

TP: Video Graphics Array (VGA)

1 BUTS DU TRAVAIL

- ✓ **Modéliser** en **VHDL** une sortie **VGA**.
- ✓ **Concevoir** le modèle à l'aide de constantes.
- ✓ **Générer** un rectangle et un cercle sur l'écran
- ✓ **Synthétiser** le circuit et tester sur le circuit
- ✓ **Vérifier** le bon fonctionnement du circuit par **simulation et synthèse**.

2 DESCRIPTION DU CIRCUIT

Une description du fonctionnement d'une interface VGA peut être trouvée sur internet, par exemple sur :

Entité du bloc VGA :

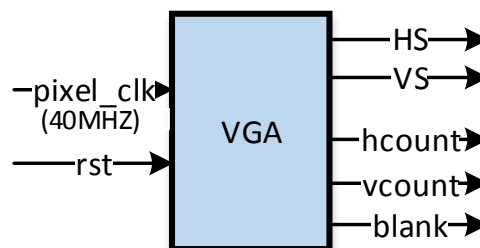
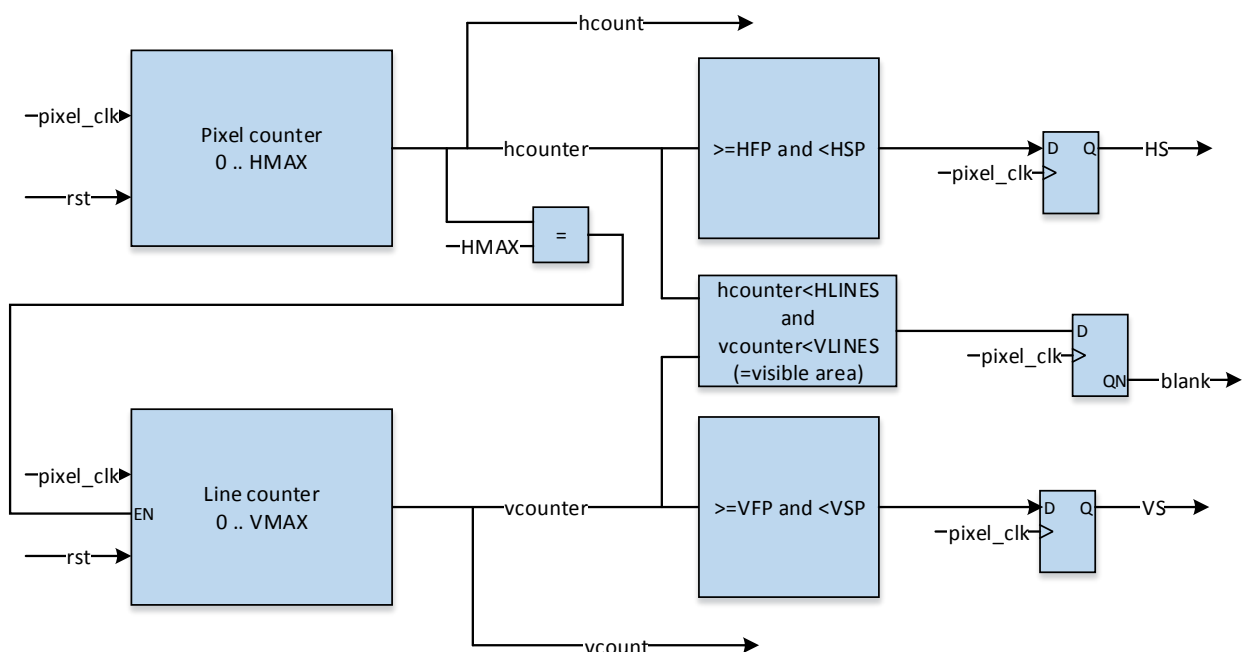
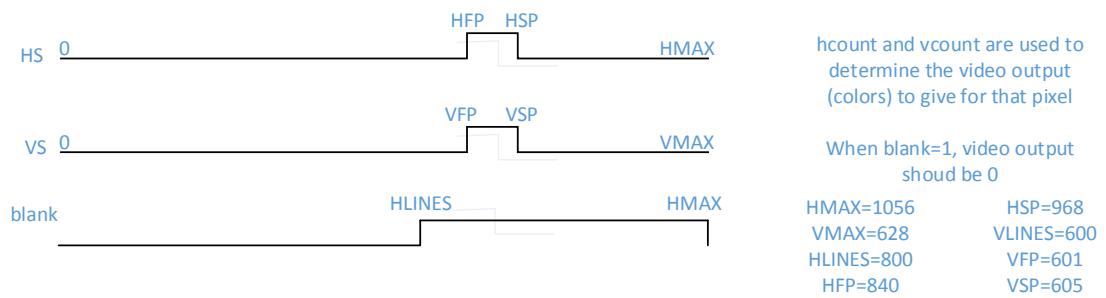


Schéma bloc du composant :





Description des signaux de l'entité:

Entrées :

Pixel_clk : clock de la FPGA (pixel), 40MHZ

Rst : reset de la FPGA

Sorties :

HS : Impulsions de synchronisation horizontale

VS : Impulsions de synchronisation verticale

Hcount,

Vcount : Coordonnées X,Y du pixel

Blank: Lorsque =1, la sortie video doit être nulle

Remarque: Ce bloc VGA ne contient pas la génération du signal vidéo lui-même, en fonction de hcount et vcount. Ce signal sera généré par un autre bloc qui recevra hcount, vcount et blank.

3 FICHIER DE CONTRAINTE .ucf

A l'aide du schéma du board Nexys 3, vous devez écrire le fichier de contrainte .ucf afin de définir les entrées/sorties sur le FPGA.

ANNEXE

VGA Timing :

SVGA Signal 800 x 600 @ 60 Hz timing

General timing

Screen refresh rate	60 Hz
Vertical refresh	37.878787878788 kHz
Pixel freq.	40.0 MHz

Horizontal timing (line)

Polarity of horizontal sync pulse is positive.

Scanline part	Pixels	Time [μs]
Visible area	800	20
Front porch	40	1
Sync pulse	128	3.2
Back porch	88	2.2
Whole line	1056	26.4

Vertical timing (frame)

Polarity of vertical sync pulse is positive.

Frame part	Lines	Time [ms]
Visible area	600	15.84
Front porch	1	0.0264
Sync pulse	4	0.1056
Back porch	23	0.6072
Whole frame	628	16.5792

Source : <http://tinyvga.com/vga-timing/800x600@60Hz>

TP: Générateur de mire VGA

1. BUTS DU TRAVAIL

- ✓ **Modéliser** en VHDL le générateur de mire en utilisant **l'interface VGA**.
- ✓ **Vérifier** le bon fonctionnement du circuit par **simulation, synthèse et test sur la carte**.

2 DESCRIPTION DU CIRCUIT

Entité du bloc :

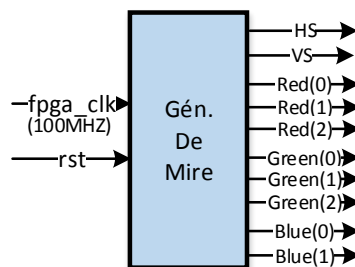
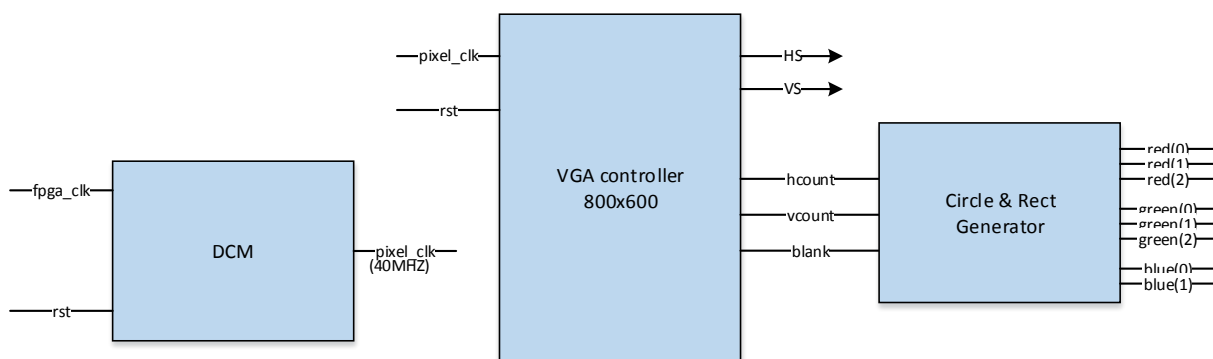


Schéma bloc :



3 FICHIER DE CONTRAINTE .ucf

A l'aide du schéma du board Nexys 3, vous devez écrire le fichier de contrainte .ucf afin de définir les entrées/sorties sur le FPGA.

4 TRAVAIL A EFFECTUER

- I. Modéliser en VHDL le bloc générateur de mire en instanciant le bloc interface VGA, pour générer un rectangle, puis un cercle, puis les 2.
- II. Simuler votre modèle
- III. Synthétiser votre code
- IV. Tester sur la carte Nexys 3

ANNEXE 1 : Génération et configuration du DCM

Le DCM (Digital Clock Management) permet de convertir la clock de 100MHz en 40MHz.

Pour l'ajouter au projet ISE, procédez comme suit :

Dans ISE : Projet > New Source > IP core

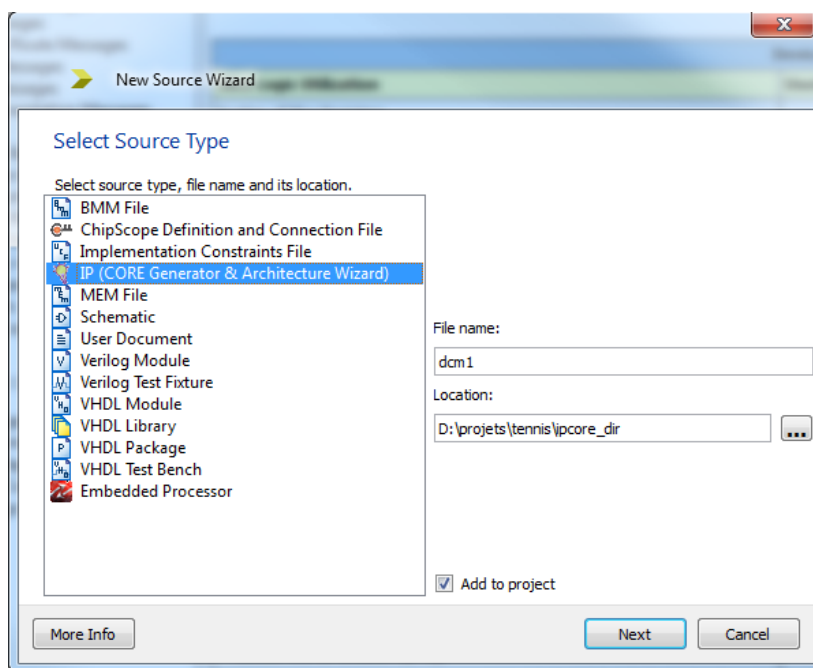
Donner un nom, par exemple dcm1 et un répertoire, par exemple <projet_dir>\ipcore_dir

Vérifier/activer l'option add to project

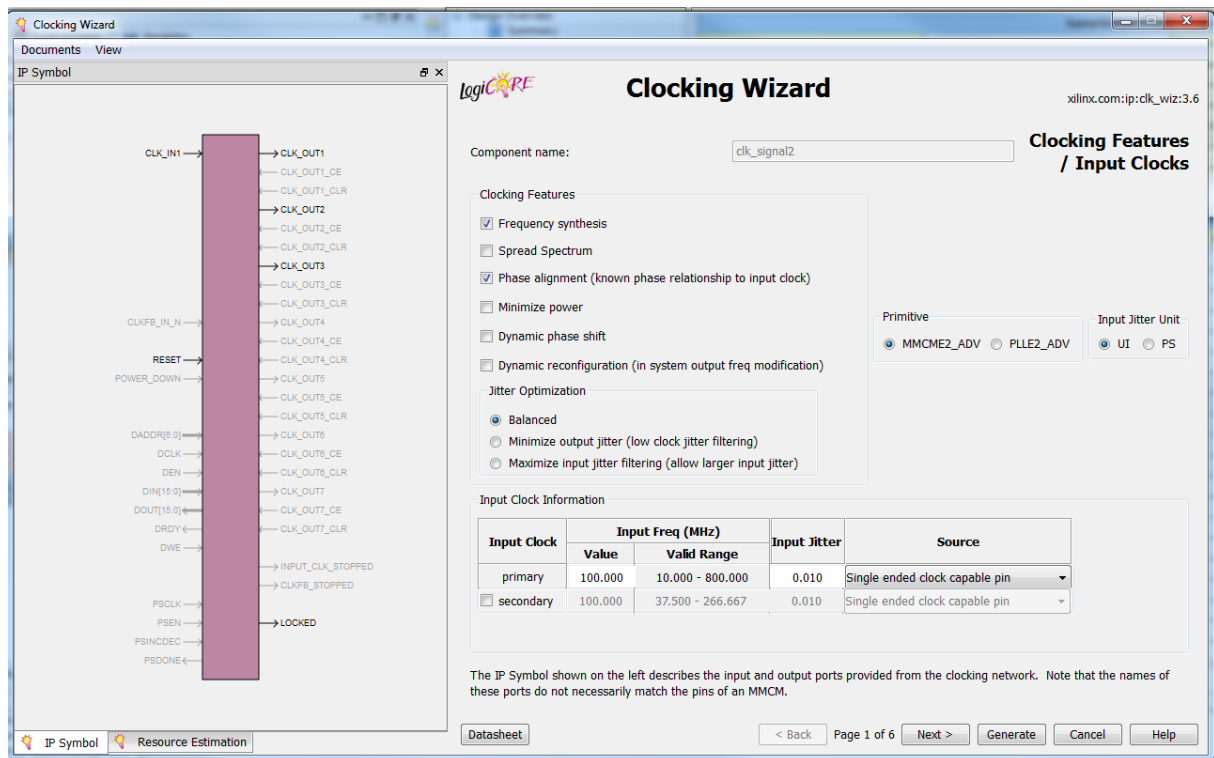
Cliquer Next

Dans la fenêtre qui s'ouvre, taper clocking wizard dans search IP catalog

Sélectionner le clocking wizard trouvé et Next, puis Finish



Configurer comme suit :



Clocking Wizard

Component name:

Clocking Features

- ☒ Frequency synthesis
- ☐ Spread Spectrum
- ☒ Phase alignment (known phase relationship to input clock)
- ☐ Minimize power
- ☐ Dynamic phase shift
- ☐ Dynamic reconfiguration (in system output freq modification)

Jitter Optimization

- ☒ Balanced
- ☐ Minimize output jitter (low clock jitter filtering)
- ☐ Maximize input jitter filtering (allow larger input jitter)

Primitive

☒ MMCME2_ADV ☐ PLLE2_ADV

Input Jitter Unit

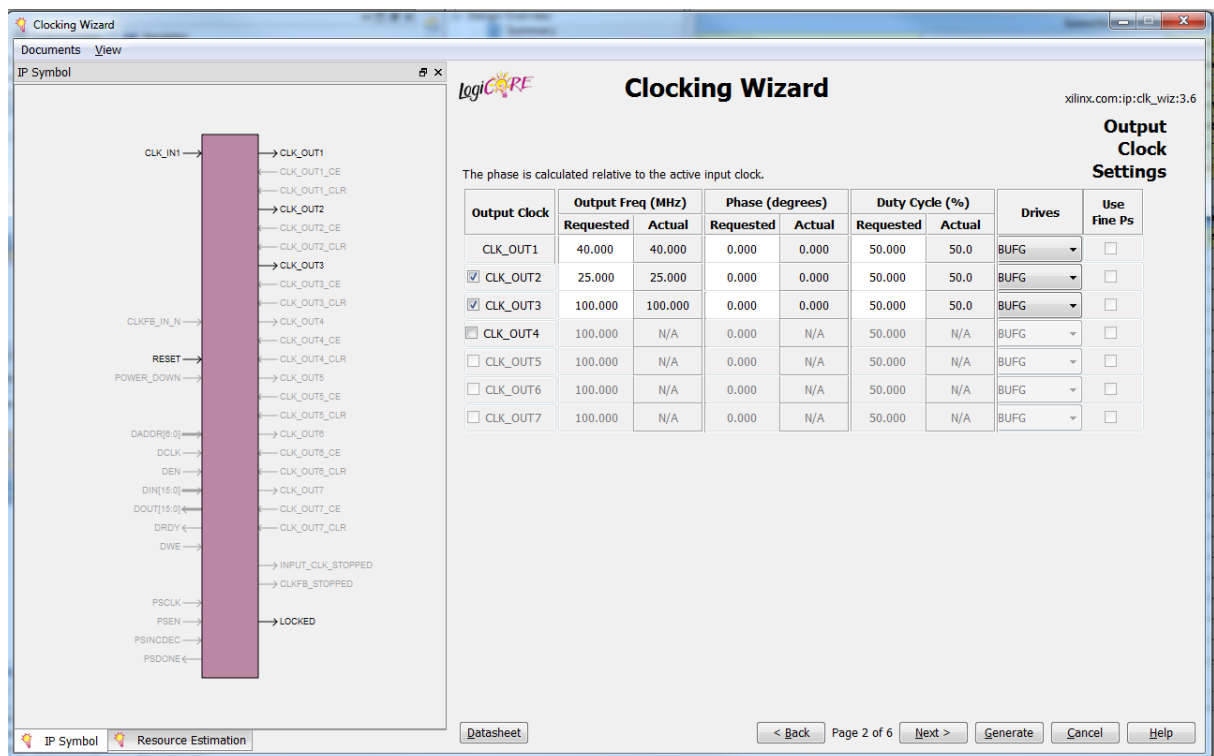
☒ UI ☐ PS

Input Clock Information

Input Clock	Input Freq (MHz)		Input Jitter	Source
	Value	Valid Range		
primary	100.000	10.000 - 800.000	0.010	Single ended clock capable pin
<input type="checkbox"/> secondary	100.000	37.500 - 266.667	0.010	Single ended clock capable pin

The IP Symbol shown on the left describes the input and output ports provided from the clocking network. Note that the names of these ports do not necessarily match the pins of an MMC.

[Datasheet](#) < Back Page 1 of 6 Next > Generate Cancel Help



Clocking Wizard

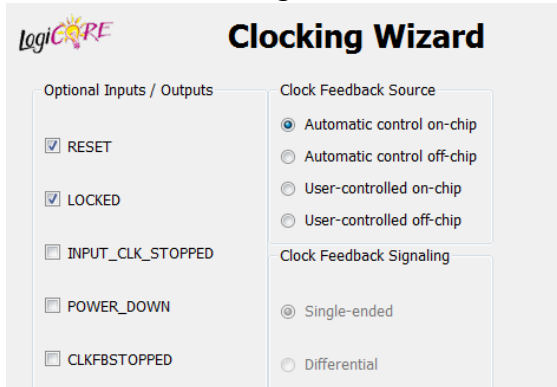
The phase is calculated relative to the active input clock.

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine Ps
	Requested	Actual	Requested	Actual	Requested	Actual		
<input type="checkbox"/> CLK_OUT1	40.000	40.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input checked="" type="checkbox"/> CLK_OUT2	25.000	25.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input checked="" type="checkbox"/> CLK_OUT3	100.000	100.000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>
<input type="checkbox"/> CLK_OUT7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>

[Datasheet](#) < Back Page 2 of 6 Next > Generate Cancel Help

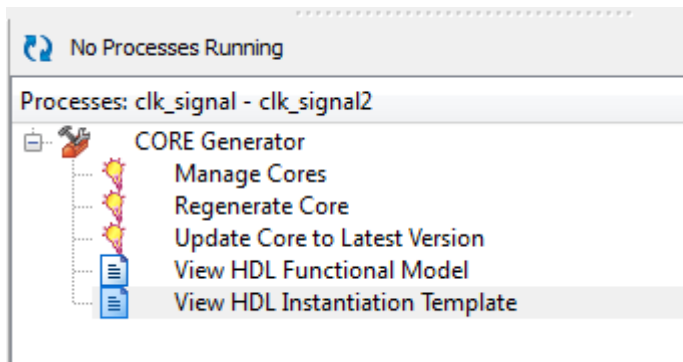
Remarque : pour notre TP actuel, nous n'avons besoin que de la sortie CLK_OUT1

Next, contrôler/corriger :



Ensuite cliquer Generate

Pour instantier ce bloc dans votre code VHDL, sélectionner le bloc DCM dans le design viewer, puis double-cliquer sur View HDL instantiation template :



Vous pouvez ensuite copier-coller la déclaration du composant et l'exemple de mapping qu'il vous faudra corriger avec vos signaux.

ANNEXE 2 : Génération d'un rectangle

Le signal video est constitué des signaux rouge, verts et bleus (red, green, blue). Ce signal doit être généré en fonction des coordonnées des pixels hcount et vcount.

Par exemple, pour dessiner un rectangle compris entre $200 < \text{hcount} < 600$ et $200 < \text{vcount} < 400$, on pourra introduire le code suivant pour le bloc « Circle & Rect Generator » :

```
process(hcount,vcount)
begin
-- rectangle
if (((vcount=200) or (vcount=400)) and ((hcount>200) and (hcount<600))) or
   (((hcount=200) or (hcount=600)) and ((vcount>200) and (vcount<400))) then
    color <= "00000000";
else
    color <= "11111111";
end if;
end process;

red <= color(7 downto 5) when blank='0' else "000";
green <= color (4 downto 2) when blank='0' else "000";
blue <= color (1 downto 0) when blank='0' else "00";
```

Note: color est un std_logic_vector(7 downto 0).

ANNEXE 3 : Génération d'un cercle

La génération d'un cercle est un peu plus difficile. Les points sur la périphérie du cercle répondent à l'équation (x_c, y_c =centre du cercle, r = rayon) :

$$(hcount-xc)^2+(vcount-yc)^2=r^2$$

En VHDL, cela devient (r^2 =rayon au carré):

```
process(hcount,vcount)
variable temp : integer;
begin
  -- cercle
  color<="11111111";
  temp:=( (hcount-xc)*(hcount-xc)) + ((vcount-yc)*(vcount-yc)); -- equation du cercle
  if temp=r2 then
    color<= "00000000";
  end if;
end process;
```

Le problème est que, vu que $hcount$ et $vcount$ sont des entiers, l'équation est rarement satisfaite et seuls quelques points apparaissent. Il faudra donc tolérer une certaine marge d'erreur ou remplir l'espace entre deux cercles de rayon très proche (la condition $temp=r^2$ doit être modifiée).