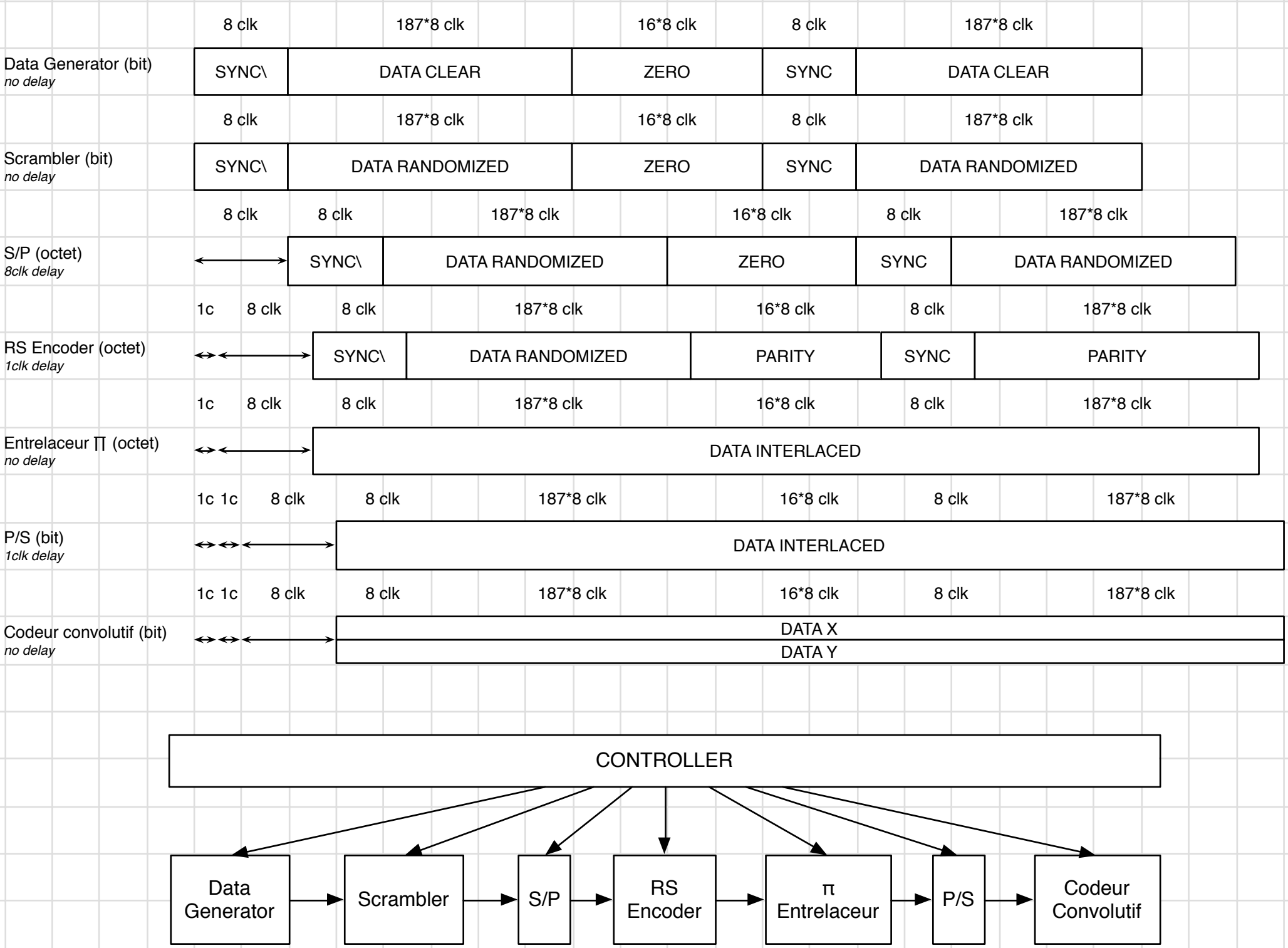
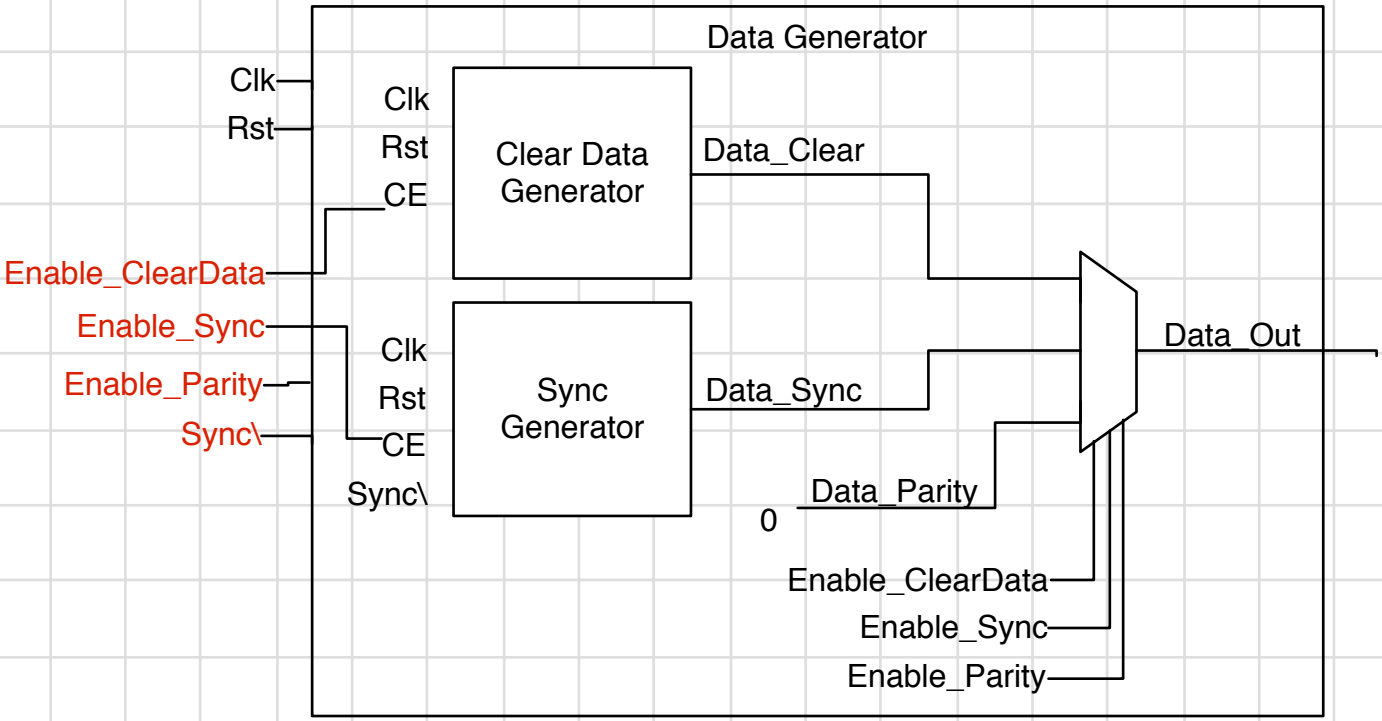
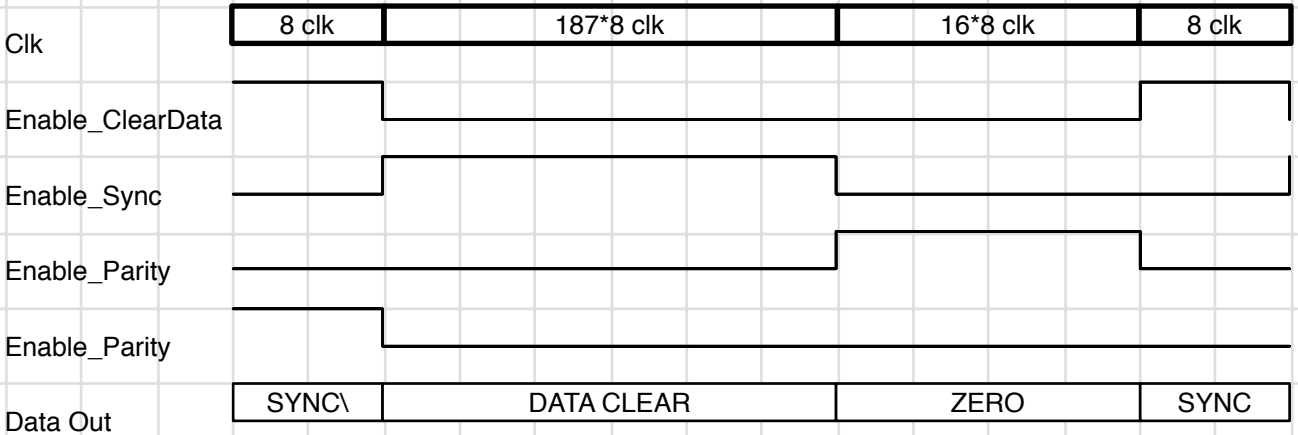
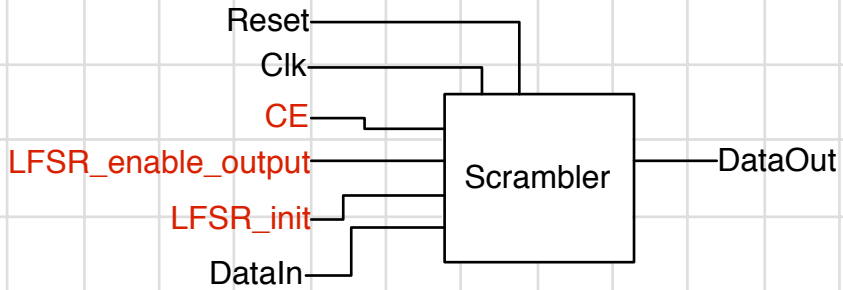

8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
Data Generator (bit)
no delay
SYNC\
DATA CLEAR
ZERO
SYNC
DATA CLEAR
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
Scrambler (bit)
no delay
SYNC\
DATA RANDOMIZED
ZERO
SYNC
DATA RANDOMIZED
8 clk
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
S/P (octet)
8clk delay
SYNC\
DATA RANDOMIZED
ZERO
SYNC
DATA RANDOMIZED
1c
8 clk
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
RS Encoder (octet)
1clk delay
SYNC\
DATA RANDOMIZED
PARITY
SYNC
PARITY
1c
8 clk
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
Entrelaceur ∏ (octet)
no delay
DATA INTERLACED
1c 1c
8 clk
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
P/S (bit)
1clk delay
DATA INTERLACED
1c 1c
8 clk
8 clk
187*8 clk
16*8 clk
8 clk
187*8 clk
Codeur convolutif (bit)
no delay
DATA X
DATA Y
CONTROLLER
Data Generator
Scrambler
S/P
RS Encoder
π Entrelaceur
P/S
Codeur Convolutif



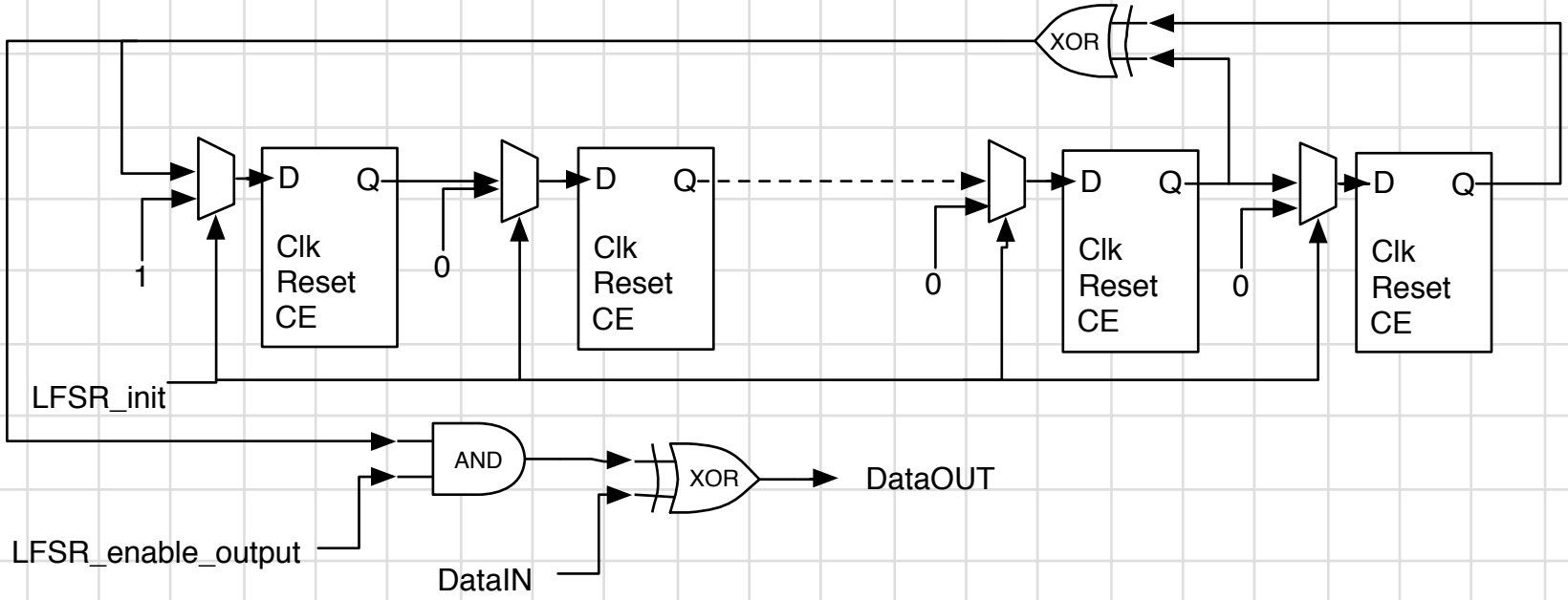
Principe : Générer les données pré-formatées pour le reste de la chaine
Les « Enable » permettent de choisir quel type de donnée est générée
Fonctionne au bit

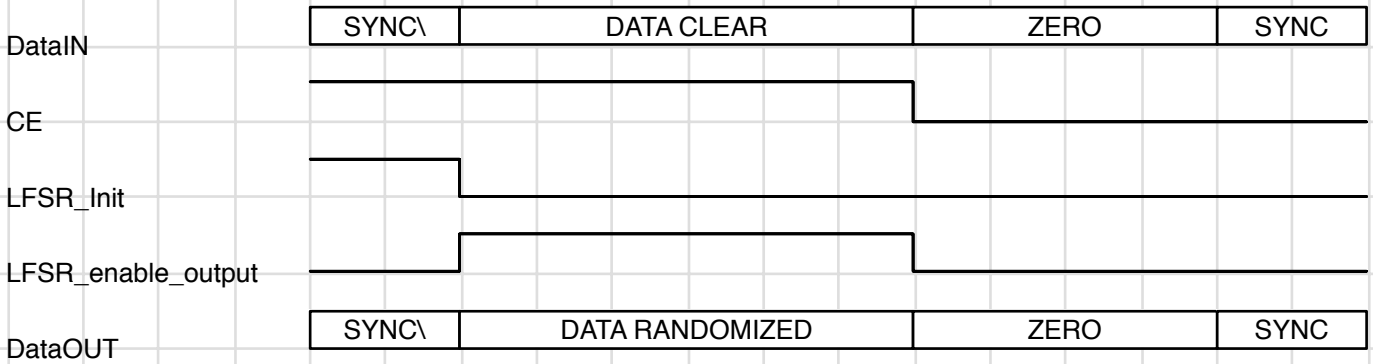


Rouge = provenance du controleur

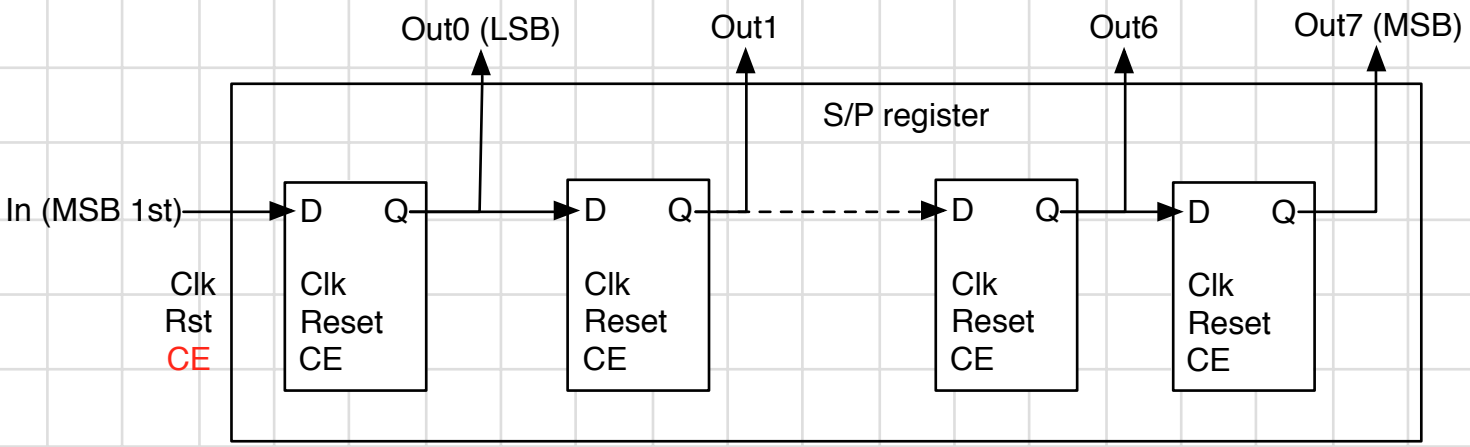


Principe : Une valeur pseudo aléatoire est générée par les bascules. Un XOR est effectué avec la data (et seulement elle). Puis lorsque *CE* et *LFSR_init* sont à '1' (pendant SYNC\ uniquement) une valeur d'init est chargée dans chaque bascule.





Rouge = provenance du controleur



		MSB							LSB								
In	X	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	X
Out0 (LSB)	X	X	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
Out1	X	X	X	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1
Out2	X	X	X	X	1	0	0	1	0	1	1	0	1	0	0	1	0	1
Out3	X	X	X	X	X	1	0	0	1	0	1	1	0	1	0	0	1	0
Out4	X	X	X	X	X	X	1	0	0	1	0	1	1	0	1	0	0	1
Out5	X	X	X	X	X	X	X	1	0	0	1	0	1	1	0	1	0	0
Out6	X	X	X	X	X	X	X	X	1	0	0	1	0	1	1	0	1	0
Out7 (MSB)	X	X	X	X	X	X	X	X	X	1	0	0	1	0	1	1	0	1

Donnée NOK

Donnée non exploitable pendant 7 clk

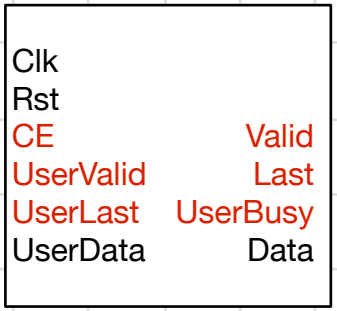
Donnée OK

Donnée non exploitable pendant 7 clk

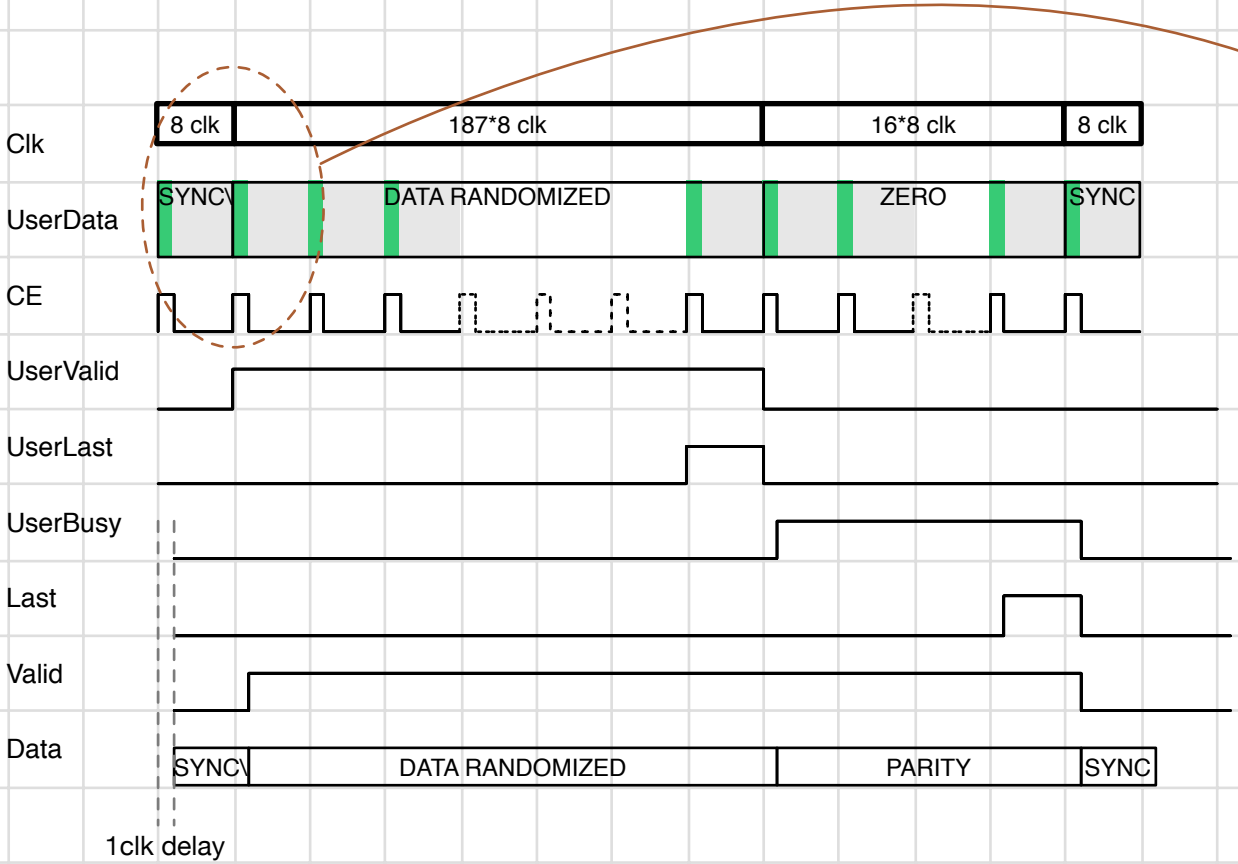
Donnée OK

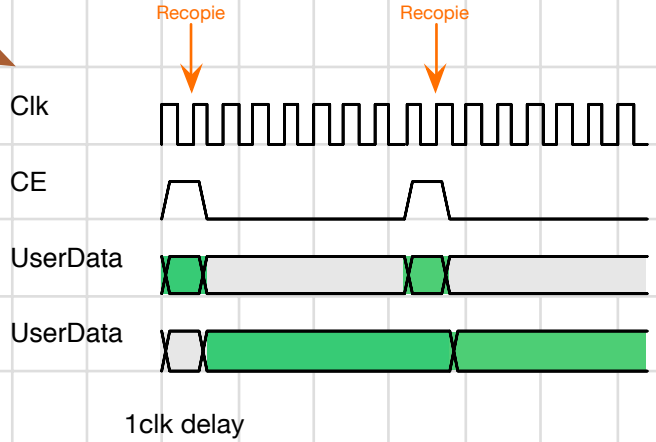
Au démarrage/reset, 1er octet non exploitable = 8 clk de latence

Rouge = provenance du controleur



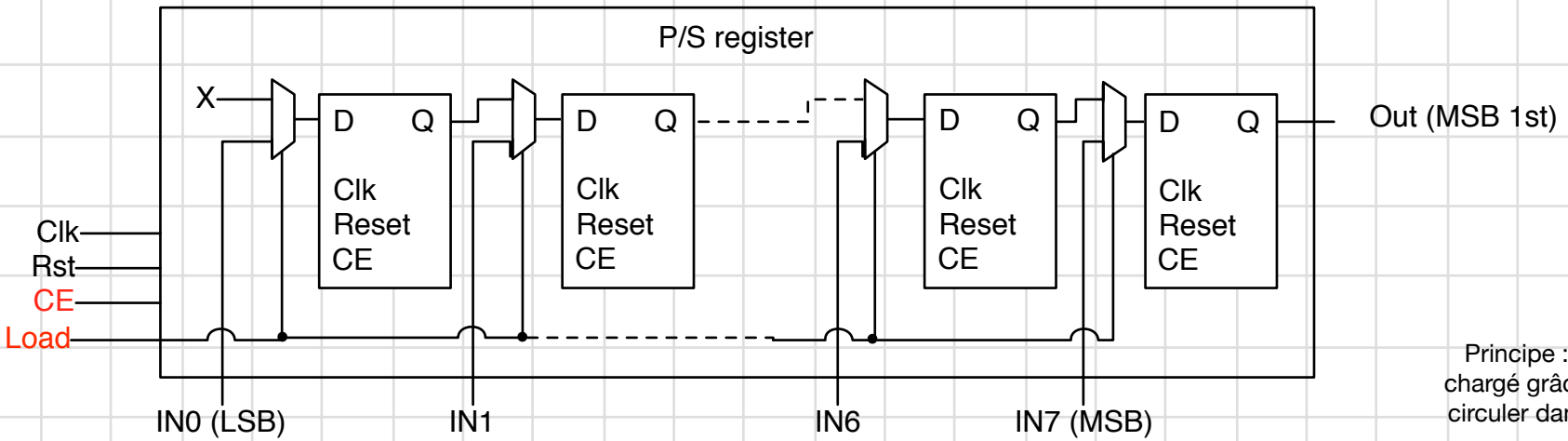
Principe : la pariée est calculée sur la data (UserValid) puis est insérée dans le signal après la data (UserLast)
Le circuit ne doit fonctionner que sur les octets valides (verts) : ajout d'un CE à '1' sur 1clk sur 8
UserValid est positionné à '1' pendant la data pour le calcul de la parité
UserLast est positionné à '1' pendant le dernier octet de data
Circuit séquentiel : 1clk de retard
Le circuit doit aussi fonctionner pendant les sync (sans UserValid ni UserLast) pour recopie entrée sur sortie
Les signaux de sorties sont envoyés au controler
À cause des 8clk de latence du s/p, UserValid, UserLast et CE doivent (au reset) débuter avec 8clk de retard



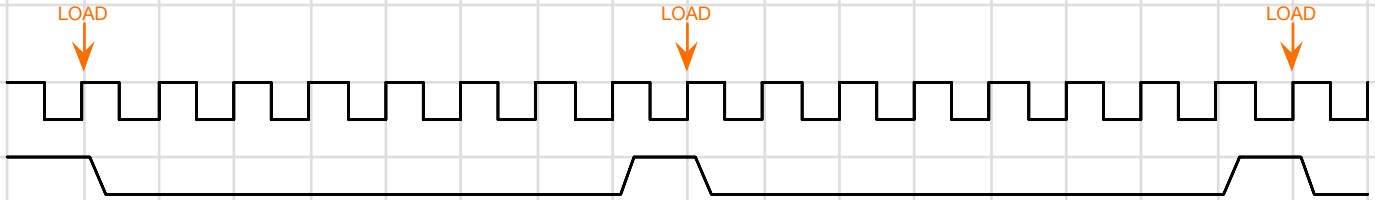


Rouge = provenance du controleur

Rouge = provenance du controleur



Principe : sérialise l'octet préalablement chargé grâce au signal « load » en le faisant circuler dans les bascules. Introduit 1 coup d'horloge de retard.



In0 (LSB)	1	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	1	X
In1	0	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	0	X
In2	0	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	0	X
In3	1	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	1	X
In4	0	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	0	X
In5	1	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	1	X
In6	1	X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	1	X
In7 (MSB)	0	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	0	X
Out	X	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	0

1clk delay

MSB

LSB

MSB

LSB

Rouge = provenance du controleur

Rouge = provenance du controleur