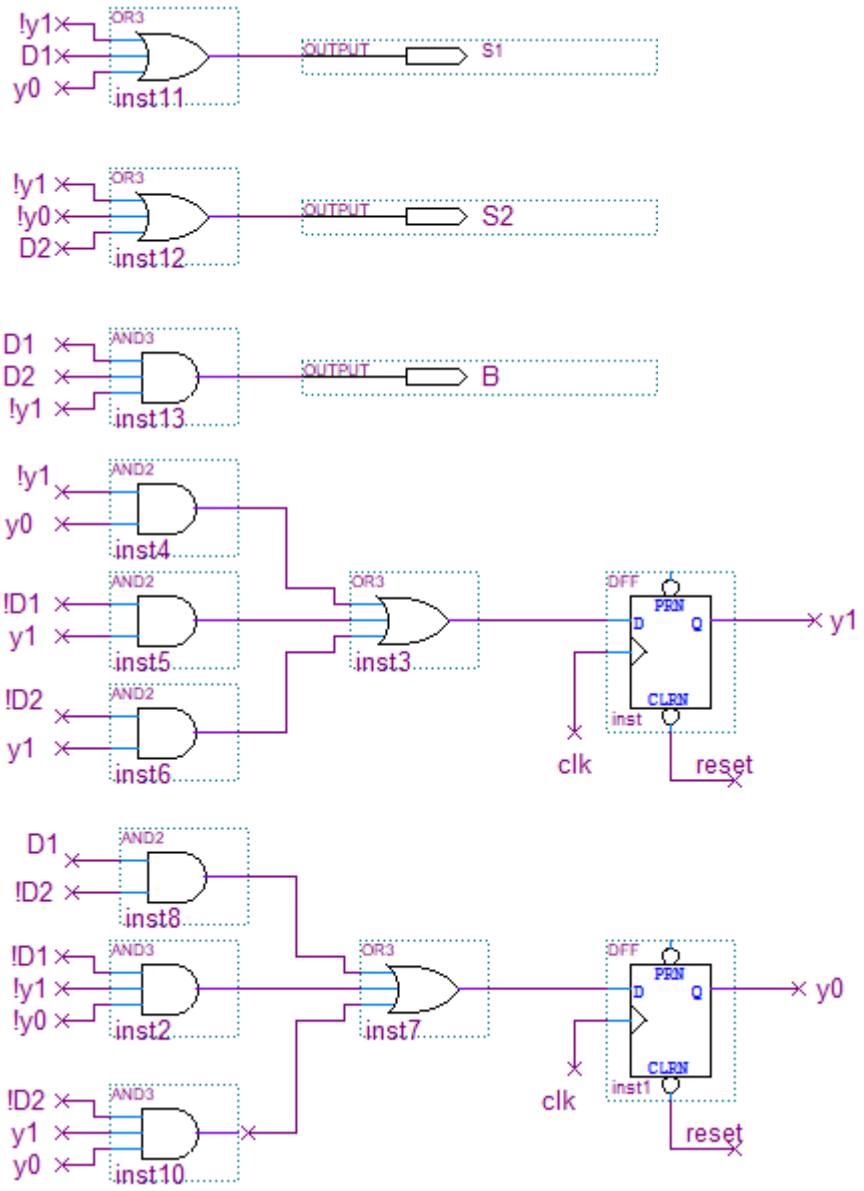
SOLUCIÓN

B

$y_1 y_0$ \ $D1\ D2$	00	01	11	10
00	0	0	1	0
01	0	0	X	0
11	0	0	0	0
10	0	0	0	0

B = D1D2!y1

Circuito



SOLUCIÓN

PROBLEMA 2

SECUENCIA

MODULE: EJ_RTL
INPUTS: data, init
OUTPUTS: out_0[6], out_1[6], out_2[6], out_3[6], ready, blk
MEMORY: Cont[3], Aux, Mem_out_0[6], Mem_out_1[6] , Mem_out_2[6], Mem_out_3[6], Dir[2], Word[5]

0 Mem_out_0[5..0] ←000000
Mem_out_1[5..0] ←000000
Mem_out_2[5..0] ←000000
Mem_out_3[5..0] ←000000

1 Cont[2..0] ← 000
→ (init, /init) / (2, 1)

2 Dir[1..0] ← Dir[0], bit_in
Cont[] ← inc (Cont[])
→ (Cont=001, /Cont=001) / (3, 2)

3 Word[4..0] ← Word[3..0], bit_in
Cont[] ← inc (Cont[])
Mem_out_0[5..0] * /Dir[1]./Dir[0] . (Cont[]=111) . /fin ← Word[4..0], bit_in
Mem_out_1[5..0] * /Dir[1].Dir[0] . (Cont[]=111) ./fin ← Word[4..0], bit_in
Mem_out_2[5..0] * Dir[1]./Dir[0] . (Cont[]=111) ./fin ← Word[4..0], bit_in
Mem_out_3[5..0] * Dir[1].Dir[0] . (Cont[]=111) ./fin ← Word[4..0], bit_in
→ (/Cont[]=111) , (Cont[]=111) . /fin , (Cont[]=110) . fin)/ (3,4,5)

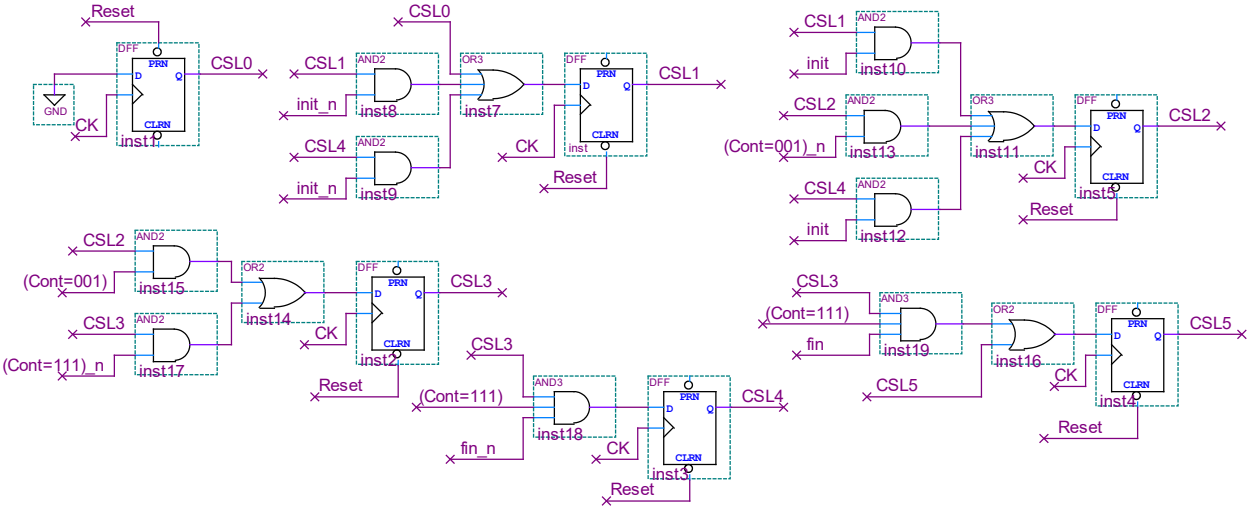
4 ready = 1
→ (init, /init) / (2, 1)

5 blk = 1
→ (5)

END SEQUENCE
CONTROL RESET(0)
bit_in = Aux xor data
Aux ← data
out_0[] = Mem_out_0[]
out_1[] = Mem_out_1[]
out_2[] = Mem_out_2[]
out_3[] = Mem_out_3[]
fin = ((Word[4..0], bit_in) = 111111))
END

SOLUCIÓN

BLOQUE DE CONTROL



BLOQUE DE DATOS