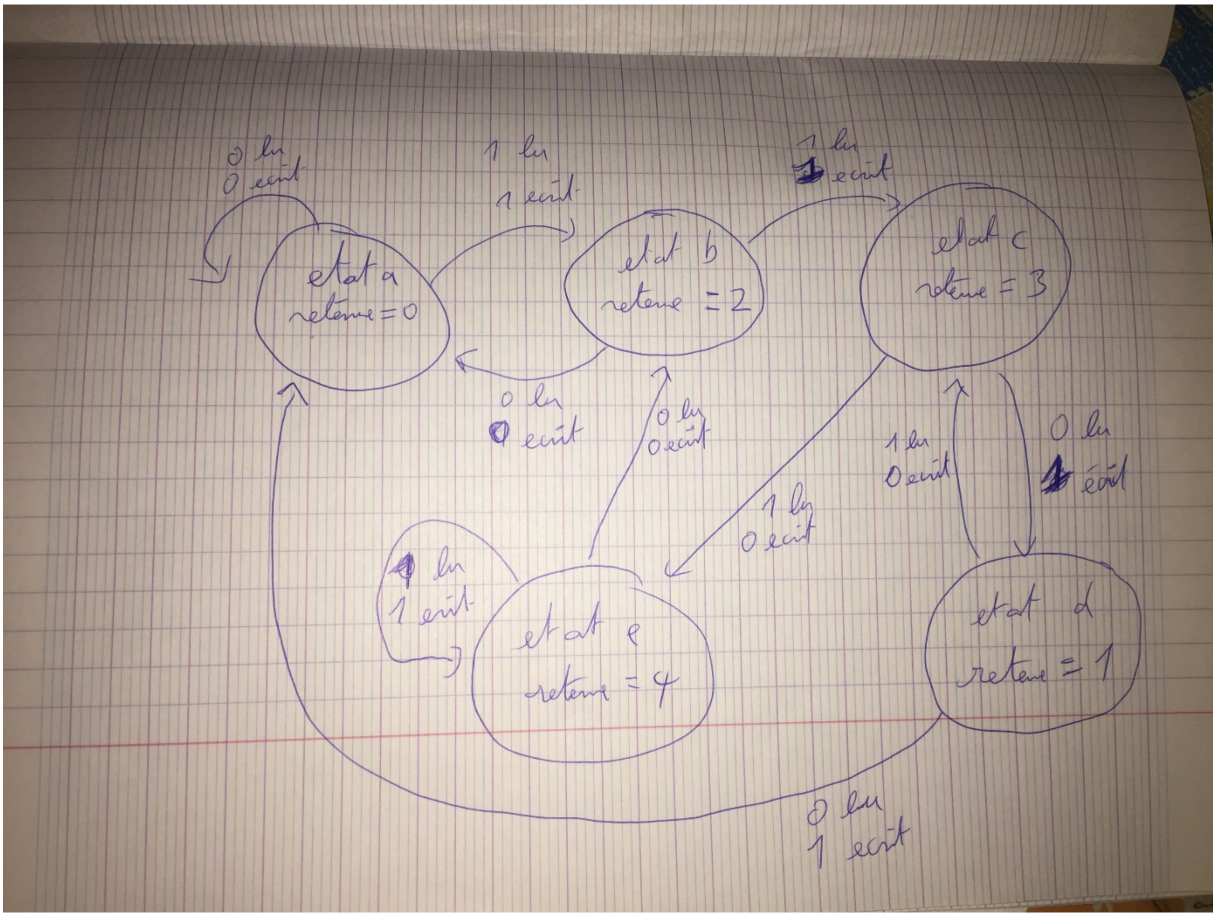
1.Voici donc le graph et la table de verité de cet automate

Etat Courant	e2	e1	e0	input	Etat Suivant	n2	n1	n0	output
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	2	0	1	0	1
1	0	0	1	0	0	0	0	0	1
1	0	0	1	1	3	0	1	1	0
2	0	1	0	0	0	0	0	0	0
2	0	1	0	1	3	0	1	1	1
3	0	1	1	0	1	0	0	1	1
3	0	1	1	1	4	1	0	0	0
4	1	0	0	0	2	0	1	0	0
4	1	0	0	1	4	1	0	0	1
5	1	0	1	0					
5	1	0	1	1					
6	1	1	0	0					
6	1	1	0	1					
7	1	1	1	0					
7	1	1	1	1					

Entrée	Retenue	Resultat(10)	Resultat(2)	Bit produit	Retenue(2)	Retenue(10)
0	0	0	0	0	0	0
1	0	5	101	1	10	2
0	1	1	1	1	0	0
1	1	6	110	0	11	3
0	2	2	10	0	0	0
1	2	7	111	1	11	3
0	3	3	11	1	1	1
1	3	8	1000	0	100	4
0	4	4	100	0	10	2
1	4	9	1001	1	100	4



3. Grâce au tableau on peut établir d’abord une équation évidente bits de sortie

F(Output) = i.~e2.~e1~.~e0 + ~i.~e2.~e1~.e0 + ~i.e2.e1~.e0 + ~i.~e2.e1.e0 + i.e2.~e1~.e0

Un peu barbare et confus avec des eN qui traine, donc pour la suite on va prendre la notation suivante :

i = a
e2 = b
e1 = c
e0 = d

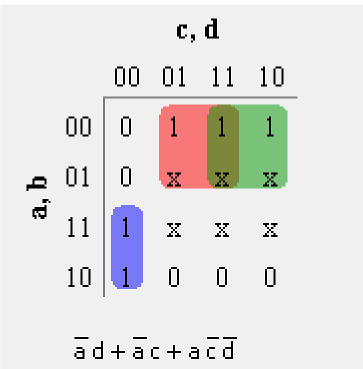
Ce qui nous donne :

F(output) = a.~b.~c.~d + ~a.~b.~c.d + a.~b.c.d + ~a.~b.c.d + a.b.~c.~d

De plus on peut aussi faire l’équation des états 5 6 et 7,

5 = b.~c.d
6 = b.c.~d
7 = b.c.d

Grâce logisim, on dessiner facilement le tableau de Karnaugh où x représente les cases d’états non utilisées.



Bon je triche un peu comme il manque la partie raisonnement avec logisim, un peu d’explication pour me justifier…
Pour la partie bleue, nous avons a.~b.~c.~d + a.b.~c.~d, on voit bien que b n’est pas une condition nécessaire, donc on obtient a.~c.~d

Pour la partie rouge, nous avons d’abord les cas ~a.~b.~c.d + ~a.~b.c.d mais on peut aussi choisir d’intégrer les cas inutilisées que sont ~a.b.~c.d + ~a.b.c.d

F(rouge)= ~a.~b.~c.d + ~a.~b.c.d + ~a.b.~c.d + ~a.b.c.d

b n’est donc pas nécessaire

= ~a.~c.d + ~a.c.d + ~a.~c.d + ~a.c.d

= ~a.~c.d + ~a.c.d

c est aussi inutile donc nous avons au finale ~a.d

Pour la partie verte nous avons

F(verte)= ~a.~b.c.d +~a.~b.c.~d + ~a.b.c.d +~a.b.c.~d

b n'est pas nécessaire

= ~a.c.d +~a.c.~d + ~a.c.d +~a.c.~d

=~a.c.d +~a.c.~d

Même raisonnement pour d

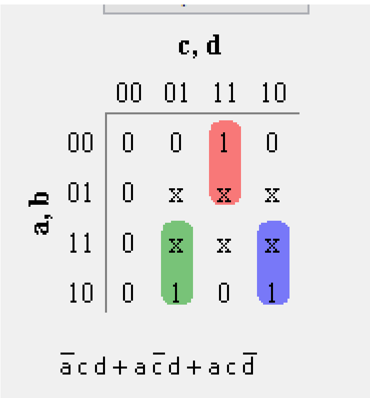
= ~a.c

F(Out)= a.~c.~d + ~a.d+ ~a.c

Passons au cas n0

e2	e1	e0	input	n0
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1

F(n0)= a.~b.~c.d + a.~b.c.~d + ~a.~b.c.d



on inclue les cas inutilisées où b est vrai pour éliminer b.

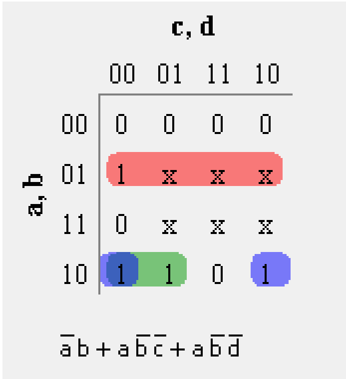
F(n0)= a.~b.~c.d + a.~b.c.~d + ~a.~b.c.d

F(n0)= a.~b.~c.d + a.~b.c.~d + ~a.~b.c.d +a.b.~c.d + a.b.c.~d + ~a.b.c.d

F(n0)= a.~c.d+ a.c.~d +~a.c.d

Passons maintenant au cas n1

e2	e1	e0	input	n1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	
1	0	1	1	



F(n1)= a.~b.~c.~d + a.~b.~c.d + a.~b.c.~d + ~a.b.~c.~d

1[er] terme a un terme complémentaire avec le 2eme et 3eme terme, on peut donc éliminer d et c

F(n1)= (a.~b.~c.~d + a.~b.~c.d) + (a.~b.~c.~d+ a.~b.c.~d)+ ~a.b.~c.~d

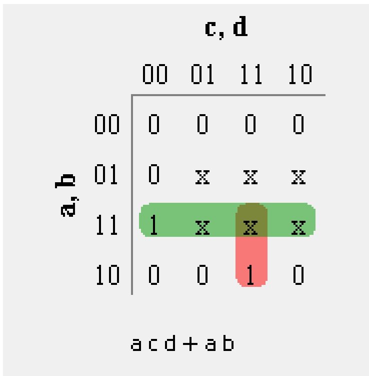
F(n1)= a.~b.~c + a.~b.~d + ~a.b.~c.~d

enfin on introduit les termes inutilisés

F(n1)= a.~b.~c + a.~b.~d +(~a.b.~c.~d + ~a.b.~c.d+ ~a.b.~c.~d+ ~a.b.c.~d)

d'où :

F(n1)= a.~b.~c + a.~b.~d + ~a.b



e2	e1	e0	input	n2
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1

F(n2)=a.~b.c.d+ a.b.~c.~d

F(n2)= (a.~b.c.d+ a.b.c.d) + (a.b.~c.~d + a.b.~c.d +a.b.c.d +a.b.c.~d)

F(n2)= a.c.d + a.b

F(Out)= a.~c.~d + ~a.d+ ~a.c

F(n2)= a.c.d + a.b

F(n1)= a.~b.~c + a.~b.~d + ~a.b

F(n0)= a.~c.d+ a.c.~d +~a.c.d

Le circuit en question est dans le fichier exo1.circ, en esperant que je ne suis pas trompé au niveau des entrée/sortie de la bascule D…

2.Chenillard

>>Pas entrée

>>4 sortie abcd

peut etre 6 etats ? et donc 3 bits ?

Bref j'ai pas compris l'énoncé..