UNIVERSITÉ DE TOULOUSE
Faculté sciences et ingénierie

Université de Toulouse

Faculté Sciences et Ingénierie

Simulation et synthèse des matériels

BLDC controller

Compte rendu projet

par

Paul CLUZEL
Iban LEGINYORA

Encadrant : Dr THIEBOLT François

Année universitaire 2024 - 2025

Table des matières

Introduction

Dans le cadre du projet de simulation et de synthèse des matériels, nous avons été amenés à concevoir et tester un contrôleur de moteur brushless (BLDC). Ces moteurs sont largement utilisés dans les systèmes embarqués, les drones, ou encore les véhicules électriques, en raison de leur efficacité énergétique et de leur réponse rapide. Le contrôleur développé, nommé `bldc_controller`, a été implémenté en VHDL et testé au moyen d'un banc d'essai également codé en VHDL, `test_bldc_controller`. Ces deux fichiers constituent la base de notre architecture matérielle. Le `bldc_controller` a pour rôle de gérer la séquence d'activation des phases du moteur afin d'en assurer un fonctionnement fluide et stable, tout en prenant en compte des paramètres tels que le rapport cyclique (duty) et les signaux issus d'un capteur Hall. La simulation du fonctionnement a été réalisée via l'analyse d'un fichier VCD généré, permettant une visualisation détaillée des signaux internes et de la progression du système au cours des différents scénarios de test. Ce rapport présente la structure du composant, la logique interne de traitement, ainsi que les différents scénarios de test qui ont permis de valider son comportement.

Chapitre 1

Structure du composant

La structure du composant `bldc_controller` se compose de différents éléments essentiels qui permettent son bon fonctionnement et son intégration dans un système plus large. Ce chapitre décrit les différentes interfaces, ainsi que les signaux internes associés au composant.

1.1 Ports

Les ports du composant représentent les interfaces principales par lesquelles celui-ci communique avec l'extérieur. Ils sont classés en entrées et sorties, et chaque port joue un rôle précis dans la gestion du contrôleur. La table suivante présente les ports du composant et leur utilité.

Nom	**Direction**	**Taille**	**Description**
clk	IN	1 bit	Horloge du système
rst	IN	1 bit	Signal de reset asynchrone
en	IN	1 bit	Signal d'activation du PWM (non implémenté)
h	IN	1 bit	Capteur Hall pour déterminer si le rotor a fait un tour complet
duty	IN	8 bits	Rapport cyclique
U	OUT	1 bit	Broche de mise à l'état haut de la phase U
Un	OUT	1 bit	Broche de mise à l'état bas de la phase U
V	OUT	1 bit	Broche de mise à l'état haut de la phase V
Vn	OUT	1 bit	Broche de mise à l'état bas de la phase V
W	OUT	1 bit	Broche de mise à l'état haut de la phase W
Wn	OUT	1 bit	Broche de mise à l'état bas de la phase W

Table 1.1 – Ports du composant `bldc_controller`

1.2 Signaux internes

En plus des ports, `bldc_controller` dispose également de plusieurs signaux internes qui sont utilisés pour le traitement et le contrôle internes du système. Ces signaux permettent de gérer les différentes étapes du processus de commande et d'oscillation. La table suivante présente ces signaux internes et leur fonction.

Nom	Type	Description
counter	integer	Compteur interne pour le timing des phases montantes et descendantes
step	integer	Etape de la séqeunce de commande (Pour la machine à état)
cmd	integer	Etape lors d'une rampe d'une phase
cmdBefore	integer	Précédente étape lors d'une rampe d'une phase
rampUp	integer	Valeur max d'une rampe lors de la phase de montée
rampDown	integer	Valeur max d'une rampe lors de la phase de descente
rapport	integer	Rapport entre la montée et la descente
stepRamp	integer	Etape de la rampe
delta	integer	Valeur de l'oscillation autour de la commande
sens	integer	Sens de l'oscillation (+1 ou -1)
hPrev	std_logic	Gestion du front montant de h

Table 1.2 – Signaux internes du composant `bldc_controller`

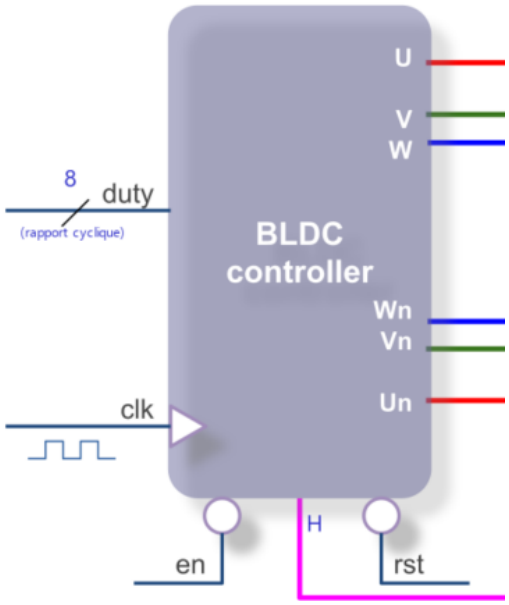

Figure 1.1 – Schéma du composant BLDC controller

Chapitre 2

Traitement et contrôle interne du système

Le contrôleur a une logique de fonctionnement synchrone, donc il va effectuer ses traitements sur le front montant de l'horloge (`clk`) du composant.

2.1 Calcul de la valeur de base (`res`) en fonction du `duty`

Le calcul de la valeur de base (`res`) dans le contrôleur dépend directement du rapport cyclique (`duty`) qui est passé en entrée avec une taille de 8 bits. Cette valeur est utilisée pour ajuster le comportement du moteur en fonction du signal d'entrée, qui représente le pourcentage de la période pendant laquelle une phase du moteur est activée.

Définition des variables et constantes

La constante `MAX_CPT` est définie comme la valeur maximale du compteur, calculée par la formule suivante :

$$\text{MAX\_CPT}(MHz) = \frac{\text{FREQ\_CLK}}{\text{MOTOR\_CYCLE}}$$

où :

- `FREQ_CLK` (Hz) : la fréquence de l'horloge du système ;
- `MOTOR_CYCLE` (Hz) est la fréquence de fonctionnement du moteur.

Calcul de `res` en fonction du `duty`

La valeur de `duty` est une valeur 8 bits qui représente le rapport cyclique, c'est-à-dire le pourcentage de temps où la phase du moteur est activée. Le calcul de la valeur de base `res` se fait selon la formule suivante :

$$\text{res} = \text{MAX\_CPT} - \left(\frac{\text{MAX\_CPT} \times \text{to\_integer(unsigned(duty))}}{256} \right)$$

Dans cette formule, `duty` est multiplié par `MAX_CPT`. La valeur obtenue est ensuite soustraite de `MAX_CPT` pour déterminer la durée pendant laquelle une phase du moteur sera activée.

Contrôle des limites de `res`

Afin d'éviter des valeurs extrêmes qui pourraient conduire à un mauvais contrôle du moteur, la valeur de `res` est limitée à une plage spécifique. Si `res` est inférieure à 50% de `MAX_CPT`, elle est ajustée à cette valeur minimale :

$$\text{res} \geq \frac{\text{MAX\_CPT}}{50}$$

Si `res` dépasse 90% de `MAX_CPT`, elle est réduite à cette valeur maximale :

$$\text{res} \leq \frac{9 \times \text{MAX\_CPT}}{10}$$

Ces ajustements garantissent que la valeur de `res` reste dans une plage acceptable pour un contrôle stable du moteur.

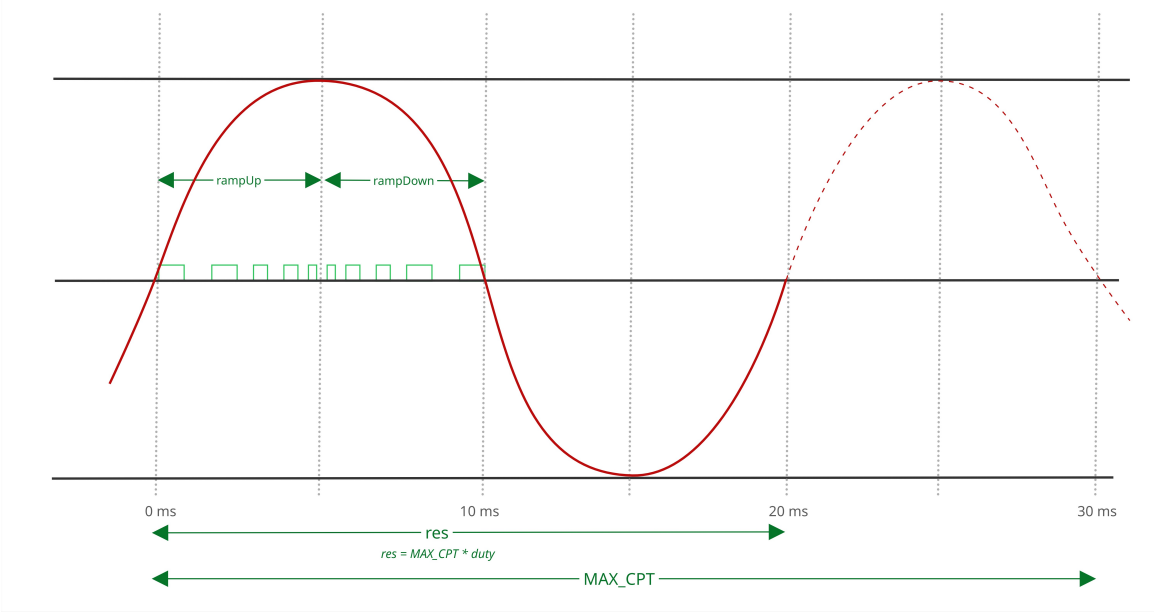

Figure 2.1 – Schéma d'une durée d'une phase sur un cylce `res`

2.2 Oscillation autour de la valeur de base (res)

Pour introduire une variation dynamique dans la commande du moteur, une oscillation est ajoutée autour de la valeur de base res. Cette oscillation permet d'éviter des comportements trop rigides du moteur, notamment lorsque le duty reste constant pendant une longue période. Elle est implémentée à l'aide de deux signaux : delta, qui représente l'écart ajouté à res, et sens, qui indique la direction de l'oscillation (positive ou négative).

Principe de l'oscillation

L'oscillation suit une logique simple : à intervalle régulier, on incrémente ou décrémente la valeur de delta, ce qui modifie la commande finale cmd comme suit :

$$\texttt{cmd} = \texttt{res} + \texttt{delta}$$

Cette commande est ensuite utilisée pour déterminer la durée de la phase active dans le contrôle moteur.

Fréquence de l'ondulation

Le code utilise une condition sur le compteur principal pour contrôler la fréquence de l'oscillation :

```
if counter mod 500 = 0 then
```

Cela signifie que l'oscillation est mise à jour tous les 500 cycles d'horloge.

Amplitude de l'oscillation

L'amplitude maximale de l'oscillation est limitée à 5% de la valeur de MAX_CPT, ce qui est exprimé dans le code par :

$$\texttt{delta} \in \left[-\frac{\texttt{MAX\_CPT}}{20}, \frac{\texttt{MAX\_CPT}}{20} \right]$$

Une fois cette limite atteinte dans un sens, la variable sens est inversée pour faire osciller la valeur dans l'autre direction.

2.3 Gérer les transitions de vitesse

Un effet de montée et de descente en puissance permet de gérer les transitions entre différentes vitesses de manière progressive et fluide. Les rampes permettent d'éviter des changements brusques de vitesse qui pourraient entraîner des comportements instables ou non souhaités du moteur.

Initialisation des rampes

Lors du démarrage ou lorsque la commande de vitesse subit un changement significatif, il faut initialiser les valeurs des rampes correctement. Cela garantit une transition progressive vers la nouvelle vitesse sans secousses.

Le but est de préparer une montée progressive vers une nouvelle valeur de commande (`cmd`), en comparant la valeur actuelle avec l'ancienne (`cmdBefore`).

Dans un premier temps, si la commande précédente (`cmdBefore`) vaut zéro, cela signifie qu'on est dans un cas d'initialisation. On considère alors que le système part de la valeur maximale (`MAX_CPT`). La montée vers la valeur de commande actuelle se fait donc à partir de cette valeur maximale. On calcule l'écart entre cette valeur maximale et la commande (`rapport`), puis on le divise en 40 étapes pour créer une montée progressive. Aucun paramètre de descente n'est pris en compte ici, car il n'y a pas eu de commande précédente.

Ajustement des rampes

Lorsque la commande (`cmd`) change en cours de fonctionnement, il est nécessaire d'ajuster les rampes pour gérer correctement l'augmentation ou la diminution de la vitesse. Ce mécanisme permet d'assurer une transition fluide entre les différentes vitesses :

— `cmd` > `cmdBefore` : Si la nouvelle commande est supérieure à l'ancienne, la vitesse doit augmenter progressivement. Dans ce cas, la rampe montante est ajustée à la valeur précédente de `cmdBefore`, et le rapport d'augmentation est calculé comme la différence entre `cmd` et `cmdBefore`.

— `cmd` < `cmdBefore` : Si la nouvelle commande est inférieure, la vitesse doit diminuer. La rampe descendante est alors ajustée pour réduire la vitesse de manière progressive.

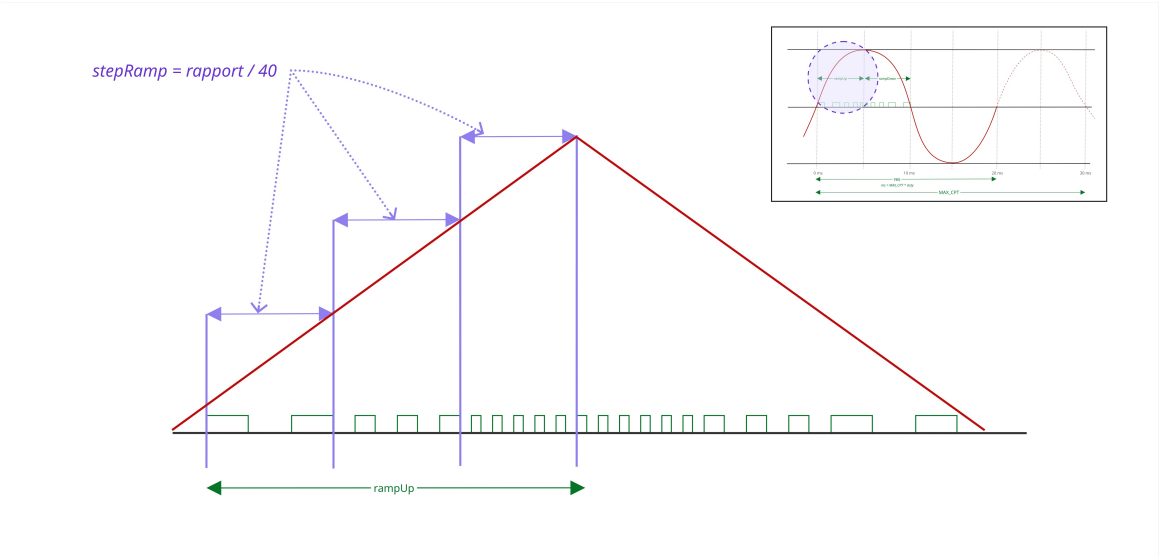

Figure 2.2 – Schéma des étapes montantes d'une phase

2.4 Gestion du compteur et progression des étapes

La gestion du compteur est un élément central du fonctionnement du composant `bldc_controller`. Elle permet de faire évoluer les broches appliquées aux phases du moteur en fonction du temps.

Activation conditionnelle

La gestion du compteur est conditionnée par la présence d'un rapport non nul :

```
if rapport /= 0 then
```

Cela signifie qu'une transition de vitesse (accélération ou décélération) est en cours, et qu'il faut donc gérer les rampes associées.

Cas de la rampe montante

Lorsque la rampe descendante est nulle (`rampDown = 0`), cela signifie que le système est en phase d'accélération :

```
if counter < rampUp then
    counter <= counter + 1;
```

- Le compteur `counter` est incrémenté à chaque cycle d'horloge jusqu'à atteindre la valeur de la rampe (`rampUp`).
- Cette valeur correspond à une durée d'attente, et donc à une vitesse de commutation des phases.

Une fois la valeur atteinte :

```
else
    counter <= 0;
    if rampUp - stepRamp >= cmd then
        rampUp <= rampUp - stepRamp;
    end if;
```

- Le compteur est remis à zéro pour redémarrer un nouveau cycle.
- La valeur de la rampe est réduite progressivement à chaque étape (`rampUp := rampUp - stepRamp`), ce qui augmente la vitesse de transition.

Cas de la rampe descendante

Lorsque `rampUp = 0`, on est en phase de décélération :

```
if counter < rampDown then
    counter <= counter + 1;
```

Le compteur augmente jusqu'à la valeur de la rampe descendante (`rampDown`), ce qui ralentit la commutation des étapes.

Une fois cette valeur atteinte :

```
else
    counter <= 0;
    if rampDown + stepRamp <= cmd then
        rampDown <= rampDown + stepRamp;
    end if;
```

— Le compteur est remis à zéro.

— La valeur de la rampe est augmentée à chaque cycle, ralentissant ainsi progressivement le rythme.

Avancement des étapes de commande

Dans les deux cas (montée ou descente), après chaque cycle complet du compteur, on avance à l'étape suivante de la séquence :

```
if step < 6 then
    step <= step + 1;
else
    step <= 1;
end if;
```

2.5 Gestion du basculement des phases

Pour gérer la synchronisation de l'activation/désactivation des phases en suivant un pattern, un process a été utilisé.

Il représente une machine à états qui gère les signaux de commande pour trois phases : `U`, `V` et `W`, ainsi que leurs compléments respectifs (`Un`, `Vn`, `Wn`).

Le fonctionnement est basé sur une variable appelée `step`, qui détermine l'état actif à un instant donné. Pour chaque valeur de `step` (de 1 à 6), un ensemble de sorties est activé pour générer la bonne séquence de commande. Cela permet de faire tourner le moteur selon une logique de commutation.

Voici les principales étapes :

— **Étape 1** (0° à 60°) : Activation des phases `U` et `Vn`.

— **Étape 2** (60° à 120°) : Activation des phases `U` et `Wn`.

— **Étape 3** (120° à 180°) : Activation des phases `V` et `Wn`.

— **Étape 4** (180° à 240°) : Activation des phases `V` et `Un`.

— **Étape 5** (240° à 300°) : Activation des phases `W` et `Un`.

— **Étape 6** (300° à 360°) : Activation des phases `W` et `Vn`.

— **Autres cas** : Toutes les sorties sont désactivées par sécurité.

Cette séquence permet de générer une onde triphasée approximée pour commander un moteur pas à pas ou brushless.

Voici la représentation de ces étapes au niveau du PWM, de la rotation électrique et de l'état des transistors :

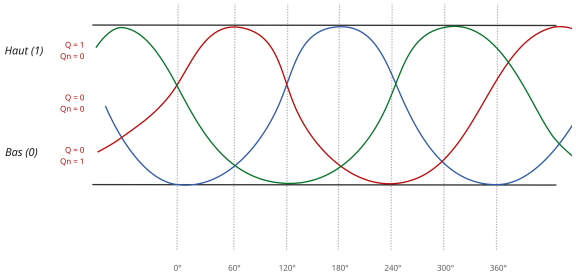

Figure 2.3 – Représentation du PWM des 3 phases.

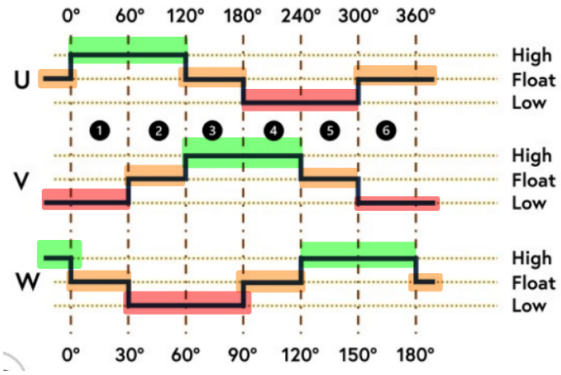

Figure 2.4 – Représentation des rotations électriques.

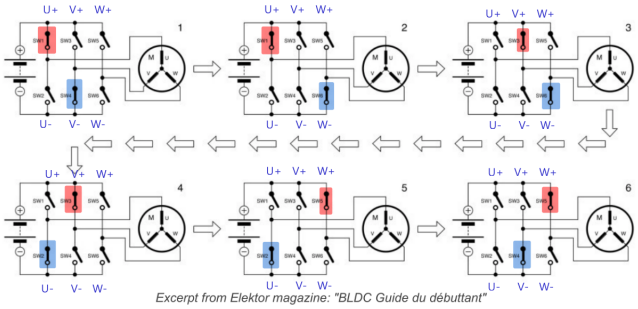

Figure 2.5 – Schéma électronique au niveau des transistors connectés aux broches de sorties du composant `bldc_controller`.

2.6 Logique du capteur Hall

Le capteur Hall est un dispositif utilisé pour détecter la position du rotor dans le moteur BLDC. En réalité, le moteur utilise trois capteurs Hall, qui sont positionnés sur le stator du moteur. Le rotor est équipé d'aimants permanents qui génèrent un champ magnétique. À chaque rotation du rotor, les capteurs Hall détectent les changements dans le champ magnétique lorsque les aimants passent devant eux.

Dans notre cas, on utilisera 1 seul capteur Hall qui va détecter si le rotor a fait un tour complet. Le signal du capteur Hall est utilisé pour synchroniser et mettre à jour les phases de commutation du moteur.

2.6.1 Utilisation du signal du capteur Hall

En VHDL, l'entité principale du contrôleur BLDC utilise le capteur Hall comme une entrée pour déterminer la position du rotor. Le signal du capteur Hall, représenté par un bit `h`, est surveillé dans le processus principal.

Lorsqu'un front montant du signal `h` est détecté, cela indique que le rotor a effectué un tour complet. À ce moment, le contrôleur réinitialise l'état du moteur et met à jour la séquence de commutation des phases.

Chapitre 3

Scénarios de tests effectués

Dans cette section, nous décrivons les scénarios de tests réalisés pour évaluer le fonctionnement du contrôleur BLDC. Ces tests permettent de valider le bon comportement du système en simulant différents cas de figure. Les scénarios de tests suivants ont été exécutés :

3.1 Scénario 1 (nominal) : Test avec un cycle de travail de 50% sans signal Hall

Le premier scénario consiste à tester le contrôleur avec un cycle fixé à 50%, ce qui correspond à une valeur de 128 dans le vecteur de commande de type `std_logic_vector(7 downto 0)`. Dans ce test, le signal Hall est désactivé, simulant ainsi une absence de détection de rotation.

- **Objectif :** Vérifier le comportement du contrôleur lorsqu'il fonctionne à 50% de son cycle de travail.
- **Détails :**
  - Le signal `duty` est configuré à '10000000' (128, soit 50%).
  - Le signal Hall `h` est maintenu à `'0'`, indiquant l'absence de détection.
- **Durée :** 500 ms
- **Résultat attendu :** Le contrôleur doit ajuster les sorties `U, V, W` en fonction du cycle spécifié.

3.2 Scénario 2 : Test avec un cycle de travail de 25%

Le deuxième scénario teste le contrôleur avec un cycle réduit à 25%. Ce test permet de valider que le système peut fonctionner efficacement à un niveau de puissance plus faible.

- **Objectif :** Vérifier le comportement du contrôleur avec un cycle de travail de 25%.
- **Détails :**
  - Le signal `duty` est configuré à '01000000' (64, soit 25%).
  - Le signal Hall `h` est maintenu à '0'.
- **Durée :** 500 ms
- **Résultat attendu :** Le contrôleur doit ajuster les sorties de manière correcte en fonction du cycle `duty` de 25%, sans changement dans la séquence de phases.

3.3 Scénario 3 : Test avec un signal Hall actif et un cycle de travail de 75%

Le troisième scénario est une situation plus dynamique où un cycle de 75% est combiné avec une détection du signal Hall, simulant ainsi un tour complet du moteur. Ce test permet de valider le bon fonctionnement du détecteur Hall et de vérifier que le contrôleur ajuste correctement les phases de commutation en réponse à la détection du signal Hall.

- **Objectif :** Tester la détection du signal Hall et la réaction du contrôleur avec un cycle de travail de 75%.
- **Détails :**
  - Le signal `duty` est configuré à "11000000" (192, soit 75%).
  - Le signal Hall `h` est activé périodiquement pour simuler la détection de tours complets (chaque tour simulé par une impulsion Hall).
- **Durée :** 120 ms par tour, répété 5 fois.
- **Résultat attendu :** Le contrôleur doit détecter les impulsions du signal Hall et ajuster la séquence des phases de manière appropriée. Chaque impulsion de `h` déclenche un changement de phase.

3.4 Arrêt automatique du test après 10 secondes

Après l'exécution des trois scénarios, un délai de 10 secondes est observé pour s'assurer que le système continue de fonctionner correctement sur une période prolongée. Ce délai permet également de vérifier que le test s'arrête proprement après la période spécifiée.

3.5 Conclusion des tests

3.5.1 Résultat du chronogramme obtenu pour le scénario 1

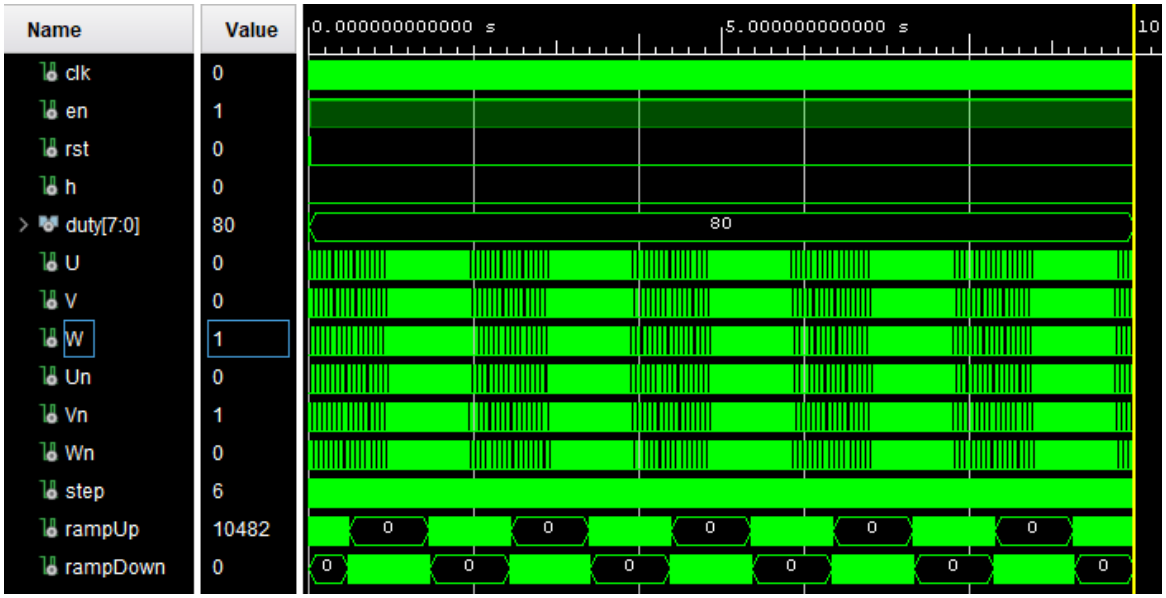

Figure 3.1 – Vu d'ensemble du test du scénario 1

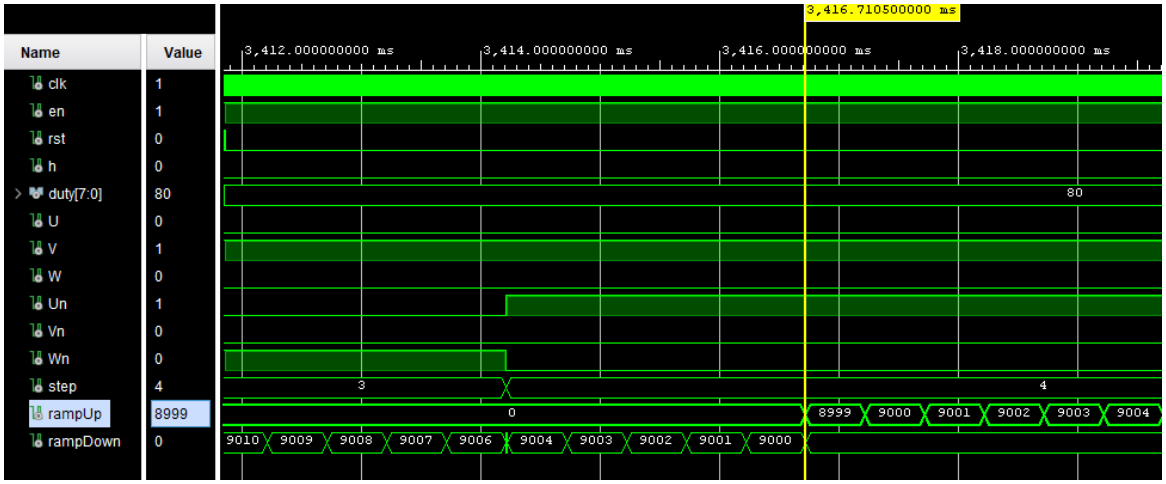

Figure 3.2 – Zoom au niveau du sommet d'une phase

Interprétation

On peut remarquer qu'il y a bien une évolution constante au niveau de la durée des cycles des rampes montantes et descente. De plus, on peut remarquer qu'il y a bien une incrémentation au niveau des steps montants et une décrémentation au niveau des steps descendants.

3.5.2 Résultat du chronogramme obtenu pour le scénario 2

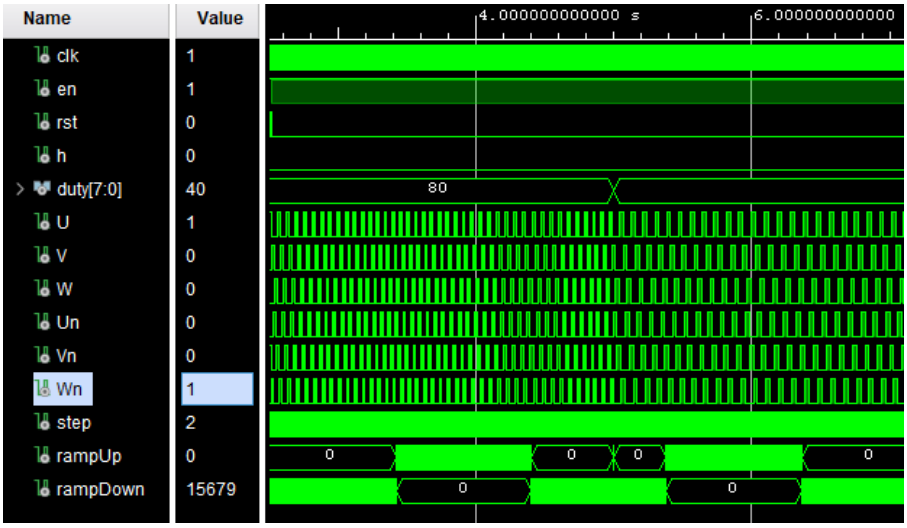

Figure 3.3 – Vu d'ensemble du test du scénario 2

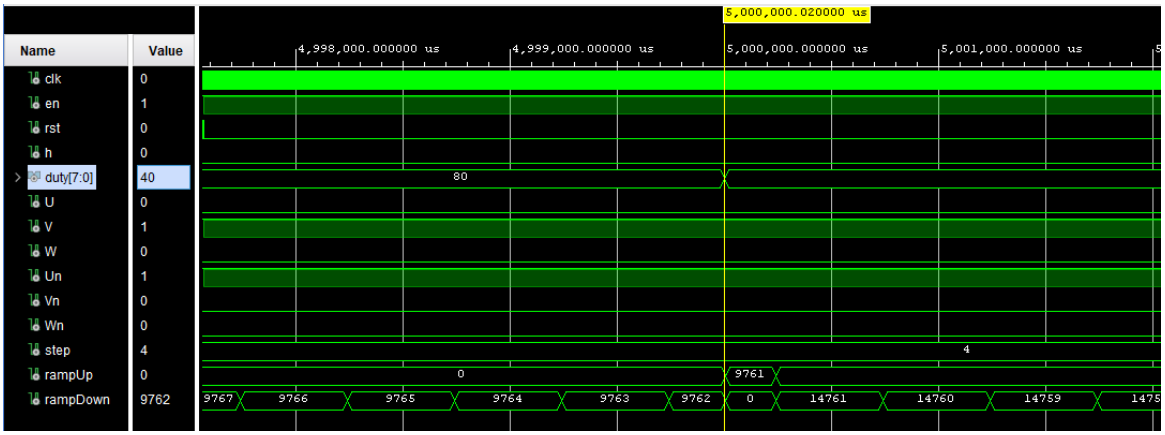

Figure 3.4 – Zoom sur le basculement entre l'ancien et le nouveau `duty`

Interprétation

On peut constater que le changement s'est bien effectué est que les rampes sont réadaptées en fonction du nouveau cycle de travail.

3.5.3 Résultat du chronogramme obtenu pour le scénario 3

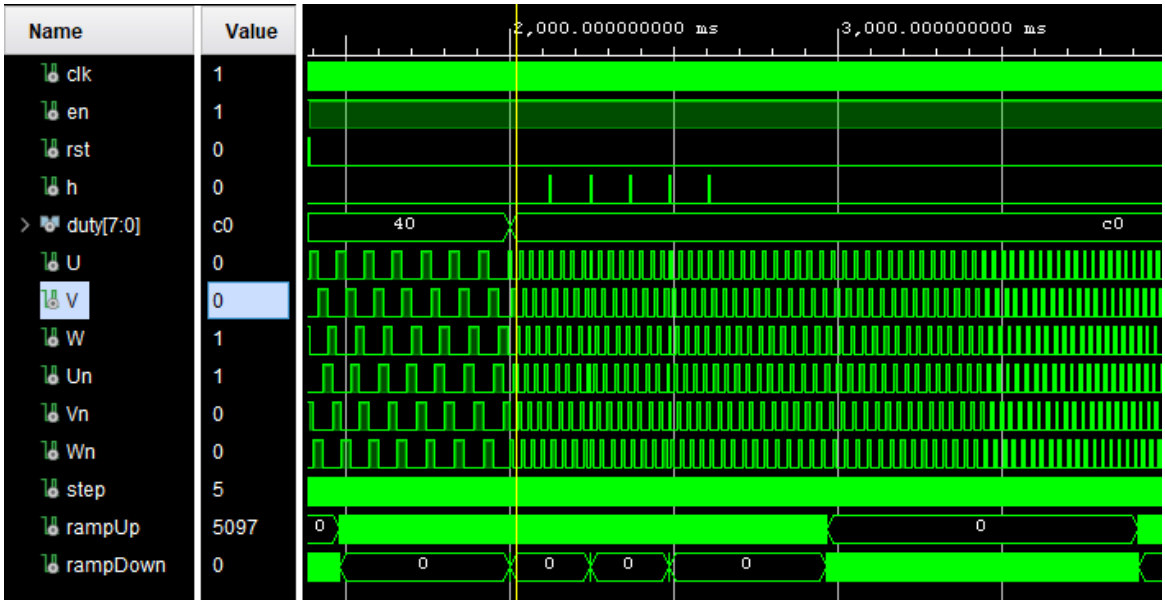

Figure 3.5 – Vu d'ensemble du test du scénario 3

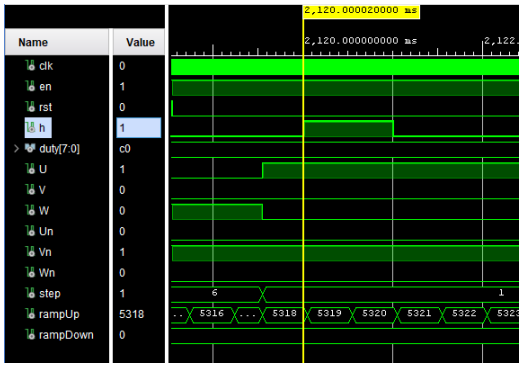

Figure 3.6 – Réception d'un signal Haut du capteur `Hall` sans influence sur l'exécution

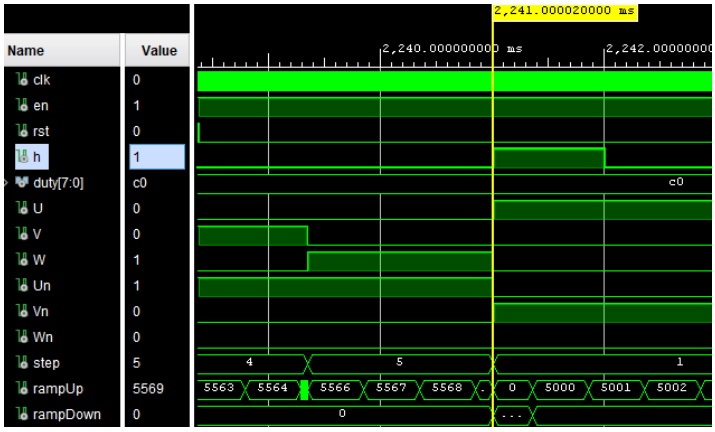

Figure 3.7 – Réception d'un signal Haut du capteur `Hall` avec une influence sur l'exécution

Interprétation

On peut constater que le capteur `Hall` est bien détecté est pris en compte si et seulement si le step de la machine d'état est n'est plus synchronisé.

Conclusion

Le développement du contrôleur BLDC nous a permis de mettre en œuvre une architecture complète de commande, intégrant à la fois le calcul du signal PWM en fonction du rapport cyclique, l'introduction d'une oscillation dynamique pour améliorer la réactivité, ainsi que la gestion fluide des transitions de vitesse via des rampes. Les tests réalisés ont démontré que le système est capable de s'adapter à différents cycles de travail et de répondre efficacement à la détection d'un signal Hall, assurant ainsi une commutation correcte des phases du moteur. Ce projet a été une excellente occasion d'approfondir notre compréhension des systèmes de commande temps réel et de leur modélisation en VHDL.

Table des figures