Introduction

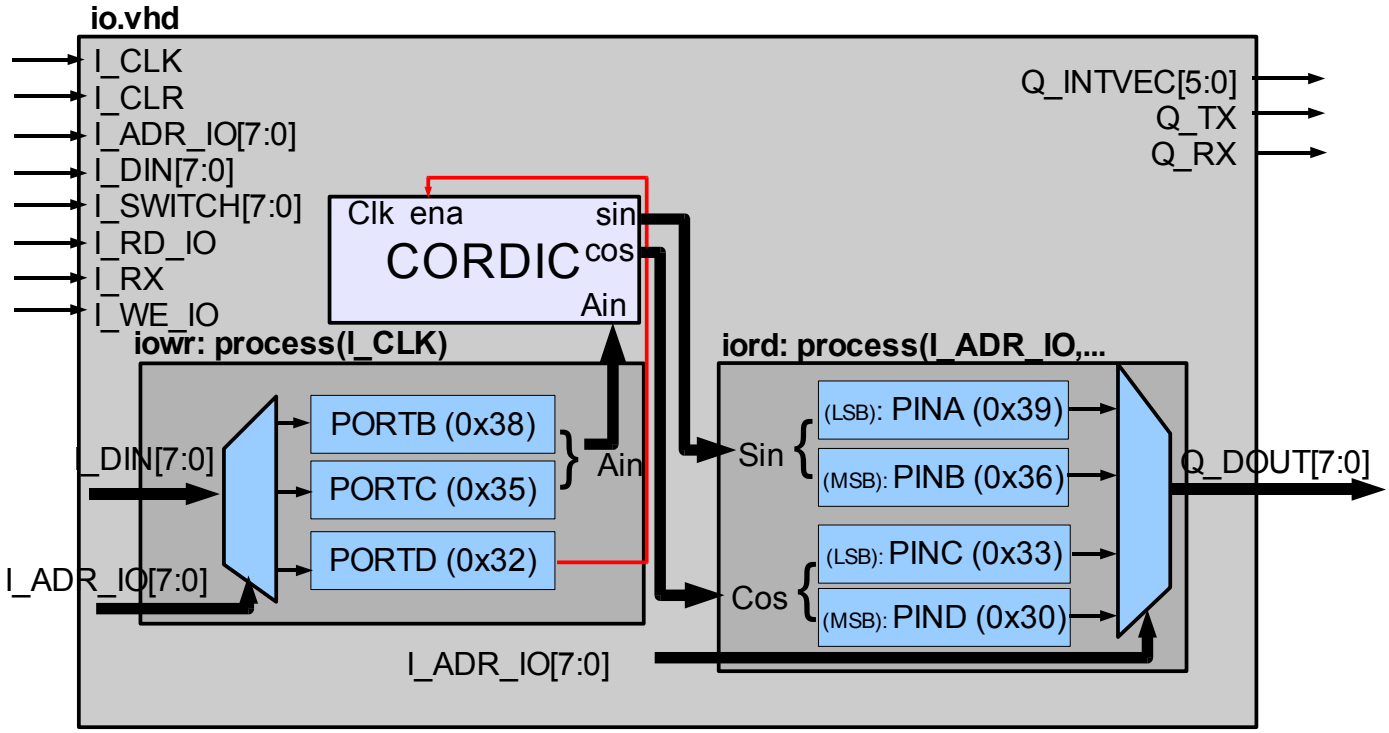
Nous allons partir d'une version corrigée de l'exercice 3 du TP sur CORDIC et en faire un certain nombre de modifications. Dans cette épreuve , on ne vous demande pas de vous poser des questions sur l'utilité ou non des modifications mais seulement de les exécuter et de montrer que le travail résultant fonctionne toujours. Le but de l'épreuve est ainsi d'évaluer comment vous appréhendez des modifications à partir d'un schéma complexe décrit en VHDL.
Vous disposez d'une feuille réponse sur laquelle vous répondez et sur laquelle l'enseignant notera ce qu'il a vu fonctionner et sinon, quelques indications sur l'état d'avancement de votre travail.
Les ressources se trouvent dans le répertoire "final2013" du fichier ressources utilisé en TP.

Travail à réaliser (Préparation logicielle exo0)

Faire fonctionner l'ensemble CORDIC avec le code source cordicExo3.c (dans répertoire "Exam2013Start/soft"). Quelle est la taille du code (sur feuille réponse ?)

I. Réalisation d'un compteur/décompteur d'angle

Dans l'exercice précédent exo0 sur CORDIC (sa réalisation matérielle), l'angle était fourni par le processeur par l'intermédiaire de deux PORTs comme le montre le schéma ci-dessous :



Travail à réaliser (Préparation logicielle exo1)

On vous demande de trouver les valeurs extrémales des angles à réaliser en format Q3.13 si l'on veut rester dans le domaine $\{\frac{-\pi}{2}, \frac{\pi}{2}\}$. Pour cela vous utiliserez un programme C sous Linux avec la fonction de conversion du TD :

```
1             int float2HexQ3_13(float val){  //conversion float vers Q3.13
2               int temp;
3               char i;
4               float f_temp;
5               if (val < 0) f_temp = -val; else f_temp = val;
```

```
6               temp = ((int) floor(f_temp)<<13);
7               f_temp = f_temp - floor(f_temp);
8               for (i=0;i<13;i++) {
9                 temp|=((int)floor(2*f_temp)<<(12-i));
10                f_temp = 2*f_temp - floor(2*f_temp);
11               }
12               if (val < 0) return -temp; else return temp;
13            }
```

- Ajouter un main() pour calculer les valeurs demandées
- Compiler l'algorithme sous linux : `gcc -o final2013_exo1 final2013_exo1.c -lm`
- Essayer le programme : `./final2013_exo1`

Notez les limites trouvées avec votre programme en hexadécimal sur votre feuille réponse. Attention un "%x" dans un printf semble afficher systématiquement sur 32 bits (sauf pour les zéros à gauche) ! Vous garderez donc les seuls 16 bits de poids faibles pour le nombre négatif.

Travail à réaliser (Préparation matérielle : avec processeur exo2)

2-1) Un composant VHDL capable de compter jusqu'au maximum, puis décompter sans jamais dépasser le minimum est donné ci-dessous. Le maximum et minimum ont été calculés en question précédente. Compléter donc les constantes "MAX" et "MIN" dans le VHDL ci-dessous (ou récupéré dans les ressources "cmptAngle.vhd").

Sa sortie sera l'angle Ai sur 16 bits en format Q3.13.

```
1             -- fichier cmptAngle.vhd dans "Resources2013.zip"
2             library IEEE;
3             use IEEE.std_logic_1164.all;
4             use IEEE.std_logic_arith.all;
5             use IEEE.std_logic_unsigned.all;
6             ENTITY cmpt_Angle is
7               PORT(
8                 clk,reset,en : IN std_logic;
9                 inc_dec_i : IN std_logic_vector(7 DOWNTO 0);
10                -- 3243 < angle_o < CDBD
11                angle_o : OUT std_logic_vector(15 downto 0));
12            END cmpt_Angle;
13            ARCHITECTURE Behavioural OF cmpt_Angle IS
14            constant MAX : std_logic_vector(15 downto 0) := x"????";
15            constant MIN : std_logic_vector(15 downto 0) := x"????";
16            -- =1 si incrementation =0 si decrementation
17              signal increment : std_logic;
18              signal s_angle : std_logic_vector(15 downto 0);
19            BEGIN
20              PROCESS(clk,reset) BEGIN
21                IF reset='1' THEN
22                  s_angle <=MAX;
23                  increment <= '1';
24                  ELSIF rising_edge(clk) THEN
25                    IF en = '1' THEN
26                      IF increment = '1' THEN
27                        s_Angle <= s_Angle + inc_dec_i;
28                        IF s_Angle >= MIN and s_Angle(15)='0' THEN
29                           s_angle <= MIN;
30                           increment <= '0';
31                        END IF;
32                      ELSE
33                        s_Angle <= s_Angle - inc_dec_i;
34                        IF s_Angle <= MAX and s_Angle(15)='1' THEN
35                           s_angle <= MAX;
36                           increment <= '1';
37                        END IF;
```

```
38                      END IF;
39                    END IF;
40                  END IF;
41                END PROCESS;
42                angle_o <= s_angle;
43              END Behavioural;
```

Pour pouvoir tester ce nouveau périphérique, on vous demande de l'interfacer comme ci-dessous au processeur. L'objectif est d'en faire un périphérique qui compte ou décompte (c'est lui-même qui choisi) à partir d'un incrément fourni dans un seul PORT (ce sera PORTA et donc sur 8 bits).

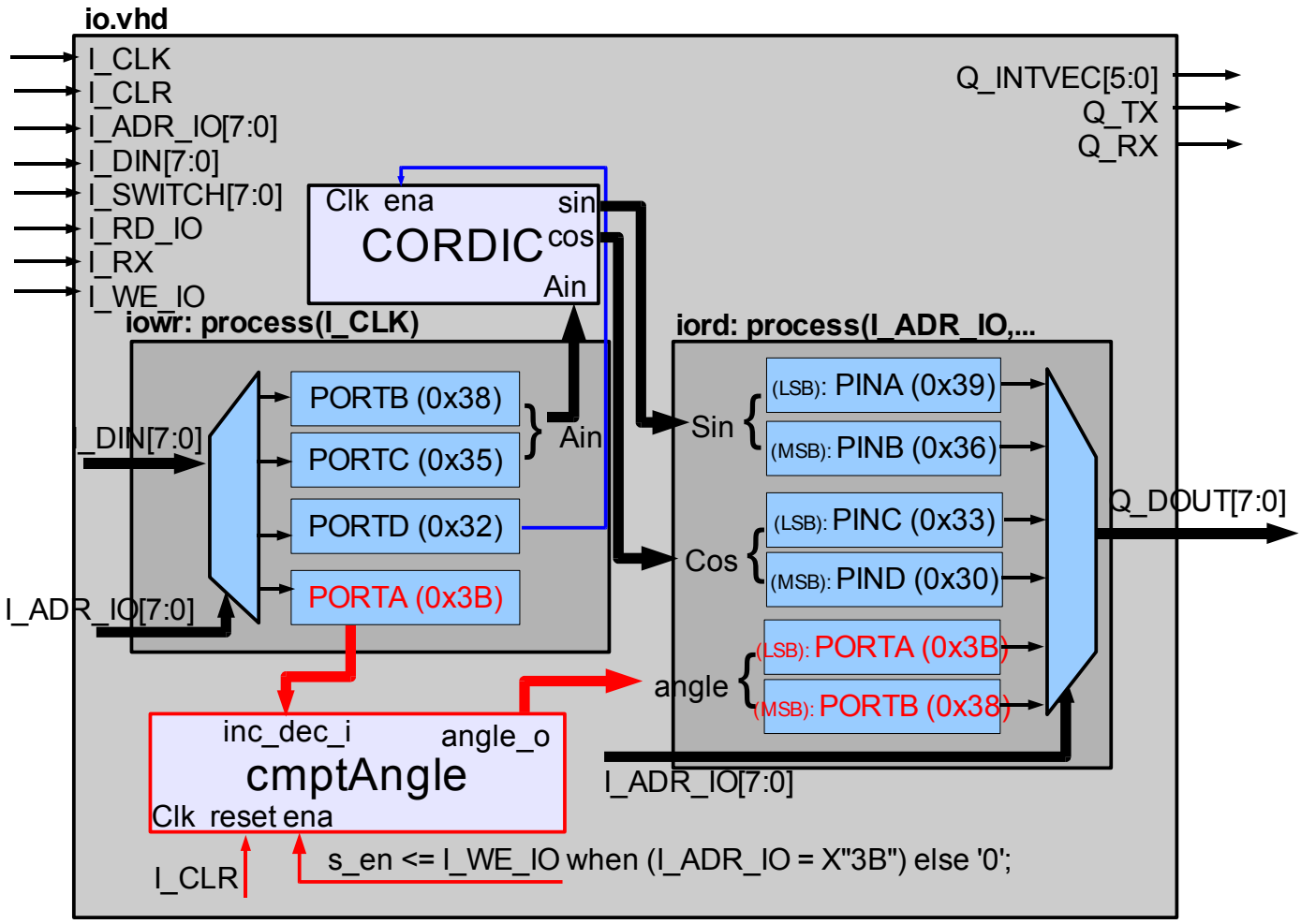
Ce que vous avez à réaliser est en rouge :

- ajouter le PORTA en écriture

- ajouter le PORTA et le PORTB en lecture

- ajouter le compteur d'angle donné

- ajouter les signaux et câbler.



2-2) Écrire un programme complet en C capable de tester ce nouveau périphérique en sortant les angles successifs en format lisible dans GTKTerm. L'ensemble des sous-programmes nécessaires pour ces tests peut être trouvé dans le programme "cordicExo3.c" des ressources.

Voici par exemple un programme main qui ferait l'affaire. Expliquez ce que vous voyez dans GTKTerm.

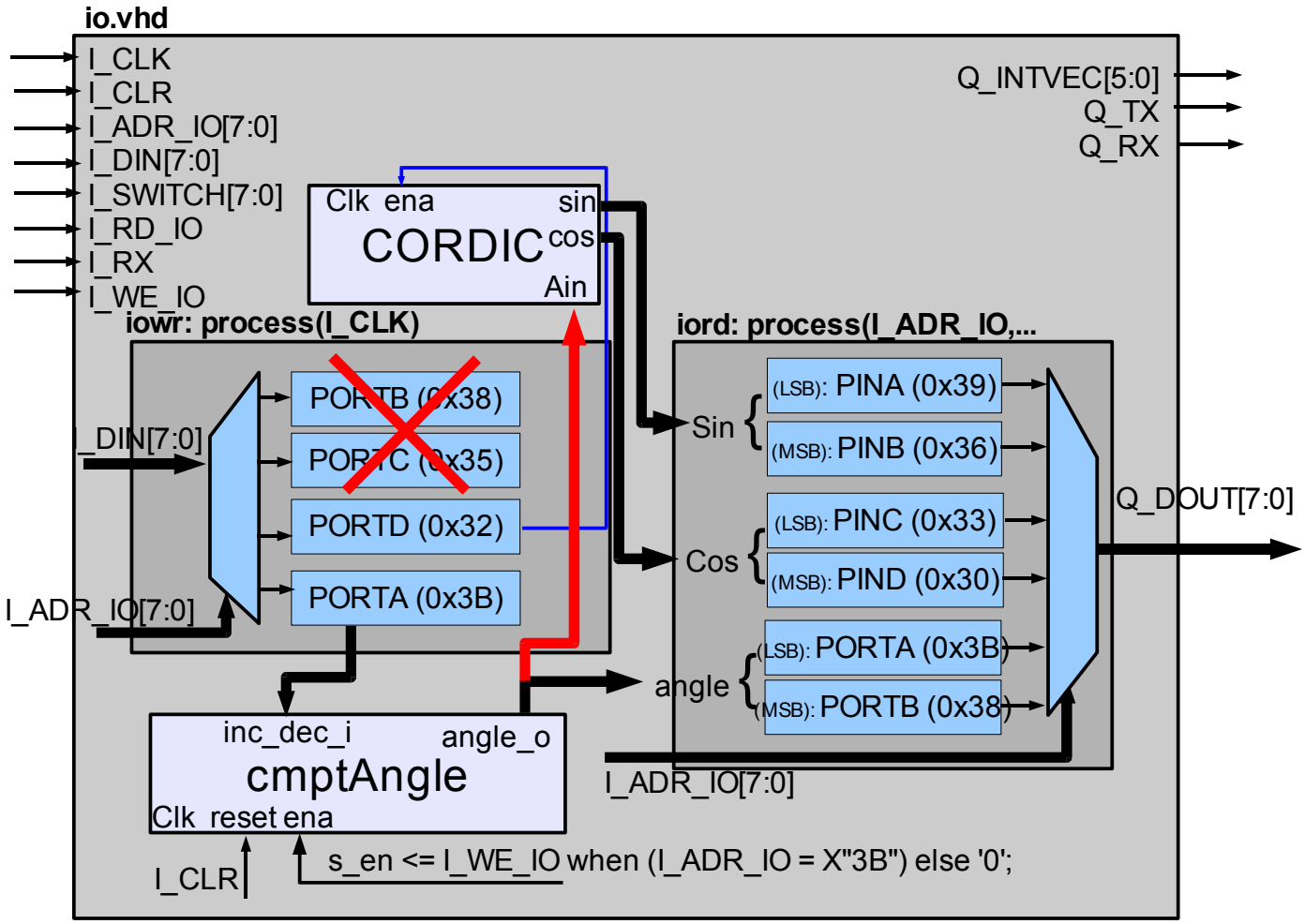
```
1               /* Port A */
2               #define PINA   _SFR_IO8(0x19)
3               //#define DDRA   _SFR_IO8(0x1A)
4               #define PORTA  _SFR_IO8(0x1B)
```

```
5               int main(void)
6               {
7                   int8_t i=0;
8                   char str[10];
9                   int16_t x;
10                  usart_init(); // pour pouvoir déclencher interruption RS232
11                while(1){
12                  PORTA = 0xFF; //increment compteur d'angle
13                  x = PORTB; //lecture resultat
14                  x = (x<<8) + PORTA;
15                  HexQ3_13ToString(x,str);
16                  i=0;
17                  while(str[i]!=0) {
18                    usart_send(str[i]);
19                    i++;
20                  }
21                  usart_send(0x0D);usart_send(0x0A);
22                  _delay_ms(500);
23                }
24              }
```

Que fait l'instruction "PORTA=0XFF;" ? Quelle est la valeur de l'incrément correspondante en valeur décimale ? Répondre sur la feuille réponse.

Travail à réaliser (compteur d'angle pour générer un angle : exo3)

3-1) Interfacer le composant de l'exercice 2 au processeur ATMega8 comme il devrait l'être : le composant de l'exercice précédent fournit l'angle au composant CORDIC comme l'indique la flèche rouge.



N'oubliez pas de débrancher l'ancien angle relié aux PORTB et PORTC qui sont devenus inutiles et ainsi retirés comme indiqué ici en rouge dans le dessin.

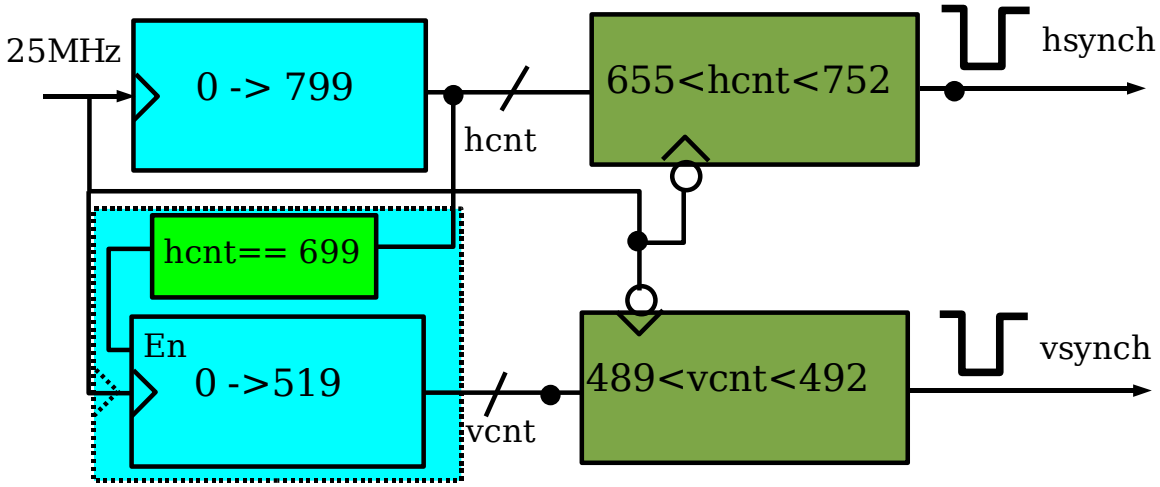
3-2) Réaliser un petit programme d'essai dans l'AVR qui affiche les sinus et cosinus dans GTKTerm au fur et à mesure de la variation des angles.

II. Affichage sur écran VGA en mode graphique

On désire réaliser un affichage sur écran VGA en mode graphique. Pour cela il nous faut gérer 5 signaux que l'on va décrire maintenant.

Partie synchronisation

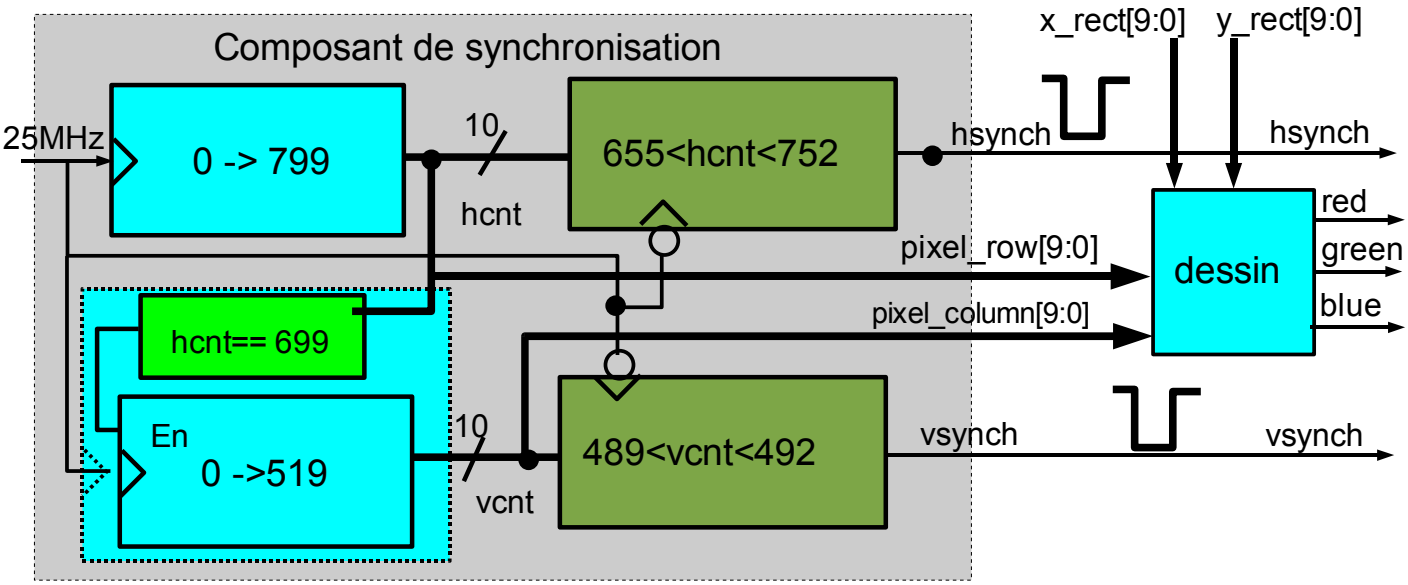
Son objectif est de gérer deux signaux appelés hsynch et vsynch à l'aide de deux compteurs.



Le programme VHDL qui réalise cette partie vous est donné.

Partie dessin sur l'écran

Dessiner sur un écran est très simple. La figure ci-dessous vous explique comment on peut faire.
On a ajouté à droite un composant appelé dessin qui est combinatoire puisqu'il n'a pas d'horloge. Dessiner un rectangle consiste instancier un composant rectangle donné. Si vous voulez un deuxième rectangle, vous faites de même et vous faites un OU logique entre les couleurs avant de les sortir.



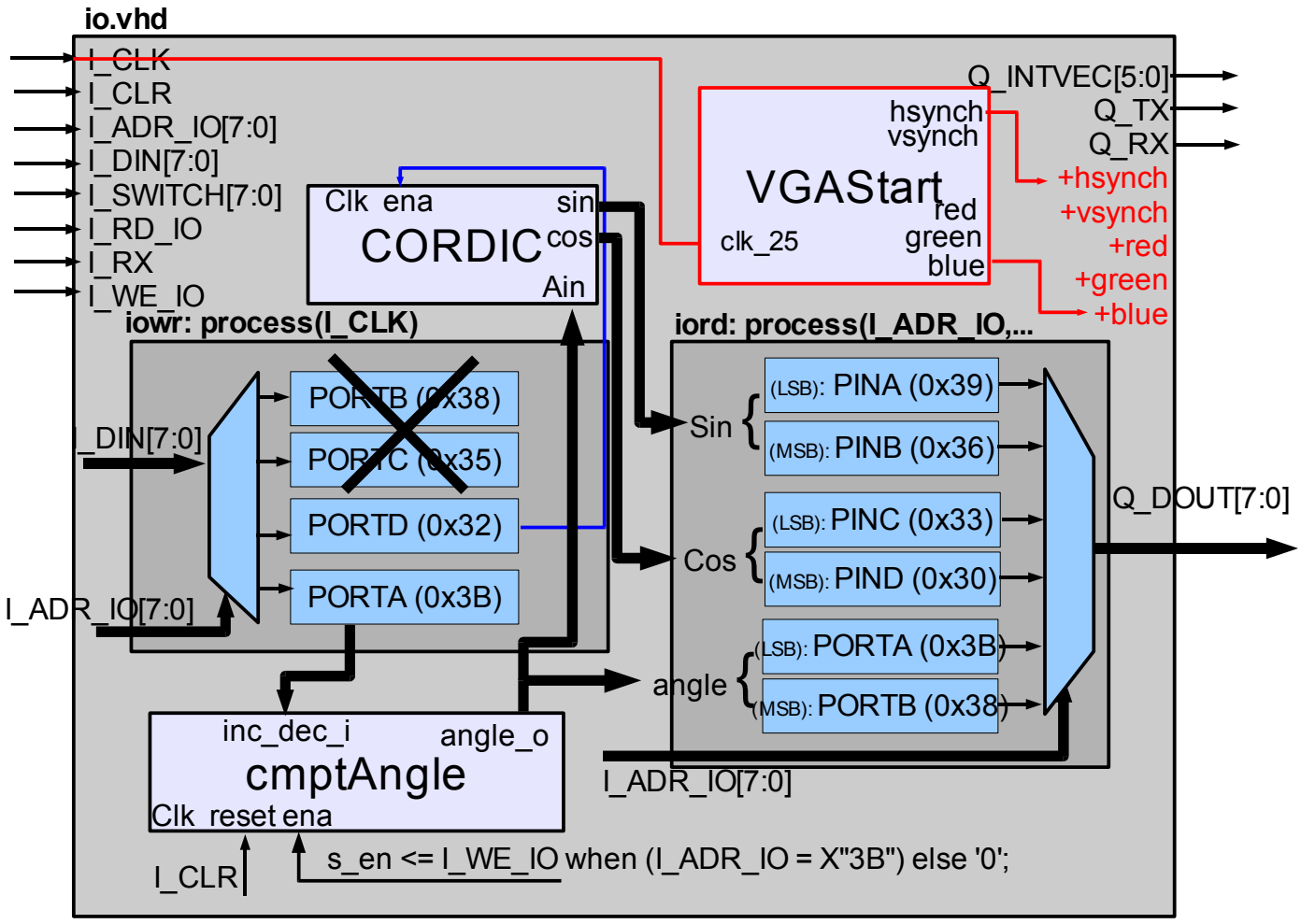
Vous avez l'ensemble disponible dans la ressource "resource2013.zip" en répertoire "final2013" et dans le fichier "VGAStart.vhd". C'est exactement comme le dessin ci-dessus mais sans les entrées x_rect et y_rect.

Voir aussi les explications dans :
http://fr.wikiversity.org/wiki/Very_High_Speed_Integrated_Circuit_Hardware_Description_Language/Interfaces_VGA_et_PS/2

Travail à réaliser (dessin VGA sans interaction processeur exo4)

Contrairement à ce que pourrait laisser penser le titre de cette section, nous allons commencer par installer le cœur VGA dans le processeur. Mais le processeur n'aura aucune interaction avec celui-ci. Cela rallonge fortement les temps de compilation mais permet de préparer petit à petit le travail de l'exo5.

4-1) A partir de la ressource "resource2013.zip" en répertoire "final2013" le fichier "VGAStart.vhd", on vous demande d'interfacer le composant VGATop comme ci-dessous :



Comme d'habitude le travail à réaliser est en rouge. N'oubliez pas de propager les sorties en rouge dans votre composant complet ("avr_fpga_spartan3.vhd").
N'oubliez pas d'écrire le fichier ucf. Montrer que l'on a bien dessiné un rectangle bleu sur notre écran VGA.

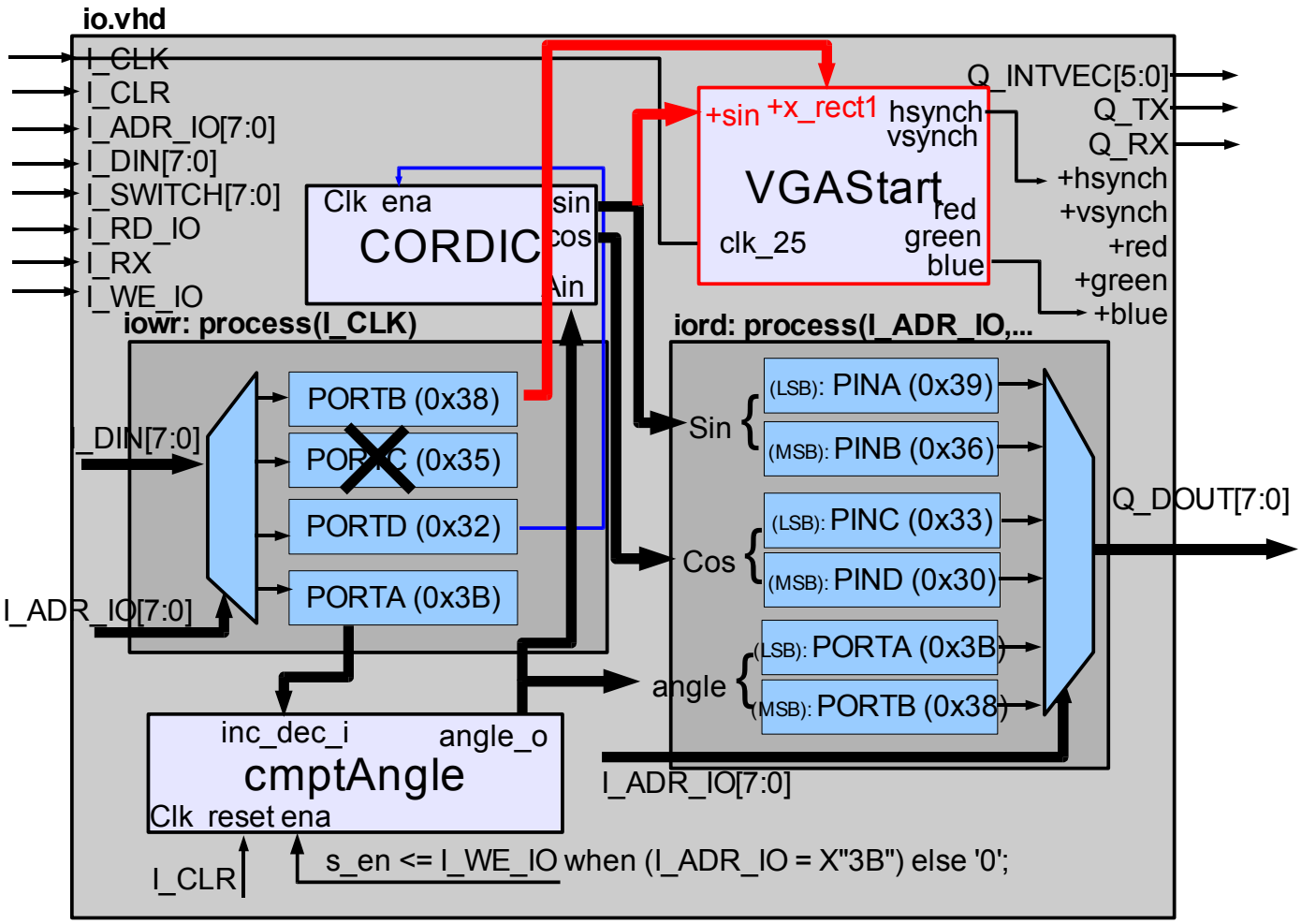
4-2) Modifier le programme "VGAStart.vhd" pour dessiner un rectangle vert en plus du bleu (un port map sans oublier le ou final sur les couleurs)

4-3) Garder l'ensemble précédent en diminuant la taille des rectangles à 5 pixels de côté pour la suite.

Travail à réaliser (dessin du sinus sur VGA exo5)

5-1) On vous demande d'interfacer l'écran VGA comme ci-dessous. Les coordonnées y des rectangles sont fournies par le cœur CORDIC (sin) tandis que les coordonnées x son fournies par le processeur à travers un PORT 8 bits (le bit de poids faible et de poids fort sont perdus et fixés à 0 par le matériel).

Ajouter ce qui est en rouge sur le schéma ci-dessous sans vous préoccuper pour le moment de ce qui sera fait des entrées ajoutées ("sin[15:0]" et "x_rect[9:0]") dans VGA start.



Indications :

Voici comment interfacer x_rect1 pour perdre les bits de poids fort et faible.

```
vga:VGAtop port map (
       clk_25 => I_CLK,
          sin(15 downto 8) => sinMSB, -- sinMSB à déclarer
          sin(7 downto 0)  => sinLSB, -- sinLSB à déclarer
          x_rect1(9) => '0',
          x_rect1(8 downto 1) => s_PORTB, -- s_PORTB à déclarer
          x_rect1(0) => '0',
          hsynch => hsynch,
          vsynch => vsynch,
          red => red,
          green => green,
          blue => blue);
```

5-2) Le passage de la valeur de sinus à une coordonnée y sur l'écran VGA nécessite un peu de réflexion, particulièrement à cause des nombres négatifs. Ce passage se fait à l'aide du code VHDL suivant (qui devra se trouver dans "VGAStart.vhd") :

```
1               signal srow,scol,y : STD_LOGIC_VECTOR(9 DOWNTO 0);
2               signal sina2 : STD_LOGIC_VECTOR(15 DOWNTO 0);
3               begin
4                 sina2 <= (not(sin) ) + 1;
5                 process(sin) begin
6                    if sin(15) ='0' then
7                          y <= "0100000000" - ("00" & sin(13 downto 6));
8                    else
9                          y <= "0100000000" + ("00" & sina2(13 downto 6));
10                   end if;
11                 end process;
12                -- on assemble notre composant de synchronisation
13                i1:vga_sync port map(clock_25Mhz =>clk_25, horiz_sync_out=>hsynch,
14                          vert_sync_out=>vsynch, pixel_row=>srow, pixel_column=>scol);
15                -- on assemble maintenant le composant rect
16                rectangle:rect port map(row=>srow, col=>scol, red1=>red, green1=>green,
17                          blue1=>blue, colorRGB=>"001",
18                          delta_x=>"0000000101",delta_y=>"0000000101", -- 100 et 100
19                          x_rec => x_rect1,
20                          y_rec => y
21                   );
22
```

Pouvez-vous calculer la coordonnée y sur l'écran pour les angles suivants :

$-\pi/2, +\pi/4 et +\pi/2$ à l'aide des lignes 4 à 11 du programme ci-dessus. Répondre sur la feuille réponse.

Indications : calculer les valeurs correspondantes des sinus en format Q3.13 à l'aide d'un programme sous linux.

5-3) Écrire un programme qui envoie toujours une incrémentation et qui ainsi déplace le petit rectangle verticalement. Choisir une temporisation permettant de voir ce déplacement sur l'écran.

Indications :

Une boucle comme ci-dessous devrait faire l'affaire

```
1               do {
2               //incrementation/decrementation angle
3                 PORTA = 0xFF;
4               // attente obligatoire du calcul :
5                 _delay_ms(1);
6               // positionnement horizontal du rectangle :
7                 PORTB = y;
8                 y += 1;
9               // pour avoir le temps de voir sur l'écran :
10                _delay_ms(10);
11              } while(1);
```

Travail à réaliser (dessins sinus et cosinus exo6)

A l'aide de la primitive rectangle et ses port map, ajouter un petit quadrillage de l'écran d'une couleur de votre choix.

Travail à réaliser (dessins sinus et cosinus exo7)

On vous demande de généraliser ce qui a été fait en exercice 5 et de montrer les problèmes qui apparaissent alors pour ce nouveau cosinus. Peut-on facilement les résoudre ?

Indications :

On vous donne le schéma d'implantation suivant :