Département de génie électrique
et génie informatique

UNIVERSITÉ LAVAL

GIF-4202 : Conception de systèmes VLSI
Automne 2022
Laboratoire 3: Protocole UART et gestion de flux d'image

Professeur : A. Miled

---

1. Directives particulières

1. Durée du laboratoire : 2 semaines du 12 octobre au 7 novembre à 8h30.

2. **La présence aux séances de laboratoire est obligatoire. Toute information complémentaire présentée verbalement durant les séances de laboratoires et cours ou sur le site web du cours fait partie de l'énoncé écrit et doit être prise en considération.**

3. Une pénalité de 10% sera appliquée pour chaque jour de retard.

4. Les rapports doivent être soumis en ligne à travers le portail des cours.

5. Vous devez soumettre un seul répertoire compressé et nommé LAB3_VOTRE_NOM_ET_PRENOM.

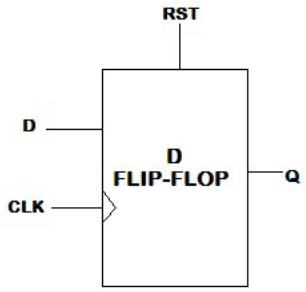
6. Pour chaque exercice, vous devez créer un projet dont le nom est le numéro d'exercice. Normalement, Vivado place chaque projet dans un répertoire ayant le même nom que le projet. Assurez-vous que c'est le cas.

7. Pour chaque exercice, vous devez effectuer la simulation des modules. **CHAQUE EXERCICE DONT LA VALIDATION N'EST PAS OBLIGATOIRE SERA VALIDÉ EN SIMULATION.** Veuillez respecter le gabarit d'entrée/sortie fourni pour ces exercices, car les modules seront testés avec un testbench préconçu. Le non-respect de ce gabarit peut entraîner une perte de 10% sur la note finale de ce laboratoire. La validation en simulation se fera lors de la correction et non lors de la validation en laboratoire. Notez qu'il est judicieux de tester tout de même les modules qui ne seront pas évalués en simulation.

8. Pour chaque exercice, insérez le rapport sommaire du projet contenant le rapport de timing et l'utilisation des ressources du FPGA.

9. Pour chaque exercice, vous devez fournir un schéma expliquant les connexions entre les différents circuits.

10. Vous pouvez faire les diagrammes blocs fonctionnels manuscrits ou avec un logiciel spécialisé comme Microsoft VISIO. Les étudiants qui soumettent des diagrammes fonctionnels faits avec un logiciel auront **5% bonus.**



11. Tous les fichiers des diagrammes, figures et réponses aux questions doivent être soumis en ligne directement dans le répertoire du lab. Nommez chaque fichier comme l'exemple suivant: DOCUMENTATION_NUMERO_EXERCICE.pdf

12. TOUS LES DOCUMENTS EXPLICATIFS ET RÉPONSES AUX QUESTIONS S'IL Y A LIEU; SOUMIS EN LIGNE; DOIVENT ÊTRE EN **PDF. TOUT AUTRE FORMAT NE SERA PAS CONSIDÉRÉ.**

13. Les travaux et rapports sont **OBLIGATOIREMENT** en équipe de 2 pour ce laboratoire.

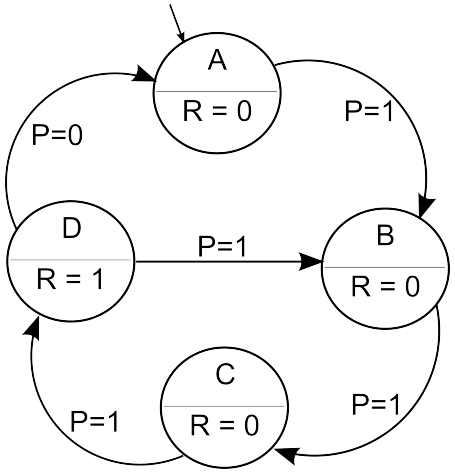
14. Il vous est recommandé d'utiliser le forum pour poser vos questions et suggérer des réponses aux questions posées.

15. La carte Zybo est munie d'une horloge de 125 MHz. Cette horloge est connectée à l'entrée K17 sur le FPGA.

16. L'utilisation des **variables** et des **fonctions** en VHDL est interdite.

17. Si les exercices avec **VALIDATION OBLIGATOIRE** ne sont pas validés lors des séances de validation, l'étudiant perdra 50% de la note attribuée à l'exercice.

18. Vous devez fournir un diagramme explicatif de chaque machine à états. Ci-dessous, un exemple de machine à états finis.



2. Politique sur le plagiat et la fraude académique

2.1. Règles disciplinaires

Tout étudiant qui commet une infraction au Règlement disciplinaire à l'intention des étudiants de l'Université Laval dans le cadre du présent cours, notamment en matière de plagiat, est passible des sanctions qui sont prévues dans ce règlement. Il est très important pour tout étudiant de prendre connaissance des articles 28 à 32 du Règlement disciplinaire. Celui-ci peut être consulté à l'adresse suivante:
https://www.fsg.ulaval.ca/fileadmin/fsg/documents/PDF/POLITIQUE_EN_MATIERE_DE_PLAGIAT.pdf
https://www.ulaval.ca/fileadmin/Secretaire_general/Reglements/Reglement_disciplinaire_a_l_intention_des_etudiants_CA-2016-91.pdf

2.2. Plagiat

Tout étudiant est tenu de respecter les règles relatives au plagiat. Constitue notamment du plagiat le fait de:

- copier textuellement un ou plusieurs passages provenant d'un ouvrage sous format papier ou électronique sans mettre ces passages entre guillemets et sans en mentionner la source;
- résumer l'idée originale d'un auteur en l'exprimant dans ses propres mots (paraphraser) sans en mentionner la source;
- traduire partiellement ou totalement un texte sans en mentionner la provenance;
- remettre un travail copié d'un autre étudiant (avec ou sans l'accord de cet autre étudiant);
- remettre un travail téléchargé d'un site d'achat ou d'échange de travaux scolaires.
  L'Université Laval étant abonnée à un service de détection de plagiat, il est possible que l'enseignant so umette vos travaux pour analyse.

3. Objectifs du laboratoire

1. Comprendre le fonctionnement d'une caméra
2. Comprendre le fonctionnement de certains IP core
3. Maîtriser l'utilisation de la commande generic/generate
4. Découvrir certains aspects du traitement d'image
5. Comprendre le fonctionnement du module de transmission UART
6. Introduire l'étudiant à l'utilisation de l'outil de « Block Design » (Vivado)

4. Registre à décalage multifonction N bits VALIDATION EN SIMULATION (2 points)

Dans cet exercice, vous développerez un registre à décalage permettant à la fois de charger une donnée et de faire le décalage des bits. Ce registre à décalage servira à l'implémentation du module de transmission UART.

Important : **Veuillez respecter le nom du fichier et des entrées/sorties du registre à décalage.**

Contraintes :

C4.1 Vous devez obligatoirement implémenter le registre à décalage avec des registres à 1 bit et implémenter les multiplexeurs dans un **composant indépendant.**
C4.2 Vous devez **obligatoirement** utiliser les instructions **generic/generate** afin de créer un registre à décalage dont la longueur est variable.
C4.3 Lors de l'utilisation de ce registre pour le module de transmission UART, celui-ci aura une longueur de 8 bits. Par contre, la valeur par défaut de **N** doit être 12.

- Créez un fichier nommé **rdc_load_Nbits.vhd**. Ce fichier contiendra le registre à décalage multifonction. Implémentez le registre à décalage multifonction sur N bits à l'aide des opérateurs **generic** et **generate**. Le registre à décalage possède une entrée **RESET** permettant la réinitialisation des registres. L'entrée reset doit être **à logique positive**, donc lorsque le reset est à '1', les registres sont initialisés à '0'. Une entrée **CLK** permet le contrôle du passage des données. Le'entrée **ENABLE** permet d'activer ou de désactiver le registre. L'entrée **MODE** permet de choisir le mode du registre à décalage. Lorsque cette entrée est à '0', le registre est en mode décalage. Lorsque cette entrée est à '1', le registre est en mode chargement. L'entrée **INPUT** est sur 1 bit et est l'entrée du registre à décalage. L'entrée **LOAD** est sur N bits et représente la donnée à charger dans le registre à décalage. La sortie du registre à décalage multifonction se nomme **OUTPUT** et est sur 1 bit.

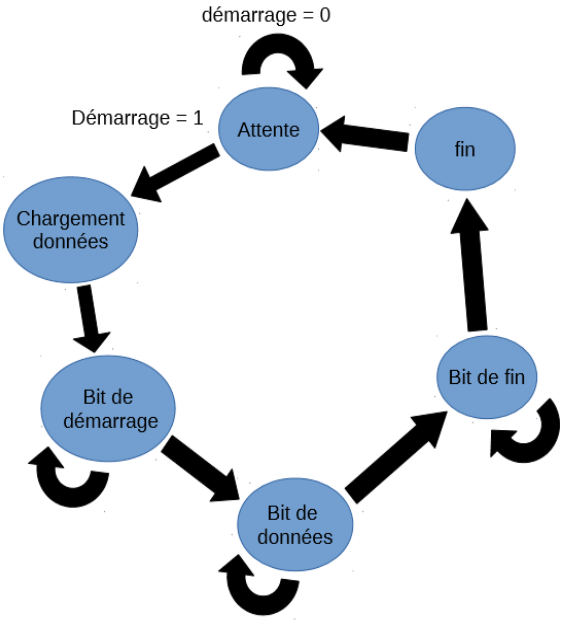
- Simulez votre architecture et testez son implémentation physique.

5. Implémentation du module de transmission UART VALIDATION OBLIGATOIRE (4 points + 1 point bonus)

5.a Inspirez-vous du module de réception du protocole UART développé au laboratoire 2 afin de programmer le module de transmission. Relisez attentivement la section **7.a** afin de bien comprendre le fonctionnement de ce protocole. Vous implémenterez le protocole permettant de transmettre 8 bits de données. Il est fortement conseillé de s'inspirer de la machine à état présentée plus bas afin de programmer votre module.

Machine à état

Un diagramme de la machine à états du transmetteur est proposé afin d'en faciliter l'implémentation. Vous devez tout de même fournir un diagramme d'états de votre machine à états et celui-ci doit concorder avec votre code VHDL.



Module de transmission UART

- L'état **attente** est l'état de démarrage de la machine à état. Cet état est quitté lorsque le démarrage d'un envoi de 8 bits est amorcé.
- L'état **chargement de données** permet de charger les données à envoyer dans le registre à décalage servant à l'envoi.
- L'état **bit de démarrage** permet d'envoyer sur la ligne **TX** le bit de démarrage.
- L'état **bit de données** permet d'envoyer les bits de données sur la ligne **TX**.
- L'état **bit de fin** permet d'envoyer le bit de fin sur la ligne **TX**.
- L'état **fin** permet de signaler que l'envoi est terminé.

Contraintes :

C5.1 Vous devez **obligatoirement** utiliser le registre à décalage multifonction développé plus haut afin de charger les données à envoyer. Le registre à décalage doit avoir une longueur de 8 bits.

- Créez un fichier **transmetteur_UART.vhd** et implémentez le transmetteur UART. Le module possède une entrée **START**, connecté à **SW0** de la carte ZYBO, qui permet de démarrer le transfert des données. Une entrée **CLK** permet de contrôler le passage des données et le fonctionnement de la machine à états. Une entrée **RESET**, connecté à **SW1** de la carte ZYBO, permet de réinitialiser la machine à états et les registres. De plus, ce module possède une entrée **DATA_IN** qui correspond aux données à envoyer. Les **DATA(7:2)** sont connectés **SW7** à **SW2** et **DATA_IN(1:0)** sont connecté à **0**.

  Important : Vous remarquerez que ce ne sont pas tous les bit de l'octet à envoyer qui sont connecté à des switch et cela dû au fait qu'il n'y a que 8 interrupteurs qui sont disponibles.

- Un signal de sortie **OCCUPE** est actif lorsque le transfert des données s'effectue**.** Ce signal est connecté à la **LED0**. Un signal de sortie **TERMINE** permet de signaler la fin du transfert des données. Ce signal est connecté

à la **LED1**. Pour finir, un signal de sortie **TX** correspondant à la ligne de transmission des bits est connecté au pin **W15** sur la carte ZYBO.

- Vous pouvez utiliser le schéma de la machine à état qui est fourni ou le modifier comme bon vous semble. Par contre, assurez-vous que **le schéma de votre machine à états** concorde avec le code que vous développez. De plus, il est **fortement** conseillé de développer une architecture avec une approche modulaire.

- Simulez votre architecture et testez l'implémentation avec **hercules**. Afin d'afficher les données reçues, vous devez activer les caractères hexadécimaux en effectuant un clic droit et en cochant l'option **HEX Enable**. Faites valider votre travail par l'assistant de laboratoire.

Port	Description	Pin
RX	Données en réception (PC --> FPGA)	T11
TX	Données en transmission (FPGA --> PC)	W15

5.b (Bonus) Utilisez un générique et des opérations mathématiques afin de rendre votre architecture paramétrable. Donc, votre architecture doit s'adapter à n'importe quel **baudrate** et le nombre de bits de données à envoyer doit être paramétrable. Par conséquent, la longueur des compteurs, des registres et le temps d'attente pour l'envoi d'un bit doivent s'ajuster en fonction du **baudrate** et du nombre de bits de données choisis lors de la synthèse du module. Afin d'obtenir votre point bonus, démontrez à l'assistant de laboratoire que votre architecture est paramétrable et montrez le bon fonctionnement.

6. Caméra vidéo : introduction au traitement de signal *(*4 points)

Les couleurs en vidéo

La couleur dépend de la longueur d'onde de la lumière, mais aussi de la définition subjective que notre œil lui donne. Pour comprendre comment les couleurs se forment dans notre œil, il faut d'abord s'intéresser à son fonctionnement.

Petit rappel : définition de la lumière

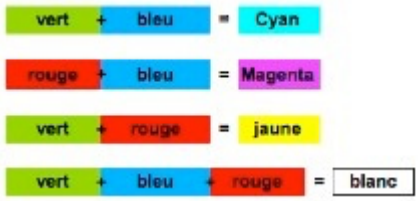
La lumière est une source d'énergie composée d'un ensemble d'ondes électromagnétiques et de particules élémentaires appelées photons. La couleur est caractérisée par la longueur d'onde de la lumière.

L'œil et son fonctionnement

L'œil ne perçoit pas toutes les longueurs d'onde de la lumière. Il n'est sensible qu'à certaines radiations électromagnétiques comprises entre 380 nm et 780 nm. En dessous de 380 nm, on parle de lumière ultraviolette. Au-dessus de 780 nm, on parle de lumière infra rouge.
On appelle la lumière visible par l'œil humain, le spectre lumineux.

Synthèse de la couleur
Synthèse additive :

On parle de synthèse additive lorsque l'on additionne les longueurs d'onde de plusieurs sources émettrices lumineuses colorées.Exemple : si vous prenez un spot lumineux vert et un spot lumineux rouge et que vous combinez les 2 faisceaux lumineux, vous obtenez la couleur complémentaire jaune.
Les couleurs utilisées dans cette synthèse sont les 3 couleurs primaires : rouge, vert et bleu qui, lorsqu'elles vont s'additionner 2 par 2, vont donner naissance aux couleurs complémentaires (secondaires) cyan, magenta et jaune.



Synthèse soustractive :

La matière d'un objet (support qui n'est pas source de lumière) est composée de telle sorte qu'elle va absorber (soustraire) une partie de la lumière (longueur d'onde) pour ne diffuser que les couleurs restantes. Le citron est jaune, car les pigments qui le composent absorbent la couleur bleue et diffusent les autres couleurs c'est-à-dire le vert et le rouge. On parle de synthèse soustractive lorsque vous combinez l'effet d'absorption de plusieurs couleurs sur une surface blanche et que vous obtenez alors une nouvelle couleur : le cyan mélangé au jaune va donner du vert. La synthèse soustractive est principalement utilisée en imprimerie.
Les couleurs généralement utilisées sont les couleurs fondamentales (secondaires lorsqu'on parle de synthèse additive) : le cyan, le jaune et le magenta.

Références : http://www.bemediasprod.com/fiches-techniques/les-couleurs-en-video/ft21/

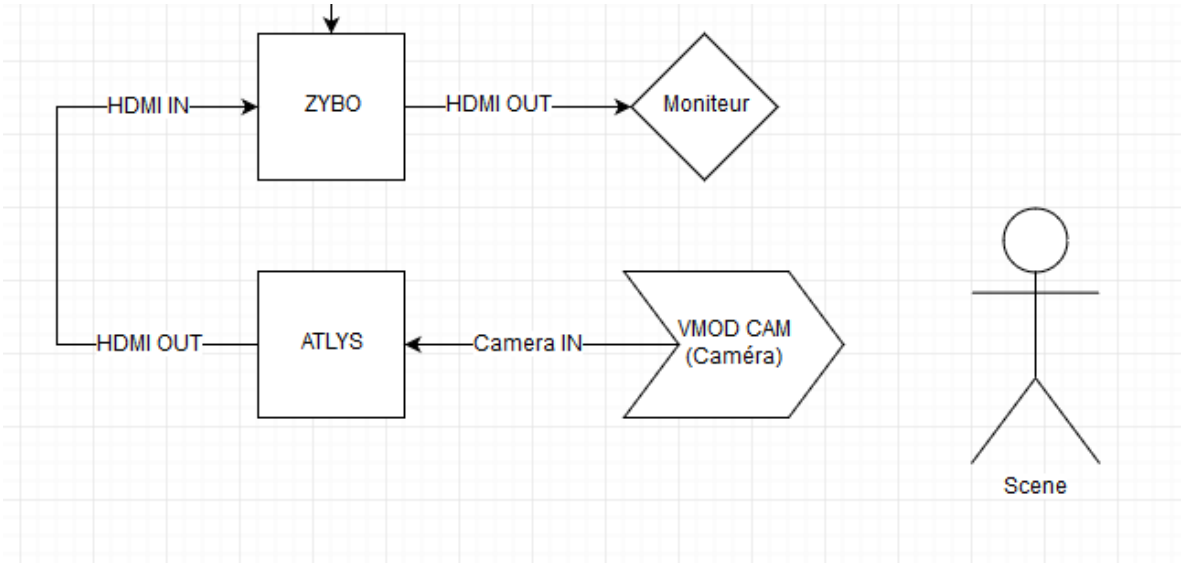
Description du matériel

Pour ce laboratoire, vous utiliserez une autre carte FPGA, la carte ATLYS. Sur cette carte, une carte est connectée. Voici la fiche technique de la caméra. Un exemple de code permet de capter l'image de la caméra et de la diffuser sur un port HDMI. Ce code est déjà implémenté dans la carte ATLYS et il ne sera pas nécessaire de la programmer.

La carte ZYBO contient une entrée HDMI et une sortie HDMI. Par conséquent, la sortie HDMI de la carte ATLYS est connectée à l'entrée HDMI de la carte ZYBO et la sortie HDMI de la carte ZYBO est connectée à un moniteur. Ce montage est déjà effectué pour vous au laboratoire.

Un exemple de code est fourni sur la carte ZYBO et permet de capter l'entrée du port HDMI et de diffuser le flux sur le port HDMI de sortie. Il est possible de télécharger l'exemple de code complet à partir du site du cours au laboratoire 3.

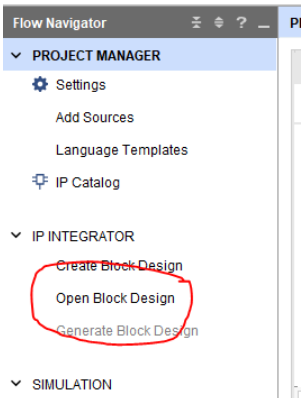
La figure ci-dessous montre une vue d'ensemble du montage disponible au laboratoire.



Exploration de l'architecture

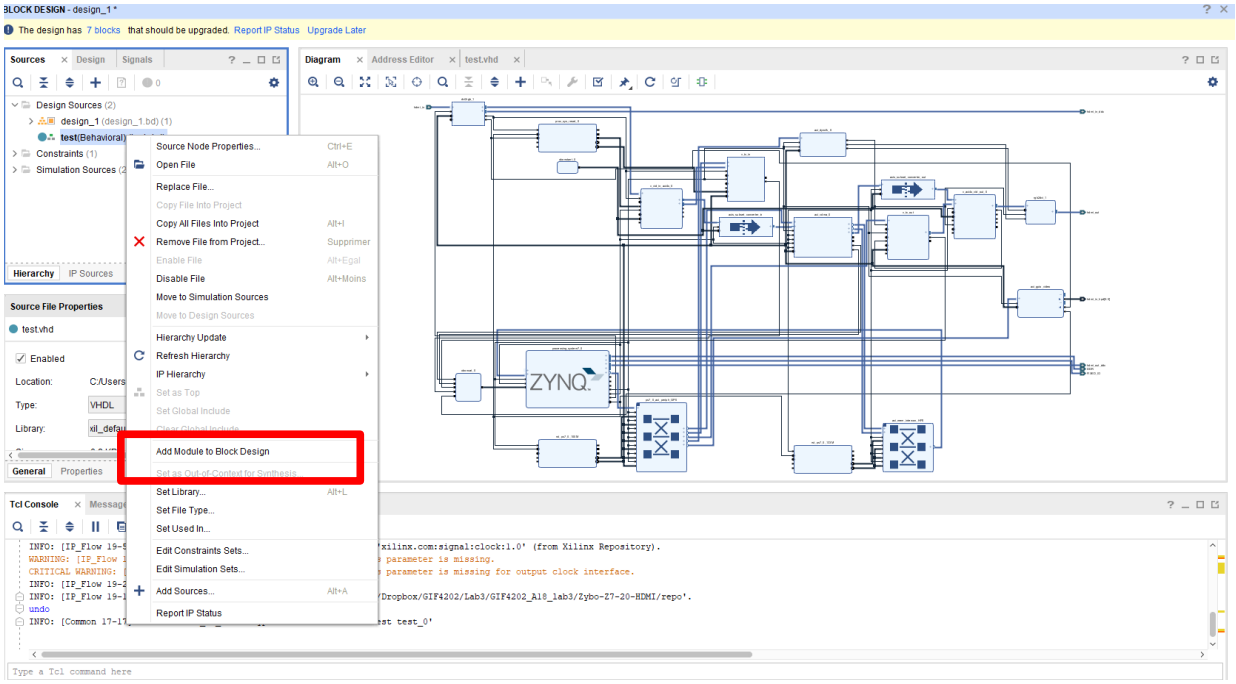
Afin de faciliter l'exploration de l'architecture de l'exemple de code, il est plus commode d'utiliser la vue « diagramme bloc ». Cet outil est aussi appelé **Block Design** ou **IP Integrator**.

Pour ce faire, ouvrez l'exemple de code fourni et appuyez sur **Open Block Design** dans le menu de droite, comme le montre la figure ci-dessous.

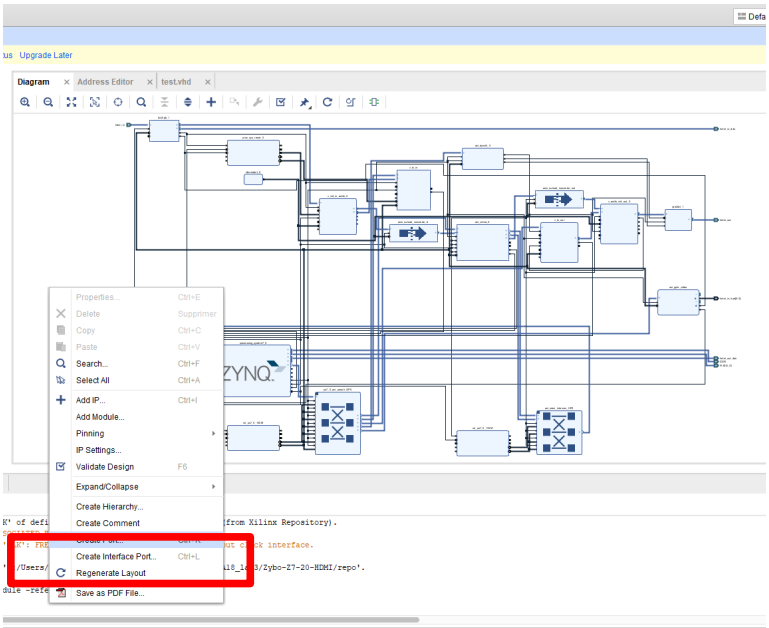


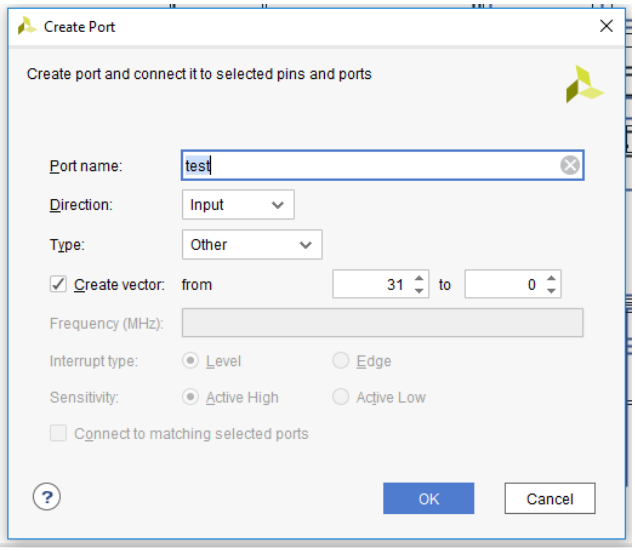
Le code vous sera alors présenté sous la forme d'un diagramme bloc. Cela permet de mieux visualiser l'architecture.

Pour ajouter un fichier VHDL au **Block Design**, ouvrez celui-ci, effectuez un clic droit sur votre fichier lorsque le et appuyez sur **Add module to Block Design,** comme le montre la figure ci-dessous. Si votre fichier ne contient pas d'erreur, il devrait s'ajouter sans problème. Ne vous souciez pas des alertes pouvant s'afficher à l'écran.



Pour ajouter un port d'entrée/sortie au **Block Design**, ouvrez celui-ci, effectuez un clic droit sur un espace vide. Par la suite, appuyez sur **Create Port**, comme le montre la figure ci-dessous. Un menu s'affichera pour vous permettre de confiture ce port d'entrée/sortie. Il sera possible par la suite, de connecter ce port à un module personnalisé ajouté au **Block Design**.





Pour plus d'informations sur l'utilisation du **Block Design**, consultez ce lien (https://reference.digilentinc.com/vivado/getting-started-with-ipi/start)

6.a Questions

Explorez l'architecture de l'exemple de code et la fiche technique de la caméra afin de répondre aux questions suivantes.

- Quelles sont les caractéristiques principales de la VmodCAM?
- Expliquez brièvement l'architecture de l'exemple de code.
- Quel est le format RGB utilisé dans l'exemple de code?
- Quelle est l'utilité du port HPD?

7. Multiplexage des couleurs (3 points) VALIDATION OBLIGATOIRE

7.a Modifiez l'architecture de l'exemple de code de référence afin de permettre le multiplexage des canaux rouge, vert et bleu du flux vidéo. Autrement, l'interrupteur **SW2** activera le canal rouge dans l'image, **SW1** le canal bleu et **SW0** le canal vert. Par exemple, lorsque seul l'interrupteur SW0 est à '1', l'image devrait avoir une teinte verte.

- Assurez-vous de bien comprendre l'architecture des exemples afin de faciliter l'implémentation de votre code.
- Ajouter un module RTL et les ports d'entrées de **SW0, SW1 et SW2** au **Block Design**.

7.b Dessinez et implémentez votre architecture afin de la faire valider par l'assistant de laboratoire.

8. Détection des couleurs (5 points) VALIDATION OBLIGATOIRE

8.a Modifiez l'architecture de l'exemple de code de référence afin de permettre la détection des couleurs. **Assurez-vous de faire cet exercice avec un exemple de code vierge**, donc, qui n'a pas été modifié auparavant.

- Pour ce faire, les systèmes au laboratoire sont équipé d'un cube sur une table tournante donc les faces sont de couleur **rouge**, **vert** et **bleu** et celui-ci sera présentée devant la caméra. Votre architecture doit effectuer la détection des couleurs rouge, verte et bleue.
- Lorsque la couleur rouge est détectée, la **LED0 et LED1** s'allument. La **LED2 et LED3** s'allument lorsque le vert est détecté et la **LED5 et LED 5** s'illuminent lorsque la couleur bleue est détectée**.**

- La détection est active, donc le FPGA est constamment à l'oeuvre et détecte une couleur dès que celle-ci est présente dans l'image.

IMPORTANT :

- Il est possible que plusieurs couleurs soient présentés à la fois dans l'image.
- Il est évident qu'une image est composée d'une combinaison de rouge, vert et bleu. Par conséquent, vous devez définir des seuils afin de détecter uniquement une couleur précise.
- Ces seuils doivent être assez flexibles afin de détecter la bonne couleur. Les codes de couleur utilisés pour le test de validation sont présentés ci-dessous.

➔ Le code RGB888 utilisé pour la couleur rouge est **0xFF0000.**
➔ Le code RGB888 utilisé pour la couleur bleue est **0x0000FF.**
➔ Le code RGB888 utilisé pour la couleur verte est **0x00FF00.**

- Ajouter un module RTL et les ports de sorties de **LED0, LED1 et LED2** au **Block Design**.

8.b Dessinez et implémentez votre architecture afin de la faire valider par l'assistant de laboratoire.

9. Initiation au concept système sur puce (2 points) VALIDATION OBLIGATOIRE

Modifiez l'architecture du numéro précédent pour remplacer la partie de la gestion des LED par un algorithme simple sur le processeur intégré ARM Cortex-A9 au Zynq (communément appelé partie PS).
Le processeur aura comme entrée un mot sur 3 bits indiquant la couleur dominante (001 pour rouge, 010 pour vert et 100 pour bleu). Par la suite le processeur allumera les LEDs selon le pattern de l'exercice précédent.

- Pour construire la partie qui servira à connecter votre design PL à la PS, commencez par adapter votre code de l'exercice précédent afin de produire la sortie désirée dans la PL (le mot à 3 bit nécessaire au processeur tel que décrit précédemment), puis suivez le tutoriel fournit en annexe pour découvrir comment connecter votre design au PS en utilisant le protocole AXI.

- Vous pourrez ensuite vous servir du driver minimaliste fournit GPIO_min_driver pour allumer les LEDs selon le pattern de l'exercice précédent.

10. Annexe 1 – Exemple de remise