

Projet Vivado avec HDMI sur Zedboard

Frédéric Fortier

Eva Terriault

INF3610 : Laboratoire 4

Département de génie informatique et génie logiciel

Polytechnique Montréal

Montréal (Québec), Canada

{frederic.fortier,eva.terriault}@polymtl.ca

2 novembre 2017

Table des matières

1	Création du projet Vivado	3
2	Importation dans le SDK	11
3	Ajout du filtre de Sobel à la plateforme matérielle	12
3.1	Dans Vivado HLS	12
3.2	Dans Vivado	12
3.3	Dans le SDK	14
4	Problèmes avec la mise à jour du SDK	15
5	Mise à jour du filtre de Sobel	15
5.1	Dans Vivado HLS	15
5.2	Dans Vivado	15

1 Création du projet Vivado

Note : Ce tutoriel suppose que vous avez déjà acquis quelques connaissances du fonctionnement de Vivado au lab 2. Si vous avez besoin de plus de détails, consultez le tutoriel du lab 2.

Note 2 : Cette section du tutoriel est en bonne partie une mise à jour et traduction du tutoriel HDMI d'AVNET pour Zedboard [1], inclus dans le dossier AVNET_ZED_HDMI.

Note 3 : Une bonne partie du système (surtout vers la fin de cette section) est générée automatiquement avec des scripts qui ont besoin de noms exacts pour différentes composantes. N'ignorez donc pas les instructions demandant de donner un nom particulier ou de renommer une composante.

1. Créez le dossier `C :/TEMP/3610_4/matricule1_matricule2` (aka **ne travaillez pas sur le réseau.**)
2. Copiez le sous-dossier `ip_repo` fourni dans avec cet énoncé (dans CodeFourni) dans le dossier créé à l'étape précédente.
3. Créez un nouveau projet Vivado 2017.2 dans `C :/TEMP/3610_4/matricule1_matricule2`, ciblant le Zedboard.
4. Créez un nouveau *block design*, appelé `design_1` et exécutez son automatisation de connexions.
5. Configurez le Zynq pour qu'il génère 3 horloges différentes (figure 1).
 - (a) FCLK_CLK0 à 75 MHz (pour le contrôle des différents périphériques).
 - (b) FCLK_CLK1 à 150 MHz (pour les composants permettant le transfert de la mémoire vers la sortie HDMI).
 - (c) FCLK_CLK2 à 100 MHz (pour par la suite générer une horloge plus précise à 148.5 MHz pour le périphérique HDMI¹).

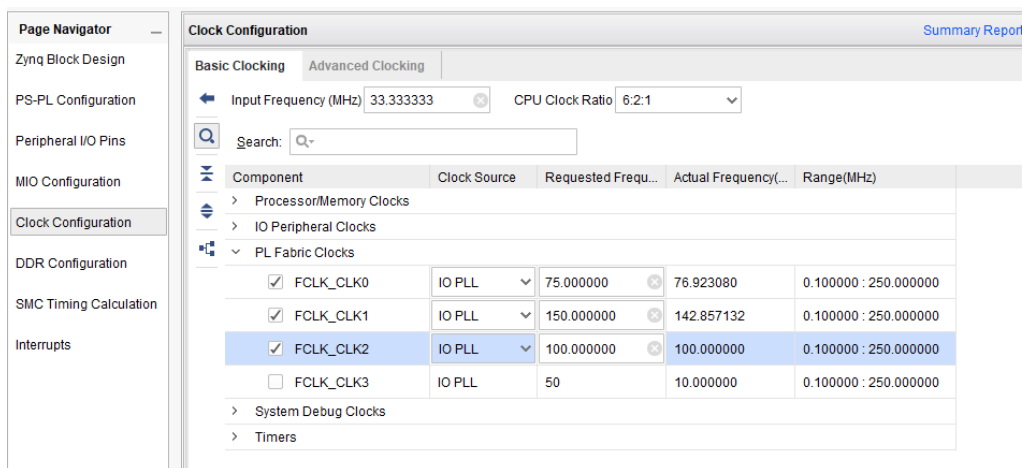


FIGURE 1 – Horloges à générer sur le Zynq

6. Activez aussi le signal de reset FCLK_RESET1_N dans *PS-PL Configuration/General/Enable Clock Resets*.

1. Sans entrer dans les détails, dans le protocole HDMI, la résolution et le taux de rafraîchissement de l'image envoyée au moniteur détermine la fréquence de ce transfert, et donc la fréquence du périphérique gérant ce transfert.

7. Ajoutez une instance de *clocking wizard*, connectez son entrée à l'horloge FCLK_CLK2, puis configurez le bloc pour ressortir une horloge à 148.5 MHz (figure 2), sans reset ni *lock*.

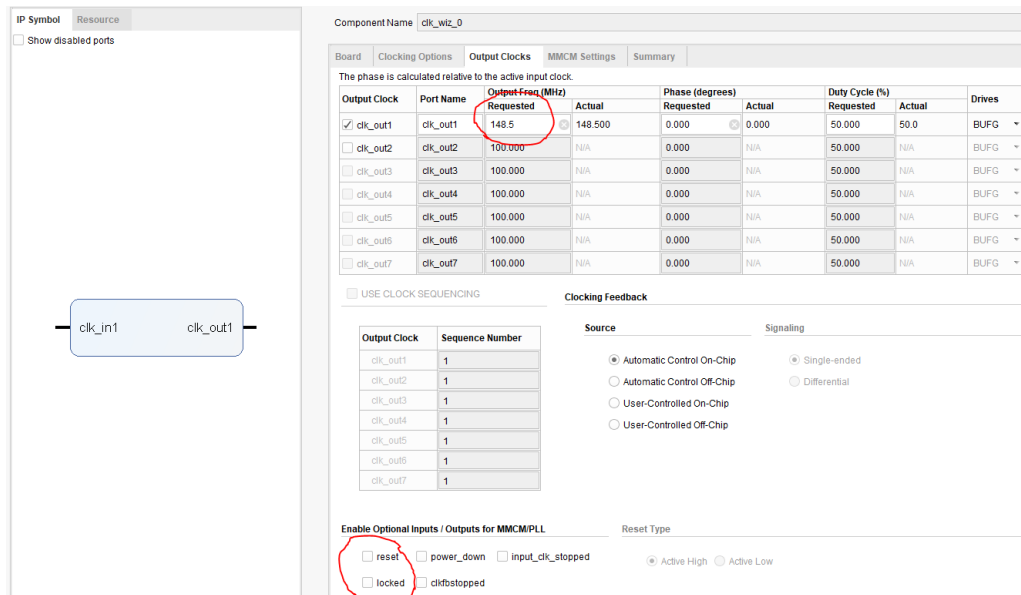


FIGURE 2 – Génération d'horloge à 148.5 MHz

8. Ajoutez un contrôleur I2C (*AXI IIC*), qui sera utilisée pour contrôler le périphérique HDMI et **renommez là pour *zed_hdmi_iic_0***, comme vu à la figure 3

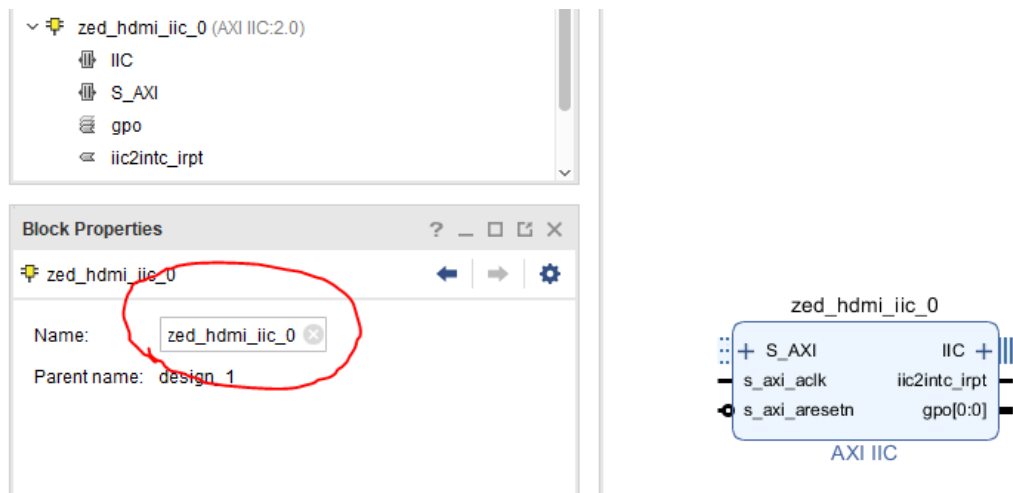


FIGURE 3 – Renommer le bloc I2C

9. Automatisez la connexion de ce bloc de manière à ce que la source des horloges soit **FCLK_CLK0** (figure 4).

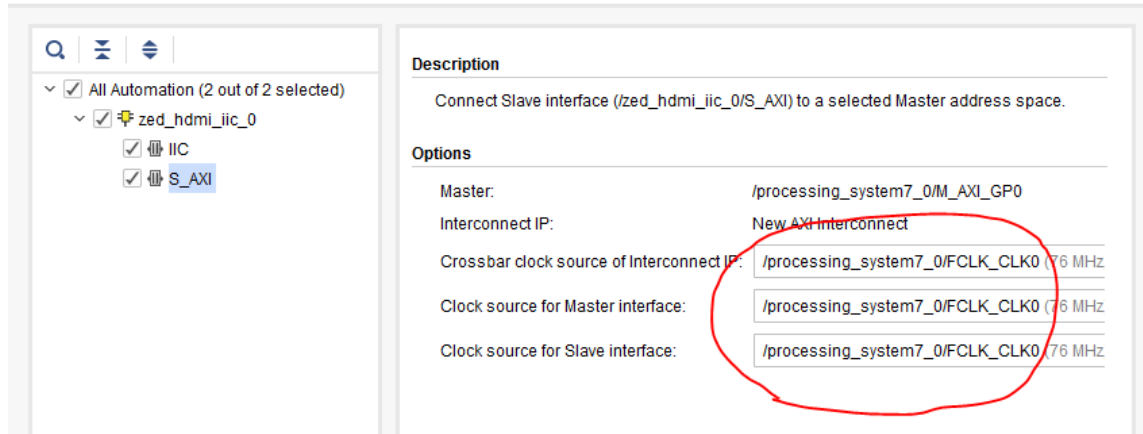


FIGURE 4 – Assignment automatisée à l'horloge FCLK_CLK0

10. **Renommez** la sortie *iic_rtl* générée pour *zed_hdmi_iic*.
11. Validez que votre design est correct² (Menu Tools/Validate Design). Il devrait ressembler à la figure 5.

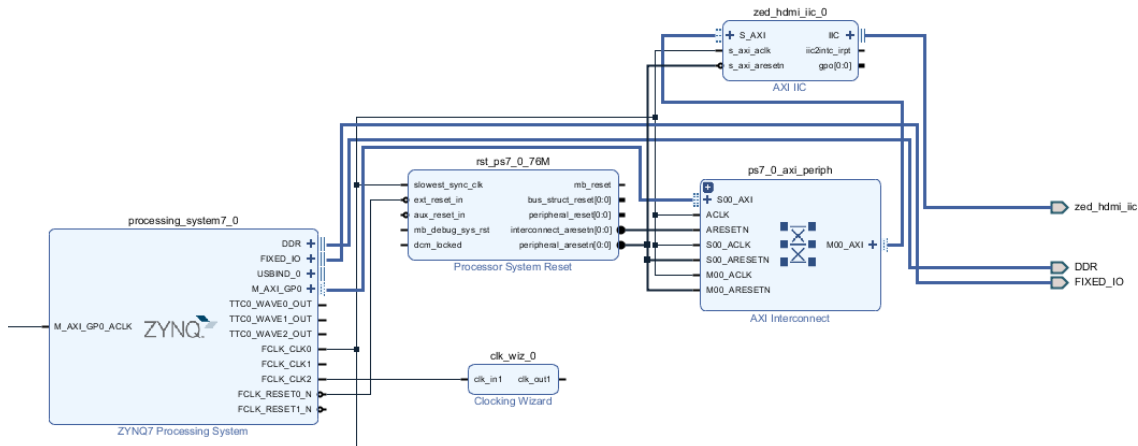


FIGURE 5 – Design avant l'ajout de l'HDMI

12. Pour poursuivre, il est nécessaire d'ajouter des modules qui sont fournis par Avnet (le fabricant
2. l'avertissement sur le *clock skew* de la DDR est normal

du Zedboard) plutôt que Xilinx. Pour se faire, allez dans Tools/Settings et ajoutez une *IP repository* pointant sur le *ip_repo* préalablement copié, comme à la figure 6.

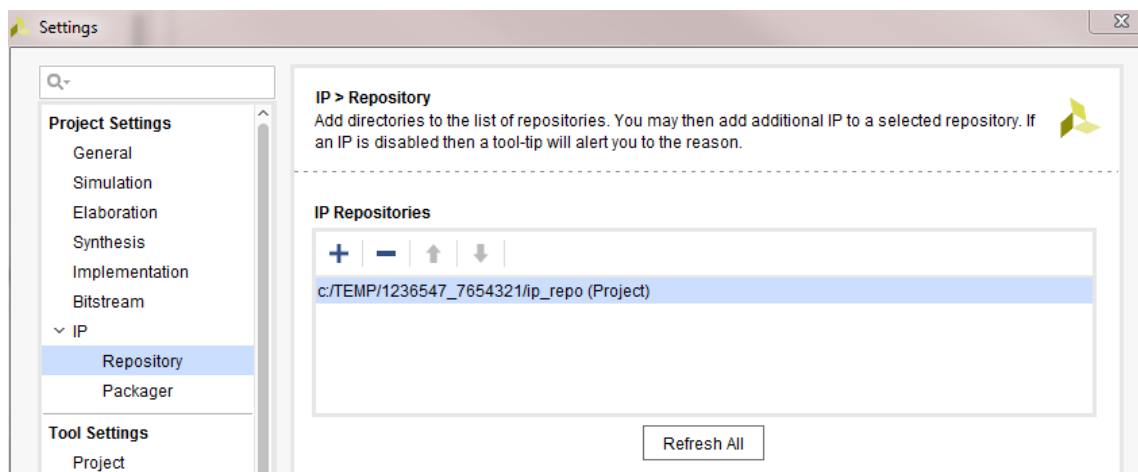


FIGURE 6 – Ajout du répertoire contenant le *core* HDMI

13. Pour simplifier, toutes les composantes permettant la sortie vidéo sont générées par le script *zed_hdmi_display.tcl* dans *ip_repo/AVNET_ZED_HDMI/scripts*. Tapez donc, dans la console TCL, la commande source chemin-vers-le-script/*zed_hdmi_display.tcl* (figure 7). Vous devriez vous retrouver avec un nouveau module nommé *zed_hdmi_display*.



FIGURE 7 – Exécution du TCL

14. *zed_hdmi_display* est en fait un groupe de modules, que vous pouvez voir plus en détail en double-cliquant dessus.
15. Sur le diagramme principal, connectez l'horloge **axi4lite_clk** au port **FCLK_CLK0** du PS et le signal **axi4lite_aresetn** au port **peripheral_arestn[0 :0]** du bloc *Processor System*

Reset.

16. Cliquez sur *Run connection automation* pour connecter l'horloge du `zed_hdmi_display/vtc_ctrl` et du `zed_hdmi_display/vdma_ctrl` à l'horloge **FCLK_CLK0** (figure 8).

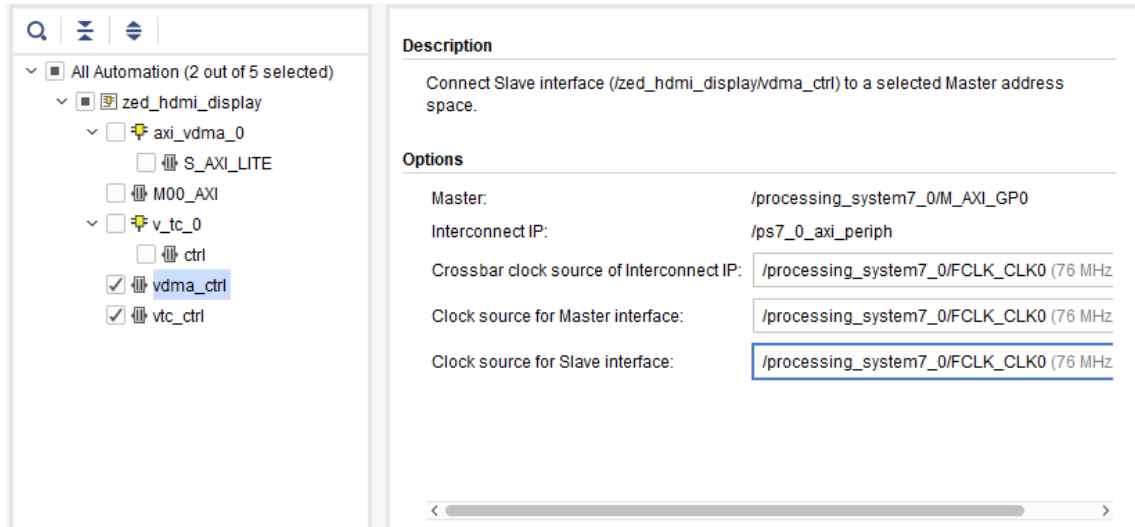


FIGURE 8 – Connexion des contrôleurs à l'horloge **FCLK_CLK0**

17. Connectez l'entrée d'horloge `axi4s_clk` à l'horloge **FCLK_CLK1** du PS et le signal `axi4_resetn` au port **FCLK_RESET1_N** du PS.
18. Connectez l'horloge `hdmio_clk` à la sortie à 148.5 MHz du *Clocking Wizard*.
19. Reconfigurez le PS pour activer le port haute-performance esclave **HP0**, comme à la figure 9.

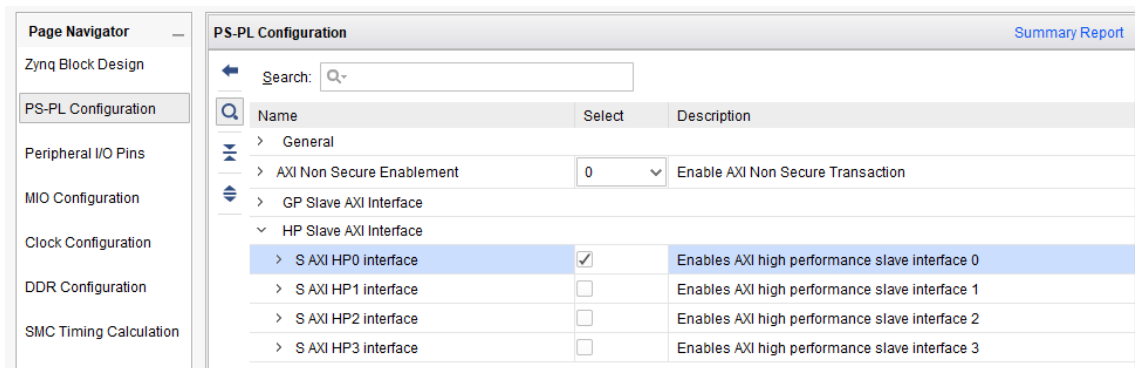


FIGURE 9 – Activation du port esclave HP0

20. Connectez l'entrée d'horloge de ce port (**S_AXI_HP0_ACLK**) à l'horloge **FCLK_CLK1**.
21. Connectez le maître **M00_AXI** du module *zed_hdm_display* à l'esclave **S_AXI_HP0**.
22. Définissez comme externe le port **hdmio_io**, avec un clic droit sur le port, comme à la figure 10.

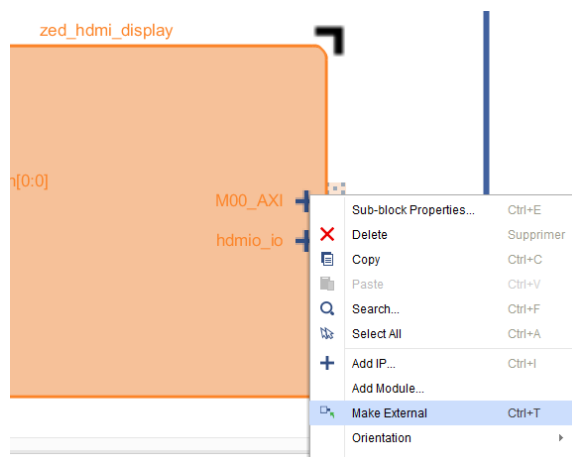


FIGURE 10 – HDMIO externe

23. Allez dans l'éditeur d'addresses, et assignez une plage d'addresses au DMA vidéo (figure 11). Cette plage d'adresse devrait correspondre à l'entièreté de la mémoire DDR du Zedboard (bien que le périphérique n'en utilise que quelques méga-octets de manière non-exclusive pour son *framebuffer*).

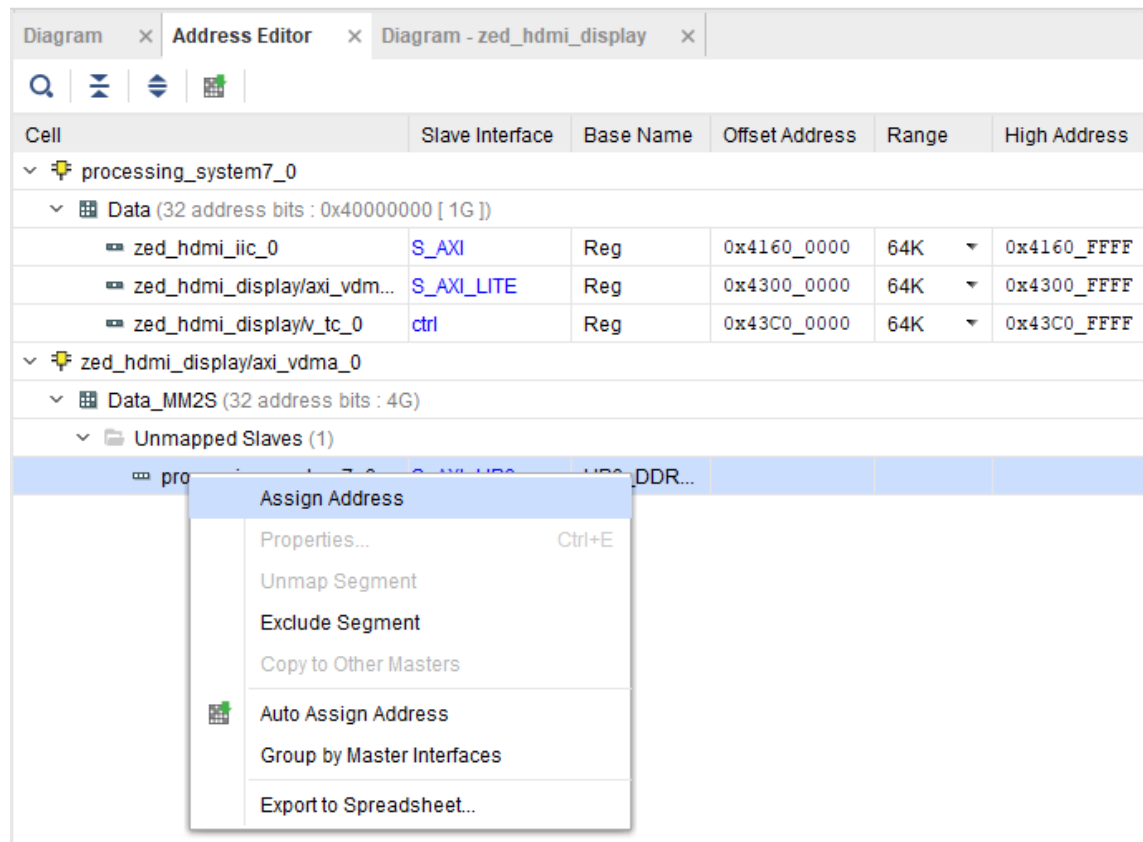
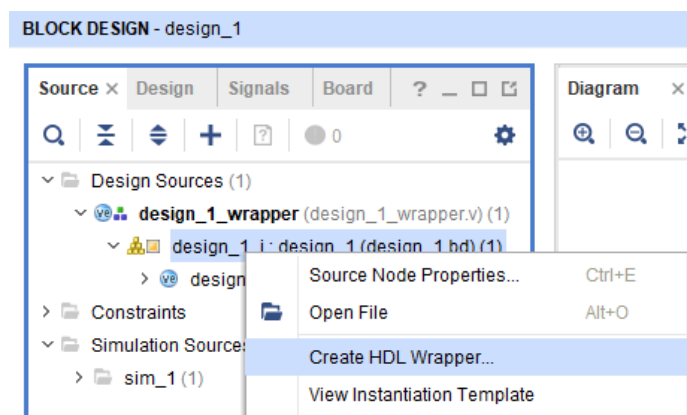


FIGURE 11 – Assignment automatique d'une plage d'adresses.

24. Revalidez le design, puis créez un *wrapper HDL* (comme à la figure 12).

FIGURE 12 – Génération d'un *wrapper* pour le *block design*

25. Il faut maintenant importer le fichier de contrainte, qui assigne les différentes sorties du bloc HDMI (le port `hdmio` et `I2C`) aux bonnes broches du FPGA qui sont connectées à la puce HDMI externe (ADV7511). Pour ce faire :
 - (a) Cliquez sur *Add Sources* dans le *Flow Navigator* (à gauche).
 - (b) Vous voulez créer ou ajouter des contraintes (*Add or create constraints*).
 - (c) Ajoutez le fichier `ip_repo/AVNET_ZED_HDMI/constraints/zedboard_hdmi_display.xdc` (figure 13).
 - (d) Cliquez sur *Finish*.

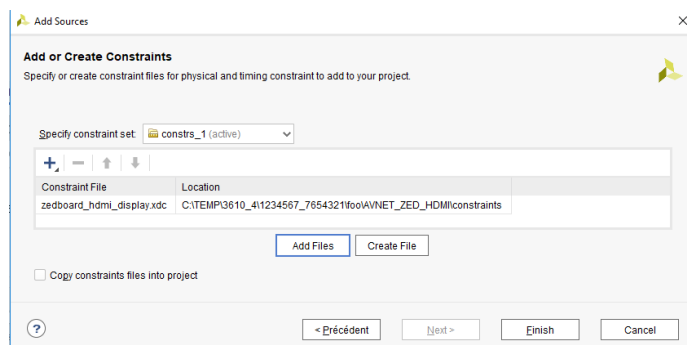


FIGURE 13 – Ajout du fichier de contraintes

26. Générez le bitstream, puis exportez-le vers le SDK.
27. Ouvrez le SDK **via Vivado** pour qu'il importe correctement la plateforme matérielle puis **fermez le SDK**³.

3. Il s'agit ici de contourner le même problème qu'au lab 2, c'est-à-dire que le SDK ne créera pas correctement le BSP s'il est ouvert depuis Vivado. Cependant, le SDK doit être ouvert depuis Vivado pour importer correctement la plateforme matérielle.

2 Importation dans le SDK

Note importante : Ce lab n'en étant qu'à sa première itération, certaines choses ne fonctionnent qu'en majorité. C'est entre autres le cas de la configuration du pipeline d'affichage HDMI (ce qui est à l'intérieur de `zed_hdmi_display` dans Vivado) qui fonctionne bien tant que l'on n'essaie pas de le redémarrer... Donc **vous devez préalablement reprogrammer le bitstream du FPGA (*Program FPGA*) à chaque fois que vous voulez redémarrer une séance de débogage ou d'exécution de votre code.**

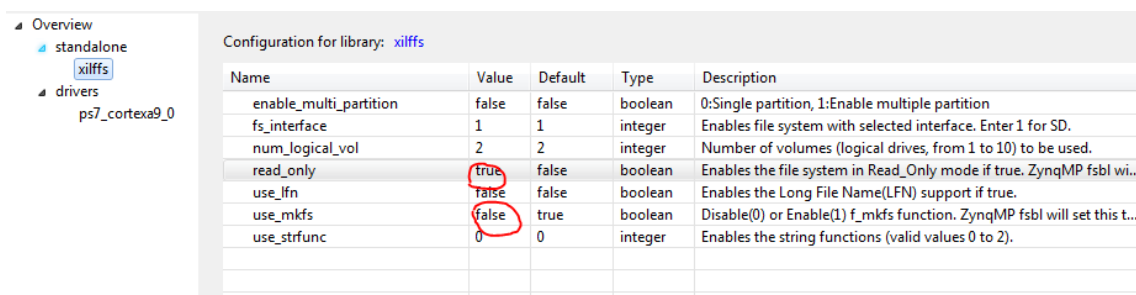
Note très importante 2 : Le lab n'a aussi été que testé à Poly le 2 novembre, pour constater que les écrans dans lequel les Zedboards sont branchés ne supportent pas la résolution de 1920x1080. Après avoir brièvement vérifié que l'année courante était bien 2017 (tirant sur 2018), nous sommes parvenus au *workaround* suivant : **débranchez la sortie HDMI/adaptateur DVI du Zedboard de l'écran 4 :3. Puis, branchez là dans l'écran 16 :9. Pour rester courtois, effectuez la manipulation inverse avant de partir.**

1. Ouvrez le SDK, la workspace pointant vers le dossier créé par l'exportation de Vivado (par ex. `C :/TEMP/1236547_7654321/project_1/project_1.sdk`)
2. Une fois la plateforme matérielle automatiquement importée, créez un nouveau BSP *bare-metal* ciblant cette plateforme.
 - (a) En incluant la librairie *xilffs* (figure 14).
 - (b) En configurant cette librairie comme *read only* et en désactivant la fonctionnalité de création de système de fichiers⁴.
 - (c) Pour faciliter le débogage vous pouvez, comme au lab 2, ajouter `-O0 -g3 -DDEBUG` aux options de compilateur additionnelles.

Name	Version	Description
<input type="checkbox"/> libmetal	1.2	Libmetal Library
<input type="checkbox"/> lwip141	1.8	LwIP TCP/IP Stack library: lwIP v1.4.1
<input type="checkbox"/> openamp	1.3	OpenAmp Library
<input checked="" type="checkbox"/> xilffs	3.6	Generic Fat File System Library
<input type="checkbox"/> xilflash	4.3	Xilinx Flash library for Intel/AMD CFI compliant paral...
<input type="checkbox"/> xilisf	5.8	Xilinx In-system and Serial Flash Library
<input type="checkbox"/> xilmfs	2.3	Xilinx Memory File System
<input type="checkbox"/> xilpm	2.1	Power Management API Library for ZynqMP
<input type="checkbox"/> xilrsa	1.3	Xilinx RSA Library
<input type="checkbox"/> xilskey	6.2	Xilinx Secure Key Library

FIGURE 14 – Inclure la librairie du système de fichier FAT32.

4. Vous optionnellement activer la fonctionnalité de long noms de fichiers (*use_lfn*), qui permet la lecture de fichiers dont le nom est plus long que 8 caractères (+ 3 caractères pour l'extension). Cette option devrait marcher, mais n'a pas été testé. Il peut être plus simple de vérifier que le nom de votre vidéo ne dépasse pas plus de 8 caractères.



Name	Value	Default	Type	Description
enable_multi_partition	false	false	boolean	0:Single partition, 1:Enable multiple partition
fs_interface	1	1	integer	Enables file system with selected interface. Enter 1 for SD.
num_logical_vol	2	2	integer	Number of volumes (logical drives, from 1 to 10) to be used.
read_only	true	false	boolean	Enables the file system in Read_Only mode if true. ZynqMP fsbl wi...
use_lfn	false	false	boolean	Enables the Long File Name(LFN) support if true.
use_mkfs	false	true	boolean	Disable(0) or Enable(1) f_mkfs function. ZynqMP fsbl will set this t...
use_strfunc	0	0	integer	Enables the string functions (valid values 0 to 2).

FIGURE 15 – Configuration de xilffs

3. Créer un nouvel *Application project*, nommé *Lab4* et ciblant le bsp créé précédemment.
4. Ouvrez l'explorateur de fichiers Windows, et remplacez le dossier *project_1.sdk/Lab4/src* par celui fourni *CodeFourni/SDK/src*.
5. De retour dans le SDK, faites un clic-droit sur le projet *Lab4* et cliquez sur *Refresh*.
6. Validez que votre deuxième écran est sur la sortie HDMI du Zedboard.
7. Configurez le FPGA avec son bitstream.
8. Envoyez et démarrez votre application en débogage. Des barres de couleurs devraient apparaître sur l'écran après quelques secondes.

Note : Vous pouvez maintenant retourner à l'énoncé du laboratoire, les sections suivantes ne servant que plus tard dans le lab.

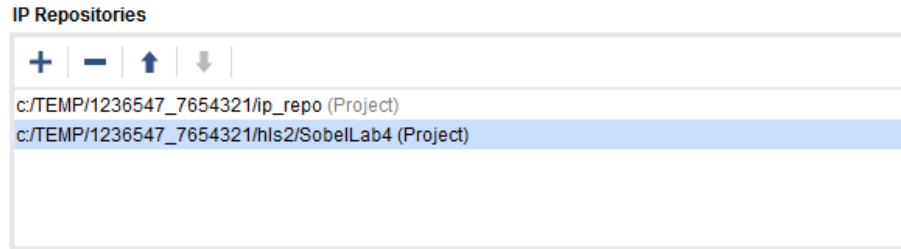
3 Ajout du filtre de Sobel à la plateforme matérielle

3.1 Dans Vivado HLS

1. Synthétisez votre solution, assurez vous que tout est correct.
2. Cliquez sur Export RTL (ou Solution/Export RTL). Laissez les options par défaut.
3. Attendez que la console affiche "Finished export RTL."

3.2 Dans Vivado

1. Ajoutez un *IP Repository* (Tools/Settings/IP) pointant vers votre projet HLS (ex. *C :/TEMP/3610_4/matricule1_matricule2/HLS/SobelLab4*, figure 16)

FIGURE 16 – *IP Repository* avec Sobel

2. Ajoutez une instance de Sobel
3. Dans l'automatisation des connexions, ne sélectionnez que **s_axi_AXILiteS** pour le connecter à **FCLK_CLK0**, comme à la figure 17.

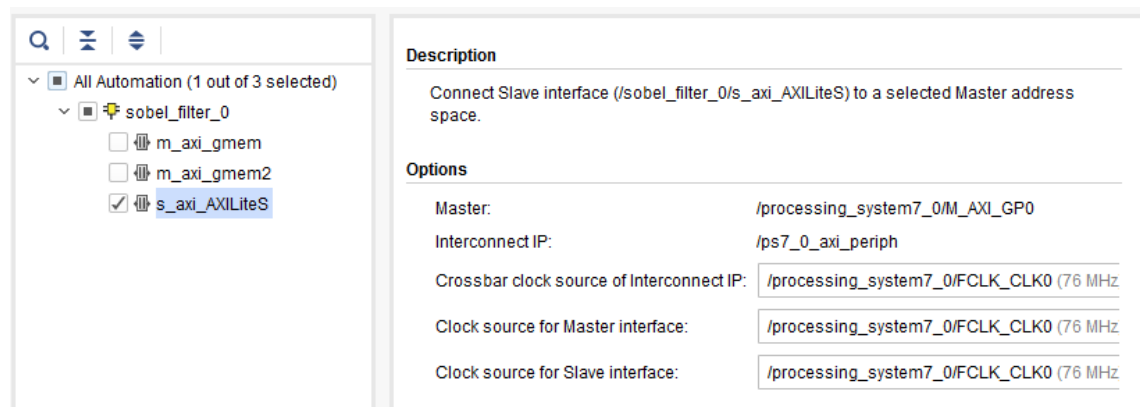


FIGURE 17 – Connexion AXI-Lite automatisée

4. Modifiez la configuration du PS (*Zynq7 Processing System*)
 - (a) Dans *PS-PL configuration*, activez les interfaces esclaves HP1 et HP2.
 - (b) Dans *Clock configuration*, activez *FCLK_CLK3* à 100 MHz.
5. Dans l'automatisation des connexions :
 - (a) Pour **S_AXI_HP1**, configurez son maître comme **sobel_filter_0/m_axi_gmem**, avec un *New AXI Smart Connect* comme interconnexion et sur **FCLK_CLK3**.
 - (b) Faites la même chose pour **S_AXI_HP2**, mais en ayant **sobel_filter_0/m_axi_gmem2** comme maître (figure 18).

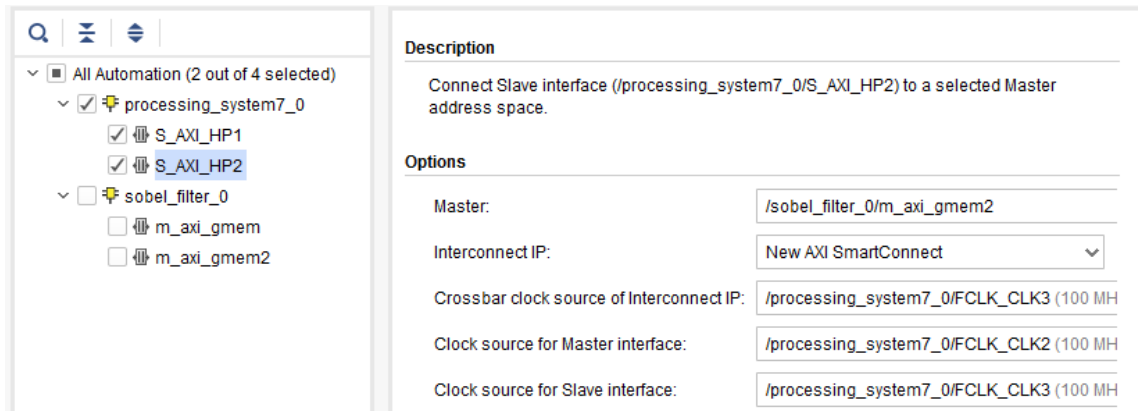


FIGURE 18 – Connexion des ports HP1 et 2

6. Validez le design.
7. Générez un nouveau bitstream.
8. Allez dans le dossier *design_1_wrapper_hw_platform_0* du SDK
(*C:/TEMP/1236547_7654321/project_1/project_1.sdk/design_1_wrapper_hw_platform_0*)
et supprimez le fichier **design_1_wrapper.bit**.
9. Exportez le design vers le SDK (en incluant le bitstream).

3.3 Dans le SDK

1. Le SDK devrait vous mentionner que la plateforme a été mise à jour. Le fichier .bit que vous avez récemment supprimer devrait avoir été recopié.
2. Validez que les paramètre du BSP sont toujours corrects. Notamment, vérifier que stdin/stdout sont toujours sur *ps7_uart_1* (figure 19).

standalone		Configuration for OS: standalone			
xilffs					
drivers					
ps7_cortexa9_0					
Name	Value	Default	Type	Description	
hypervisor_guest	false	false	boolean	Enable hypervisor guest support	
stdin	ps7_uart_1	none	peripheral	stdin peripheral	
stdout	ps7_uart_1	none	peripheral	stdout peripheral	
zynqmp_fsbl_bsp	false	false	boolean	Disable or Enable Optimization f	
microblaze_exceptions	false	false	boolean	Enable MicroBlaze Exceptions	
enable_sw_intrusive_profiling	false	false	boolean	Enable S/W Intrusive Profiling or	

FIGURE 19 – *stdin/out* sur UART

3. Votre BSP mis à jour devrait maintenant contenir le fichier *xsobel_filter.h*, qui contient un driver auto-généré vous permettant de configurer et de faire fonctionner votre filtre. Les paramètres nécessaires à l'initialisation de votre filtre devraient maintenant se retrouver dans le fichier *xparameters.h*.

4 Problèmes avec la mise à jour du SDK

Il arrive souvent et de manière pseudo-aléatoire que la mise à jour du bitstream pose problème/ne soit pas remarquée par le SDK. Si c'est le cas :

1. Fermez le SDK
2. Copiez le dossier *project_1.sdk/Lab4/src* en sécurité.
3. Supprimez le dossier *project_1.sdk*.
4. Refaites l'exportation vers le SDK (incluant le bitstream) dans Vivado.
5. Rouvrez le SDK, recréez le bsp et le projet *Lab4*.
6. Supprimez le nouveau *project_1.sdk/Lab4/src* et remplacez le par vos sources
7. Faites un clic droit sur le projet Lab4 (dans le SDK) et cliquez sur *Refresh*.

Note : cette procédure semble déplaisante, mais vous aurez probablement à la faire 3-4 fois durant le lab. Après la deuxième fois, on s'y fait et ça va beaucoup plus vite.

5 Mise à jour du filtre de Sobel

5.1 Dans Vivado HLS

1. Synthétisez votre solution, assurez vous que tout est correct.
2. Cliquez sur Export RTL (ou Solution/Export RTL). Laissez les options par défaut.
3. Attendez que la console affiche "Finished export RTL."

5.2 Dans Vivado

1. Vivado devrait afficher la notice *IP Catalog is out of date* (figure 20). **Ne cliquez pas sur Refresh** avant que Vivado HLS ait fini d'exporter le RTL ou les portes de l'enfer s'ouvriront sous vos pieds.⁵

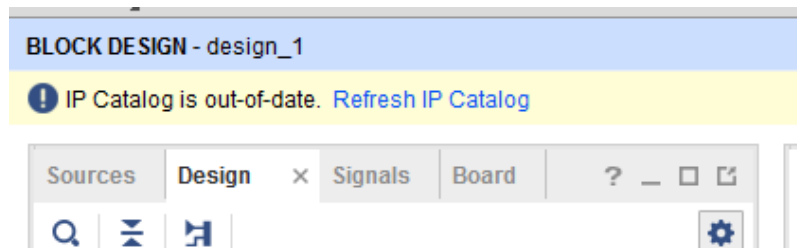


FIGURE 20 – Notification d'*IP out of date*

2. Cliquez sur *Refresh IP Catalog*.

⁵ Ces portes peuvent être refermées en supprimant votre instance de *sobel_filter*, en supprimant la référence à celui-ci dans *l'IP Catalog* (dans les paramètres) puis en remettant le projet dans le catalogue et en recréant une instance de *sobel_filter*.

3. Cliquez sur *Upgrade Selected* (figure 21).

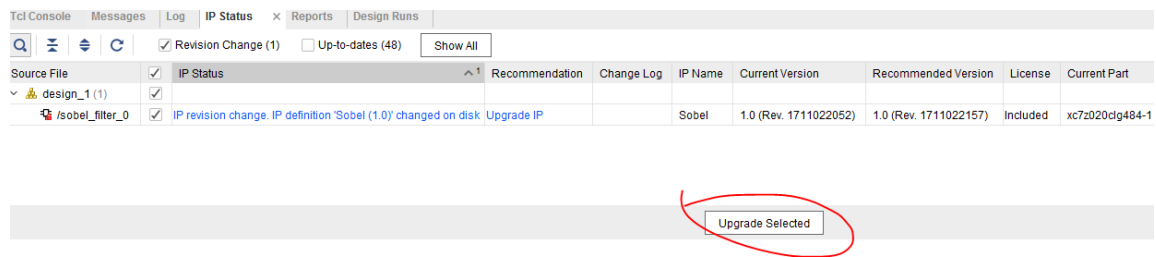


FIGURE 21 –

4. Régénérez le bitstream, ré-exportez vers le SDK. N'oubliez pas de supprimer le fichier .bit dans le sous-dossier de la plateforme matérielle du SDK pour être assurés d'avoir un bitstream à jour.

Références

[1] AVNET, *Zedboard hdmi display controller tutorial*, Version 1.3.