

Laboratoire VSN

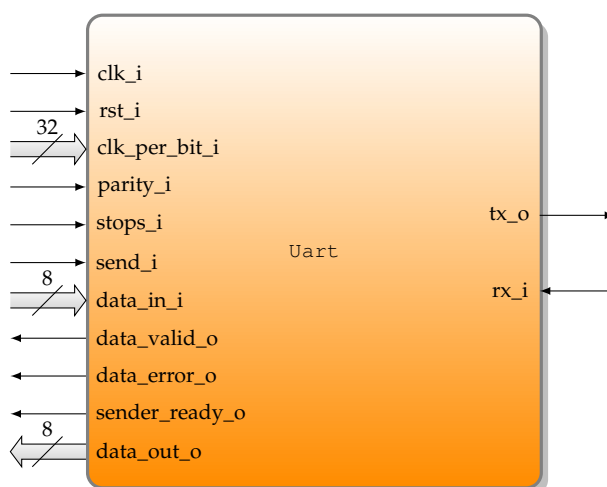
semestre de printemps 2020 - 2021

Exercices de vérification : Plan de vérification

Plan de vérification d'un UART

Contexte

Soit un UART dont l'entité est la suivante :



Cet UART permet d'envoyer des octets sur une ligne série, et d'en recevoir également.

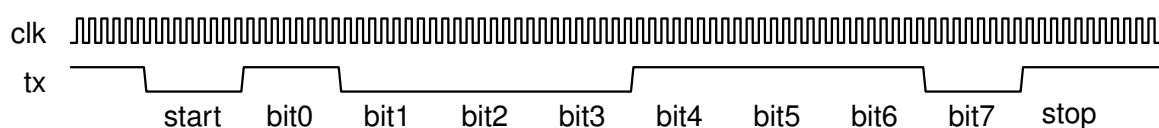
Pour l'envoi, il suffit de placer `send_i` à '1' pendant un cycle d'horloge, et à ce moment-là la donnée présente sur `data_in_i` au même instant sera ensuite envoyée en série sur la sortie `tx_o`. Attention, une donnée ne peut être envoyée que lorsque `sender_ready_o` est active. Si elle est inactive, la donnée ne sera simplement pas traitée.

Pour la réception, les données sérielles reçues sur `rx_i` sont ensuite transmises sous forme d'octet, via `data_out_o`. La donnée reçue est considérée comme valide lorsque `data_valid_o` est à '1', et ce pour un seul cycle d'horloge.

La sortie `data_error_o` passe à '1' lorsqu'il y a une erreur détectée sur un start bit ou un stop bit. Autant pour l'envoi que pour la réception, `parity_i` permet d'indiquer la présence d'un bit de parité ('1') ou non ('0'). `stop_bit_i` indique le nombre de stop bit : 1 stop bit si '0', et 2 stop bits si '1'. Enfin, `clk_per_bit_i` donne, sur 32 bits, le nombre de cycles d'horloges nécessaires à l'envoi/réception d'un bit.

Finalement, un paramètre générique, `FIFOSIZE`, permet de choisir la taille de la FIFO présente dans le composant UART en transmission. Cette FIFO interne permet de bufferiser les données et de les envoyer ensuite.

Le chronogramme suivant donne l'exemple d'un envoi de la valeur 0x71 avec `parity_i` à '0', `stops_i` à '0' et `clk_per_bit_i` à 8.



Exemples de bits de parités :

data_i	b _{parity}
00000000	0
00000001	1
00101101	0
01101101	1

Exemples d'envois

data_i	parity_i	stops_i	Observé sur la ligne
01110011	0	0	0 11001110 1
01110011	1	0	0 11001110 1 1
00110011	1	0	0 11001100 0 1
01110011	0	1	0 11001110 11
01110011	1	1	0 11001110 1 11

Exercice

A partir de la description ci-dessus, proposez un plan de vérification. Pour ce faire remplissez le fichier `uart_testplan.xml` fourni avec des tests permettant de vérifier des *features* du système. Pour chaque test indiquez la *feature* mise en avant, sa priorité, le scénario à jouer et ce qui doit être vérifié.

Le fichier est formaté pour QuestaSim. Pour l'éditer il faut utiliser Excel (pas LibreOffice, mais bien Excel). En théorie lors de la sauvegarde Excel devrait garder le format XML. Si ce n'est pas le cas il faut le forcer.

⚠ Avant de modifier le fichier, vérifiez les étapes suivantes :

Un fichier `default.rmdb` vous est proposé, et peut être lancé par le Verification Run Manager. La vérification complète peut être lancée de différentes manières :

1. En pure ligne de commande. Dans le terminal Linux, en étant dans le répertoire `code`, exécuter :

```
vrun directed
```

2. En ouvrant la version graphique du Verification Run Manager. Dans le terminal Linux, en étant dans le répertoire `code`, exécuter :

```
vrun -gui directed
```

3. Depuis QuestaSim. Dans la version graphique de QuestaSim, depuis le répertoire `code`, exécuter :

```
vrun directed
```

La base de données générée par la vérification est indiquée dans le fichier `default.rmdb`, et en l'occurrence sera sauvegardée dans le fichier `VRMDATA/merge.ucdb`.

Pour visualiser ensuite l'état du plan de vérification, dans QuestaSim graphique (ou dans un terminal Linux), exécuter :

```
vsim -viewcov VRMDATA/merge.ucdb
```

Pour cet exercice vous pouvez lancer les deux commandes à la suite dans QuestaSim. Placez-vous dans le répertoire `code` fourni, et exécutez les commandes suivantes :

```
vrun directed
vsim -viewcov VRMDATA/merge.ucdb
```

Affichez ensuite le plan de test et sa couverture grâce au menu *View* → *Verification Management* → *Tracker*. Vous devriez observer ceci :

Verification Management Tracker									
Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status	Weight	Link Status	
0	★ testplan	Testplan	9.21%	-	9.21%	<div></div>	1	Partial	
1	★ Some section title 1	Testplan	0%	100%	0%	<div></div>	1	Unlinked	
1.1	★ <unknown>	Testplan	0%	100%	0%	<div></div>	1	Unlinked	
2	★ Some section title 2	Testplan	36.85%	100%	36.85%	<div></div>	1	Partial	
2.1	★ Une feature	Testplan	100%	100%	100%	<div></div>	1	Clean	
2.2	★ Une autre feature	Testplan	0%	100%	0%	<div></div>	1	Unlinked	
2.3	★ Coverage DUV	Testplan	19.11%	100%	19.11%	<div></div>	1	Clean	
2.4	★ Coverage TB	Testplan	28.3%	100%	28.3%	<div></div>	1	Clean	
3	★ Some section title 3	Testplan	0%	100%	0%	<div></div>	1	Unlinked	
4	★ Some section title 4	Testplan	0%	100%	0%	<div></div>	1	Unlinked	

Observez la correspondance entre le champ *Link* du plan de test, et le nom des *runnable* dans le fichier `default.rmdb`.

Après avoir modifié le plan de test, vous pouvez relancer les deux commandes listées, et vous devriez pouvoir observer les résultats dans le tracker. Evidemment, les tests ne seront pas implémentés, et donc vous aurez une couverture plutôt faible. C'est normal, et pour cet exercice nous en resterons là.

A rendre

Cet exercice peut être réalisé par groupes de deux personnes au maximum. Rendez simplement le fichier `uart_testplan.xml` sur Cyberlearn. Nous discuterons en classe des différentes propositions.