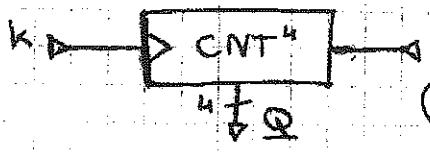
ETUDE D'UN COMPTEUR BINAIRE SYNCHRONE

1. SPEC d'Interface :



(Le comportement est décrit dans le sujet)

2. ANALYSE

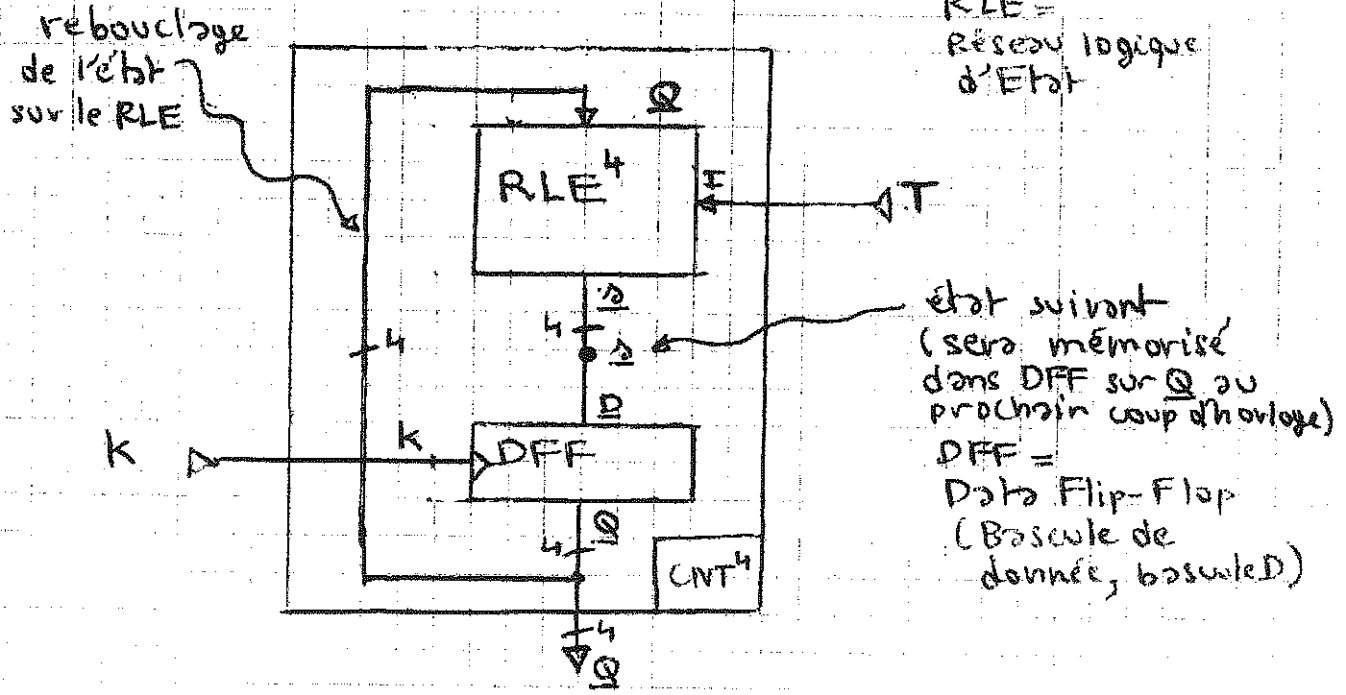
2.1. Décomposition en logique synchronisée (RTL)

Il s'agit d'un opérateur* séquentiel (puisque la sortie dépend de l'historique des entrées)

* synchrone (puisque les changements sont synchronisés par une horloge, front montant ici.)

Nous allons donc tenter de le réaliser avec une machine de Moore simplifiée, c'est à dire un automate dont la sortie est égale au numéro d'état. Le réseau logique de sortie est donc la fonction identité et est donc omis: sortie $\underline{Q} = \underline{S}$ = état.

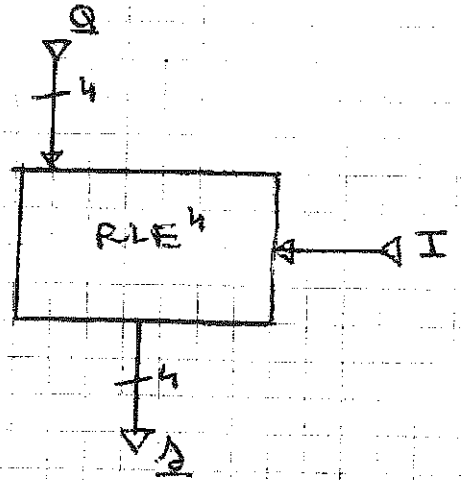
Nous montrons cette logique au niveau de détail ou apparaît la logique synchronisée (par des bascules D), Registered Logic Level (RTL)



Toutes les entrées D, T (données et commandes) vont sur le RLE à sortie de 4 bits. qu'il faut spécifier :

2.2.a/

* interface



(On note que l'entrée de commande T du CNT4 est branchée sur l'entrée I du RLE4 : les noms du RLE4 peuvent être quelconques).

2.2.b/

* comportement

Au coup d'horloge (front montant de R : R↑) la bascule D adjoint l'état suivant Δ et le mémorise sur sa sortie Q

$$k\uparrow \quad \Longmapsto \quad Q \leftarrow \Delta$$

Or, la spécification du comportement indique que par ex. pour T = 1

$$T=1 \quad \Longmapsto \quad Q \leftarrow Q \# 1$$

En identifiant on voit donc que pour T=1 Δ doit être égal à Q#1.

On déduit donc le comportement du RLE4 à partir de celui du CNT4, en remplaçant $Q \leftarrow$ par $\Delta =$

$$k\uparrow \wedge T = 1 \quad \Longmapsto \quad Q \leftarrow Q \# 1$$

règle spécification automate

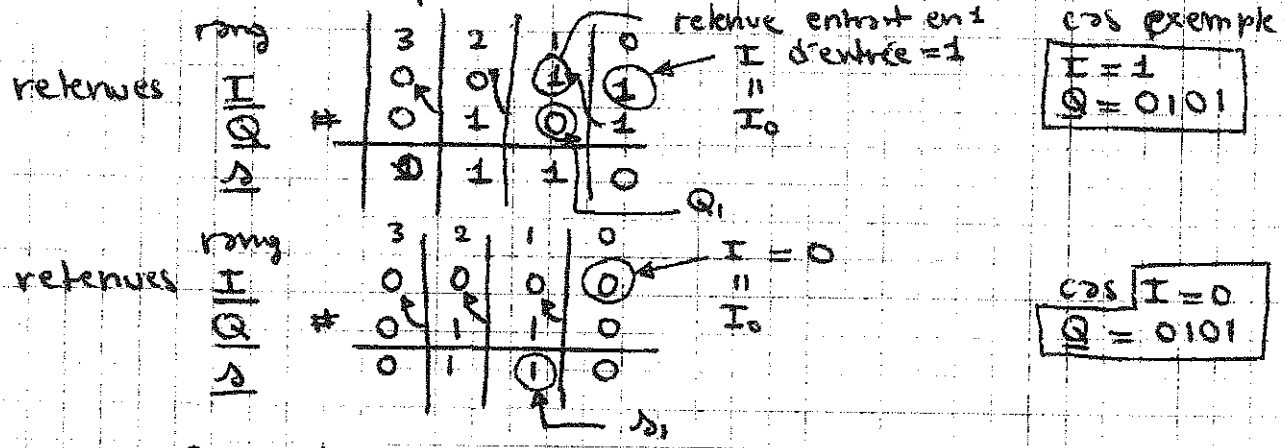
état suivant

$$k\uparrow \wedge I = 1 \quad \Longmapsto \quad \Delta = Q \# 1$$

règle de spéc. RLE

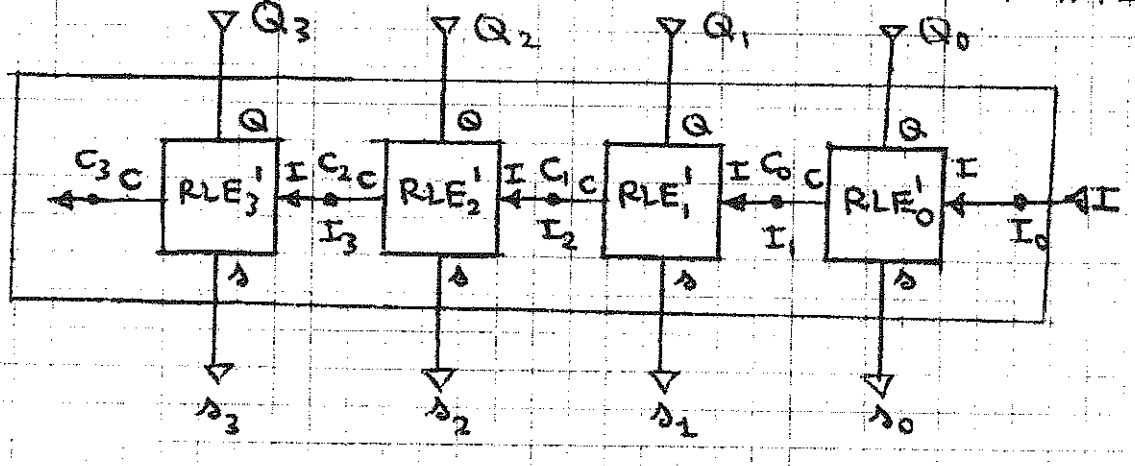
2.3. Structure en tranches

Nous allons donc le décomposer en tranches, comme pour l'additionneur :



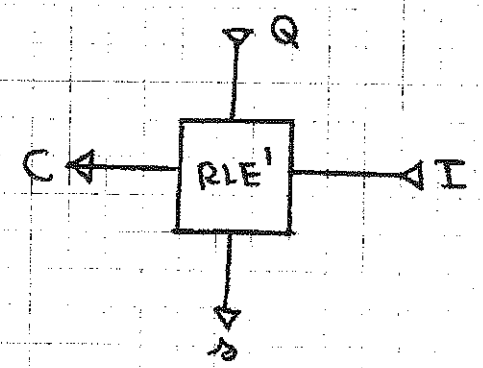
On voit que l'on effectue en réalité l'opération en tranches naturellement, chaque tranche de rang n° n sont :

- le bit de somme S_n,
- le bit de retenue de sortie C_n qui est le bit de retenue d'entrée I_{n+1} de la tranche suivante n° n+1



Du fait que chaque tranche est identique, elle est une instance d'un opérateur RLE' que nous devons maintenant spécifier.

2.4 a/ interface :



2.4b/ Comportement :

$CS = Q \# I$ (# addition)

On va étudier toutes les valeurs possibles de I et Q :

I	0	0	1	1
# Q	# 0	# 1	# 0	# 1
C S	0 0	0 1	0 1	1 0
				↑ retenue

On va établir les tables de vérité (= de Karnaugh ici) pour c et pour s :

c

I \ Q	0	1
0	0	0
1	0	(1)

s

I \ Q	0	1
0	0	(1) ②
1	(1) ①	0

correspond à la table de vérité du ET (AND).

correspond à la table du ou exclusif XOR ⊕

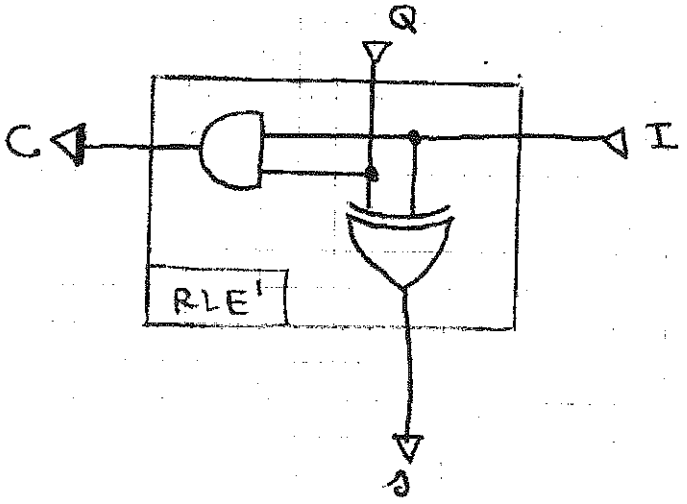
2.5.

$$c = Q \cdot I$$

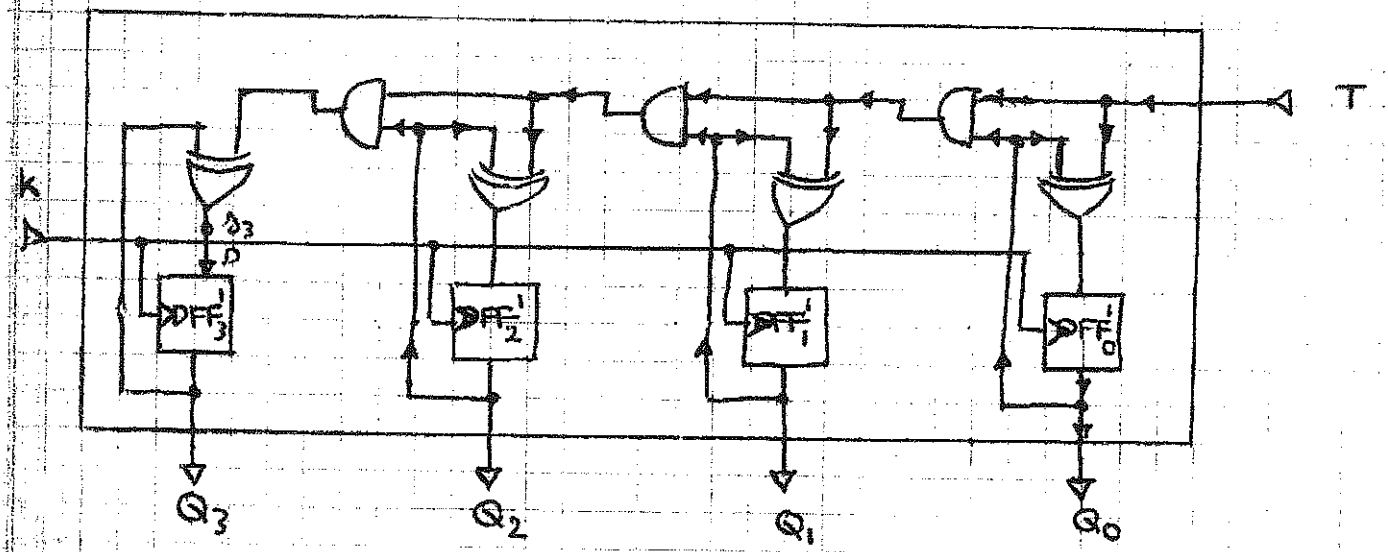
$$s = Q \oplus I = \overbrace{\overline{Q} \cdot I}^{①} + \overbrace{Q \cdot \overline{I}}^{②}$$

3. SYNTHÈSE

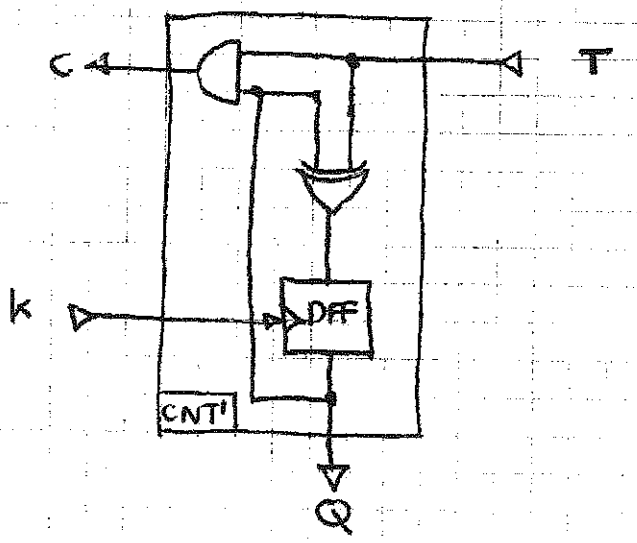
Avec des portes AND et XOR :



Pour bien comprendre, voici le compteur complet dans le détail (avec portes et bascules D). La bascule D à 4 bits DFF' a été décomposée en tranches qui sont des bascules D à 1 bit.



On peut décomposer ce compteur en tranches complètes CNT' comportant la bascule D:



4.

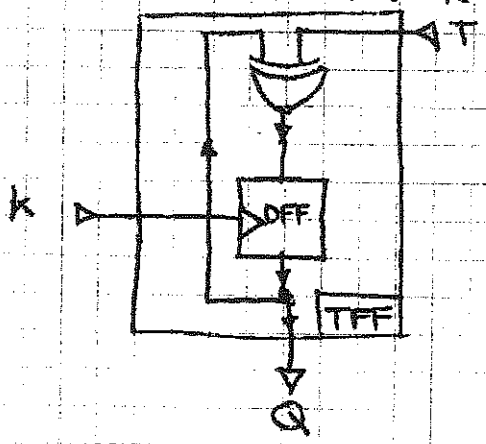
Mais comment garantir l'état du compteur au départ? Et comment repartir?

Il faudrait pouvoir l'initialiser, par exemple à zéro avec une commande de Remise Zéro (Reset) R.

Ce serait bien aussi de pouvoir le recharger avec une valeur précise comme un registre avec une commande de chargement (Load) L, comme un registre.

Note pour l'enseignement: (éviter d'en parler aux étudiants)

Notons que la bascule D et le XOR forment une bascule T ("Toggle Flip-Flop TFF)



$= \begin{cases} \overline{Q} & \text{qd } T=1 \\ Q & \text{qd } T=0 \end{cases}$

qui a donc pour spécification : $k\uparrow \Rightarrow Q \leftarrow \underbrace{Q \oplus T}$ soit :

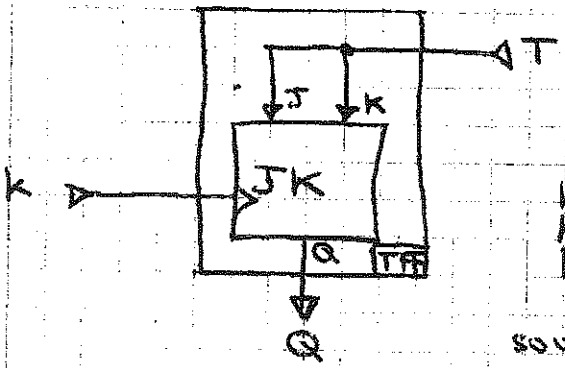
$k\uparrow \wedge T=1 \quad \Rightarrow \quad Q \leftarrow \overline{Q}$ (d'où le nom "Toggle")

(dans l'autre cas où $T=0$, Q ne change pas).

C'est ce qui explique que de nombreux ouvrages mentionnent qu'un compteur est réalisé avec des bascules T.
L'inconvénient est que :

* d'un point de vue pédagogique, je préfère une méthode générale permettant de générer tout automate, donc en utilisant des bascules D, sans en apprendre d'autres.
* les circuits intégrés programmables incorporent pour cette raison des bascules D et non des bascules T (bien que certains ont des bascules configurables en D, L ou T, ou encore des XOR bouclés)

La bascule T étant un cas particulier de la bascule JK quand $J=T$ et $K=T$, certains ouvrages mentionnent aussi ce cas.

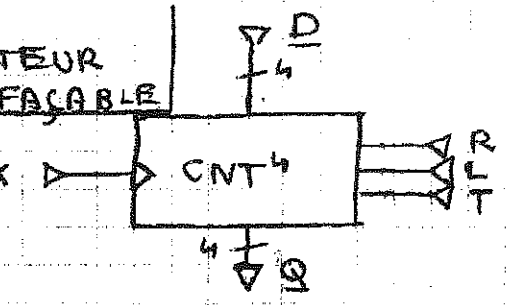


Spécification du JK :

Evénement	Action
$k\uparrow \wedge J=0 \wedge K=1$	$\Rightarrow Q \leftarrow 0$
$k\uparrow \wedge J=1 \wedge K=0$	$\Rightarrow Q \leftarrow 1$
$k\uparrow \wedge J=1 \wedge K=1$	$\Rightarrow Q \leftarrow \overline{Q}$

sous-entendu : pas de changement quand $J=0 \wedge K=0$

ETUDE D'UN COMPTEUR RECHARGEABLE EFFAÇABLE



1. SPEC. D'INTERFACE

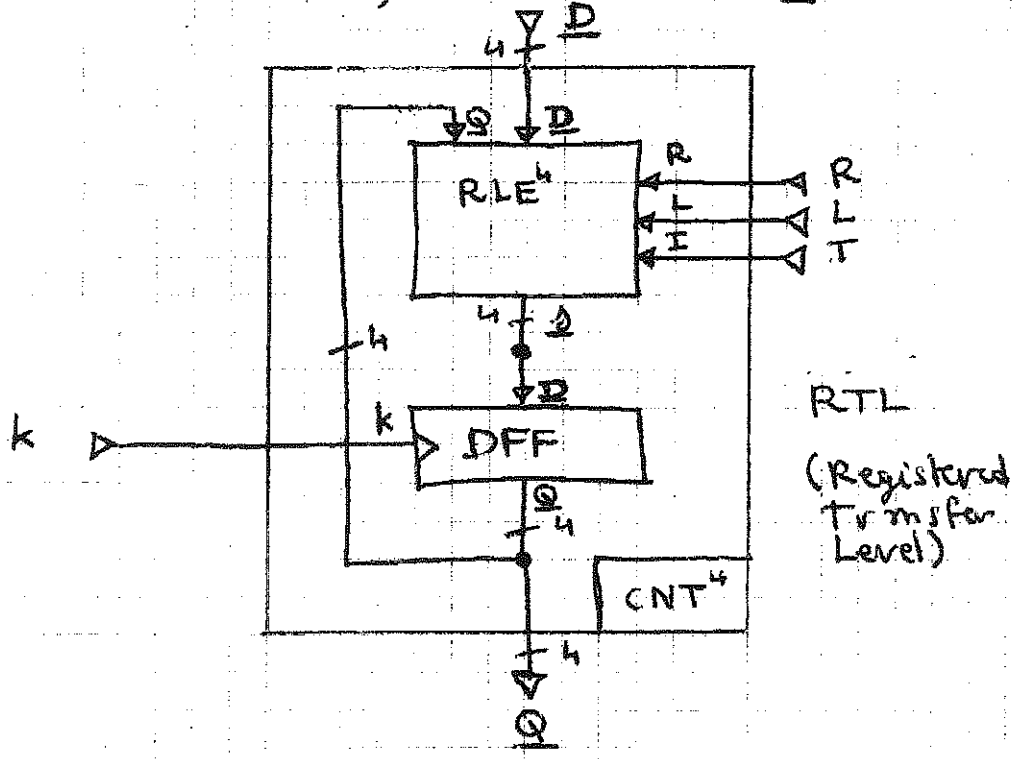
2. ANALYSE

2.1 Décomposition RTL

On va décomposer structurellement le compteur.
Il s'agit d'un opérateur séquentiel (puisque la sortie dépend de l'historique des entrées et non pas seulement de l'entrée à l'instant t)

* synchrone (car les changements sont synchronisés avec le front montant de l'horloge ici).

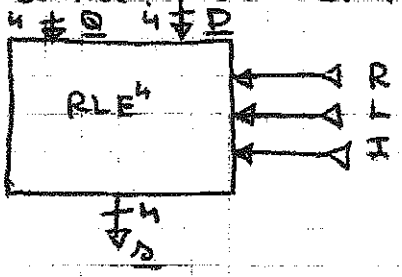
Nous allons donc tenter de le réaliser avec une machine de Moore simplifiée, c'est à dire un automate dont la sortie est égale au numéro d'état interne. Le réseau logique de sortie est donc la fonction identité, il est donc omis, donc la sortie Q = S état.



Les entrées D, R, L, T vont toutes sur le Réseau Logique d'Etat qui calcule l'état suivant S en fonction de l'état actuel S = Q.

La bascule DFF est connue. En revanche, il faut spécifier RLE4 :

2.2 Interface :



2.2 b comportement

Au coup d'horloge (front montant de h : h↑) la bascule D (Data Flip-Flop) acquiert l'état suivant $\underline{\Delta}$ et le mémorise sur sa sortie $\underline{Q}$:

$$h\uparrow \Longmapsto \underline{Q} \leftarrow \underline{\Delta}$$

Or, la spécification du comportement indique que par exemple pour $R=0 \wedge L=1$, on veut que :

$$R=0 \wedge L=1 \wedge h\uparrow \Longmapsto \underline{Q} \leftarrow \underline{D}$$

En identifiant, on voit donc que l'on doit avoir $\underline{\Delta} = \underline{D}$ pour $R=0 \wedge L=1$

On en déduit le comportement du RLE à partir de celui du compteur complet CNT :

$$\begin{cases} R=1 \Longmapsto \underline{\Delta} = \underline{\emptyset} & (1) \\ R=0 \wedge L=1 \Longmapsto \underline{\Delta} = \underline{D} & (2) \\ R=0 \wedge L=0 \wedge I=1 \Longmapsto \underline{\Delta} = \underline{Q} * 1 & (3)\ ** \\ \text{autres cas} : \Longmapsto \underline{\Delta} = \underline{Q} & (4)\ * \text{ (sous-entendu)} \end{cases}$$

* Le seul autre cas est ici : $R=0 \wedge L=0 \wedge I=0$;

On veut alors (mais c'est sous-entendu) que $\underline{Q}$ ne change pas, même au coup d'horloge, ce qui signifie alors :

$$R=0 \wedge L=0 \wedge I=0 \wedge h\uparrow \Longmapsto \underline{Q} \leftarrow \underline{Q}$$

et donc $\underline{\Delta}$ doit être égal à $\underline{Q}$. Donc (4) devient :

$$R=0 \wedge L=0 \wedge I=0 \Longmapsto \underline{\Delta} = \underline{Q} \quad (5)$$

** Notons aussi l'astuce habituelle pour $R=0$ et $L=0$, on a finalement pour (3) et (5)

$$R=0 \wedge L=0 \wedge \begin{cases} I=0 \Rightarrow \underline{\Delta} = \underline{Q} = \underline{Q} * 0 & (3) \\ I=1 \Rightarrow \underline{\Delta} = \underline{Q} * 1 & (5) \end{cases}$$

On peut donc poser que pour $R=0$ et $L=0$: $\underline{\Delta} = \underline{Q} * I$ (6)

↑ Incrém.

On a donc finalement :

$$\begin{cases} R=1 \Longmapsto \underline{D} = \underline{\emptyset} & (1) \\ R=0 \wedge L=1 \Longmapsto \underline{S} = \underline{D} & (2) \\ R=0 \wedge L=0 \Longmapsto \underline{S} = \underline{Q} \# \underline{I} & (6) \end{cases}$$

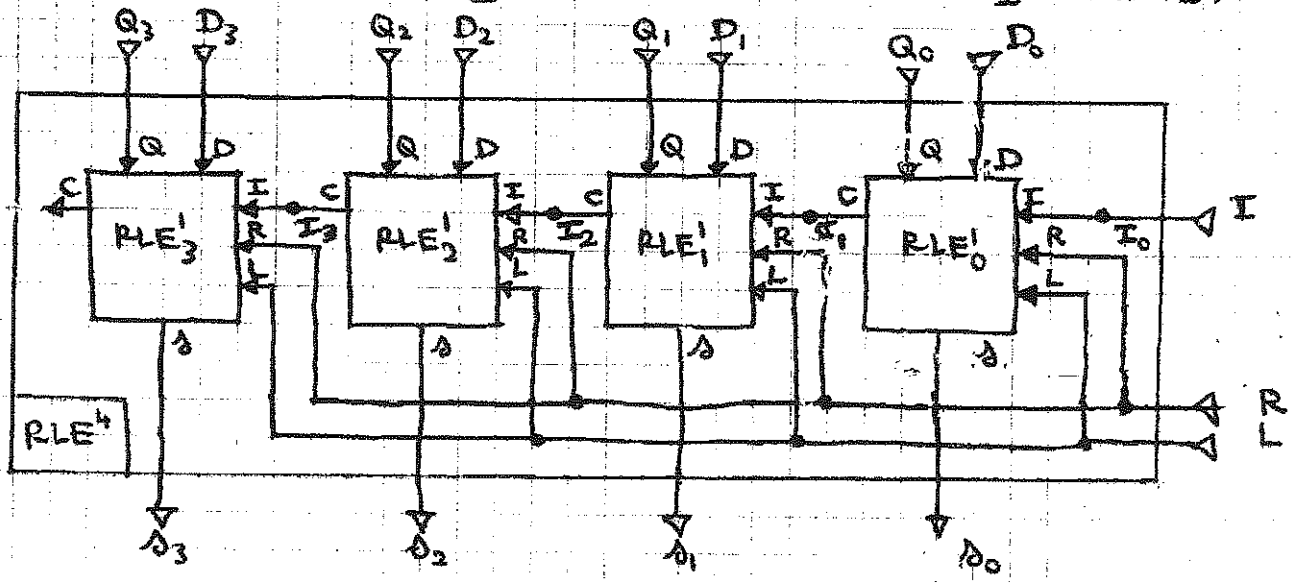
On peut recopier tableau comme habituellement : (Valeurs de $\underline{S}$ pour chaque valeur de RL)

	RL	$\underline{S}$
(6)	0 0	$\underline{Q} \# \underline{I}$
(2)	0 1	$\underline{D}$
(1)	1 1	$\underline{\emptyset}$
	1 0	$\underline{\emptyset}$

2.3. Décomposition en tranches du RLE

Le RLE ressemble à une petite ALU que nous allons décomposer en tranches, chaque tranche fournissant un bit de sortie S_n.

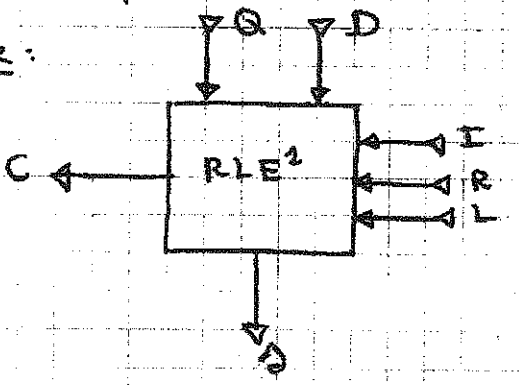
Du fait que $\underline{S} = \underline{Q} \# \underline{I}$ pour RL = 00, il faudra penser à faire propager les retenues, en s'inspirant d'un additionneur pour lequel $\underline{S} = \underline{A} \# \underline{B} \# \underline{I}$ (ici $\underline{B} = \underline{\emptyset}$ et $\underline{A} = \underline{Q}$)



On a donc du ... utiliser 4 exemplaires RLE_0^1, RLE_1^1, RLE_2^1, RLE_3^1 d'un modèle de RLE à 1 bit RLE^1 (on l'a donc instancié 4 fois). Le schéma synoptique indique comment par exemple l'entrée D de RLE_2^1 est connectée à l'entrée D_2 du RLE^4.

Il faut donc spécifier ce nouvel opérateur RLE^1 :

2.4a interface :



2.4b Comportement :

On déduit du comportement du RLE^4 le comportement de RLE^1.

En particulier, pour RL = 00, on voit que

$CS = Q \# I$.

— Effectuons cette addition pour toutes les valeurs de Q et I :

I	0	0	1	1
Q	# 0	# 1	# 0	# 1
CS	00	01	01	10

↑ retenue

on voit que :

S :

Q \ I	0	1
0	0	1
1	1	0

c'est la table de vérité du XOR

⇒ $S = Q \oplus I$

C

Q \ I	0	1
0	0	0
1	0	1

c'est la table de vérité du AND

⇒ $S = Q \cdot I$

— On peut aussi le déduire de l'expression algébrique d'un additionneur complet : $CS = A \# B \# I$

On sait que chaque tranche est telle que :

$$\begin{cases} S = A \oplus B \oplus I \\ C = A \cdot B + A \cdot I + B \cdot I \end{cases}$$

On veut que : $\Delta = Q \# I$.

En identifiant :

RLE'	ADD'
Δ	= S
A	= Q
B	= 0
I	= I

On déduit :

$$\begin{cases} \Delta = Q \oplus 0 \oplus I = Q \oplus I \\ C = Q \cdot 0 + Q \cdot I + 0 \cdot I = Q \cdot I \end{cases}$$

On a donc le tableau :

RL	C (sortie)	Δ
0 0	Q·I	Q⊕I
0 1	—	D
1 1	—	0
1 0	—	0

valeur de commande → (0 1)

expression algébrique → (Q⊕I)

chaque case indique la valeur de la sortie de la colonne par une expression algébrique pour la valeur de commande de la ligne.

2.5 Polynômes booléens réduits des sorties du RLE'

C : pour RL ≠ 00, C ne sert à rien et peut donc prendre n'importe quelle valeur.

Or le plus simple est de toujours avoir Q.I.

Donc : $C = Q \cdot I \quad \forall RL \in \mathbb{B}^2$

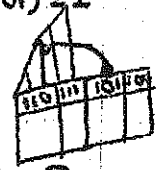
Δ : — Pour Δ, on va remplir la table de Karnaugh en s'aidant de la table de comportement ou directement à partir des relations $CS = Q \# I$.

Une table de Karnaugh est aussi une spécification de comportement.

Table de Karnaugh de s :

RL \ DQI	000	001	011	010	110	111	101	100
00	0	① 1	0	① 1	② 1	0	① 1	0
01	0	0	0	0	③ 1	1	1	1
11	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0

Pour le monôme ①, on note que dist(001, 101) = 1 et que donc ces cases doivent être considérées contiguës (Le tableau est replié autour de la médiane verticale)



De cette table il vient :

$$s = \overbrace{\overline{R}\cdot\overline{L}\cdot\overline{Q}\cdot I}^{①} + \overbrace{\overline{R}\cdot\overline{L}\cdot Q\cdot\overline{I}}^{②} + \underbrace{\overline{R}\cdot L\cdot D}_{③}$$

Il y a un risque d'aléa pour les transitions RL DQI

$0 \to 110$ et $0 \to 101$ (Flèches sur la table)

mais c'est sans importance puisque la machine est synchrone (on attend que s soit stabilisé avant d'envoyer le coup d'horloge).

— On peut aussi utiliser une méthode algébrique à partir du tableau de comportement, mais cette méthode ne permet pas toujours une bonne réduction :

pour RL = 00, le résultat est $Q \oplus I$
RL = 01, le résultat est D
pour les autres cas on a $\emptyset$.

Donc : $s = \overline{R}\cdot\overline{L}\cdot(Q \oplus I) + \overline{R}\cdot L\cdot D$

on voit bien que pour RL = 00, $\overline{R}\cdot\overline{L} = 1$ et les autres monômes $\emptyset$. Alors, $s = 1\cdot(Q \oplus I) = Q \oplus I$.
Pour RL = 01, $s = 1\cdot D = D$.

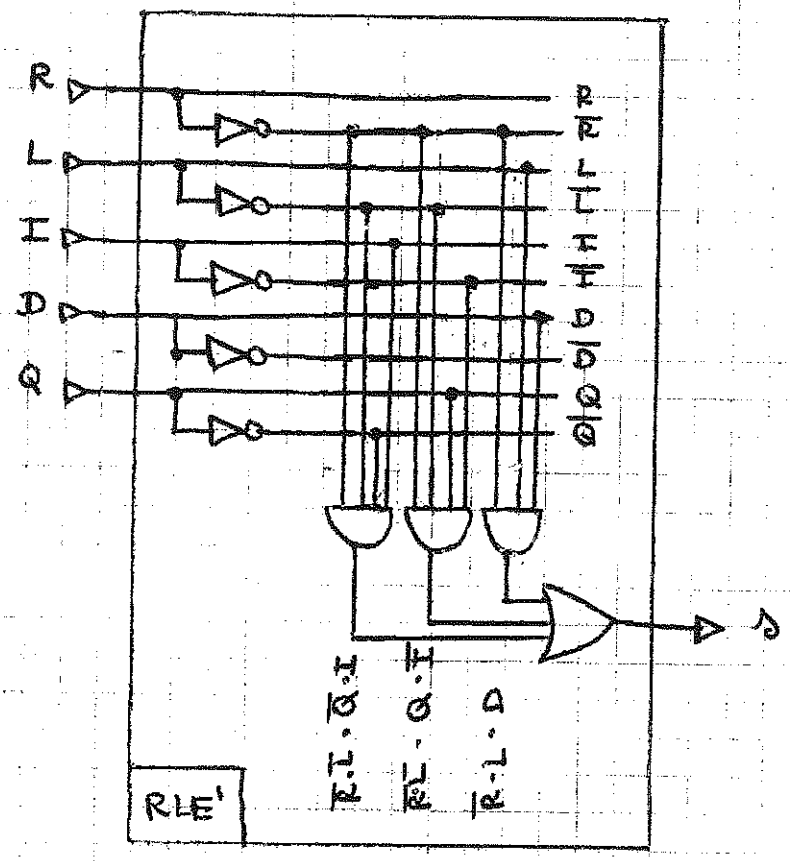
On peut développer $Q \oplus I = \overline{Q}\cdot I + Q\cdot\overline{I}$

$$\Rightarrow s = \overline{R}\cdot\overline{L}\cdot(\overline{Q}\cdot I + Q\cdot\overline{I}) + \overline{R}\cdot L\cdot D$$

$$= \overline{R}\cdot\overline{L}\cdot\overline{Q}\cdot I + \overline{R}\cdot\overline{L}\cdot Q\cdot\overline{I} + \overline{R}\cdot L\cdot D.$$

3. SYNTHESE

3.1 Synthèse avec un réseau logique ordonné de NOT, AND, OR :



3.2 Synthèse avec NANDs

On peut évidemment faire une synthèse avec des NANDs en remplaçant les AND et OR par des NANDs.

En effet :

$$S = \overline{\overline{S}} = \overline{\overline{\overline{R}\cdot\overline{L}\cdot\overline{Q}\cdot\overline{I} + \overline{R}\cdot\overline{L}\cdot Q\cdot\overline{I} + \overline{R}\cdot L\cdot D}} =$$

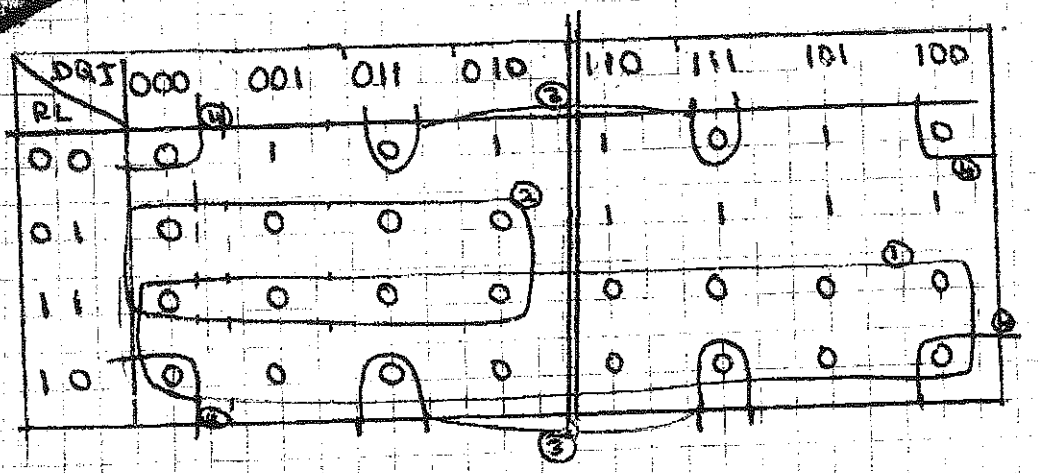
$$= \overline{\overline{\overline{R}\cdot\overline{L}\cdot\overline{Q}\cdot\overline{I}} \cdot \overline{\overline{R}\cdot\overline{L}\cdot Q\cdot\overline{I}} \cdot \overline{\overline{R}\cdot L\cdot D}} =$$

$$= (\overline{R}\uparrow\overline{L}\uparrow\overline{Q}\uparrow\overline{I}) \uparrow (\overline{R}\uparrow\overline{L}\uparrow Q\uparrow\overline{I}) \uparrow (\overline{R}\uparrow L\uparrow D)$$

3.3 Synthèse avec NORs

Avec des NORs, il faudrait partir de

$\overline{S}$ = polynôme(R, L, I, Q, D)



$$\overline{D} = \underbrace{R}_{①} + \underbrace{L\cdot\overline{D}}_{②} + \underbrace{\overline{L}\cdot Q\cdot I}_{③} + \underbrace{\overline{L}\cdot\overline{Q}\cdot\overline{I}}_{④}$$

$$\delta = \overline{\overline{D}} = R + L\cdot\overline{D} + \overline{L}\cdot Q\cdot I + \overline{L}\cdot\overline{Q}\cdot\overline{I}$$

$$= \overline{R + \overline{\overline{L}+D} + \overline{L+\overline{Q}+\overline{I}} + \overline{L+Q+\overline{I}}}$$

$$= R \downarrow (\overline{L}\downarrow D) \downarrow (L\downarrow\overline{Q}\downarrow\overline{I}) \downarrow (L\downarrow Q\downarrow I)$$

$\downarrow$ = symbole de Shaeffer du NOR
(utilisé en Logique formelle, comme "ou" $\equiv \vee$)

$\uparrow$ = symbole de Shaeffer du NAND
(comme "et" barré ... avec "et" $= \wedge$)