Chapitre 4

Développement d'un simulateur temps réel sur FPGA

Le présent chapitre explicite le travail accompli pour le développement d'un simulateur temps réel à pas multiple sur FPGA. Les détails de l'implémentation du réseau simulé seront présentés au prochain chapitre. La première partie présente le montage simulé dans le cadre de ce projet ainsi que les principes d'opération de base d'un tel dispositif. Par la suite, l'élaboration des différents aspects du simulateur rapide est présentée. Puis le simulateur à long pas de calcul est décrit et finalement la dernière partie de ce chapitre porte sur le couplage entre les deux parties du simulateur.

4.1 Système STATCOM

Afin de démontrer la faisabilité de la simulation temps réel sur FPGA, la simulation d'un convertisseur de puissance triphasé est réalisée. Le système de puissance simulé est similaire à un système STATCOM (*STATic synchronous COMpensator*) qui est utilisé pour améliorer l'écoulement de puissance ainsi que la stabilité dynamique d'un réseau électrique [50].

À l'instar d'un compensateur synchrone (voir figure 4.2), un STATCOM produit ou absorbe de la puissance réactive selon la différence de tension entre la tension du réseau (V_R) et la tension produite par le convertisseur de puissance du système STATCOM (V_O).

X est la réactance de fuite du transformateur. Le fonctionnement simplifié présenté ici ne tient pas compte des pertes.

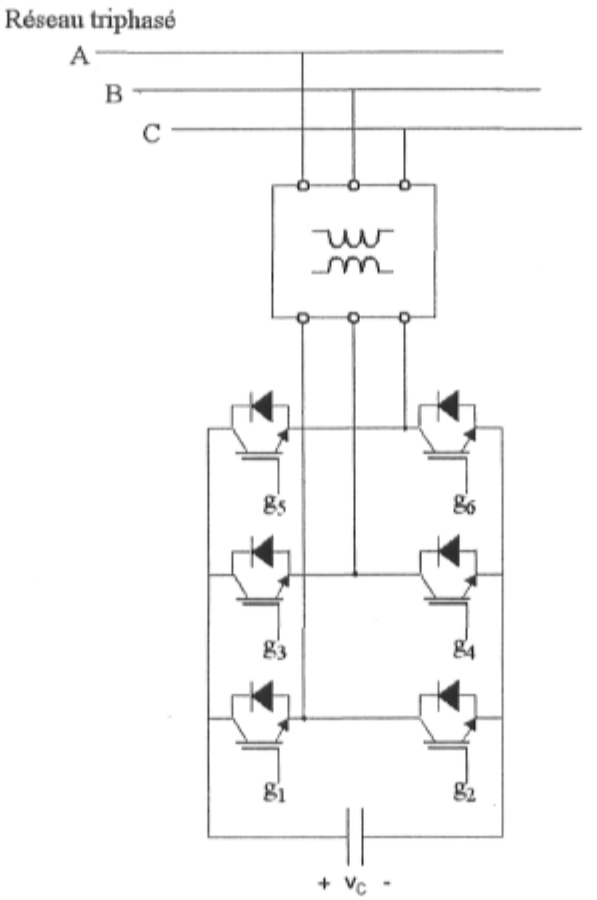


Figure 4.1 Représentation simplifiée d'un système STATCOM.

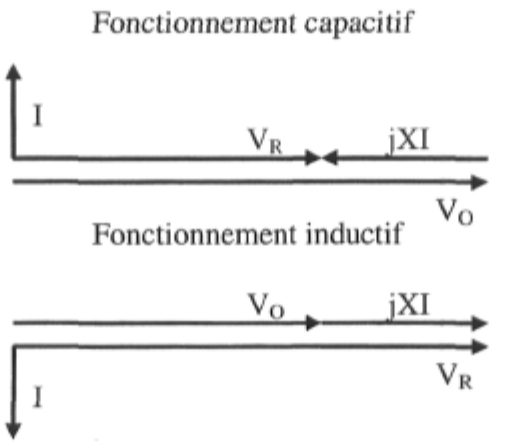


Figure 4.2 Représentation vectorielle simplifié du fonctionnement du fonctionnement du STATCOM [50].

Dans le système réel et tel qu'illustré à la figure 4.1, un condensateur est utilisé pour stocker l'énergie nécessaire au fonctionnement de l'installation. Un système de régulation assure que la tension aux bornes du condensateur demeure à un certain niveau en agissant sur la synchronisation du système d'allumage des interrupteurs du pont d'interrupteur. Selon l'avance ou le retard de phase du signal utilisé pour effectuer la modulation MLI, le niveau de la tension dans le condensateur augmente ou diminue. Dans le cadre de ce projet, il a été décidé de faire fonctionner le convertisseur en boucle ouverte et d'assumer une boucle de régulation qui maintient la valeur de la tension aux bornes du condensateur constante, ce qui permet de représenter le condensateur comme une source de tension continue.

Commande par modulation de la largeur d'impulsion (MLI)

La commande de l'allumage des interrupteurs du convertisseur s'effectue selon une modulation de la largeur d'impulsion. Les deux interrupteurs constituant un bras du convertisseur sont utilisés de manière complémentaire. Un signal sinusoïdal est comparé à un signal triangulaire : lorsque ce dernier est inférieur à la sinusoïde, l'interrupteur du haut est en conduction tandis que c'est celui du bas qui est en conduction lorsque l'onde triangulaire est supérieure à la sinusoïde. La même onde triangulaire est utilisée pour les trois phases. La fréquence de l'onde triangulaire est la fréquence de la porteuse MLI. L'indice de modulation m est défini comme le rapport entre l'amplitude du signal modulant et l'amplitude de la porteuse triangulaire.

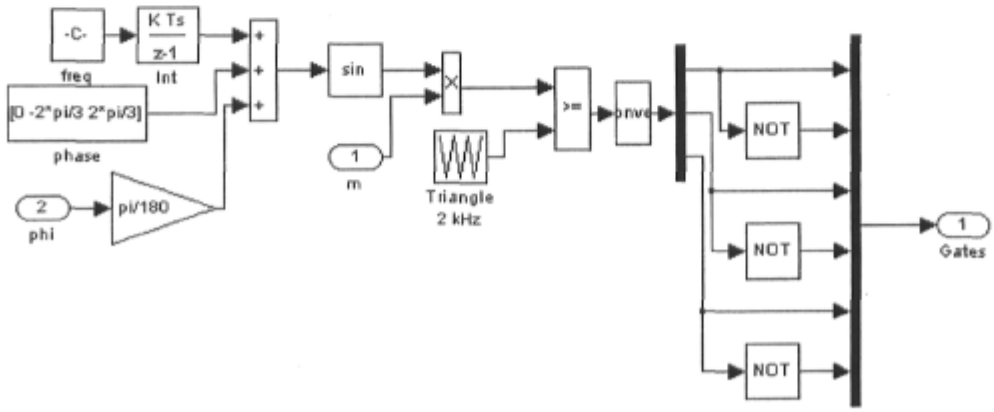


Figure 4.3 Implémentation de la modulation de la largeur d'impulsion

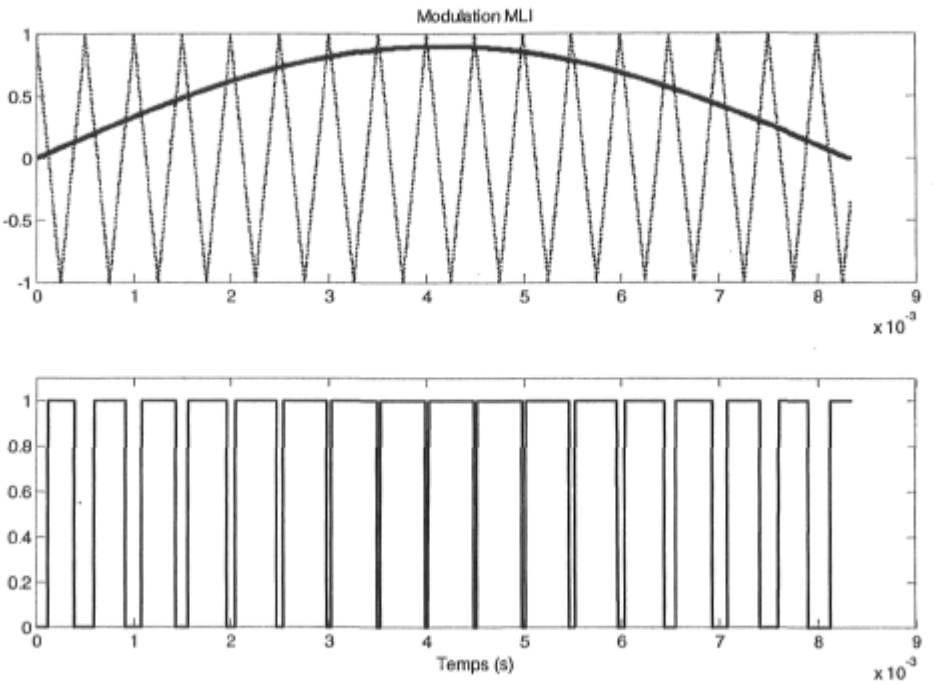


Figure 4.4 Exemple de modulation de la largeur d'impulsion (porteuse : 2 kHz; modulante sinusoïdale : 60 Hz, phase A, m = 0.9).

4.2 Représentation des nombres

Tous les nombres dans le noyau de simulation sont représentés en virgule fixe en complément 2 avec une partie entière de 12 bits et une partie fractionnaire de 20 bits. Ce choix de la représentation numérique a été effectué afin de répondre à plusieurs considérations.

En premier lieu, le choix du nombre de bits utilisés pour la représentation est basé sur la largeur des registres du PPC qui est de 32 bits. De plus, puisque le PPC est un processeur de nombre entier, cela a évidemment influencé le choix de la notation.

En second lieu, la notation à virgule flottante entraîne une latence relativement importante lors des opérations arithmétiques, si le traitement est effectué en pipeline, ou une diminution marquée de la fréquence d'opération pour une implémentation non segmentée. Un simulateur de convertisseur de puissance a été réalisé en virgule flottante et en virgule fixe pour explorer les différentes notations. Le temps d'exécution, en

utilisant la même fréquence d'horloge, était plus de quatre fois plus élevé pour l'implémentation en notation à virgule flottante (1.075 µs versus 0.25 µs). Par surcroît, même si la plage dynamique est beaucoup plus grande, la précision au niveau fractionnaire est variable, ce qui peut être embêtant pour le traitement de systèmes rigides présentant des constantes de temps très grandes et très petites. De plus, avec les deux simulateurs de validation, des oscillations numériques divergentes ont été observées et la divergence était plus marquée pour le système en virgule flottante. Cette accélération de la divergence provient de la précision variable de la notation à virgule flottante. Ces oscillations étaient dues à la méthode d'intégration trapézoïdale lors de commutation.

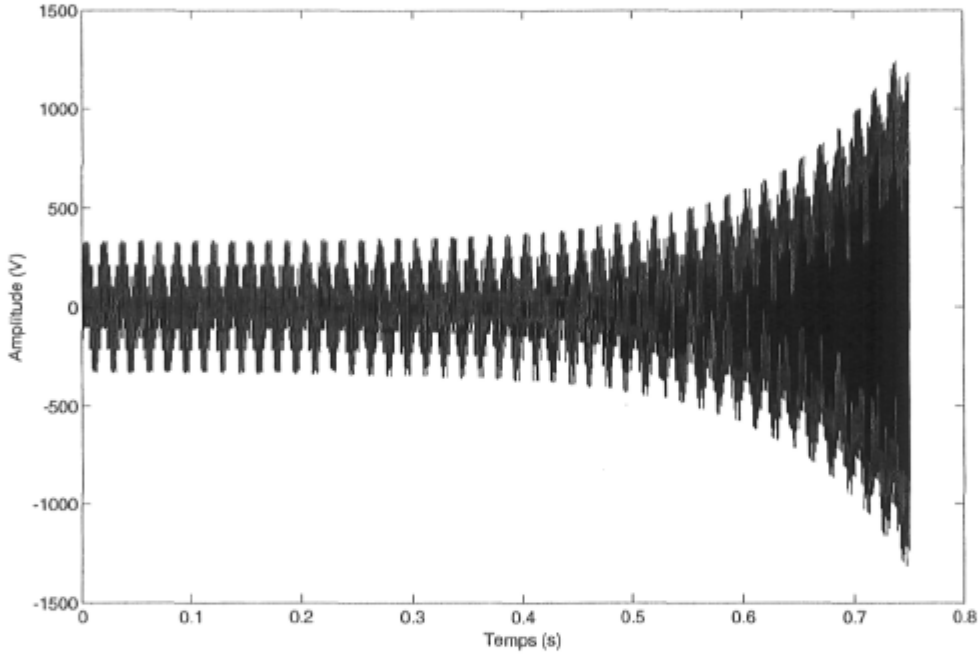


Figure 4.5 Oscillations divergentes provenant de la méthode d'intégration numérique.

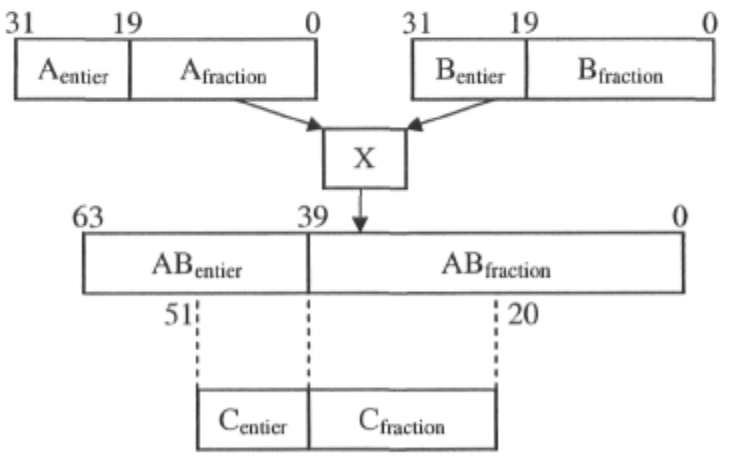
L'utilisation de la notation *per-unit* (p.u.) permet de réduire l'ampleur de la plage dynamique nécessaire, permettant d'obtenir une précision plus élevée de la partie fractionnaire, mais alors la notation à virgule flottante perd de son intérêt si les bornes de la plage dynamique sont ainsi réduites.

Suite à toutes ces considérations, il a été décidé d'utiliser une notation à virgule fixe, pour sa rapidité et sa simplicité d'implémentation, de 32 bits afin d'être directement

compatible avec le PPC et les bus de communication. En utilisant 20 bits pour la partie fractionnaire, cela donne une résolution de 2^{-20}, soit un peu moins qu'un micron ($0{,}95367431640625 \times 10^{-6}$). Il reste alors 12 bits pour la partie entière et en les utilisant avec une notation en complément 2, afin de pouvoir représenter les nombres négatifs, cela permet une plage de valeur représentable allant de -2048 à ($2048 - 2^{-20}$) avec une résolution de 2^{-20}.

Précision des résultats intermédiaires

À noter que le résultat d'une multiplication compte deux fois plus de bits que les opérandes, si elles sont de même taille. Dans le cas présent, le résultat de la multiplication compte 24 bits pour la partie entière et 40 pour la partie fractionnaire. Pour ne pas commettre d'erreur de représentation il faudrait conserver la totalité des bits du résultat. Afin de pouvoir réutiliser au maximum chaque élément arithmétique de 32 bits, tous les résultats sont ramenés sur 32 bits. Donc dans le cas des résultats de multiplication, on conserve uniquement les 12 bits inférieurs de la partie entière et les 20 bits supérieurs de la partie fractionnaire.



$$A \times B = AB \approx C$$

Figure 4.6 Troncature effectuée suite à une multiplication.

4.3 Description du noyau de simulation

Le noyau de simulation exécute les calculs nécessaires pour la simulation du convertisseur de puissance. La méthode nodale est utilisée avec la modélisation EMTP. Cependant, la méthode Euler arrière est utilisée. Cette méthode a été choisie afin d'éliminer les oscillations numériques causées par les commutations des interrupteurs de puissance. Puisque le pas de calcul utilisé est très faible, 5 µs, la précision de la méthode Euler arrière est acceptable. Pour un pas de calcul plus grand, on ne pourrait pas faire la substitution aussi aisément sans occasionner une perte de précision notable.

En plus de contenir le nécessaire pour la résolution du système d'équation relatif au convertisseur, le noyau de simulation contient un générateur MLI qui commande l'état des interrupteurs du convertisseur ainsi qu'un circuit de commande qui cadence le noyau de simulation afin d'exécuter la simulation en temps réel. Ces deux modules sont décrits dans les deux prochaines sections.

Algorithme de simulation

L'algorithme de simulation utilisé est illustré à la figure 4.7. La première étape consiste à mettre à jour le vecteur d'injection de courant. Les sources de tension, les sources de courant d'historique, les tensions provenant du simulateur lent ainsi que l'état actuel du convertisseur sont utilisés pour la mise à jour de l'injection de courant. La seconde étape est la multiplication de la matrice d'admittance inverse avec le vecteur d'injection de courant, soit (2-44). C'est l'étape la plus gourmande en ressource et en temps. La troisième partie de l'algorithme est le calcul des courants circulant dans les éléments réactifs, (2-24) et (2-28) avec la méthode trapézoïdale ou (2-26) et (2-30) avec la méthode Euler arrière, et finalement la dernière étape est la mise à jour des sources de courant commandées qui modélisent l'historique des éléments réactifs, (2-25) et (2-29) ou (2-27) et (2-31) avec la méthode trapézoïdale et Euler arrière respectivement.

Une fois ces quatre étapes complétées, le noyau de simulation se place en mode d'attente. Initialement le noyau de simulation est en mode attente et c'est dans ce mode qu'il est

configuré et que l'on peut accéder au contenu de ses registres. Afin de débuter la simulation, il est nécessaire de charger en mémoire les matrices d'admittance inverses, la valeur de l'admittance équivalente des éléments réactifs nécessaires au calcul de leur courant et de leur source de courant historique ainsi que les valeurs des autres sources contenues dans le système simulé.

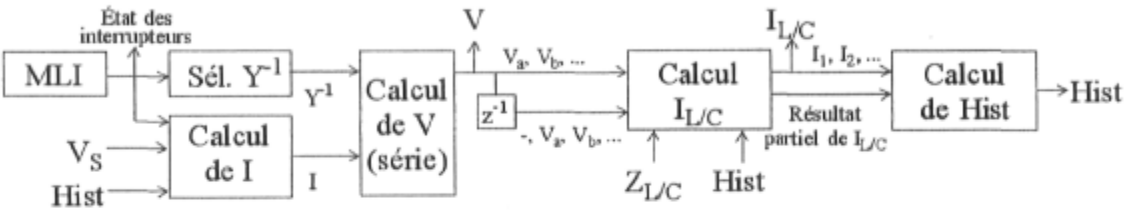


Figure 4.7 Diagramme fonctionnel de l'algorithme de simulation.

Réalisation FPGA

L'implémentation d'un algorithme arithmétique sur FPGA peut être réalisée de façon séquentielle, parallèle ou une combinaison des deux. Ces deux méthodes sont totalement opposées au niveau des ressources utilisées et du temps d'exécution. Dans ce cas-ci, un compromis entre ces deux méthodes a été choisi. Afin de limiter les ressources utilisées, une série de multiplicateurs et d'additionneurs/soustracteurs est créée et elle servira pour réaliser toutes les opérations mathématiques de l'algorithme.

La mise à jour du vecteur d'injection de courant consiste en la sommation algébrique des courants entrant et sortant du nœud en question. Dans les deux cas, on tient compte uniquement des courants provenant de sources de courant ou de tension. Pour les courants provenant d'une source de courant, cela implique uniquement une addition ou une soustraction mais dans le cas des sources de tension, il est nécessaire de faire la multiplication de la valeur de la source de tension par l'admittance la reliant au nœud traité. Le temps nécessaire pour faire la mise à jour du vecteur d'injection de courant dépend directement du nombre de sources et d'éléments réactifs. Dans l'implémentation finale, cette étape nécessite trois coups d'horloge de 80 MHz.

Une fois que le vecteur d'injection de courant est calculé, les tensions nodales sont calculées séquentiellement à l'aide des multiplicateurs et de l'arbre d'addition illustrés à la figure 4.8.

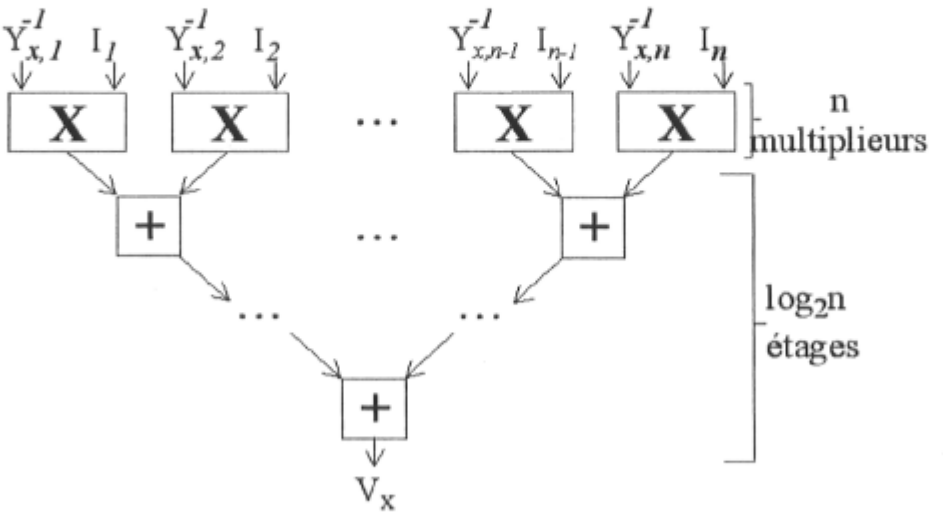


Figure 4.8 Module arithmétique servant au calcul des tensions nodales.

À chaque coup d'horloge, une ligne de la matrice d'admittance inverse $\boldsymbol{Y}^{-1}$ est injectée dans la structure illustrée à la figure 4.8. Le terme approprié du vecteur d'injection de courant constitue la seconde entrée des multiplicateurs et il y reste pour le calcul de toutes les tensions nodales. Chaque nœud de l'arbre d'addition contient un registre pour stocker un résultat partiel. Cela augmente la latence de l'opération mais permet de la cadencer plus rapidement. Dans le simulateur final, la multiplication nécessite un coup d'horloge et puisqu'il y a quatre nœuds dans le système simulé cela implique un arbre de multiplication de deux niveaux, donc trois coups d'horloge avant d'avoir la première tension nodale et donc au total six coups d'horloge avant d'avoir toutes les tensions nodales.

Le simulateur de validation utilisé pour effectuer le choix de la notation implémentait le calcul des tensions nodales en parallèle : au lieu d'effectuer le calcul une tension à la fois, toutes les tensions étaient traitées simultanément, soit n^2 multiplications suivies de n arbres d'addition eux aussi implémentés en parallèle pour n tensions nodales. Il est évident que cette approche nécessite n fois plus de ressources et selon n et les ressources disponibles sur le FPGA utilisé, il est probable que cette approche soit impossible à

réaliser par manque de multiplicateurs dédiés ou tout simplement par manque de ressources logiques.

Il est important de noter que le calcul des courants dans les éléments réactifs débute dès que l'information nécessaire est disponible, le début de cette étape est donc imbriqué avec la fin de l'étape de calcul des tensions nodales. Puisque les tensions nodales sont disponibles séquentiellement, cette étape nécessite peu de ressources : avec la modélisation utilisée, cette étape requiert un additionneur, un soustracteur et un multiplicateur. Chaque opération nécessite un coup d'horloge, soit une latence de trois coups d'horloge avant d'obtenir le premier résultat. Le système simulé compte trois éléments inductifs, donc cette étape nécessite cinq coups d'horloge.

Puisque le calcul de la source de courant historique requiert la valeur du courant circulant dans l'élément auquel elle se rattache, cette étape aussi est imbriquée avec l'étape précédente. De plus, selon la modélisation utilisée, il est possible d'utiliser un résultat partiel calculé lors de l'étape précédente, ce qui réduit le calcul de l'historique à l'addition du résultat partiel avec la valeur de courant qui vient d'être calculée. Donc une latence d'un coup d'horloge et un temps total de trois coups d'horloge pour cette étape.

On constate que la multiplication de la matrice d'admittance et du vecteur d'injection de courant est l'opération la plus gourmande en temps et que cette opération impose le nombre d'additionneurs et de multiplicateurs requis pour le circuit si une stratégie de partage de ressources est utilisée. L'ajout de logique pour injecter dans les multiplicateurs ou les additionneurs différents signaux à différents instants implique des délais de routage additionnels. Ainsi le parcours critique qui fixe la fréquence maximale d'opération à un peu plus de 80 MHz se situe entre deux registres en passant par de la logique de routage puis par un multiplicateur 32 bits par 32 bits. Puisque les multiplicateurs dédiés acceptent des opérandes de 18 bits seulement, il faut en combiner plusieurs pour réaliser un multiplicateur acceptant des opérandes de plus grande taille. Dans le cas présent, trois multiplicateurs dédiés sont nécessaires pour réaliser un multiplicateur 32 par 32, ce qui implique des délais supplémentaires de routage qui diminuent la fréquence maximale.

4.4 Description du modulateur MLI implémenté

Le modulateur MLI triphasé se compose d'un générateur d'ondes sinusoïdales triphasées équilibrées, d'un générateur d'onde triangulaire, de trois multiplicateurs et de trois comparateurs (voir figure 4.3). Une fois le signal triphasé généré, il est multiplié par l'indice de modulation m puis chaque phase est comparée à l'onde triangulaire afin de déterminer l'état de chaque bras. Lorsque le signal d'une phase est supérieur ou égal à l'onde triangulaire, l'état du bras est mis à 1, signifiant que l'interrupteur du haut est en conduction tandis que celui du bas est bloqué. Lorsque le signal d'une phase est inférieur, l'état du bras est mis à 0, indiquant que l'état des interrupteurs est inversé. Les états des bras sont réunis pour former le signal d'état des interrupteurs, qui est utilisé pour déterminer l'injection de courant.

Générateur de sinusoïdes et d'onde triangulaire

Les générateurs d'onde sinusoïdale utilisés pour produire le signal triphasé sont constitués d'une table de lecture de 209 points qui représente un quart d'onde sinusoïdale. Le cycle complet compte 832 points et la valeur de chaque sinusoïde est mise à jour à chaque quatre pas de calcul pour un cycle total de 3328 pas de calcul de 5 µs, soit une fréquence d'environ 60,0962 Hz.

Le générateur d'onde triangulaire se constitue lui aussi d'une table de lecture représentant un quart d'onde mais elle compte 26 termes et la valeur de l'onde est mise à jour à tous les pas de calcul. Ainsi un cycle complet compte 100 pas de 5 µs, soit une fréquence de 2 kHz.

Le nombre de points dans les tables de lecture a été choisi afin de tenir compte de la longueur du pas de calcul et de la période de rafraîchissement pour obtenir un cycle proche de 60 Hz pour l'onde sinusoïdale et 2 kHz pour l'onde triangulaire tout en minimisant l'espace mémoire requis.

Réalisation FPGA

La réalisation FPGA du générateur d'ondes est illustrée à la figure 4.9. La mise à jour des signaux sinusoïdaux et de l'onde triangulaire est effectuée à la fin du pas de calcul pour le prochain pas tandis que la multiplication par l'indice de modulation est effectuée au début de chaque pas de calcul pour tenir compte d'un éventuel changement d'indice de modulation. Ainsi, l'état des interrupteurs du convertisseur est établi juste avant le calcul de l'injection de courant.

La notation numérique utilisée pour le générateur de MLI diffère du reste du circuit. Puisque toutes les valeurs sont comprises entre -1 et 1, le nombre de bits pour la partie entière a été réduit à deux afin de pouvoir représenter la partie négative également, tandis que la partie fractionnaire en compte 11. Les tables de conversion contiennent des valeurs non signées afin de réduire la quantité de mémoire requise pour leur implémentation. Le signe est rajouté après la lecture et, dans le cas des ondes sinusoïdales, juste avant la multiplication avec l'indice de modulation. Le choix de cette notation a été effectué afin de réduire l'espace mémoire requis pour l'implantation des tables de conversion tout en gardant une précision intéressante (2^{-11}).

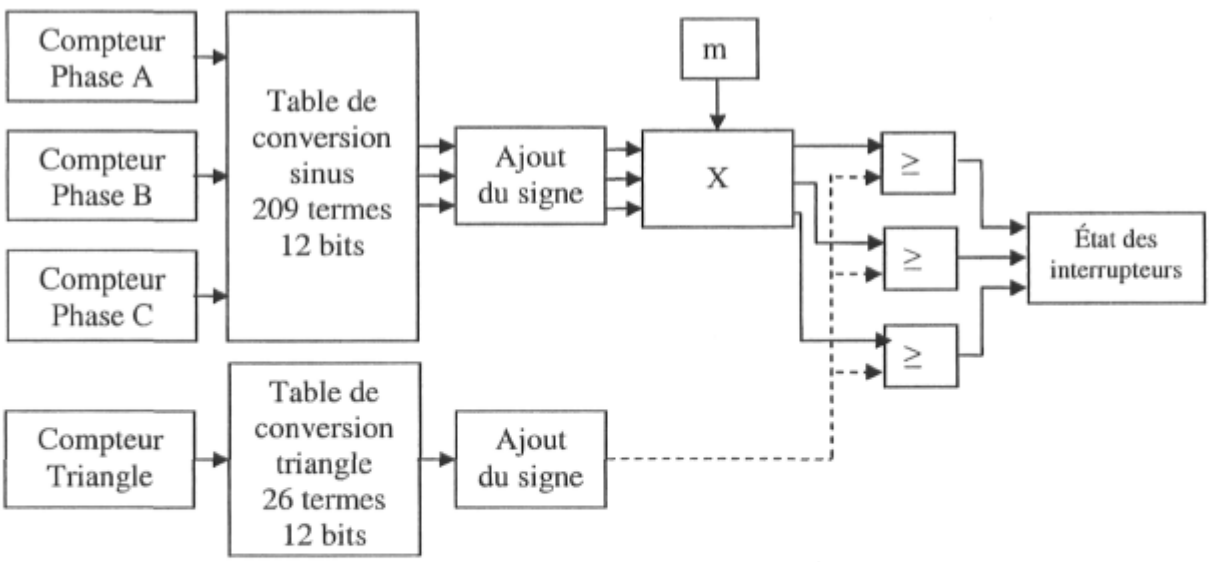


Figure 4.9 Réalisation FPGA du générateur d'onde sinusoïdale et triangulaire.

4.5 Description du cadenceur

Le cadenceur régit le fonctionnement du simulateur rapide et assure le fonctionnement en temps réel. Il comporte essentiellement des compteurs qui cumulent le nombre de coups d'horloge depuis le début de la séquence. Chaque séquence consiste en dix pas de calcul de 5 µs. Au début de chaque pas de calcul, le cadenceur active le noyau de calcul et le modulateur MLI. Ces deux modules sont ensuite placés en mode d'attente une fois leur travail effectué. Puisque le noyau de simulation utilise l'horloge du bus local au processeur, qui a une fréquence de 80 MHz, les calculs sont effectués à chaque 400 coups d'horloge. Une fois les dix pas de calcul effectués, le simulateur entier passe en mode attente pour le transfert de données avec le simulateur à long pas.

Réalisation FPGA

La figure 4.10 illustre les machines à états utilisées pour implémenter le cadenceur. Le diagramme qui y est présenté est très explicatif et il représente le simulateur implanté pour le système test présenté au chapitre 5. La machine à états est implémentée à l'aide d'un registre stockant l'état actuel ainsi que de comparateurs pour déterminer quand activer les différents modules. Pour amorcer la séquence de calcul, le registre « slv_reg0 » doit être chargé avec le code de départ. Une fois la séquence lancée, les compteurs sont activés. Lorsque le signal « start_ts » est actif, l'algorithme de simulation est exécuté. Le signal « finish_ts » devient actif lorsque l'algorithme a terminé de s'exécuter. À noter que la séquence prend fin dès que les calculs du dixième pas de calcul sont terminés. Le noyau de simulation retombe alors en mode attente. Cette manière de faire a pour but de palier au léger délai entre le début du pas de calcul du PPC et l'activation de la séquence du noyau de simulation.

4.6 Simulateur à long pas PPC

Le PPC embarqué dans le Virtex II Pro est utilisé pour implémenter le simulateur à long pas. Malgré ses performances limitées ce processeur a été jugé suffisant pour démontrer

la méthode à pas multiple. Avec un pas de calcul de 50 µs, ce simulateur est en mesure de représenter adéquatement les phénomènes basses fréquences qui caractérisent un réseau électrique opérant à une fréquence inférieure à 200 Hz.

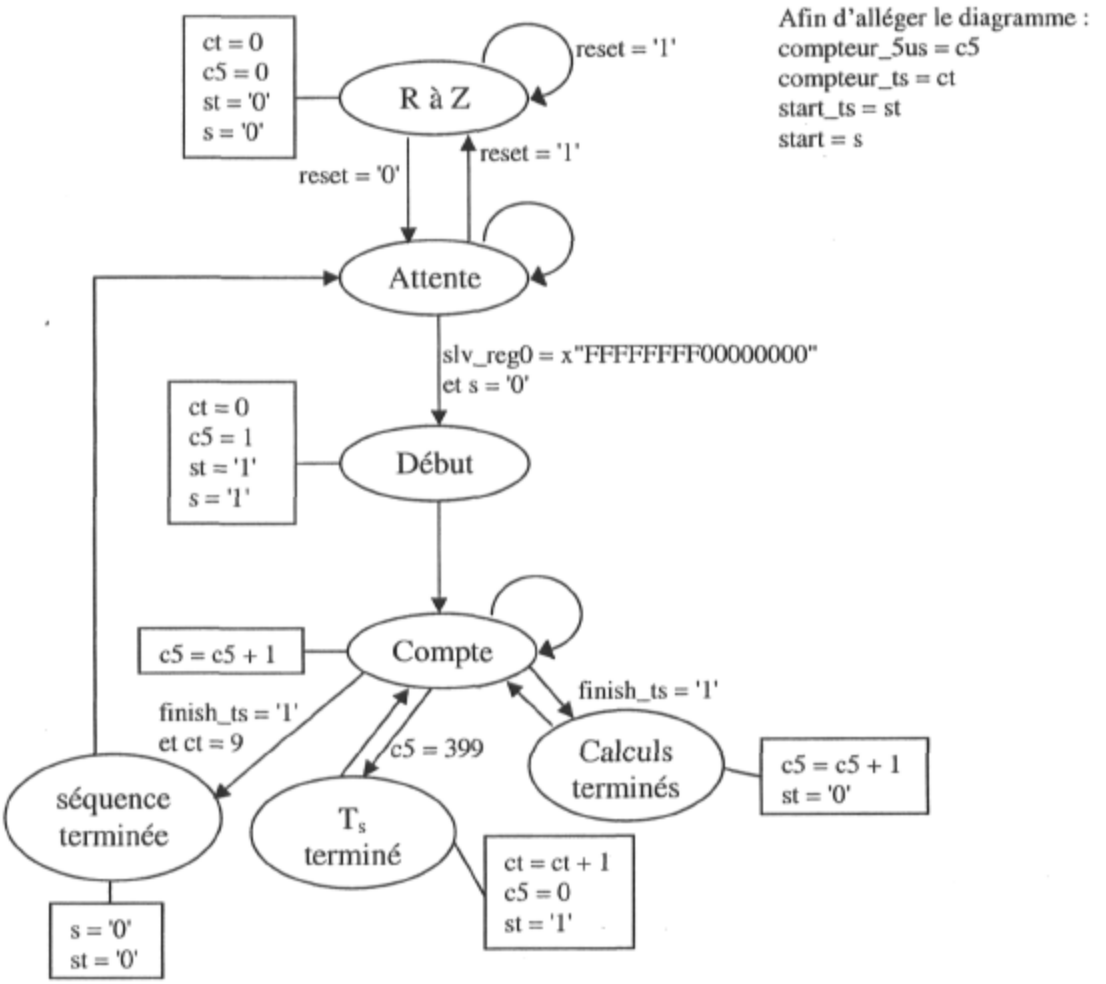


Figure 4.10 Diagramme d'état du cadenceur du noyau de simulation.

Particularités du PPC embarqué

Le PPC embarqué, le *PowerPC 405D5*, est en fait un noyau dur de *PowerPC 405D4* gravé dans le silicium avec un procédé de 130 nm. L'implémentation de ce processeur est optimisée au niveau de la consommation d'énergie, soit une consommation de 0.9 mW/MHz. Cette faible consommation permet de cadencer le processeur à des fréquences

supérieures à 300 MHz. La fréquence maximale d'opération dépend du niveau de vitesse du FPGA. Pour le niveau -6 elle est de 350 MHz tandis que pour le niveau -7 elle est de 400 MHz, ce qui est la fréquence maximale atteignable pour cette architecture [47]. Le système de base utilisé dans le cadre de ce projet est celui provenant du fabricant de la carte de développement FPGA et le PPC est cadencé à 240 MHz. Il a été tenté d'augmenter la fréquence d'opération du processeur mais l'augmentation de cette fréquence entraîne inévitablement l'augmentation de la fréquence d'opération du bus local au processeur (Processor Local Bus, PLB) et du bus des périphériques embarqués (On-chip Peripheral Bus, OPB) ce qui créait des problèmes de minutage et de routage pour certains périphériques, rendant ainsi le système de base inopérant.

Le *PPC 405D5* est un processeur RISC de 32 bits. Il est programmé en C et supporte les instructions *PowerPC User Instruction Set Architechture* (UISA) ainsi que les instructions d'arithmétiques à virgule fixe de 32 bits. Toutes les opérations de 64 bits, les instructions aux processeurs auxiliaires ainsi que les opérations en virgule flottante sont émulées au niveau logiciel par le compilateur [47]. Cependant l'émulation de ces opérations entraîne un temps d'exécution extrêmement long dans le cadre d'une application qui doit s'exécuter en moins de 50 µs. Par exemple, l'exécution de $\sin(x)$, où x est une valeur de 64 bits en virgule flottante, nécessite plus de 174 µs. Il est alors évident qu'il est impossible d'utiliser ce processeur pour traiter des nombres en virgule flottante dans le cadre de ce projet.

Tel que mentionné précédemment, la représentation des nombres utilisée dans le noyau de simulation est une notation de 32 bits en virgule fixe en complément à deux. Puisque le PPC n'est en mesure que de traiter des nombres en virgule fixe et que c'est un processeur de 32 bits, il a été décidé de travailler avec la même représentation dans le simulateur à long pas que dans le noyau de simulation. Ainsi, aucune conversion, et aucun délai outre le temps de communication, ne seraient nécessaire entre les deux simulateurs.

Algorithme de simulation du PPC

Le PPC exécute un algorithme nodal avec la même modélisation que dans le noyau de simulation à l'exception de la méthode d'intégration. À cause de la largeur du pas de calcul, 50 µs, la méthode Euler arrière ne peut être utilisée. La méthode trapézoïdale permet d'obtenir une précision acceptable avec cette valeur de pas de calcul et puisqu'il n'y a pas de commutation dans le réseau simulé, les oscillations numériques sont absentes. L'algorithme implémenté est illustré à la figure 4.11 On constate aisément la similitude avec l'algorithme du noyau de simulation. L'activation du noyau de simulation correspond à l'envoi du code d'activation qui amorce la séquence de simulation du noyau (voir figure 4.10). Cette opération a lieu avant le début de l'algorithme du PPC. Ce dernier doit générer les ondes sinusoïdales des sources de tension alternatives qu'il contient. Un développement en série, jusqu'au sixième terme, est utilisé pour calculer la valeur de $\sin(x)$. Le reste de l'algorithme est similaire à celui du noyau de simulation à l'exception que toutes les étapes sont réalisées en série. Le transfert d'information entre les deux parties a lieu à la fin de chaque pas de calcul puis tous les résultats de simulation sont stockés dans une mémoire RAM. Le contenu de cette mémoire est transféré à une station de travail via le port série à la fin de la simulation. Les données obtenues sont ensuite converties et analysées dans l'environnement *MATLAB*. Finalement, la synchronisation de toutes les opérations est effectuée à l'aide du compteur d'intervalle programmable.

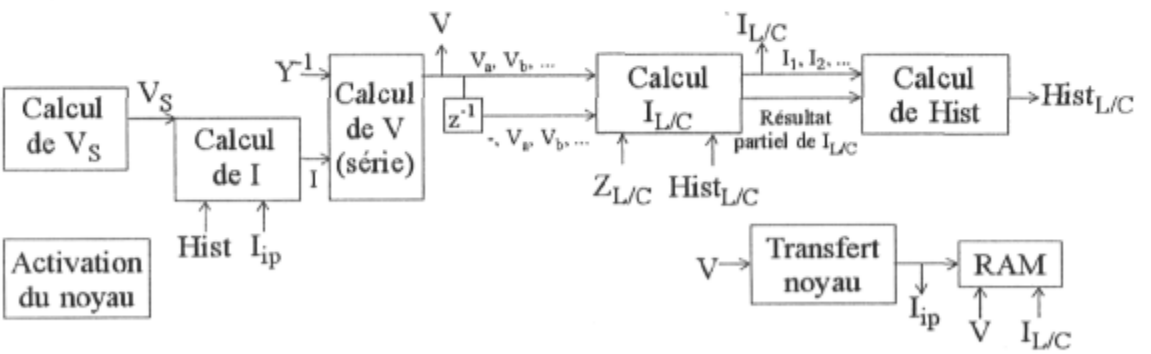


Figure 4.11 Diagramme fonctionnel de l'algorithme de simulation du PPC.

Puisque le *PowerPC 405D5* n'est pas un processeur très performant, il a fallu optimiser certaines opérations de base afin d'exécuter l'algorithme de simulation en temps réel. Ces

opérations ont été codées directement en assembleur afin de minimiser leur temps d'exécution telles les multiplications, qui, à cause de la notation utilisée, nécessitent plusieurs décalages ainsi que les communications avec le noyau de simulation et avec la mémoire RAM.

4.7 Couplage entre le PPC et le noyau de simulation

Le transfert des variables de tension et de courant entre les deux sous-systèmes est effectué à l'aide de sources de courant et de tension commandées tel qu'illustré à la figure 4.12. Cette méthode est similaire à celle utilisée pour effectuer le couplage numérique-analogique dans les simulateurs hybrides. Cette technique introduit un délai d'un pas de calcul entre les deux sous-systèmes ce qui peut avoir des impacts importants sur la dynamique du système global et possiblement introduire des oscillations numériques. C'est pourquoi le point de découplage entre les deux sous-systèmes est placé à proximité d'éléments réactifs. Ainsi, si la constante de temps de ces éléments est plus longue que le délai de découplage, la dynamique de l'élément aidera à maintenir la stabilité des courants et des tensions

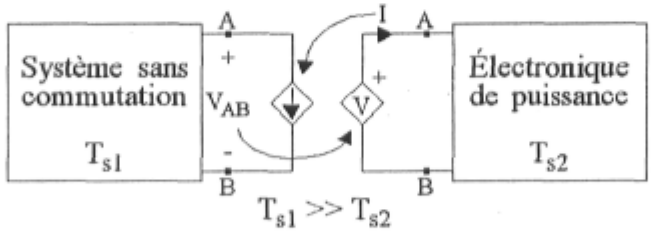


Figure 4.12 Couplage entre les deux parties du simulateur à pas multiple.

Implémentation FPGA du découplage

Afin d'implémenter le découplage, des sources commandées de tension sont placées dans le réseau simulé dans le noyau de simulation. Les valeurs de tension de ces sources commandées proviennent du PPC et correspondent aux valeurs de tension aux points de découplage. Le PPC actualise ces valeurs en les écrivant dans les registres appropriés du

noyau de simulation. Ce dernier utilise ces valeurs pour ses dix pas de calcul (5 µs) suivants. Les sources de courant commandées contenues dans le réseau sans commutation prennent la valeur du courant sortant des sources commandées de tension contenu dans le noyau de simulation. Suite à l'écriture des tensions aux points de découplage, le PPC lit dans les registres correspondant les valeurs de courant pour le prochain pas de 50 µs.

Ces transferts sont effectués à l'aide du bus de communication local au processeur (*Processor Local Bus* (PLB)) qui est un noyau *CoreConnect* de Xilinx. Le noyau n'a pas été modifié et le mode standard de communication est utilisé. Ce bus de communication est cadencé à 80 MHz. Un échange de trois tensions et de trois courants nécessite environ 2.7 µs. Ce laps de temps tient compte du temps d'accès à la mémoire du PPC en lecture, pour les tensions, et en écriture, pour les courants, en plus du temps de communication avec le noyau de simulation. Il est important de noter que les fonctions logicielles de communication ne sont pas utilisées car l'appel d'une fonction implique un délai non négligeable. En réalisant une lecture/écriture directe dans le noyau de simulation, qui est perçu par le PPC comme un périphérique adressable, il est possible d'économiser près de 0.4 µs afin d'obtenir le temps de communication de 2.7 µs mentionné précédemment.

La limite inférieure théorique de la longueur du pas de calcul de la partie rapide est égale à la somme du temps de communication entre les deux simulateurs et du temps d'exécution de l'algorithme du noyau de simulation. Puisque le noyau de simulation requiert moins de 300 ns pour exécuter son algorithme, la principale limitation dans le choix du pas de calcul de la partie rapide est le temps de communication. En effet, s'il était possible de réduire le laps de temps nécessaire pour le transfert des données, il serait possible d'augmenter la plage de fréquence représentable.