ELE3311 – Systèmes logiques programmables
Projet n°3

HORLOGE NUMÉRIQUE AFFICHAGE OLED

Hiver 2019

Introduction

Le but de ce laboratoire est d'approfondir des concepts fondamentaux des systèmes logiques programmables. Ce projet consiste à réutiliser le coeur d'une horloge numérique réalisée lors du projet 2, soit la logique des heures, minutes et secondes, ainsi que le contrôleur pour programmer cette horloge à partir de quelques boutons. À ce coeur, l'affichage sur un écran OLED est ajouté. Des fonctionnalités sont aussi comprises afin de faciliter les simulations et la vérification du système. Les concepts approfondis lors de ce projet sont :

- l'utilisation de deux domaines d'horloge;
- la génération des horloges avec le *clock wizard*;
- la remise à zéro du système (*reset*);
- l'implémentation de mémoire (par inférence);
- l'analyse de rapport de synthèse;
- l'optimisation pour la performance.

Description de la fonctionnalité

Le système à concevoir est une horloge numérique avec affichage sur l'écran OLED de la carte de développement Nexys Video. Une DEL est réservée à la trotteuse qui clignote une fois par seconde. L'affichage sur l'écran OLED se fait dans le format 24h : HH:MM|SS. Huit caractères distincts doivent être dessinés sur l'écran OLED en tout temps: deux chiffres représentant les heures, deux-points entre les heures et les minutes, deux chiffres représentant les minutes, une barre verticale séparant les minutes des secondes et, finalement, deux chiffres représentant les secondes. Le code décrivant la ROM contenant ces caractères vous est fourni. L'affichage sur l'écran doit ressembler à ceci:

La programmation de l'horloge numérique est faite à l'aide des boutons poussoirs. Le bouton mode (***mode_i***) permet de passer du mode affichage aux différentes étapes de programmation pour ensuite revenir à l'affichage tel que décrit ci-dessous:

- À la mise en service, le système est en mode **affichage**. Dans ce mode l'écran OLED affiche les heures, les minutes, les secondes en continu. Dans ce mode, l'heure incrémente avec le passage du temps, tel qu'on s'attend d'une horloge.
- Lorsque le bouton mode est appuyé une fois, le système passe à la **programmation des heures**. Pendant la programmation, le passage du temps n'incrémente pas l'heure. On peut utiliser deux boutons pour programmer l'heure, soit le bouton du haut pour faire des incréments de 1 et le bouton de gauche pour faire des incréments de 10. Un incrément de 1 doit faire passer le compteur par toutes les valeurs possibles et revenir à 0 après 23. Un incrément de 10 incrémente les dizaines, en cas de dépassement de la valeur maximale, les dizaines sont remises à 0.
- Le troisième mode est celui de **programmation des minutes**. Le fonctionnement est le même que pour la programmation des heures. Un incrément de 1 fait passer le compteur par toutes les valeurs possibles et revenir à 0 après 59. Un incrément de 10 ne modifie que les dizaines.

On cycle sur ces trois modes à l'aide du bouton du centre. De plus, lorsque l'horloge retourne en mode affichage, le compteur des secondes est redémarré.

Afin de faciliter la vérification de la fonctionnalité, deux interrupteurs permettent d'augmenter la cadence du compte. La cadence est choisie à l'aide des interrupteurs 1 et 0 de la carte :

SW[1:0]	**00**	**01**	**10**	**11**
Cadence	1 Hz	4 Hz	8000 Hz	1 MHz
Fonction	Horloge numérique	Vérification des minutes	Vérification des heures	Réduction du temps de simulation

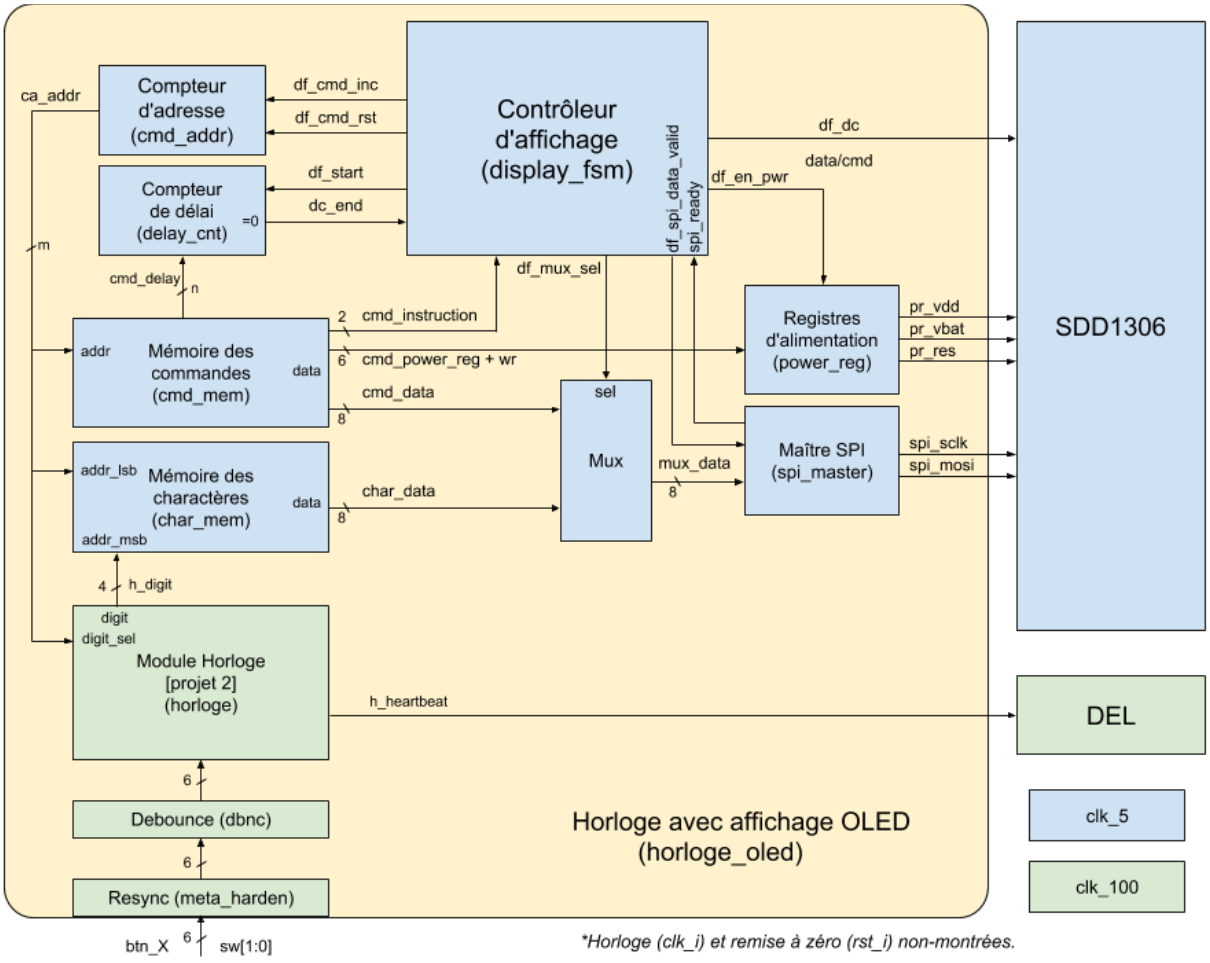
Le bouton ***cpu_resetn*** réinitialise le système (machines à états, diviseur d'horloge, compteurs).

Les entrées et sorties du niveau hiérarchique supérieur sont données dans le tableau ci-dessous.

Nom	**Largeur**	**Direction**	**Description**
rst_i_n	1	entrée	Réinitialisation du système. Bouton ***cpu_resetn***.
clk_100mhz_i	1	entrée	Horloge 100 MHz.
mode_i	1	entrée	Changement de mode (affichage, programmation). Bouton du centre (**BTNC**).
inc_one_i	1	entrée	En mode programmation, incrémente la valeur de 1. Bouton du haut (**BTNU**).
inc_ten_i	1	entrée	En mode programmation, incrémente les dizaines de 1. Bouton de gauche (**BTNL**).
speed_i	2	entrée	Contrôle de la cadence. Interrupteurs de la carte (**SW[1:0]**)

ho_dels_o	8	sortie	Diodes électroluminescentes (DEL) de la carte.
ho_df_dc	1	sortie	Data/Commande vers le OLED
ho_pr_vdd	1	sortie	Alimentation du OLED
ho_pr_vbat	1	sortie	Entrée du *charge pump* du OLED
ho_pr_res	1	sortie	Entrée *res* du OLED
ho_spi_sclk	1	sortie	SPI horloge sérielle
ho_spi_mosi	1	sortie	SPI *master out slave in*

Le schéma ci-dessous montre l'architecture du système. Les sections *Compteur d'adresse, Compteur de délai, Contrôleur d'affichage, Maître SPI et horloge_oled* doivent être implémentées. Les sections *Mux, Mémoire des commandes* et *Mémoire des caractères* sont réalisés à l'intérieur du niveau hiérarchique supérieur, soit *horloge_oled*. Vous n'avez pas à faire la section *Registres d'alimentation*.



Horloge et remise à zéro (à réaliser)

Dans le cadre de ce projet, deux horloges sont utilisées. Une horloge de 100 MHz pour les modules *horloge, debounce* et *resync* comme pour le projet 2. Toutefois, le contrôleur de l'écran OLED supporte au maximum une fréquence de 10 MHz pour l'interface utilisée, SPI, un second domaine d'horloge devra donc être utilisé pour les modules en lien avec l'écran OLED. Une fréquence d'opération de 5 MHz a été choisie.
Vous devez générer les deux horloges à partir du *Clocking Wizard* introduit au projet 1. Afin de générer deux horloges, allez sous l'onglet **Output Clocks** et sélectionnez les fréquences désirées. Vous devrez par la suite déclarer et instancier le module à partir du template généré.

La remise à zéro (***rst_i***) est utilisée pour les différents composants du module horloge, soit la machine à états *msa_horloge*, le compteur BCD, le diviseur d'horloge. Dans le cadre du projet 2, nous avons utilisé une remise à zéro asynchrone. Deux problèmes de la remise à zéro asynchrone sont:

- lors de la désactivation de la remise à zéro, s'il est n'est pas synchronisée avec l'horloge, les bascules peuvent devenir métastable;
- lors d'un long chemin de propagation du signal, deux registres pourraient sortir de la remise à zéro à des cycles d'horloge distincts.

Pour éviter ces problèmes, une solution est de synchroniser le signal de remise à zéro. Les différents modules fournis utilisent des bascules avec remise à zéro synchrone. On vous demande donc de réaliser la synchronisation du signal **rst_i_n** en entrée du module *horloge_oled*. À cette fin, vous devez instancier un module de synchronisation *sync_io_1* et ajouter un anti-rebond qui n'est pas sensible aux fronts montants, *antirebond_1b* dans le niveau hiérarchique supérieur.

Horloge (fourni)

Le module horloge est défini dans l'énoncé du projet 2. Il comprend la logique de l'horloge, de sa programmation et un multiplexeur de sortie pour choisir le caractère à afficher. La resynchronisation, et l'anti-rebond sont inclus à l'intérieur du module. La valeur de *digit_sel* correspond à la position du symbole à dessiner sur l'écran. Par exemple, si *digit_sel* vaut 0, digit correspondra à l'offset du premier chiffre représentant les heures. Si *digit_sel* vaut 2, digit correspondra à l'offset représentant le deux-points à dessiner sur l'écran.

Les entrées et sorties du module *horloge* sont données dans le tableau ci-dessous.

Nom	Largeur	Direction	Src/Dst	Description
rst_i	1	entrée	rst_gen	Réinitialisation
clk_i	1	entrée	clk_gen	Horloge 100 MHz
speed_i	2	entrée	dbnc	Sélection de la cadence de mise à jour

btn_i	4	entrée	dbnc	Boutons pour la programmation
digit_sel	3	entrée	cmd_addr	Sélection du caractère à afficher
digit	4	sortie	char_mem	Caractère à afficher en format BCD

L'entrée ***speed_i*** est utilisée pour déterminer la cadence d'activation du calcul. La cadence maximale (1 MHz) ne doit être utilisée qu'en simulation.

Contrôleur d'affichage (à réaliser)

Le contrôleur d'affichage sert à déterminer s'il faut transmettre des commandes ou des données à l'écran OLED en plus de gérer les autres modules en fonctions des commandes à réaliser.

Pour faire fonctionner l'écran OLED, une séquence d'initialisation doit être réalisée. Les commandes vous sont fournies dans la mémoire des commandes *CMD_MEM*. Cette séquence sert à mettre sous tension les différentes sources d'alimentation du contrôleur dans un ordre donné avec suffisamment de temps pour que la tension sur les pins se soit stabilisée avant de réaliser une autre commande. De plus, différents paramètres pour configurer le mode d'affichage de l'écran sont transmis au contrôleur de l'écran OLED, le SSD1306. La séquence des commandes réalisée est décrite à la section *Mémoire des commandes*. Cette séquence n'est réalisée qu'une seule fois, lors du démarrage de la carte, et n'est donc pas exécutée de nouveau lors d'un reset. Par la suite, lorsque la séquence est complétée, le contrôleur d'affichage passe à la transmission des données. Les données sont affichées en continu par la suite. Le contrôleur a donc deux modes d'opération, un mode de transmission des commandes et un mode de transmission des données.

- Le mode transmission des commandes: On exécute la séquence en mémoire jusqu'à ce que l'on ait fini de traiter la dernière commande, ce qui est identifié par le bit le plus significatif de *cmd_instruction_i* qui est à un niveau haut. On passe ensuite en mode transmission des données.
- Le mode transmission des données: Dans ce mode, on transmet en boucle les pixels à afficher sur l'écran. On ne ressort pas de ce mode.

Que l'on soit en mode de transmission des commandes ou de transmission des données, une séquence, d'une durée variable, doit être réalisée pour assurer la transmission de chacune des données. La séquence est de la forme:

- Mise à jour du compteur d'adresse.
- Lecture de la commande et transmission des signaux de contrôle aux différents modules.
- Attente que les modules *spi_master* et *delay_cnt* aient complétés leurs tâches.

Il faut utiliser le même chemin de donnée pour réaliser la séquence dans les différents modes du contrôleur d'affichage. Attention de vérifier que les données sont valides lorsque vous les lisez et que toutes les données sont transmises.

Le module reçoit en entrée deux bits de la commande en mémoire. Ces deux bits sont utilisés dans le mode de traitement des commandes et servent à indiquer si la commande est un délai, niveau bas, ou une transaction SPI, niveau haut. La machine a état doit envoyer un signal au module correspondant pour débuter la transaction, soit via les sorties *df_start_o*, pour le compteur de délai, et *df_spi_data_valid_o*, pour le maître SPI. Les entrées *dc_end_i* et *spi_ready_i* permettent de connaître l'état de chacun des deux modules, s'il a accompli sa tâche et s'il est prêt à recevoir une nouvelle donnée l'entrée sera au niveau haut.

L'état dans lequel le contrôleur d'affichage se trouve est une combinaison du mode de transmission, de l'étape à laquelle on est rendu dans la transmission et les valeurs des commandes en mémoire CMD_MEM, qui sont obtenues par le signal *cmd_instruction_i*. Les sorties dépendent donc d'une combinaison de ces trois états.

La sortie *df_dc_o* est une sortie qui est transmise à l'écran OLED et lorsqu'elle vaut 1, les données qui sont transmises à l'écran sont des données à afficher, sinon, ce sont des commandes.
La sortie *df_en_pwr_o* est valide lorsque le module *power_reg* doit prendre en compte la commande qui est lue présentement, sinon il ne change pas la valeur de ses registres.
Les sorties *df_cmd_inc_o* et *df_cmd_rst_o* permettent respectivement d'incrémenter et de réinitialiser le compteur d'adresse.

Nom	Largeur	Direction	Src/Dst	Description
clk_i	1	entrée	clk_5mhz	Horloge 5 MHz
dc_end_i	1	entrée	delay_cnt	Délai complété
spi_ready_i	1	entrée	spi_master	Prêt pour une transaction
cmd_instruction_i	2	entrée	CMD_MEM	[0]: Indique s'il s'agit d'un délai ou une transaction SPI [1]: Indique s'il s'agit de la dernière commande
df_cmd_inc_o	1	sortie	cmd_addr	Incrémenter le compteur d'adresse
df_cmd_rst_o	1	sortie	cmd_addr	Redémarrer le compte du compteur d'adresse
df_start_o	1	sortie	delay_cnt	Démarrer le compteur de délai
df_mux_sel_o	1	sortie	mux	Sélection des données pour le SPI
df_spi_data_valid_o	1	sortie	spi_master	Démarrer une transaction SPI
df_dc_o	1	sortie	sortie	Registre de commande ou de données pour l'écran OLED
df_en_pwr_o	1	sortie	power_reg	clock enable

Compteur d'adresse (à réaliser)

Le compteur d'adresse permet de sélectionner la bonne sortie de ROM. Le signal *cmd_inc_i* incrémente l'adresse de 1 et le signal *cmd_rst_i* réinitialise le compteur (l'adresse) à 0.

Nom	Largeur	Direction	Src/Dst	Description
clk_i	1	entrée	clk_gen	Horloge 5 MHz
cmd_inc_i	1	entrée	display_fsm	Incrémente l'adresse
cmd_rst_i	1	entrée	display_fsm	Réinitialise l'adresse à 0
addr_count_o	9	sortie	rom	L'adresse courante

Compteur de délai (à réaliser)

Le compteur de délai génère un délai encodé dans la ROM. Lorsque le signal *start_i* est activé, le compteur s'initialise avec la valeur de *cmd_delay_i*. Par la suite, le compteur décrémente de 1 jusqu'à ce qu'il atteigne 0. Le signal *end_o* est ensuite activé jusqu'à l'arrivée du prochain signal *start_i*.

Nom	Largeur	Direction	Src/Dst	Description
clk_i	1	entrée	clk_gen	Horloge 5 MHz
start_i	1	entrée	display_fsm	Débute le décompte
cmd_delay_i	19	entrée	rom	Nombre de coups d'horloge à attendre
end_o	1	sortie	display_fsm	Signal de complétion

Mémoire des commandes (à compléter)

La mémoire contenant les commandes d'initialisation du contrôleur de l'écran OLED. La ROM a une sortie de 64 bits. *cmd_data* correspond aux bits 7 à 0. *cmd_delay_i* correspond aux bits 26 à 8. *cmd_power_reg* correspond aux bits 37 à 32. *cmd_instruction* correspond aux bits 39 à 38. Pour initialiser l'écran OLED, la séquence de commandes suivante doit être exécutée:

1. Mise sous tension de VDD, attente de 1 ms.
2. Envoi de la commande send *display off* par SPI, valeur 0xAE.
3. Reset du SSD1306, attente de 1 ms.
4. Fin du reset du SSD1306, attente de 1 ms.

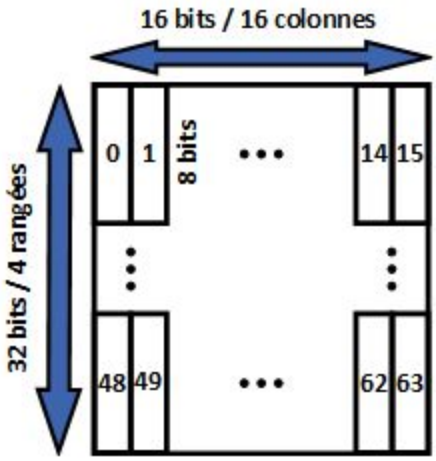
5. Envoi de la commande *enable charge pump* par SPI, valeur 0x8D.
6. Suite de la commande précédente, valeur 0x14.
7. Envoi de la commande *set pre-charge period* par SPI, valeur 0xD9.
8. Suite de la commande précédente, valeur 0xF1.
9. Mise sous tension de VBAT, attente de 100 ms.
10. Envoi de la commande *set contrast control* par SPI, valeur 0x81.
11. Suite de la commande précédente, valeur 0x0F.
12. Envoi de la commande *set segment remap* par SPI, valeur 0xA0.
13. Envoi de la commande *set COM output scan direction* par SPI, valeur 0xC0.
14. Envoi de la commande *set COM pins hardware configuration* par SPI, valeur 0xDA.
15. Envoi de la commande *set lower column start address* par SPI, valeur 0x00.
16. Envoi de la commande *set display on* par SPI, valeur 0xAF.

Ces commandes sont déjà encodées dans la ROM pour vous. Il existe des lignes commentées que vous pouvez utiliser pour la simulation (les délais sont réduits pour éviter une simulation trop longue). Toutefois, vous devez comprendre le lien entre la ROM et les différentes étapes de la séquence précédente, puisque vous devez réaliser un *process* dans le niveau hiérarchique supérieur pour aller lire la mémoire. Il s'agit de recevoir l'adresse en entrée et de fournir la donnée en mémoire à cette adresse. Il s'agit d'un *process* séquentiel.

Nom	Largeur	Direction	Src/Dst	Description
addr	1	entrée	cmd_addr	Adresse de la rom
cmd_data	8	sortie	mux	Donnée SPI
cmd_delay_i	19	sortie	delay_cnt	Délai d'attente avant la prochaine instruction
cmd_power_reg	6	sortie	power_reg	Commandes d'alimentation
cmd_instruction	2	sortie	display_fsm	Signal de complétion

Mémoire des caractères (à compléter)

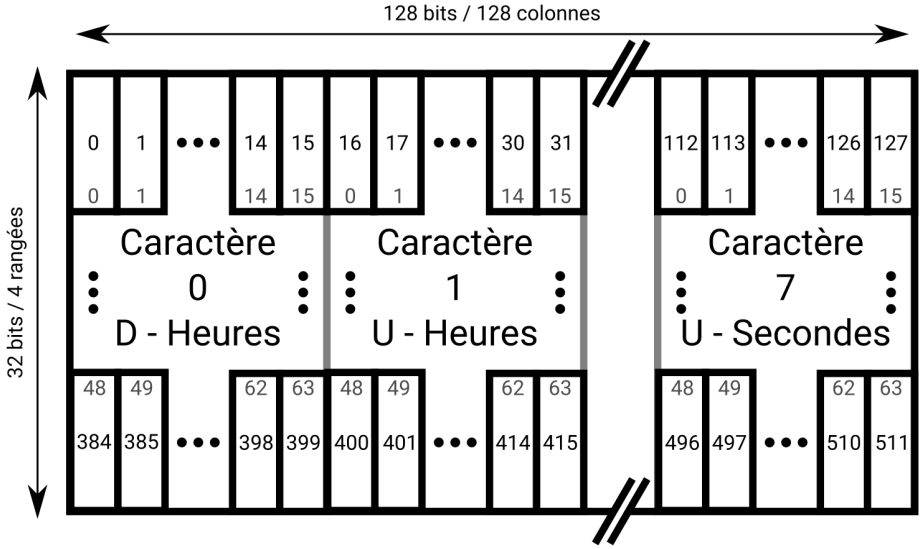
La mémoire contenant les caractères à dessiner sur l'écran OLED. L'écran est de dimension 32x128 pixels. Il est configuré pour écrire 8 pixels verticaux à la fois. L'écriture débute au coin en haut à gauche et se termine en bas à droite. Lorsque la 128e colonne est atteinte, l'écran revient à la 1ere colonne et incrémente la rangée de 1. À la fin de la 4e rangée, l'écran recommence à écrire du début. Chaque caractère est représenté sur 64 adresses (4 rangées de 8 bits par 16 colonnes de 1 bit).



Le caractère “0” débute à l’adresse 0, le caractère “1” à l’adresse 64 et ainsi de suite. Le “deux-points” à dessiner entre les heures et les minutes se situe à l’adresse 640 et la barre verticale à l’adresse 704. Le reste de la mémoire contient des “0”.

Vous devez réaliser le *process* qui ira lire la mémoire à partir des bits d’adresse fournis par le *cmd_addr* et l’*horloge*. Les bits en provenance du *cmd_addr* permettent de déterminer à quelle position, la rangée et la colonne, dans le chiffre, il faut lire les données afin que cela soit en correspondance avec la zone écrite à l’écran. Les bits en provenance de l’*horloge* permettent d’indiquer quel chiffre ou caractère doit être inscrit à cet endroit.
Un sous-ensemble de bits en provenance de *cmd_addr* sont transmis à l’*horloge* afin qu’elle transmette à la mémoire le chiffre ou le caractère à employer.

Voici un schéma de l’écran OLED avec les différents blocs pour chacun des chiffres. L’écran est parcouru rangée par rangée de gauche à droite en débutant au coin supérieur gauche.



Nom	Largeur	Direction	Src/Dst	Description
addr_lsb	6	entrée	cmd_addr	Choix de la colonne et la rangée
addr_msb	4	entrée	horloge	Choix du caractère
data	8	sortie	mux	8 bits représentant 8 pixels

Registres d'alimentation (fourni)

Module qui gère l'alimentation du contrôleur OLED. Les valeurs d'alimentation dépendent des commandes fournies par la ROM *CMD_MEM*. Les bits d'entrée *cmd_pwr_reg* sont, en partant du LSB: *VDD_SET_BIT*, *VDD_BIT*, *VBAT_SET_BIT*, *VBAT_BIT*, *RES_SET_BIT*, *RES_BIT*. Les valeurs *SET* sont des masques permettant de considérer ou d'ignorer la valeur du bit suivant. Par exemple, si *VDD_SET_BIT* est actif, *pr_vdd_o* prendra la valeur de *VDD_BIT*.

Nom	Largeur	Direction	Src/Dst	Description
clk_i	1	entrée	clk_gen	Horloge 100 MHz
en_pwr	1	entrée	display_fsm	Clock enable
cmd_pwr_reg	6	entrée	CMD_MEM	Commandes encodées dans la ROM
pr_vdd_o	1	sortie	sortie	Contrôle d'alimentation du circuit numér. (actif bas)
pr_vbat_o	1	sortie	sortie	Contrôle d'alimentation du circuit analog. (actif bas)
pr_res_o	1	sortie	sortie	Reset du contrôleur (actif bas)

Maître SPI (à réaliser)

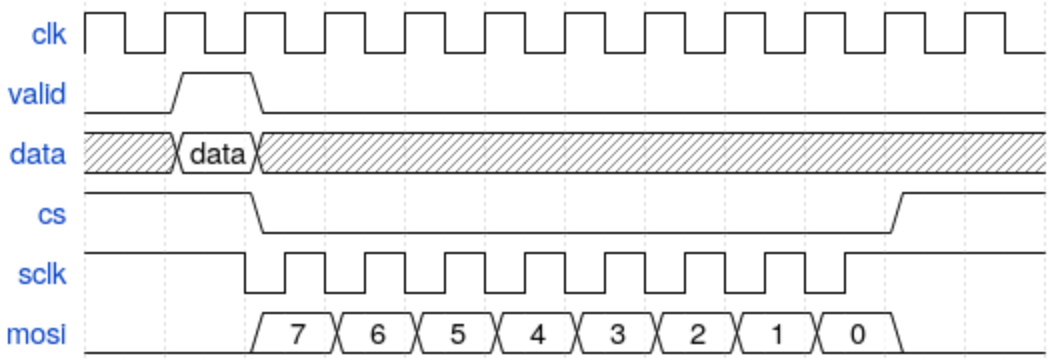
L'écran OLED possède un contrôleur SSD1306 qui permet de configurer l'écran et de fournir les données à afficher. Bien qu'il possède plusieurs interfaces de communication, la seule disponible sur la carte Nexys Video est une interface SPI (*serial peripheral interface*).

Le protocole SPI utilise un contrôleur maître et un ou plusieurs contrôleurs esclaves. Ici, il n'y a que le contrôleur SSD1306 comme esclave. Les signaux importants du protocole sont la sélection de l'esclave(CS)(pas utilisé puisqu'il n'y a qu'un seul esclave), une horloge générée par le maître(SCLK), les données transmises du maître vers l'esclave(MOSI) et les données transmises de l'esclave vers le maître(MISO)(pas utilisé dans ce projet).

De manière générale dans le protocole SPI, le contrôleur maître sélectionne l'esclave auquel s'adressent les données avec un signal CS actif bas. L'horloge transmise entre les deux modules (SCLK) est active lorsqu'une transaction SPI a lieu et elle est inactive le reste du temps (niveau haut). À cette fin, on utilise une "gated clock", c'est-à-dire qu'il s'agit de l'horloge du système,

mais que son fonctionnement est activé par une porte logique. À chaque coup d'horloge(SCLK), une donnée est transmise du maître vers l'esclave (MOSI) et de l'esclave vers le maître (MISO) (pas dans ce projet). Bien remarquer que l'horloge SCLK est inversée par rapport à l'horloge clk.

Le chronogramme suivant présente une séquence d'envoi de 8 bits pour une transaction SPI. *data* est un bus de 8 bits et *mosi* représente un seul bit, la valeur indiquée représente la position du bit dans *data* qui est transmise à ce moment.



De plus, afin de s'assurer de transmettre des données au contrôleur SPI que lorsqu'il est disponible, une interface de *handshake* est utilisée. Dans ce cas-ci, le contrôleur SPI doit fournir un signal actif haut lorsqu'il est disponible à recevoir une nouvelle séquence(ready_o). Cela permettra d'indiquer à la MSA d'attendre que cette transaction SPI soit complétée avant de pouvoir en débuter une nouvelle. Une transaction SPI est débutée lorsque le contrôleur reçoit le signal (valid_i), qui ne dure d'une période d'horloge. Ce signal signifie que les données en entrée(data_i) sont valides et qu'il faut les transmettre sériellement. C'est à dire, qu'il faut registrer ces données lors de la réception du signal valid_i et transmettre un bit à la fois en commençant par le bit le plus significatif.

Nom	Largeur	Direction	Src/Dst	Description
clk_i	1	entrée	clk_5mhz	Horloge 5 MHz
valid_i	1	entrée	display_fsm	Démarre la transmission des données en entrée
data_i	8	entrée	mux	Données à transmettre
ready_o	1	sortie	display_fsm	Le contrôleur spi est disponible
sclk_o	1	sortie	sortie	Horloge pour la communication SPI
mosi_o	1	sortie	sortie	Données transmises sériellement

Des fichiers sources sont fournis et toutes les parties à compléter sont indiquées en commentaires.

Travail au laboratoire

Séance 1 : Schéma bloc des modules *cmd_addr*, *delay_cnt* et *spi_master*. Description HDL des modules *cmd_addr*, *delay_cnt* et *spi_master*. Simulation des modules *cmd_addr* et *delay_cnt.* Ajout de ces modules au niveau hiérarchique supérieur (*horloge_oled*).

Séance 2 : Simulation du module *spi_master*. Diagramme d'états, schéma bloc et description HDL du module *display_fsm*. Ajout de *display_fsm* au niveau hiérarchique supérieur (*horloge_oled*).

Séance 3 : Instanciation au niveau hiérarchique supérieur du module *display_fsm*. Réalisation des process pour lire les mémoires ROM et compléter le niveau hiérarchique supérieur (*horloge_oled*). Débuter les simulations de *horloge_oled*.

Séance 4 : Quiz de 10 minutes à réponses courtes, évaluation du fonctionnement complet sur la plaquette et des simulations du projet complet.

Remarques

- Tout le travail doit être fait en VHDL en utilisant les fichiers fournis pour ce laboratoire.
- Veuillez ajouter le barème de la page suivante dans le rapport.
- Le point du barème de la page suivante indique les éléments à mettre dans le rapport. Ne mettez aucun autre texte qui n'est pas explicitement demandé.
- De l'information supplémentaire est disponible sur les liens suivants :
    - https://reference.digilentinc.com/reference/programmable-logic/nexys-video/reference-manual
    - https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

ELE3311 – Systèmes logiques programmables – Hiver 2019

BARÈME D'ÉVALUATION DU PROJET 3 : HORLOGE AFFICHAGE OLED

No d'équipe : _____

Noms : _____ Matricules : _____

_____ _____

_____ _____

A)	**Évaluations intermédiaires**		**/2**
	Séance 2: Simulation du module *spi_master* et schémas du module *display_fsm*		/2
B)	**Fonctionnement complet (séance 4)**		**/4**
	Simulation du niveau hiérarchique supérieur (*horloge_oled*)		/2
	Fonctionnement sur la carte et rapport de synthèse		/2
C)	**Rapport**		**/10**
	cnt_addr:	schéma bloc, simulation et description VHDL commentée	/1.25
	delay_cnt:	schéma bloc, simulation et description VHDL commentée	/1.25
	spi_master:	schéma bloc, simulation et description VHDL commentée	/2.5
	display_fsm:	diagramme d'état, schéma bloc, simulation[3] et description VHDL commentée	/2.5
	horloge_oled:	simulation, description VHDL commentée, rapport de synthèse	/2.5
D)	**Quiz individuel**		**/4**
E)	**Présentation**		**jusqu'à -2 pts**
TOTAL :			**/20**

Note 1: Pour chaque élément du rapport, lorsque applicable, les points sont séparés comme suit :
- ⅖ : schéma bloc et/ou le diagramme d'état ou rapport de synthèse
- ⅖ : simulation
- ⅕ : description VHDL commentée

Note 2: Ce rapport doit être remis la semaine après la dernière séance du laboratoire, soit le mardi, le jeudi ou le vendredi suivant. Les retards aux évaluations seront pénalisés.
Note 3: Il faut simuler horloge_oled pour montrer le fonctionnement de l'instance de display_fsm.