trx_ofdm

1) Fonctionnalités du bloc

Le bloc trx_ofdm est un bloc spécialisé dans le calcul de transformée de fourrier rapide directe (FFT) et inverse (IFFT). Il intègre par ailleurs des fonctionnalités permettant de réaliser une mise en forme des symboles OFDM entrant (framing - insertion des pilotes et des zéros) ainsi qu?une séparation des pilotes et data des symboles sortant (deframing), ce qui lui permet d?être utilisé soit en émission OFDM (tx : framing + IFFT + insertion d?intervalle de garde) soit en réception OFDM (rx : FFT + deframing)

2) Architecture du bloc

L'architecture du bloc trx_ofdm est composée de deux modules, le c?ur et le wrapper. Le c?ur effectue tous les traitements FFT et IFFT. Le wrapper sert à fournir une interface compatible SoClib CABA avec le module MWMR_controller, disponible dans la librairie SoClib.

2.1) Interface du bloc TRX_OFDM

L'interface proposé est compatible avec le module MWMR_controller sans ajouter des modifications ni adaptations. Un rappel du bloc MWMR ainsi que du protocole FIFO est détaillé dans la Section 7.

Le module OFDM dispose des interfaces suivantes:

- Deux FIFOs d'entrée de données.
- Deux FIFOs de sortie de données.
- Une FIFO d'entrée de configuration.
- Six registres de configuration (seulement en écriture).
- Cinq registres de status (seulement en lecture).

Les deux FIFO de données et celle de configuration sont du même type. Vis-à-vis du MWMR_controller, on peut considérer que le bloc TRX_OFDM dispose de trois FIFO d'entrée et deux de sortie.

2.2) Le c?ur TRX_OFDM

Le c?ur TRX_OFDM calcule des symboles OFDM, et nécessite en entrée du calcul que les blocs complets (de 32 à 2048 mots) soient présents. Pour des raisons de performance du c?ur, et afin de ne pas pénaliser les temps de calcul par les temps de communication, une solution d?implémentation en double buffering apparaît donc nécessaire. Par ailleurs, les chargements et déchargements des buffers sont configurables.

Le choix fait pour le modèle du c?ur OFDM est donc de séparer clairement les 3 phases de chargement/calcul/déchargement par un pipeline d?exécution. Le traitement d?un symbole OFDM aura donc ces trois phases configurables. Cela signifie également que 3 symboles OFDM pourront coexister dans le c?ur répartis dans chacune des 3 phases. La Figure 2 illustre ce fonctionnement.

Le c?ur trx_ofdm peut réaliser des opérations de Framing et de Deframing à des flux de données pour séparer les données et les pilotes. Pour cela, le c?ur dispose de deux FIFOs en entrée (FIFO_in 0 et 1) et deux FIFOs en sortie (FIFO_out 0 et 1).

Les opérations possibles sont:

- A partir d'un flux de données (FIFO_in 0) on peut faire une FFT et une opération de Deframing pour obtenir des données (FIFO_out 0) et des pilotes (FIFO_out 1).
- A partir d'un flux de données (FIFO_in 0) et d'un flux de pilotes (FIFO_in 1), on peut faire un Framing et ensuite une IFFT pour obtenir un flux de données (FIFO_out 0).
- A partir d'un flux (FIFO_in 0) qui contient des données et puis des pilotes, on peut faire une opération de Framing et ensuite une IFFT pour obtenir un flux de données (FIFO_out 0)

L'utilisation des opérations de Framing et Deframing nécessite la configuration des masques de données et pilotes (voir Paramètres de configuration).

2.3) Anoc_copro_wrapper OFDM

Les interfaces de communication entre les coprocesseurs développés au CEA-Leti ne sont pas directement compatibles avec les interfaces CABA proposés par SoClib. Pour permettre de faire une conversion de protocole en gardant une fonctionnalité équivalente, un wrapper à été développé.

L'interface fournie par le module MWMR controller a certaines limitations. Par exemple, le contenu des registres configuration ne peut pas être modifié par le coprocesseur. En autre, si le coprocesseur fait une copie locale de ce registre, il ne peut pas savoir quand le contenu de ces registres été mise à jour.

Pour résoudre ce problème, nous avons conçu un wrapper qui détecte le changement du contenu des registres de configuration pour détecter un changement dans ces derniers.

Pour écrire dans un registre de configuration et s'assurer que la valeur à été correctement écrite, on doit faire deux écritures avec des valeurs différentes. Une première avec une valeur quelconque et puis une deuxième avec la bonne valeur. De ce fait, le wrapper détecte un changement de la valeur et met à jour la valeur du registre interne.

2.3.1) Registres de configuration

Les registres ici détaillés correspondent aux registres de configuration accessible à travers du bloc MWMR (voir Section 7 pou plus de détail).

Le wrapper développé à été conçu pour pouvoir supporter plusieurs coprocesseurs avec plusieurs c?urs de calcul. Par contre, le bloc TRX_OFDM contient uniquement un c?ur de calcul, donc tous les registres pour les c?urs 1 à 3 ne sont pas utilisés.

Registre de configuration du MWMR	Description
@0	Execute: Début d'exécution. Bits 0 à 3, un bit par coeur
@1	SlotID pour le coeur 0
@2	SlotID pour le coeur 1 (non utilisé)
@3	SlotID pour le coeur 2 (non utilisé)
@4	SlotID pour le coeur 3 (non utilisé)
@5	Adresse de configuration: Définit à quelle adresse débute la configuration du coeur

2.3.2) Registres de status

Ces registres sont accessibles uniquement en lecture.

Registre de Status du MWMR	Description
@0	EOC (End of Compute). Bits 0 à 3 (un bit par coeur)
@1	Status du coeur 0
@2	Status du coeur 1 (non utilisé)
@3	Status du coeur 2 (non utilisé)
@4	Status du coeur 3 (non utilisé)

3) Configuration du c?ur

3.1) Liste des paramètres

Les paramètres configurables du bloc trx_ofdm sont les suivants :

Paramètre	min val	max val	Description
BYPASS_FFT	0	1	Si 1 bypass de la fonction FFT (mode de test)
FFT_TYPE	0	1	0 = FRAMING+IFFT, 1 = FFT+DEFRAMING
LOG2_SIZE_FFT	5	11	Pour size_fft = 32,64,128,256,512,1024,2048
NORMALIZE POWER FFT	0	1	Si 1, division par sqr(n) après le calcul de FFT/IFFT (n = size_fft)
GI_INSERTION	0	1	Si 1 intervalle de garde inséré
GI_SIZE	0	2047	Taille Interval de garde (pour IFFT uniquement)
SHIFT_CARRIER	0	1	Si 1, décalage des porteuses
SHIFT_PARITY	0	1	Si 1, inversion des porteuses paires/impaires
FLOC	0	2	Type de framing: 0: dp1fifo, data et pilot dans la même FIFO 0. 1: dp2fifo, data FIFO 0, pilotes FIFO 1
MASKIT	0	1	Si 1, IT non masquée
MASK_DATA	64*32 bits		Masque framing ou deframing
MASK_PILOT	64*32 bits		Masque framing ou deframing

3.2) Adresses des paramètres

Le c?ur trx_ofdm peut contenir 3 configurations différentes, chacune rangé dans un SlotID différent. Ces SlotID sont des types de traitement que nous avons enregistré dans le c?ur et que nous pouvons utiliser pour effectuer un traitement (FFT, IFFT, ...). Quand nous voulons effectuer un traitement, nous indiquons le SlotId à utiliser et ensuite nous envoyons les données. Etant donnée que le c?ur trx_ofdm est en pipeline, nous pouvons démarrer un traitement avec une SlotId et ensuite démarrer un nouveau traitement avec une autre SlotId.

Attention, il est interdit de changer la configuration d'un SlotID si celui-ci est en cours d'utilisation par le pipeline.

Les plages d'adresses des SlotId sont:

SlotId 1 0x000 => 0x084
SlotId 2 0x100 => 0x184
SlotId 3 0x200 => 0x284

Les adresses de configuration sont relatives à l'adresse du SlotId.

Relative Address (slotid)	Name	Content
0x0	MASK_DATA_0	mask_data 0 to 64 (32 bits)
...	...	
0x3F	MASK_DATA_64	
0x40	MASK_PILOT_0	mask_pilot 0 to 64
...	...	
0x7F	MASK_PILOT_64	
0x80	FFT_CFG	bypass_fft, fft_type, log2_size_fft, normalize_power_fft, shift_carrier, shift_parity
0x81	GI_CFG	gi_insertion, gi_size
0x82	FRAMING_CFG	floc
0x83	IT_CFG	mask_IT