



## Automates – Envoi d'un octet revisité

### Rappel : Composant envoyant un octet développé en TD

Nous souhaitons développer un composant synchrone sur une horloge `clk` dont le rôle est d'émettre un octet selon un protocole série.

Ce composant est par défaut au repos (ligne série `txd` à '1'), en attente d'un ordre d'émission.

Sur le premier front montant de `clk` où il voit le signal `en` (pour *enable*), à '1',

1. il stocke l'octet donnée en entrée sur le bus `data`,
2. il indique qu'il est occupé (signal `busy` à '1')
3. il envoie sur la ligne série `txd`, le bit de poids fort (7) de l'octet

puis, à chaque front montant de `clk`, il envoie les bits suivants de 6 à 0.

Une fois qu'il a envoyé les 8 bits, il remet la ligne série à '1' puis il attend, au plus 5 fronts montant de `clk`, la confirmation de la réception (signal `ack` actif à '1') pour se remettre au repos.

Si la confirmation de la réception n'arrive pas, il prévient le composant qui donne les ordres en positionnant un signal `terr` (erreur de transmission) à '1' pendant une période de l'horloge.

À tout moment, et de manière asynchrone, un `reset` (actif à '0') remet le composant au repos.



#### Code fourni :

Le code correspondant à ce composant, code distribué en TD, ainsi qu'un test et le fichier `Makefile` permettant de lancer la simulation sans Vivado sont dans le répertoire `envoi_TD`.



#### Travail à effectuer :

Vous pouvez commencer par tester le composant, visualiser les chronogrammes avec `gtkwave`<sup>a</sup> et, dans l'architecture du test, remarquer comment nous avons vérifié que les valeurs de la ligne `txd` étaient conformes à un signal de référence `txd_ref`.

<sup>a</sup>. `gtkwave output.vcd`

## 1 Modification du protocole d'envoi d'un octet

Pour rendre plus réaliste l'envoi d'un octet, on souhaite modifier la trame émise et qu'elle contienne de l'information en plus des données (l'octet) :

- la trame commence maintenant par un bit de **start** à '0' (ce qui va permettre au récepteur de détecter qu'une donnée lui est envoyée),
- puis l'**octet** de donnée comme en TD,
- un bit de **parité** permettant de détecter une erreur (**xor**<sup>1</sup> entre les bits de l'octet). Il est à noter que cette valeur peut être calculée au fur et à mesure de l'envoi de chacun des bits,
- un bit de **stop** à '1' (on remet la ligne au repos en fin d'émission)

Le reste du fonctionnement est inchangé.

### Code fourni :

Dans le répertoire `envoi_TP`, vous trouverez

1. le fichier `envoi_octet_tp.vhd` qui est une recopie du composant du TD,
2. un test correspondant au composant que vous devez développer,
3. le fichier `Makefile`

Vous pouvez commencer par lancer la simulation et constater que, cette fois-ci, des erreurs existent et indiquent la différence entre les signaux `en`, `txd` et `busy` et les signaux de référence `en_ref`, `txd_ref` et `busy_ref`.

### Travail à effectuer :

Modifier le code du composant pour respecter la nouvelle trame.



Cela va consister essentiellement à rajouter des états pour tenir compte de l'information supplémentaire envoyée en début et en fin de trame <sup>a</sup>.

Si vous simulez votre nouveau composant avec le test fourni, les messages d'erreur doivent disparaître et les chronogrammes des signaux se superposer avec ceux des signaux de référence.

- a. Vous pouvez dessiner l'automate correspondant pour vous aider

---

1. bit de parité à '0' == nombre de '1' de l'octet pair ; bit de parité à '1' == nombre de '1' impair