TP Electronique Numérique

ENSICAEN 1ere Année
isabelle.lartigau@ensicaen.fr
matthieu.denoual@ecole.ensicaen.fr

Table des matières

Chapitre 1

Fonctions combinatoires

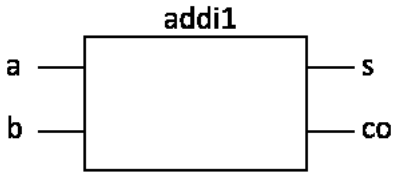
Les fonctions combinatoires sont indispensables au fonctionnement de tout système numérique. La table de vérité d'une fonction combinatoire constitue son ADN, elle fournit des informations claires sur son mode de fonctionnement.

En pratique, toute fonction combinatoire est représentée par des portes logiques simples ET, OU, INV. Pour utiliser le moins de portes possible, on passe par la table de KARNAUGH pour simplifier la fonction.

1.1 Additionneur 1 bit en logique cablée

On souhaite réaliser un additionneur 1 bit simple.

1. Compléter la table de vérité de l'additionneur :



a	b	co	s
0	0		
0	1		
1	0		
1	1		

FIGURE 1.1 – Entité d'un additionneur 1 bit + table de vérité

a	b	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

TABLE 1.1 – Table de vérité complétée de l'additionneur 1 bit

2. En déduire l'équation logique des sorties s et co.
   On en déduit que $s = \overline{a}.b + a.\overline{b}$ et que $co = a.b$
3. Réaliser le montage pratique en utilisant des portes logiques NAND (CD4011). Voir la datasheet en Annexe pour réaliser le schéma.

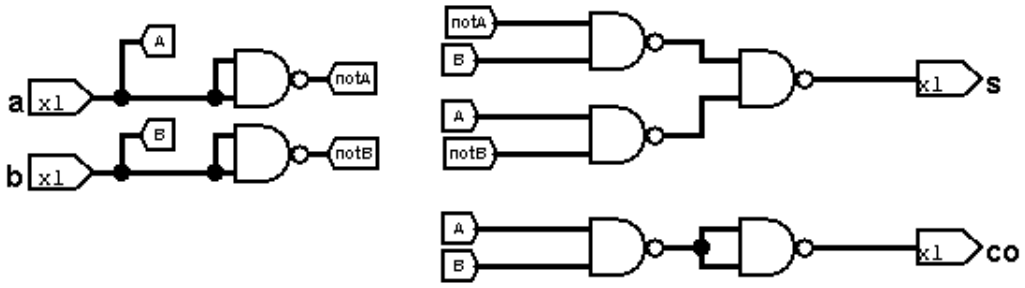


FIGURE 1.2 – Schéma de l'additionneur à base de portes NAND

4. Tester le fonctionnement pour les 4 combinaisons des entrées.

1.2 L'additionneur 1 bit

La réalisation pratique se fera maintenant sur un composant programmable de type FPGA. On utilisera le logiciel Logisim-Evolution pour décrire le circuit ou fournir la table de vérité. La table de vérité pourra être traduite en portes logiques par l'outil de synthèse. L'implémentation va générer un fichier de configuration qui, une fois chargé dans le composant, va générer les portes logiques nécessaires et les connecter comme il se doit pour réaliser l'additionneur.

1.2.1 Création d'un projet

— Démarrer le logiciel Logisim-Evolution. Le logiciel s'ouvre en créant un nouveau projet qu'il faut sauvegarder.
— *File* > *Enregistrer*

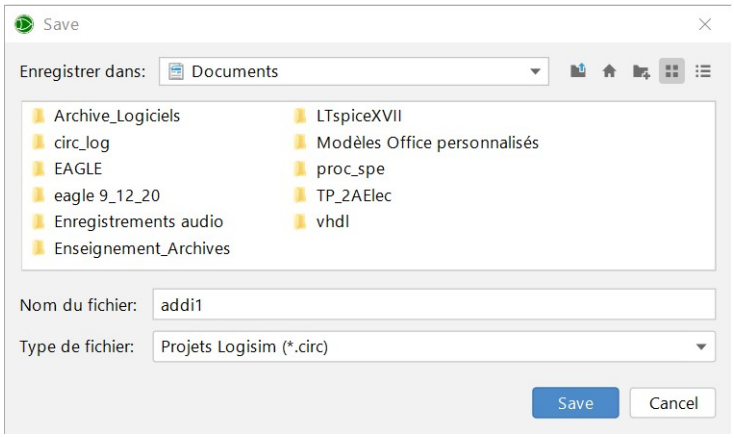


FIGURE 1.3 – Création du Projet

1.2.2 Création d'un nouveau circuit (composant)

— Renommer le circuit *main* en *addi1* : *Propriétés >> Nom du circuit >> addi1*

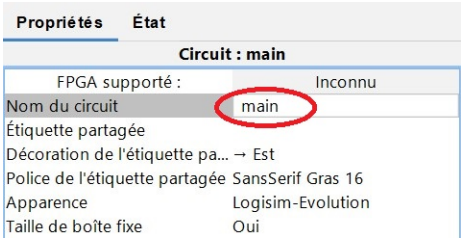


FIGURE 1.4 – Création du circuit

— Dessiner le schéma du circuit additionneur 1 bit. Le panneau de navigation permet d'accéder à tous les composants des bibliothèques pour dessiner un circuit logique. Par exemple, dans la bibliothèque *Portes logiques*, se trouvent les portes NAND nécessaires.

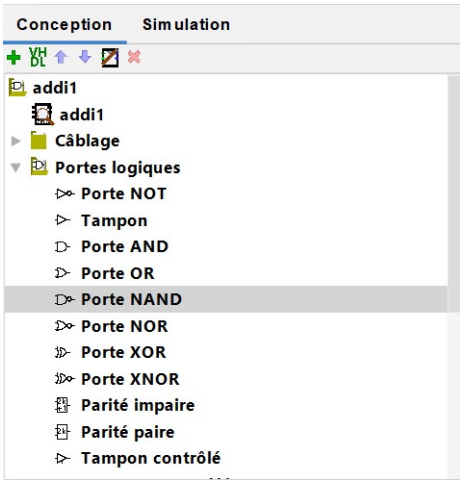


FIGURE 1.5 – Bibliothèques de Logisim

— Sélectionner et positionner les portes nécessaires.
— Puis sélectionner les broches d'entrée et de sortie dans la barre d'outils. Pour chaque entrée ou sortie, il faut indiquer son nom dans l'onglet *Etiquette*.

FIGURE 1.6 – Entrées/Sorties

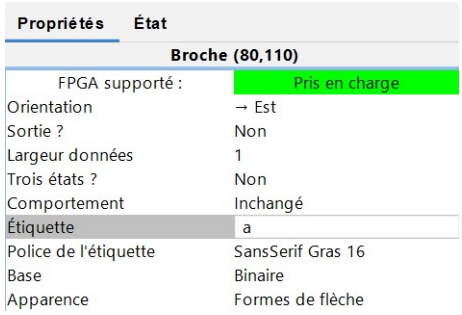


FIGURE 1.7 – Modification du nom des Entrées/Sorties

— Enfin, il faut réaliser les connectiques entre les différents éléments en sélectionnant l'outil *Câbles* dans la barre d'outils.

FIGURE 1.8 – Outil Cablage

Remarque : il est possible d'adapter le nombre d'entrées des portes logiques dans l'onglet *Nombre d'entrées*.

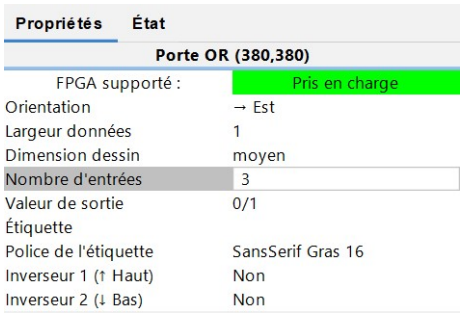


FIGURE 1.9 – Changement du nombre d'entrées d'une porte logique

1.2.3 Vérification du fonctionnement du circuit (Simulation sous Logisim-Evolution)

— Sélectionner l'onglet simulation, et vérifier que le simulation tourne en continu (premier bouton).

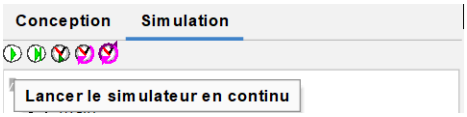


FIGURE 1.10 – Onglet Simulation

— Sélectionner dans la barre d'outils l'onglet qui permet de changer les valeurs des entrées dans le circuit. Vérifier la table de vérité en changeant les valeurs des entrées a et b.

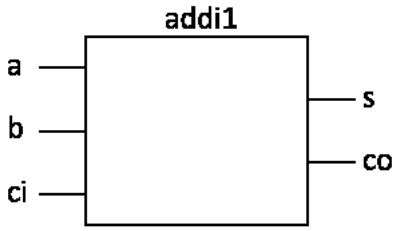
FIGURE 1.11 – Changement des valeurs d'entrée

— Fixer les valeurs des entrées pour tester toutes les combinaisons possibles des entrées a et b et valider le fonctionnement du circuit.

1.2.4 Création du circuit à partir de la table de vérité

On souhaite maintenant réaliser un additionneur complet à la place de l'additionneur simple.

— Supprimer le circuit précédent dessiné dans *addi1*.



ci	a	b	co	s
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

FIGURE 1.12 – Additionneur 1 bit complet

— Compléter la table de vérité de l'additionneur complet.

— Logisim propose la création de circuit logique à partir d'une table de vérité. On renseigne la table de vérité et on laisse à Logisim le soin de générer et dessiner le circuit correspondant. Pour cela : *Projet >> Analyser le circuit*

— Dans l'onglet *Entrées & Sorties* : Renseigner les entrées et les sorties.

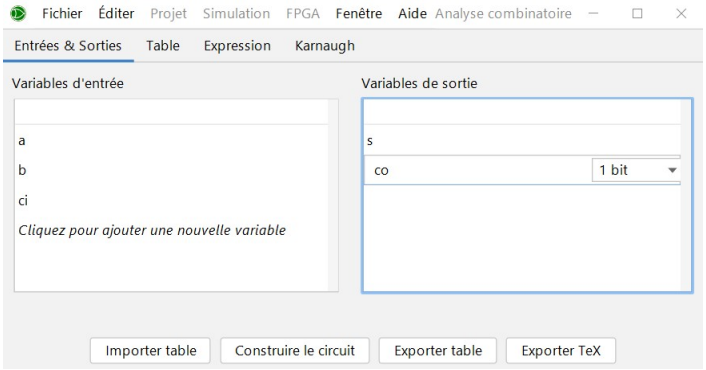


FIGURE 1.13 – Entrées et Sorties de la table de vérité

— Dans l'onglet Table : Ecrire la table de vérité. Les combinaisons des entrées sont déjà écrites.

FIGURE 1.14 – Ecriture de la table de vérité

— Vérifier l'équation logique dans l'onglet *Expression* de cette fenêtre ainsi que les tables de Karnaugh des deux sorties dans l'onglet *Karnaugh*.

— Générer le circuit : *Construire le circuit >> OK*

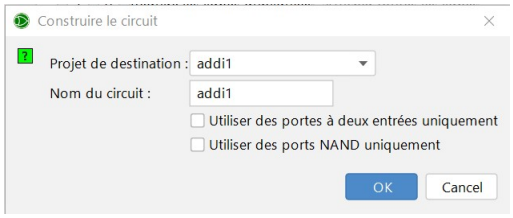


FIGURE 1.15 – Génération du circuit

— Simuler le circuit et vérifier son fonctionnement

1.2.5 Implémentation du circuit sur maquette Basys3

On souhaite implémenter ce circuit sur FPGA.

— Choisir un dossier dans lequel sera généré le projet : *Fichier >> Préférences*

— Se placer dans l'onglet *FPGA Commander parametres* et indiquer l'emplacement du projet.

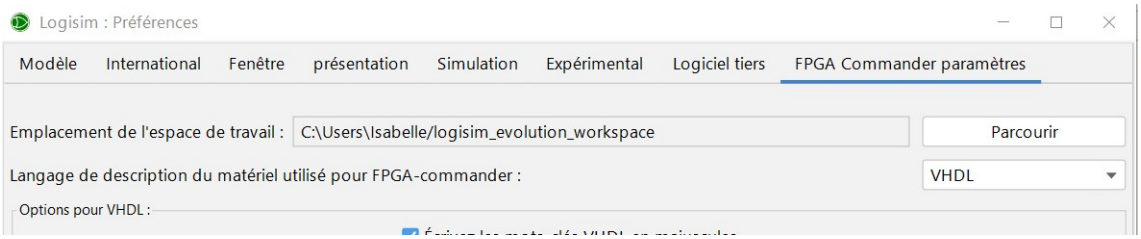


FIGURE 1.16 – Choix de l'emplacement de l'espace de travail

— Sélectionner *FPGA >> Synthétiser et télécharger.*

— Choisir la *carte Cible : BASYS3*.

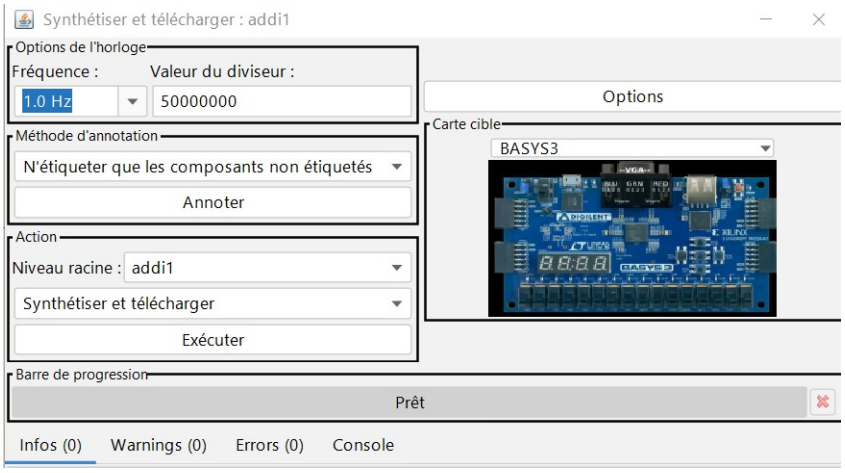


FIGURE 1.17 – Synthèse sur BASYS3

— Cliquer sur *Annoter*.
— Dans le cadre *Actions*, sélectionner
  — le niveau racine correct (*addi1*),
  — *Synthétiser et télécharger*
— Cliquer sur *Exécuter*.
— Réaliser le mappage : Associer les noms d'entrées et de sorties aux périphériques d'entrée et de sorties
  — Entrées -> Switchs
  — Sorties -> LEDs

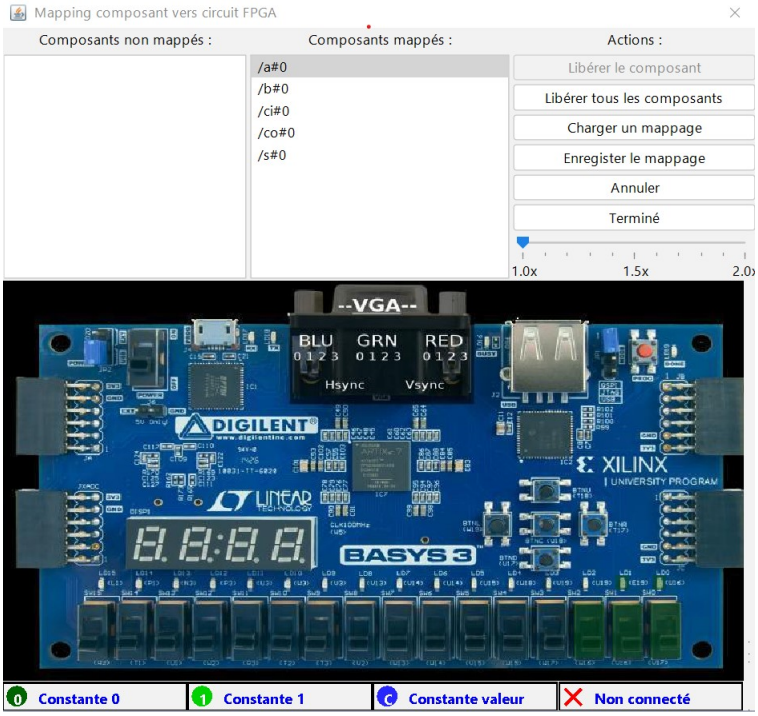


FIGURE 1.18 – Synthèse et mappage sur BASYS3

— Cliquer sur *Terminé.*
— Connecter la carte BASYS3 puis télécharger :

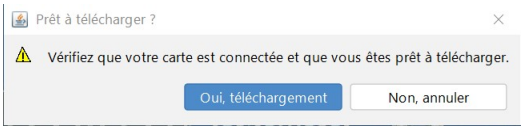


FIGURE 1.19 – Téléchargement sur BASYS3

— Tester sur la maquette.

1.2.6 Visualisation de l'implémentation avec Vivado.

— Ouvrir le projet *vp.xpr* généré par Logisim dans le dossier sélectionné précédemment. Le projet se trouve dans *Sandbox >> vp*. Le projet s'ouvre avec le logiciel Vivado.
— Dans Vivado, choisir *RTL Analysis >> Open Elaborated Design.*
— *Window >> I/O Ports* pour voir les broches d'entrées/sorties choisies sur la maquette.

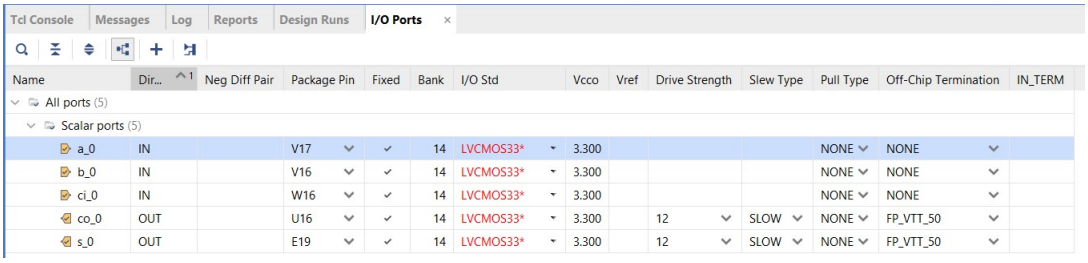


FIGURE 1.20 – Visualisation des broches sur Vivado

— Identifier dans la colonne *Package Pin* les noms des broches du FPGA choisies. Vérifier la correspondance sur la maquette BASYS3 ainsi que sur le schéma de la datasheet de la BASYS3 (voir Annexe).
— Cliquer sur *Window >> Package* pour visualiser la localisation des broches sur le boitier du FPGA.
— Cliquer sur *Implementation >> Open Implemented Design* pour visualiser l'implémentation réelle sur le composant FPGA.
— Dans l'onglet *Netlist*, sélectionner l'ensemble des Nets.

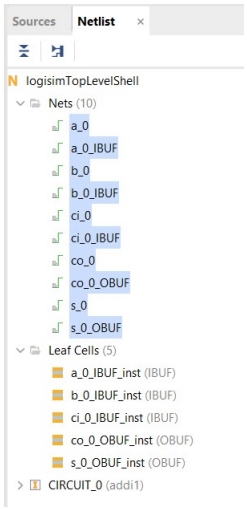


FIGURE 1.21 – Sélection des connections

— Analyser dans la fenêtre *Device* les différents éléments
  — Blocs *Inputs/Outputs (IOB)* : Pads, Buffers
  — Blocs *logiques configurables (CLB)* : Slices et LUTs : Blocs de mémoire dans lequel est implémentée la fonction logique
— Vérifier les tables de vérité implémentées dans les LUTs : Sélectionner la LUT puis *Propriétés >> Truth Table*.

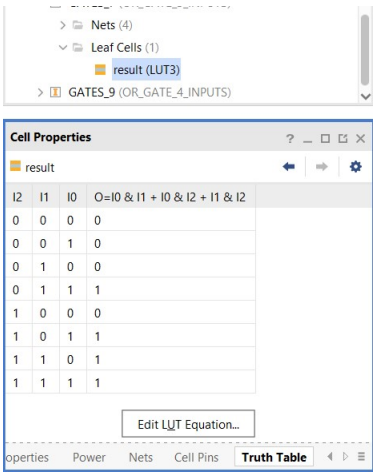


FIGURE 1.22 – Visaulisation de la table de vérité d'une LUT

Chapitre 2

L'additionneur 4 bits et décodeur 7 segments

2.1 Additionneur 4 bits

— L'additionneur 4 bits est construit avec 4 additionneurs 1 bit en cascade. Vu de l'extérieur, il ressemble à cela :

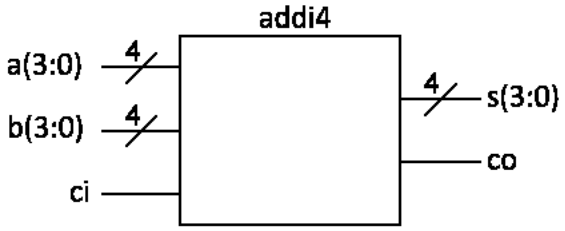


FIGURE 2.1 – Entité de l'Additionneur 4 bits

— Structure interne :

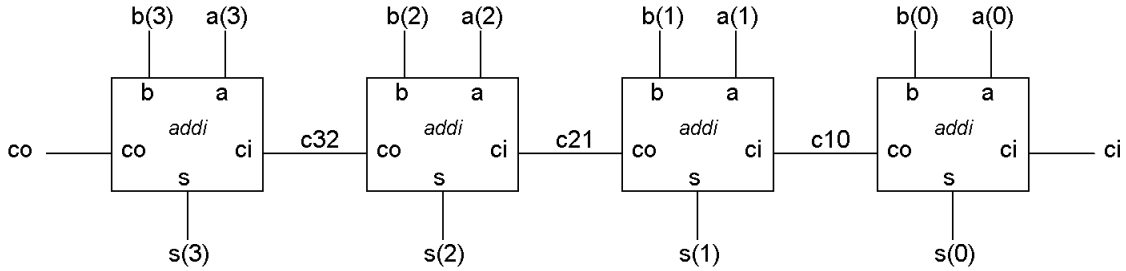


FIGURE 2.2 – Structure interne (architecture) de l'addionneur 4 bits

— Créer nouveau projet : *add_sous*.

— Récupérer le circuit *addi1* réalisé au TP précédent : *Projet >> Charger une librairie>> Librairie logisim-Evolution*

— Ajouter la librairie du premier TP : *addi1.circ*

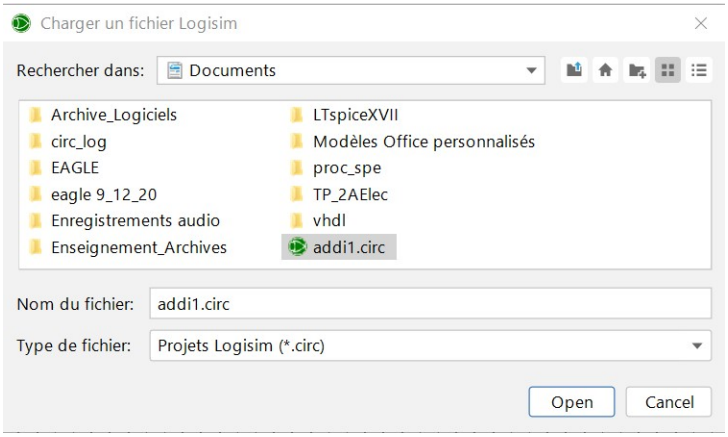


FIGURE 2.3 – Ajout d'une librairie

— La librairie *addi1* est maintenant ajoutée à la liste des bibliothèques disponibles dans Logisim.

— Renommer le main en *addi4*.

— Câbler l'additionneur 4 bits en utilisant le circuit *addi1*.

  — Les entrées sont des bus de taille 4 bits, ce paramètre est changé dans l'onglet *Largeur données*.

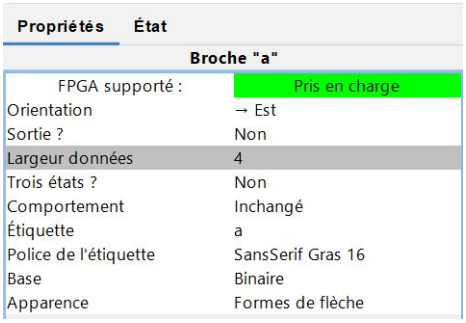


FIGURE 2.4 – Changement de la taille du bus d'entrée

  — Les bus d'entrée et de sortie doivent être découpés bit à bit pour s'adapter à l'entrée du circuit *addi1*. Pour cela, on utilise un *répartiteur (Splitter)* à sélectionner dans la librairie *Câblage*.

  — Pour ce répartiteur, indiquer son orientation, le nombre de terminaisons et la largeur du faisceau dans les propriétés.

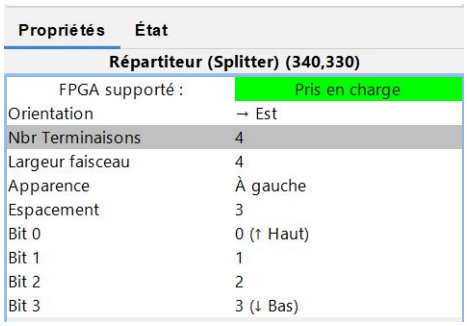


FIGURE 2.5 – Changement des propriétés du répartiteur

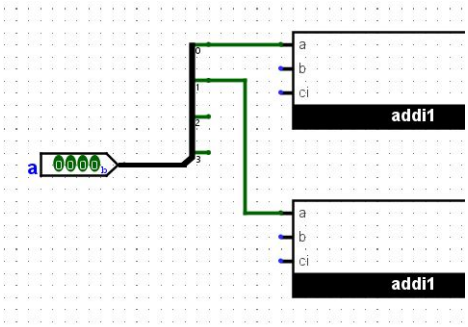


FIGURE 2.6 – Connexion du répartiteur avec le bus et le circuit addi1

— Il est possible de changer la base pour les valeurs d'entrées ou de sortie

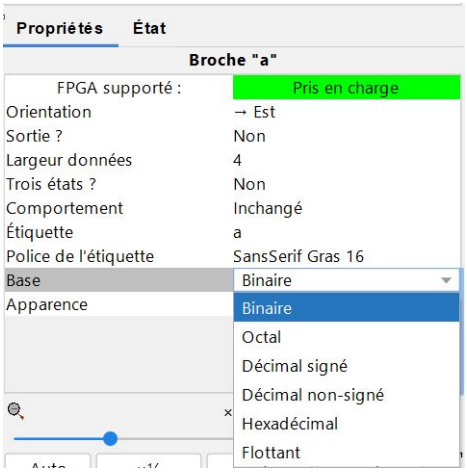


FIGURE 2.7 – Changement de base pour la représentation des nombres

— Simuler le circuit *addi4* et vérifier son fonctionnement pour quelques combinaisons des entrées.
— Implémenter et tester sur la maquette.

2.2 L'additionneur-soustracteur 4 bits

— Le soustracteur 4 bits est réalisé avec un additionneur 7 bits. Comme $A - B = A + (-B)$, on fait le compément à 2 de B, c'est à dire (le complément à 1) + 1. B est inversé pour complémenter à 1, ci=op est mise à 1 pour faire +1.

— La figure suivante donne la vue externe (l'entité) de l'additionneur soustracteur, l'entrée op permet de sélectionnet l'addition (op = 0) ou la soustraction (op = 1) :

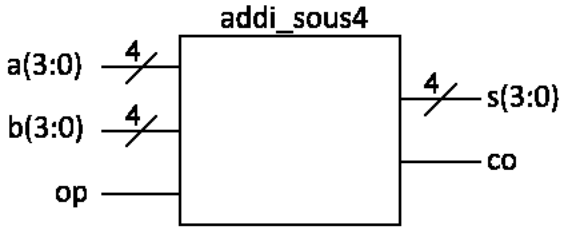


FIGURE 2.8 – Additionneur-Soustracteur 4 bits

— Structure interne (architecture) de l'additionneur soustracteur 5 bits :

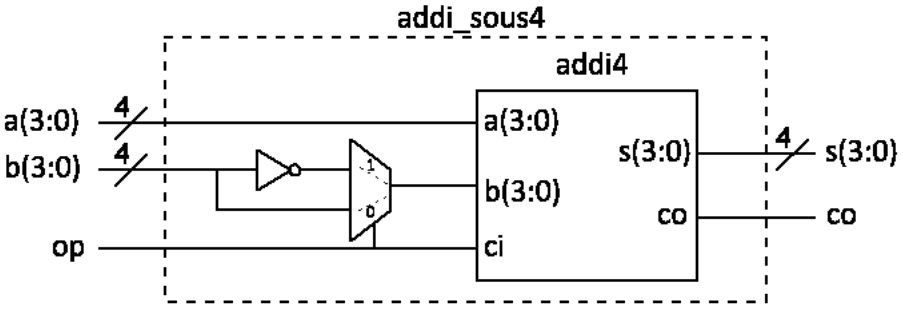


FIGURE 2.9 – Architecture de l'additionneur-soustracteur 4 bits

— Réaliser le circuit *addi_sous4*.

— Simuler le circuit et vérifier son fonctionnement pour quelques combinaisons des entrées.

— Implémenter et tester sur la maquette.

2.3 Transcodage et affichage

2.3.1 Introduction

Sur la maquette BASYS3, il y a 4 afficheurs à 7 segments qui reçoivent tous les mêmes données à afficher. Heureusement, chaque afficheur possède un signal de sélection qui permet de l'allumer ou non. En faisant tourner rapidement la sélection individuelle des afficheurs, l'œil humain a l'impression de les voir tous allumés en même temps.

— Pour commencer, il faut créer un décodeur hexadécimal vers 7 segments (circuit *decodeur_hex_7seg*) qui transforme un code binaire sur 4 bits en 7 signaux qui affichent le caractère correspondant sur l'afficheur.

— Ensuite, il faudra gérer la rotation des données sur les afficheurs (circuit *mux_4affi*), ainsi que la sélection individuelle des afficheurs (entrée *sel*)

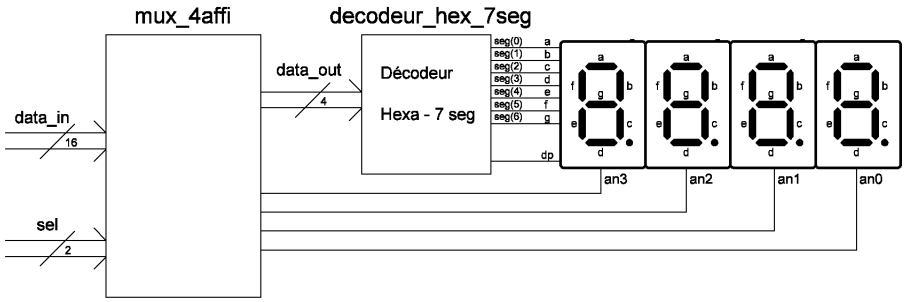


FIGURE 2.10 – Schéma de principe du transcodeur

2.3.2 Décodeur Hexa - 7 segments : circuit decodeur_hex_7seg

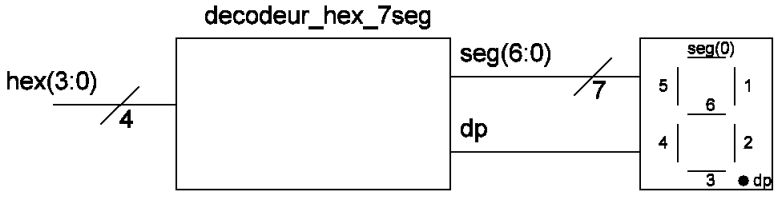


FIGURE 2.11 – Schéma de principe du décodeur héxadécimal 7 ségments

— Le décodeur reçoit une entrée sur 4 bits (code hexadécimal) et fournit 8 sorties à l'afficheur (7 segments + le point décimal).

— L'afficheur de la maquette est de type anodes communes comme le montre la figure précédente.

— Compléter la table de vérité du décodeur en considérant que le point décimal est toujours éteint.

hex(3 :0)	dp	seg(6 :0)

TABLE 2.1 – Table de vérité du décodeur héxadécimal 7 segments

— Créer un nouveau projet, nommé *affichage_multiplexe*.
— Créer un nouveau circuit nommé *decodeur_hex_7seg* dont la vue externe (entité) est représentée par la figure suivante :

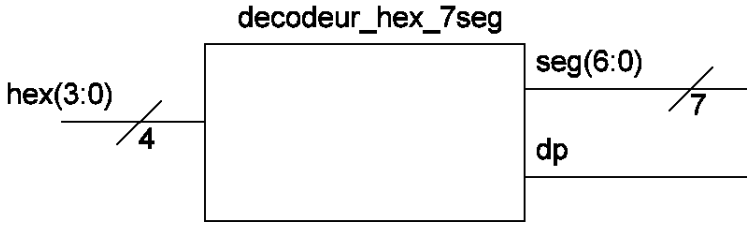


FIGURE 2.12 – Schéma de principe du décodeur hexadécimal 7 segments

— Réaliser le circuit *decodeur_hex_7seg* en renseignant sa table de vérité.
— Simuler le circuit et valider le bon fonctionnement du décodeur.
— Implémenter le circuit sur la maquette BASYS3 et tester pour toutes les combinaisons possibles. Comment fonctionnent les 4 afficheurs ?

2.3.3 Multiplexage des données : circuit mux_4affi

Pour cette application, on souhaite afficher la valeur hexadécimale des 16 switchs sur les 4 afficheurs de la maquette. Il est nécessaire de s'adresser séparément aux afficheurs par un bus de sélection. Il faut également sélectionner les 4bits de donnée que l'on souhaite afficher sur l'afficheur choisi.

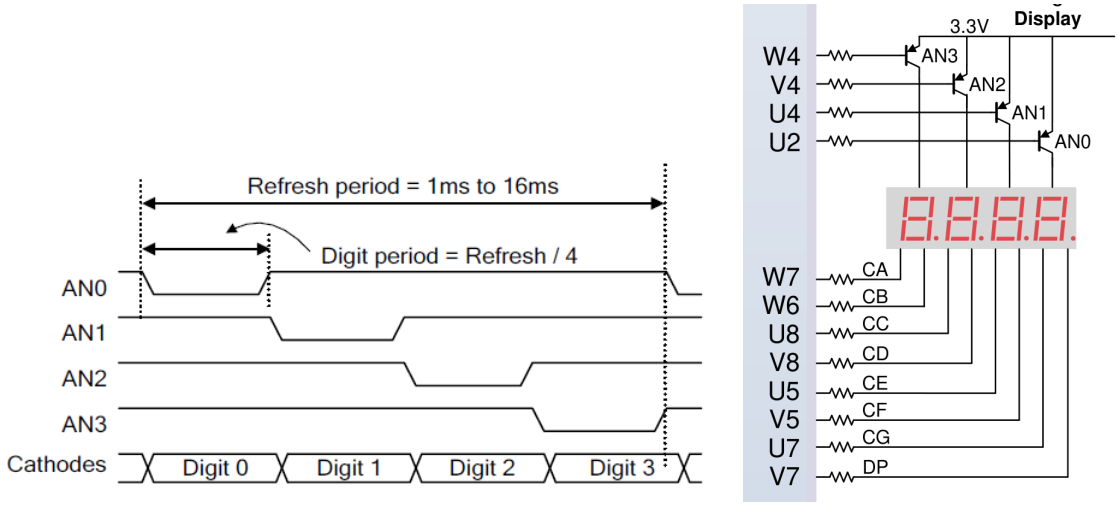


FIGURE 2.13 – Chronogramme présentant le multiplexage

— Cette étape consiste à réaliser le multiplexage des 16 signaux d'entrée nommés *data_ in(15 :0)* vers les 4 sorties *data_ out(3 :0)*. La sélection se fera par 2 fils nommés *sel(1 :0)* et le choix de l'afficheur se fait à l'aide du bus *an(3 :0)*. Le tableau suivant explique le mode de sélection.

sel(1 :0)	afficheur sélectionné	*an(3 :0)*	*data_ out(3 :0)*
00	1	1110	data_in(3 :0)
01	2	1101	data_in(7 :4)
10	3	1011	data_in(11 :8)
11	4	0111	data_in(15 :12)

TABLE 2.2 – Multiplexage de *data_ in* et sélection de *an* en fonction de l'entrée *sel*

— Créer un nouveau circuit nommé *mux_ 4affi* dont la vue externe (entité) est représentée par la figure suivante :

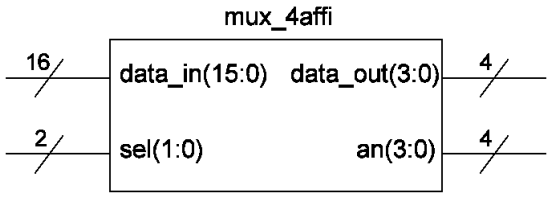


FIGURE 2.14 – Entité de mux_ 4affi

— En utilisant les multiplexeurs disponibles dans Logisim-Evolution, câbler l'architecture de l'entité *mux_ 4affi*.
— Simuler son fonctionnement.

2.3.4 Le circuit affichage complet

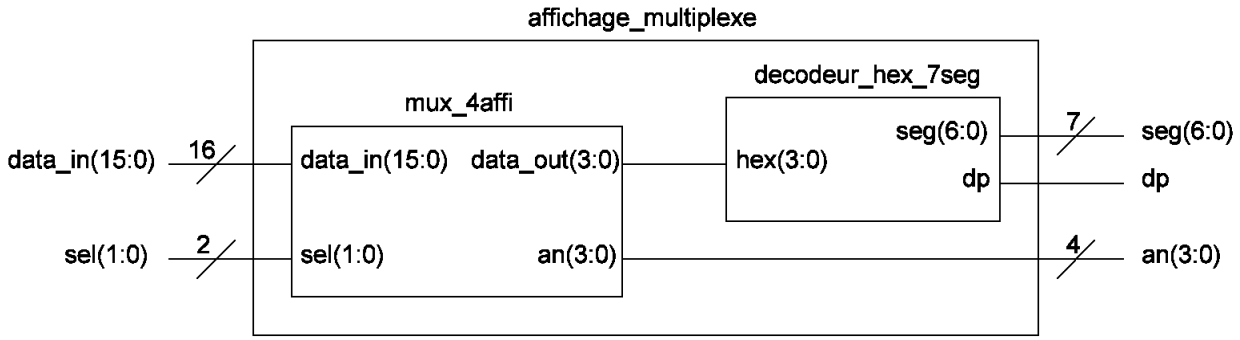


FIGURE 2.15 – Entité et architecture du circuit affichage

— Créer un nouveau circuit *affichage_multiplexe* dont l'entité et l'architecture sont représentées par la figure précédente. Câbler le circuit *affichage_multiplexe*.
— Simuler le projet.
— Programmer le composant cible et tester le fonctionnement sur la maquette. L'entrée *sel* sera réalisée par des boutons poussoirs.

Chapitre 3

Systèmes séquentiels

Les circuits logiques sont classés en deux catégories : combinatoires et séquentiels. Un circuit combinatoire se caractérise par le fait que l'état de ses sorties ne dépend que de l'état de ses entrées. C'est le cas du décodeur hexadécimal vers 7 segments étudié précédemment.

Par contre, l'état futur des sorties d'un système séquentiel dépendra de l'état des entrées mais aussi de l'état actuel des sorties. C'est le cas des feux tricolores : si le feu est actuellement vert il passera tout à l'heure à l'orange, s'il est rouge il passera au vert, ... Il passe de façon séquentielle, du vert à l'orange, de l'orange au rouge et du rouge au vert.

3.1 Le compteur décompteur sur 4 bits

On souhaite réaliser un circuit séquentiel qui compte et décompte sur 4 bits. Le système est compteur si $up/dn = 0$ et décompteur si $up/dn = 1$. Le système peut également compter ou décompter à partir d'une valeur initiale. Le schéma de principe du compteur-décompteur est représenté sur la figure suivante :

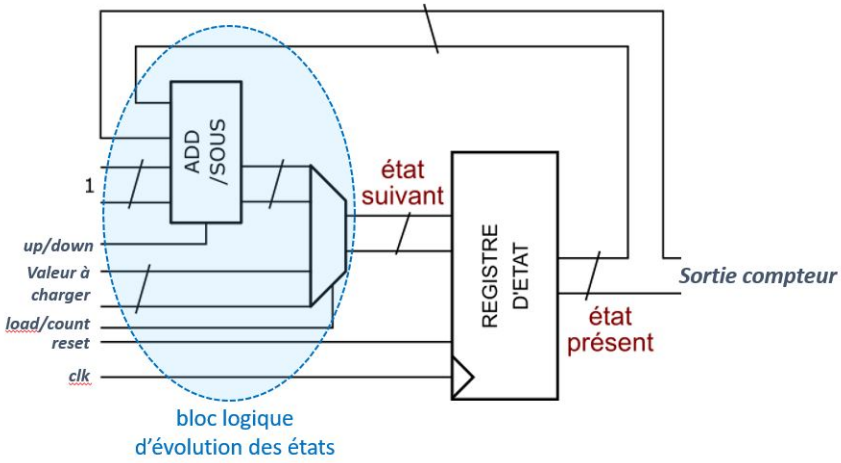


FIGURE 3.1 – Schéma de principe du compteur décompteur

— Créer un nouveau projet, nommé *compteur_decompteur*.

3.1.1 Circuit etat_suiv

— Créer un nouveau circuit nommé *etat_suiv*. Réaliser le circuit *etat_suiv* à partir du circuit *addi_sous4* et d'un multiplexeur.
— Simuler son fonctionnement.

3.1.2 Circuit registre

— Créer un nouveau circuit nommé *registre*.
— En utilisant les bascules D disponibles dans Logisim-Evolution, câbler l'architecture de l'entité *registre*.

3.1.3 Simulation du ciruit registre

On souhaite pour ce circuit séquentiel visualiser l'évolution des signaux et afficher les chronogrammes.
— Passer dans l'onglet *Simulation*. La barre d'outils propose 3 possibilités d'évolution de l'horloge :
  — le premier bouton permet de démarrer ou de stopper la simulation,
  — le troisième bouton permet d'activer le cadencement de l'horloge
  — le quatrième bouton permet de faire avancer l'horloge d'une demi-période
  — le cinquième bouton permet de faire avancer l'horloge sur une période.

FIGURE 3.2 – Possibilité d'évolution de l'horloge

— Activer le cadencement de l'horloge. Pour cette activation, il est demandé de sélectionner le signal qui servira d'horloge : Sélectionner le signal *clk*.

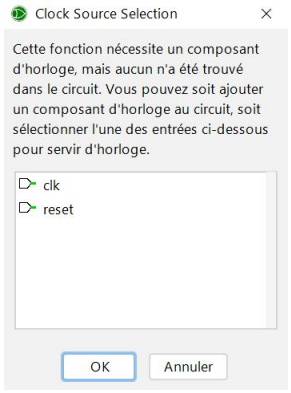


FIGURE 3.3 – Choix de l'horloge pour le circuit

— Il faut également choisir la fréquence de ce signal d'horloge : *Simulation >> Fréquence des tics >> ....*

— Tester les différentes possibilités de cadencement de l'horloge et visualiser sur le schéma l'évolution des signaux.

— On peut également visualiser le résultat de la simulation sur un chronogramme : *Simulation >> Chronogramme*

— Mettre la simuation en Pause. Dans l'onglet *Options* : Choisir *Mode continu en temps réel*

— Dans l'onglet *Chronogramme* : Choisir les signaux à afficher avec *Ajouter ou Supprimer des signaux*

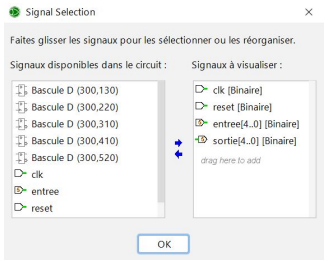


FIGURE 3.4 – Choix des signaux à afficher

— Lancer la simulation, visualiser et valider la simulation sur le chronogramme.

3.1.4 Compteur décompteur complet

— Créer un nouveau circuit nommé *compteur_decompteur.*

— Réaliser le circuit *compteur_decompteur* à partir des entités *etat_suiv, registre* selon le schéma précédent.

— Simuler le projet.

— Programmer le composant cible et tester le fonctionnement sur les LEDs de la maquette.

— Que constatez-vous ? Comment remédier à ce problème ?

3.1.5 Diviseur d'horologe

— Un compteur binaire à N bits peut être utilisé comme diviseur de fréquence.

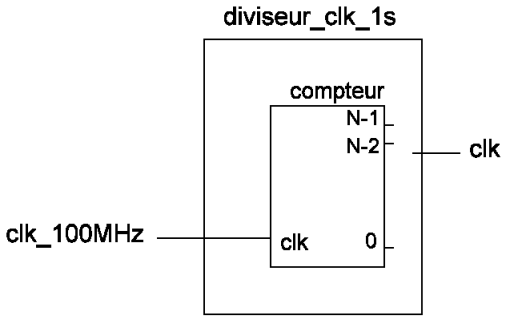


FIGURE 3.5 – Schéma de principe du diviseur

— Créer un circuit *diviseur_clk_1s*.
— Pour réaliser ce diviseur, utiliser le composant *Compteur* dans la librairie Memoire/Sequentiel de Logisim.
— Configurer et cabler le composant pour réaliser une division d'horloge par 8 en utilisant l'aide de Logisim : *Aide >> Guide Utilisateur >> Référence de la bibliothèque >> Memory Library >> Compteurs*
— Simuler et valider le fonctionnement grace aux chronogrammes.
— La fréquence de la maquette BASYS3 est 100MHz. Configurer le compteur pour avoir une fréquence de sortie de 1Hz. Utiliser dans ce cas la possibilité de fixer la valeur maximum du comptage.

3.1.6 Compteur décompteur complet avec horloge à 1Hz

— Compléter le circuit *compteur_decompteur* avec le circuit *diviseur_clk_1s*.

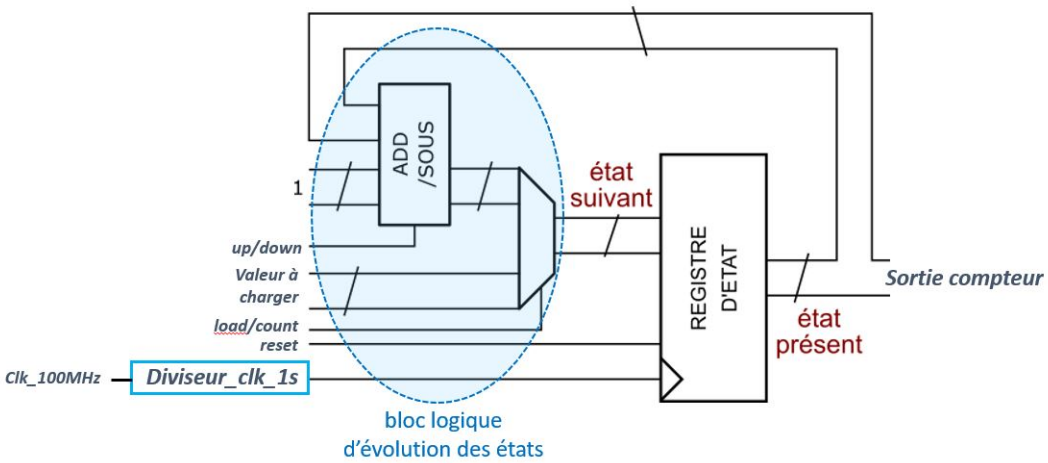


FIGURE 3.6 – Schéma de principe du compteur décompteur avec le diviseur d'horloge

— Programmer le composant cible et tester le fonctionnement sur les LEDs de la maquette.

3.2 Compteur - décompteur et Affichage

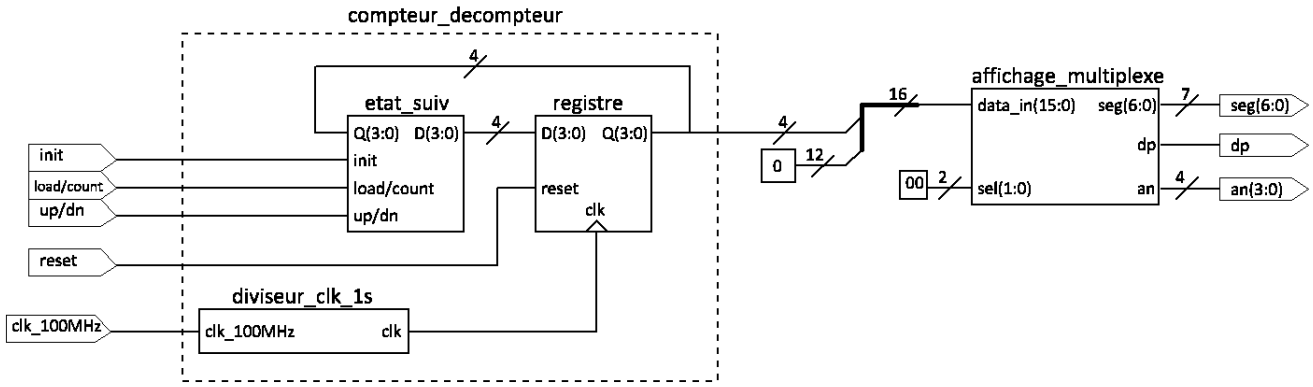


FIGURE 3.7 – Schéma de principe du compteur décompteur avec affichage

— Créer un nouveau projet *compteur_decompteur_afficheur.*

— Ajouter la librairie du TP précédent : *affichage_multiplexe.circ* dans laquelle se trouve le circuit *affichage_multiplexe.*

— Ajouter la librairie du TP précédent : *compteur_decompteur.circ* dans laquelle se trouve le circuit *compteur_decompteur.*

— Réaliser le circuit *compteur_decompteur_afficheur* à partir des entités *compteur_decompteur* et *affichage_multiplexe* selon le schéma précédent.

— Programmer le composant cible et tester le fonctionnement sur la maquette.

3.3 Retour sur le projet décodeur

On souhaite reprendre et améliorer le décodeur hexadécimal - 7 segments réalisé au TP précédent. Pour cela, on voudrait que l'entrée *sel* (qui permet de sélectionner l'afficheur) évolue régulièrement sans avoir à la commander par un bouton poussoir.

3.3.1 Génération des signaux de sélection : sel(1 :0)

L'évolution de l'entrée *sel* permet de choisir l'afficheur et les données qui le concernent. Il faut faire évoluer la valeur *sel* de façon suffisamment rapide pour balayer les 4 afficheurs sans que l'œil humain ne s'aperçoive du clignotement. On choisit une période de rafraichissement de l'ordre de 2.6

ms. Un diviseur de fréquence permet d'obtenir cette fréquence à partir de l'horloge de la maquette $f_{clk-100MHz} = 100MHz$.

— Sachant que la période de clk vaut $T_{clk} = 10ns$, une division de fréquence 2^{18} permet de fournir un signal de période $T_{sel} = 2^{18} \cdot 10ns = 2,6ms$.

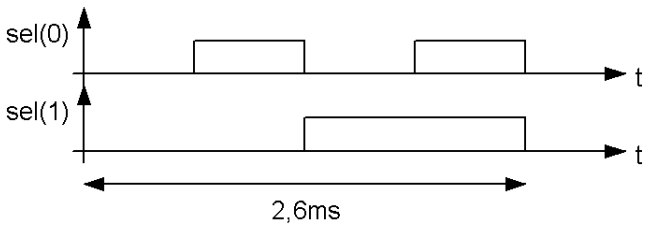


FIGURE 3.8 – Chronogramme des signaux *sel*

— Un compteur binaire à 18 bits (17 : 0) est utilisé comme diviseur de fréquence. La sortie sel(1) est connectée au bit 17 du compteur, la sortie sel(0) au bit 16.

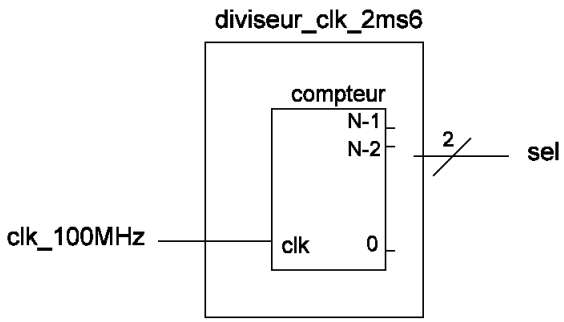


FIGURE 3.9 – Schéma de principe du diviseur

— Copier le projet *affichage_multiplexe* en *affichage_sequentiel_multiplexe.*

— Créer un nouveau circuit nommé *diviseur_clk_2ms6.*

— En utilisant un compteur disponible dans Logisim-Evolution, câbler l'architecture de l'entité *diviseur_clk_2ms6.*

3.3.2 Projet decodeur complet

— Créer un nouveau circuit *affichage_sequentiel_multiplexe* dont l'entité et l'architecture sont représentées par la figure suivante :

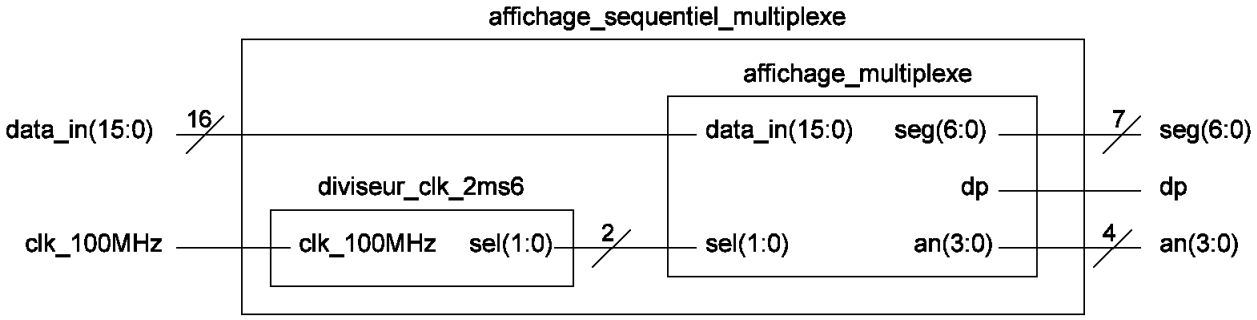


FIGURE 3.10 – Schéma de principe du décodeur complet

— Simuler le projet.
— Programmer le composant cible et tester le fonctionnement sur la maquette.

Chapitre 4

TP Moteur

Table des matières

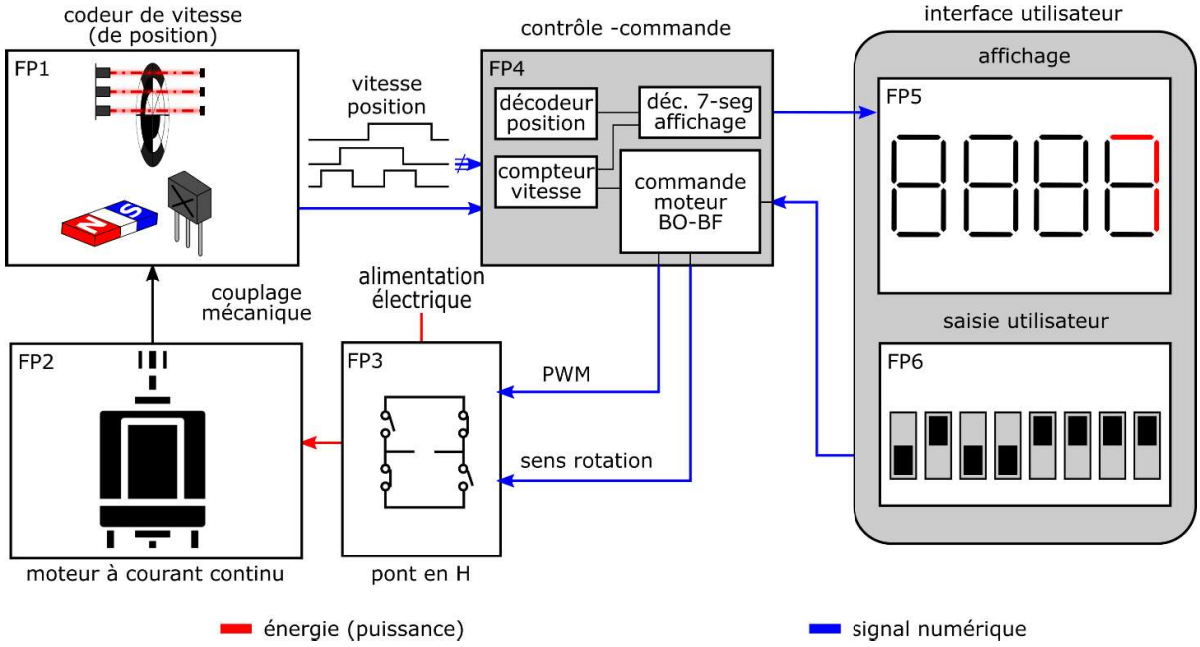


Figure 1 : Schéma fonctionnel global de l'asservissement de vitesse d'un moteur à courant continu.

FP1 : fonction principale de mesure. Mesure de la vitesse de rotation du moteur. Optionnel : mesure de la position angulaire de l'axe du moteur.

FP2 : fonction principale d'actionnement. Machine électrique à courant continu utilisée en moteur (conversion de l'énergie électrique en énergie mécanique, mouvement de rotation).

FP3 : fonction principale de conversion d'énergie électrique.

FP4 : fonction principale de contrôle-commande. Traitement des informations issus de la mesure (FP1) et génération des signaux de commande envoyés à l'étage de conversion d'énergie (FP3).

FP5 : fonction principale d'interfaçage homme-machine affichage.

FP6 : fonction principale d'interfaçage homme-machine saisie d'instruction.

Description de la carte électronique support du TP

La Figure 2 présente la carte moteur. Elle comprend un moteur à courant continu (a) associé à une roue (b). Un interrupteur (c) permet d'alimenter le moteur soit à partir d'une alimentation extérieure connectée en (e) soit à partir d'un pont en H (d) PmodHB3. Des capteurs de vitesse (f) sont reliés au connecteur d'interfaçage (g) avec la Basys3 à travers le module pont en H PmodHB3. Le strap-jumper (h) permet la sélection de l'alimentation pour le module PmodHB3 soit Basys3 (Basys3) soit alimentation extérieure (3.3V EXT).

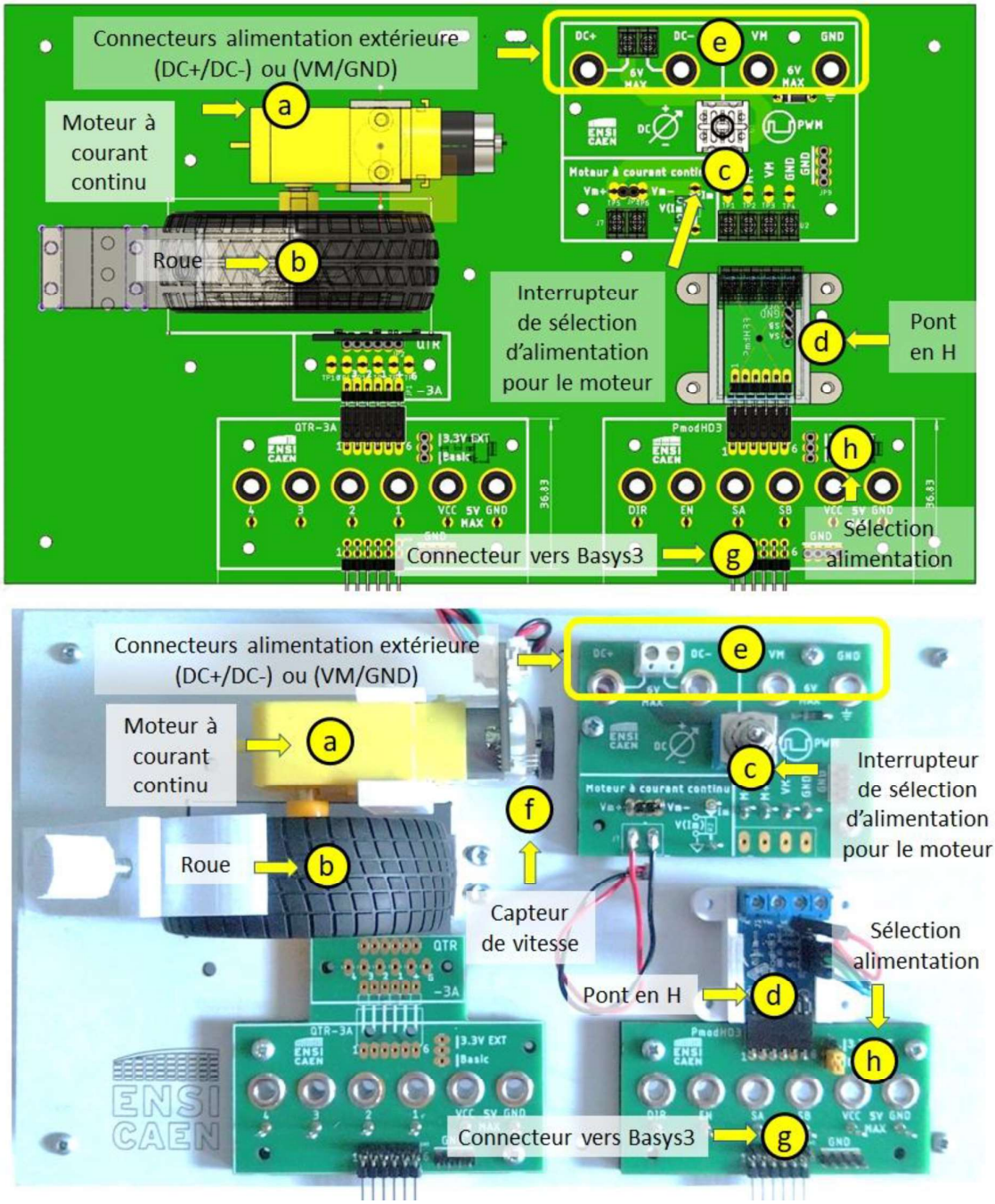
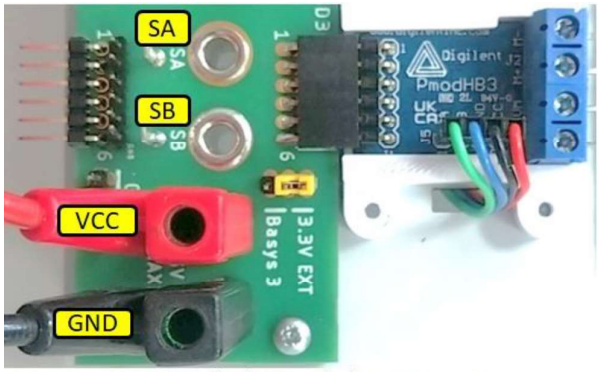


Figure 2 : (en haut) vue schématique de la carte moteur. (en bas) photographie de la carte moteur.

Partie 1 : découverte du moteur à courant continu

Objectifs : cette 1ère partie concerne l'observation du fonctionnement d'un moteur à courant continu.

- **Configurer** une alimentation continue pour délivrer un signal de 4V.
- **Connecter** l'alimentation 4V au connecteur **e** (alimentation puissance moteur, DC+/DC-).
- **Positionner** le strap-jumper sur 3.3V EXT (alimentation électronique de mesure).
- **Configurer** une seconde alimentation à 5V et la connecter sur les fiches bananes VCC et GND.
- **Connecter** les signaux **SA** et **SB** des capteurs de vitesse (**f** de la Figure 2) pour l'observation à l'oscilloscope (connecteur de la carte PmodHB3, Figure 3).
- **Positionner** l'interrupteur **c** de façon à alimenter le moteur à partir de l'alimentation continue.



Vue de dessus de la carte moteur

Vue de côté du connecteur PmodHB3

Figure 3 : Carte PmodHB3 et connecteur pour la visualisation des signaux de vitesse.

Activité 1 [Vitesse du moteur en fonction d'une tension d'alimentation continue]

Faire varier la tension d'alimentation continue et **observer** les signaux issus des capteurs de vitesse à l'oscilloscope.

Question : quel(s) paramètre(s) des signaux issus des capteurs de vitesse varie(nt) en fonction de la tension d'alimentation ?

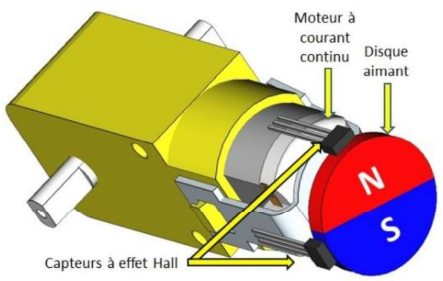


Figure 4 : Vue 3D du système de mesure de vitesse de rotation.

Expliquer la forme des signaux à partir de la Figure 4.

Identifier les réponses de chacun des capteurs.

Tracer l'évolution de la fréquence des signaux issus des capteurs en fonction de la tension d'alimentation.

Procédure de mesure :

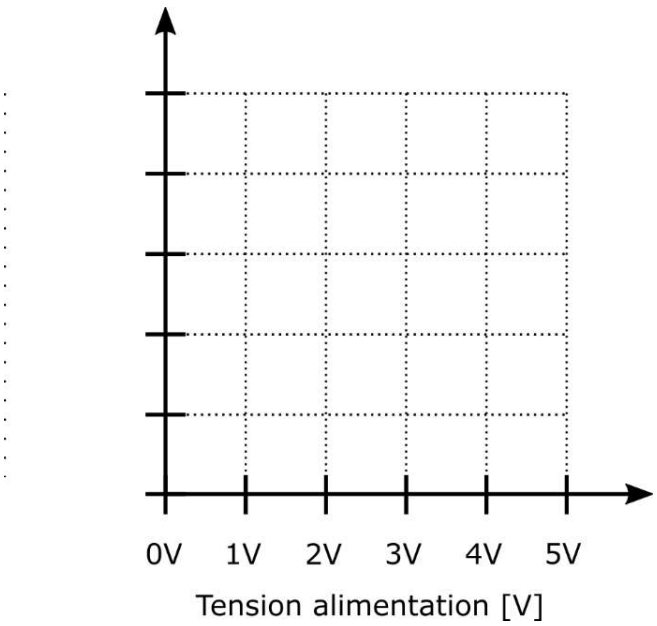


Figure 5 : Paramètre de sortie des signaux issus des capteurs de vitesse en fonction de la tension d'alimentation du moteur.

Quelle est l'expression de la vitesse de rotation du moteur en tr/min en fonction de la fréquence ?

Régler la tension d'alimentation à 5V. **Appliquer** un effort **léger** sur la roue tout en observant le courant débité par l'alimentation.

L'effort a-t-il un effet sur le courant débité ?

Activité 2 [Vitesse du moteur pour tension d'alimentation PWM]

Un signal PWM (*Pulse Width Modulation*), en modulation de largeur d'impulsion, est un signal périodique défini par 3 paramètres : une amplitude A, une période T et un rapport cyclique α. Un signal de ce type est illustré en Figure 6.

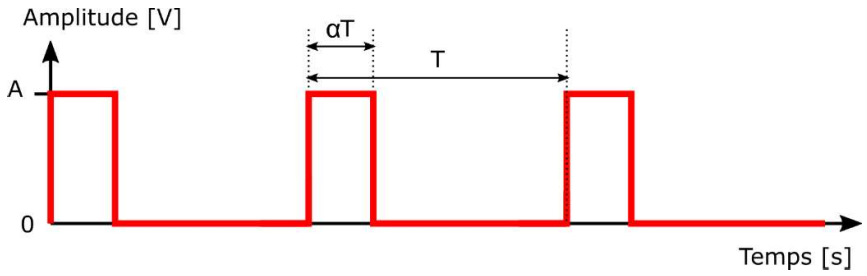


Figure 6 : Signal PWM (modulation de largeur d'impulsion).

Exprimer la valeur moyenne d'un signal PWM ?

- **Connecter** un signal (0-3.3V) généré par un générateur de fonction à l'entrée EN (*Enable*) du module PmodHB3.
- **Connecter** l'entrée DIR (*Direction*) du module PmodHB3 à la masse (0V).
- **Positionner** l'interrupteur **c** de façon à alimenter le moteur à partir du signal PWM. La carte moteur est alimentée par l'alimentation (VM/GND) connectée au connecteur **e**. **Choisir** une tension nominale de 6V.

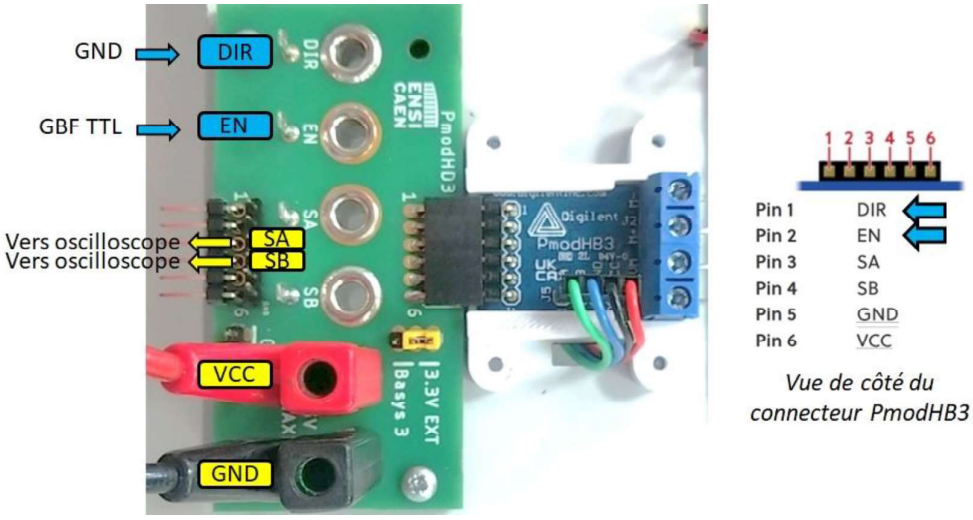


Figure 7 : Connectique du module PmodHB3 pour la commande PWM du moteur avec un générateur de fonction.

Régler la fréquence du signal PWM à 40 kHz

Faire varier le rapport cyclique du signal PWM du générateur de fonction et **observer** les signaux issus des capteurs de vitesse à l'oscilloscope.

Pourquoi la fréquence est-elle proposée à 40 kHz ?

Tracer l'évolution de la vitesse et la valeur moyenne de la tension aux bornes du moteur en fonction du rapport cyclique du signal PWM du générateur de fonction. **Régler** la fréquence du signal PWM à 40 kHz.

Procédure de mesure :

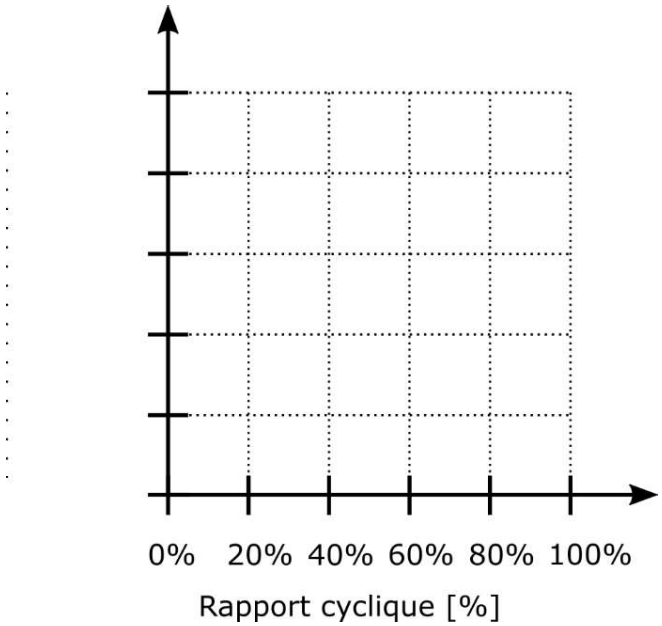


Figure 8 : Évolution de la vitesse et de la valeur moyenne de la tension aux bornes du moteur en fonction du rapport cyclique du signal PWM.

Conclusion ?

Quel élément moyenne le signal de forme PWM aux bornes d'alimentation du moteur ?

À quelle fonction élémentaire d'électronique correspond le moyennage ?

Partie 2 : Synthèse logique

La suite du travail de TP concerne la conception de fonctions logiques pour mesurer la vitesse du moteur puis pour définir une consigne de vitesse au moteur.

Déconnecter l'alimentation extérieure des fiches bananes VCC et GND. À partir de maintenant, l'électronique de mesure et de commande sera alimentée par la carte Basys3. **Positionner** le strap-jumper sur la position Basys3.

Activité 3 : [Mesure et affichage de la vitesse du moteur]

L'objectif est de mettre en forme les signaux des capteurs de vitesse de façon à afficher la vitesse sous forme numérique sur les 4 afficheurs 7-segments de la carte Basys 3. Le schéma bloc de cette fonctionnalité est illustré en Figure 9.

Quelle fonction logique utiliser pour extraire la donnée vitesse des signaux des capteurs de vitesse ?

De quoi dépendra la résolution de la mesure de vitesse ?

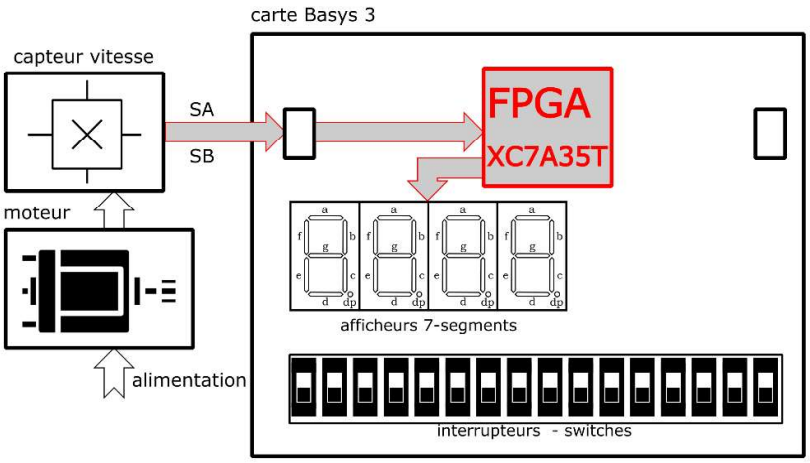


Figure 9 : Schéma bloc fonctionnel pour la mesure de vitesse du moteur.

Proposer une architecture logique permettant de mesurer la vitesse de rotation du moteur et de l'afficher sur les 4 afficheurs 7-segments de la carte Basys3.

Concevoir les éléments de l'architecture logique proposée, les **simuler** et les assembler pour créer la fonctionnalité mesure de vitesse avec affichage sur les afficheurs 7-segments. **Programmer et tester** le code sur la Basys3 en association avec la carte moteur.

Activité 4 : [Génération de consigne de vitesse]

Vérifier que DIR et EN utilisés dans la Partie 1 sont bien déconnectés pour éviter les courts-circuits. Dans cette partie, les signaux DIR et EN seront générés par la Basys3

L'objectif est de faire tourner le moteur à une vitesse définie par une consigne entrée sur les interrupteurs de la carte Basys3.

La commande de vitesse du moteur correspond aux signaux DIR et EN du module PmodHB3.

Quelle fonction utiliser pour générer un signal PWM avec le FPGA ?

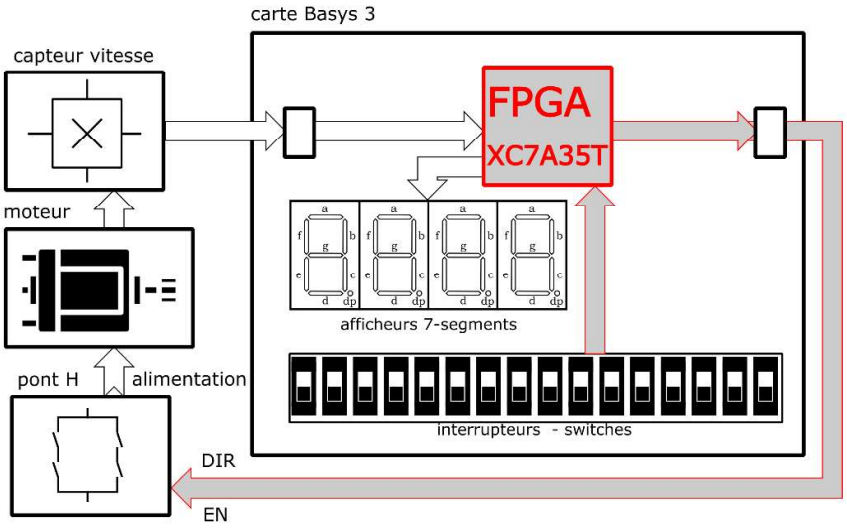


Figure 10 : Schéma bloc fonctionnel pour la consigne de vitesse.

Proposer une architecture logique pour générer les signaux PWM de consigne de vitesse.

Concevoir les éléments de l'architecture logique proposée, les **simuler** et les assembler pour créer la fonctionnalité de consigne de vitesse dont la valeur est entrée avec les interrupteurs SW15-SW0. **Programmer et tester** le code sur la Basys3 en association avec la carte moteur.

Principe capteur de vitesse à effet Hall

Les capteurs à effet Hall sont couramment utilisés pour mesurer la vitesse de rotation d'un moteur. Ils fonctionnent en détectant les variations du champ magnétique généré par des aimants présents sur le rotor du moteur. Lorsque le rotor tourne, les aimants produisent un champ magnétique qui est capté par le capteur à effet Hall. Ce dernier convertit le champ magnétique en un signal électrique de type tension. En analysant le rythme des impulsions de la tension délivrée par le capteur à effet Hall, il est possible de déterminer la vitesse de rotation du moteur avec précision. Cette information peut être utilisée pour contrôler et réguler le fonctionnement du moteur dans diverses applications, telles que l'automobile, l'industrie et l'électronique.

Les capteurs à effet Hall présentent l'avantage d'être robustes, fiables et peu sensibles aux interférences électromagnétiques. Ils sont largement utilisés dans les systèmes de contrôle de moteurs pour assurer une mesure précise et en temps réel de la vitesse de rotation.

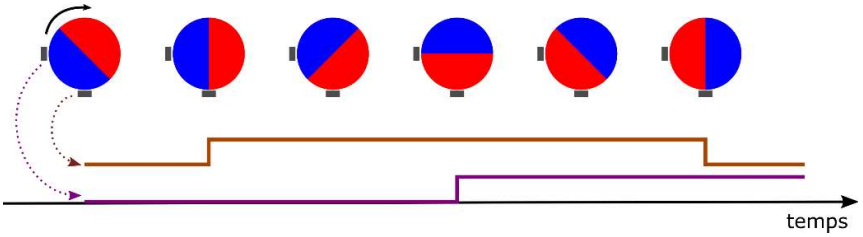


Figure 11 : Chronogramme typique de signaux issus de capteurs à effet Hall lors de la rotation

Principe du moteur à courant continu

Un moteur à courant continu à balais fonctionne grâce à l'interaction entre un champ magnétique et un courant électrique dans des bobines situées sur le rotor.

Les balais, qui sont des contacts en graphite, assurent la connexion électrique avec les bobines du rotor. Ils permettent de changer la direction du courant électrique dans les bobines en fonction de la position du rotor.

Lorsque le courant électrique (**I**) circule dans les bobines du rotor il interagit avec le champ magnétique (**B**) produit par les aimants permanents du stator, entraînant ainsi la rotation du rotor par la force magnétique (**Fm**) de Laplace.

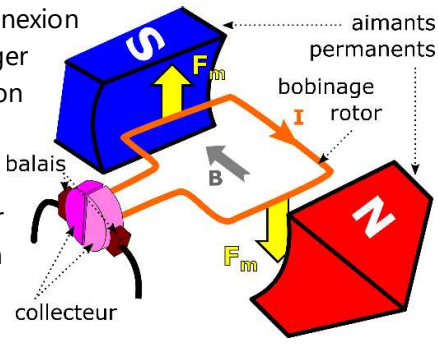
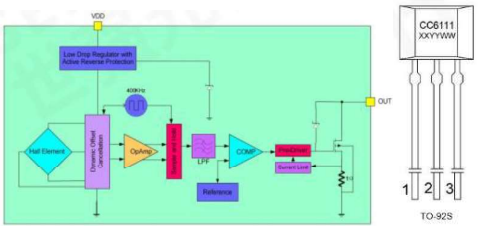
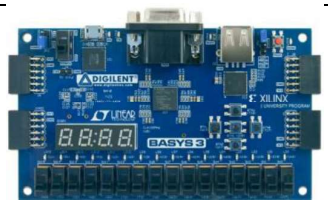


Figure 12 : Illustration du principe du moteur à courant continu.

Les balais sont en contact avec le collecteur, une pièce cylindrique solidaire de l'arbre du moteur, qui permet de commuter le courant électrique entre les différentes bobines du rotor au fur et à mesure de sa rotation, générant ainsi un mouvement continu.

Liste des éléments matériels principaux

Kit moteur + encodeur FIT0450 6 Vcc - 160 t/min - 0,8 kg.cm https://www.gotronic.fr/art-kit-moteur-encodeur-fit0450-27583.htm	
Capteur de Vitesse à effet Hall (Chopper Stabilized, High Precision, Unipolar Hall Effect Switch) https://en.sekorm.com/doc/2166553.html	
Pmod HB3 H-bridge Driver with Feedback Inputs https://fr.rs-online.com/web/p/modules-de-developpement-pour-la-robotique-la-gestion-d-alimentation-et-les-moteurs/1346445 N° Mouser : 424-PMOD-HB3 N° de fab. : 410-069 Fab. : Digilent	
Carte de développement. FPGA Digilent, Code commande RS: 134-6451, Référence fabricant: 410-183 Marque: Digilent	

Chapitre 5

Le contrôleur VGA

Dans ce dernier TP, il vous est demandé d'implémenter un circuit qui génère des signaux VGA pour dessiner un écran de couleur unie (4096 choix de couleurs) sur votre moniteur dans une résolution de 800×600 (72Hz). Les documents suivants fournissent des descriptions détaillées du fonctionnement des signaux VGA et quelques conseils sur la conception du contrôleur VGA.

5.1 Présentation du VGA

VGA signifie Video Graphics Array. Initialement, il se réfère spécifiquement au matériel d'affichage introduit pour la première fois avec l'ordinateur IBM® PS/2 en 1987. Avec son utilisation généralisée, il fait désormais généralement référence aux normes d'affichage analogiques des ordinateurs (définies par VESA®), le connecteur DB-15 (généralement connu sous le nom de connecteur VGA).

Les normes d'affichage d'ordinateur analogique sont spécifiées, publiées, protégées par copyright et vendues par l'organisation VESA (www.vesa.org). Les informations de synchronisation utilisées dans ce TP sont un exemple de la façon dont un moniteur VGA peut être piloté dans une résolution de 800 × 600.

Un connecteur DB-15, communément appelé connecteur VGA, est un connecteur subminiature D à 15 broches à trois rangées (du nom de leur blindage métallique en forme de D). Le nom de chaque broche est illustré à la Figure ci-dessous. Nous nous concentrerons uniquement sur les 5 signaux sur 15 broches dans ce projet. Ces signaux sont Red, Grn, Blue, HS et VS. Red, Grn et Blue sont trois signaux analogiques qui spécifient la couleur d'un point sur l'écran, tandis que HS et VS fournissent une référence de position de l'endroit où le point doit être affiché sur l'écran. En pilotant correctement ces cinq signaux conformément à la spécification de synchronisation VGA, nous pouvons afficher tout ce que nous voulons sur n'importe quel moniteur. Pour comprendre comment ces signaux doivent être pilotés, nous devons examiner le fonctionnement réel de nos moniteurs.

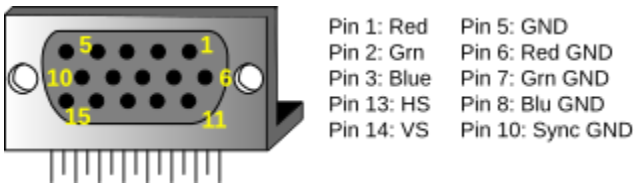


FIGURE 5.1 – Brochage d'un connecteur DB-15 (VGA)

5.2 Comment fonctionnent nos moniteurs ?

Les écrans VGA basés sur CRT (cathode-ray tube) utilisent des faisceaux d'électrons mobiles modulés en amplitude (ou rayons cathodiques) pour afficher des informations sur un écran recouvert de phosphore. Les écrans LCD utilisent un réseau de commutateurs qui peuvent imposer une tension sur une petite quantité de cristaux liquides, modifiant ainsi la permittivité de la lumière à travers le cristal pixel par pixel. Bien que la description suivante soit limitée aux écrans CRT, les écrans LCD ont évolué pour utiliser les mêmes synchronisations de signal que les écrans CRT (la discussion sur les « signaux » ci-dessous concerne donc à la fois les écrans CRT et les écrans LCD). Les écrans CRT couleur utilisent trois faisceaux d'électrons (un pour le rouge, un pour le bleu et un pour le vert) pour dynamiser le luminophore qui recouvre la face interne de l'extrémité d'affichage d'un tube à rayons cathodiques (voir Figure ci-dessous).

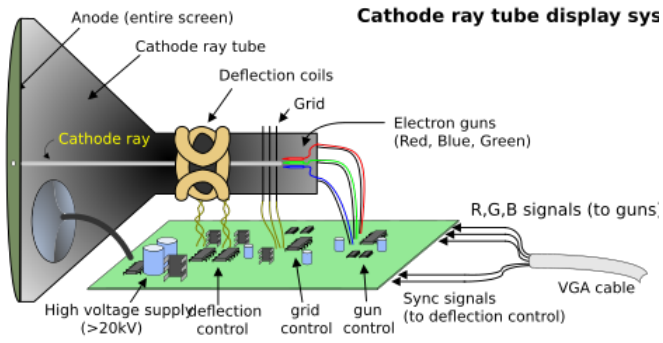


FIGURE 5.2 – Fonctionnement d'un moniteur CRT

Les faisceaux d'électrons émanent de « canons à électrons » qui sont des cathodes chauffées à pointe fine placées à proximité d'une plaque annulaire chargée positivement appelée « grille ». La force électrostatique imposée par la grille tire les rayons d'électrons sous tension des cathodes, et ces rayons sont alimentés par le courant qui circule dans les cathodes. Ces rayons de particules sont initialement accélérés vers la grille, mais ils tombent rapidement sous l'influence de la force électrostatique beaucoup plus grande qui résulte de la charge de toute la surface d'affichage recouverte de phosphore du CRT à 20 kV (ou plus). Les rayons sont focalisés en un faisceau fin lorsqu'ils traversent le centre des grilles, puis ils accélèrent pour impacter la surface d'affichage recouverte de phosphore. La surface du phosphore brille

vivement au point d'impact, et il continue à briller pendant plusieurs centaines de microsecondes après le retrait du faisceau. Plus le courant introduit dans la cathode est important, plus le phosphore brillera.

Entre la grille et la surface d'affichage, le faisceau traverse le col du CRT où deux bobines de fil produisent des champs électromagnétiques orthogonaux. Comme les rayons cathodiques sont composés de particules chargées (électrons), ils peuvent être déviés par ces champs magnétiques. Les formes d'onde de courant traversent les bobines pour produire des champs magnétiques qui interagissent avec les rayons cathodiques et les amènent à traverser la surface d'affichage selon un motif «trame», horizontalement de gauche à droite et verticalement de haut en bas, comme illustré sur la figure ci-dessous Lorsque le rayon cathodique se déplace sur la surface de l'affichage, le courant envoyé aux canons à électrons peut être augmenté ou diminué pour modifier la luminosité de l'affichage au point d'impact du rayon cathodique.

Les informations ne sont affichées que lorsque le faisceau se déplace dans le sens « avant » (de gauche à droite et de haut en bas), et non pendant le temps où le faisceau est réinitialisé sur le bord gauche ou supérieur de l'écran. Une grande partie du temps d'affichage potentiel est donc perdue dans des périodes de « suppression » lorsque le faisceau est réinitialisé et stabilisé pour commencer un nouveau passage d'affichage horizontal ou vertical. La taille des faisceaux, la fréquence à laquelle le faisceau peut être tracé sur l'écran et la fréquence à laquelle le faisceau d'électrons peut être modulé déterminent la résolution de l'affichage.

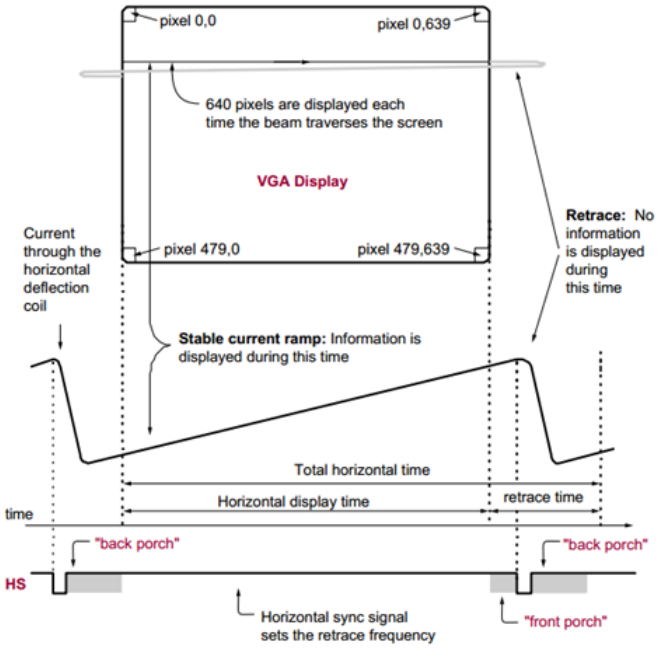


FIGURE 5.3 – Fonctionnement de l'affichage sur un moniteur CRT

5.3 Spécification de synchronisation VGA

Les écrans VGA modernes peuvent prendre en charge différentes résolutions, et un circuit de contrôleur VGA dicte la résolution en produisant des signaux de synchronisation pour contrôler les modèles de trame. Le contrôleur doit produire des impulsions de synchronisation à 3,3 V (ou 5 V) pour régler la fréquence à laquelle le courant circule dans les bobines de déviation, et il doit garantir que les données vidéo sont appliquées aux canons à électrons au bon moment. Les affichages vidéo raster (Image matricielle) définissent un certain nombre de "lignes" qui correspondent au nombre de passages horizontaux que la cathode effectue sur la zone d'affichage, et un certain nombre de "colonnes" qui correspondent à une zone sur chaque ligne qui est affectée à un "élément d'image" , ou pixel. Les affichages typiques utilisent de 240 à 1200 lignes et de 320 à 1600 colonnes. La taille globale d'un affichage et le nombre de lignes et de colonnes déterminent la taille de chaque pixel.

Les données vidéo proviennent généralement d'une mémoire vidéo (VRAM) ; avec un ou plusieurs octets attribués à chaque emplacement de pixel (la carte BASYS3 utilise 12 bits (3*4bits) par pixel). Le contrôleur doit indexer dans la mémoire vidéo lorsque les faisceaux se déplacent sur l'écran, et récupérer et appliquer les données vidéo à l'écran précisément au moment où le faisceau d'électrons se déplace sur un pixel donné.

Un circuit contrôleur VGA doit générer les signaux de synchronisation HS et VS et coordonner la livraison des données vidéo sur la base de l'horloge pixel. L'horloge pixel définit le temps disponible pour afficher un pixel d'information. Le signal VS définit la fréquence de "rafraîchissement" de l'affichage, ou la fréquence à laquelle toutes les informations sur l'affichage sont redessinées. La fréquence de rafraîchissement minimale est fonction de l'intensité du phosphore et du faisceau d'électrons de l'écran, avec des fréquences de rafraîchissement pratiques comprises entre 50 Hz et 120 Hz. Le nombre de lignes à afficher à une fréquence de rafraîchissement donnée définit la fréquence de « retracement » horizontal.

5.4 Spécification de synchronisation pour 800x600@72Hz

La figure ci-dessous et le tableau ci-dessous fournissent les spécifications de synchronisation pour une résolution de 800 × 600 à une fréquence d'images de 72 Hz :

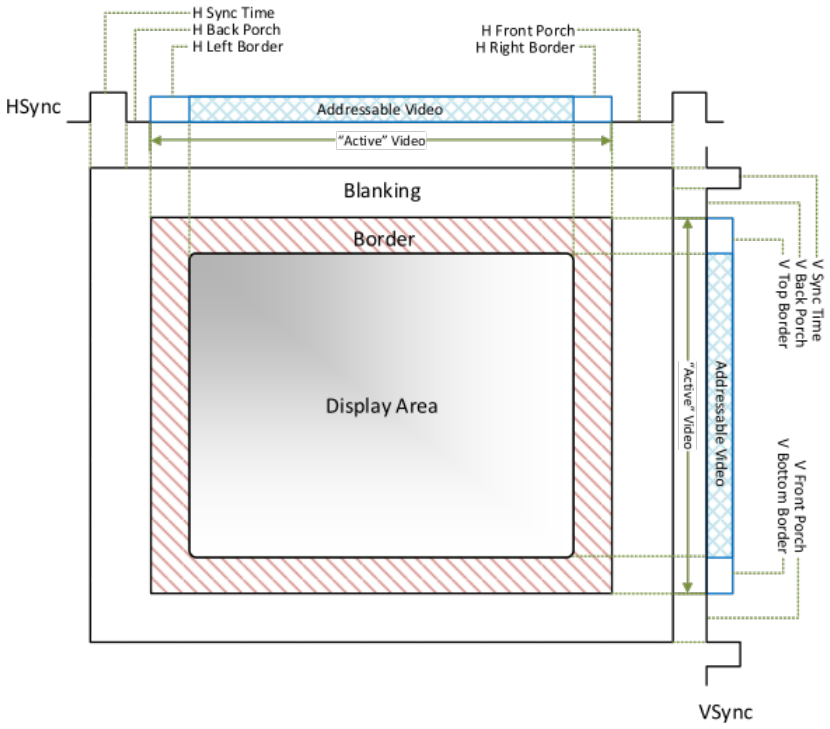


FIGURE 5.4 – Spécification des signaux HS et VS

Description	Notation	Temps	Largeur/Freq
Pixel Clk	Tpix	20 ns	50Mhz
Horizontal Sync Time	Ths	2.4µs	120 Pixels
Horizontal Back Porch	Thbp	1.28µs	64 Pixels
Horizontal Front Porch	Thfp	1.12µs	56 PIxels
Horizontal Addressable Video Time	Thaddr	16µs	800 Pixels
Horizontal Adressable L/R Border	Thbd	0µs	0 Pixels
Whole Line Time	Tl	20.8µs	1040 Pixels
Vertical Sync Time	Tvs	124,8µs	6 Lines
Vertical Back Porch	Tvbp	478.4 µs	23 Lines
Vertical Front Porch	Tvfp	769.6 µs	37 Lines
Vertical Addressable Video Time	Tvaddr	12.48 ms	600 Lines
Vertical Adressable L/R Border	Tvbd	0µs	0 Lines
Whole Frame Time	Tf	13.8528 ms	666 Lines

TABLE 5.1 – Spécifications des timings

5.5 Astuces

Tout d'abord, vous avez besoin d'un diviseur d'horloge pour générer l'horloge pixel, qui fournit une référence temporelle aux signaux HS et VS. La fréquence d'horloge des pixels est de 50 MHz dans la spécification.

Deuxièmement, vous avez besoin de deux compteurs, un compteur (compteur horizontal) pour compter les pixels dans chaque ligne et un autre compteur (compteur vertical) pour compter les lignes dans une trame. Le compteur horizontal doit se réinitialiser lorsqu'il atteint la fin de la ligne (1039 dans ce cas), et lorsqu'il se réinitialise, il doit fournir un signal à l'entrée Enable du compteur vertical afin que le compteur vertical puisse ajouter 1 lorsqu'une nouvelle ligne commence. De même, le compteur vertical doit se réinitialiser lorsqu'il atteint la fin de la trame.

Nous pouvons comparer les valeurs des compteurs aux constantes définies dans la spécification pour la généreration des signaux HS et VS.

La figure ci-dessous montre la génération HS et VS basée sur les valeurs de compteur :

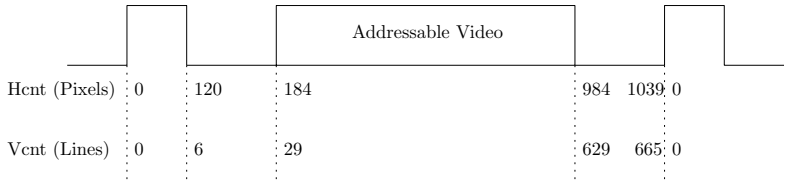


FIGURE 5.5 – Spécification des signaux HS et VS

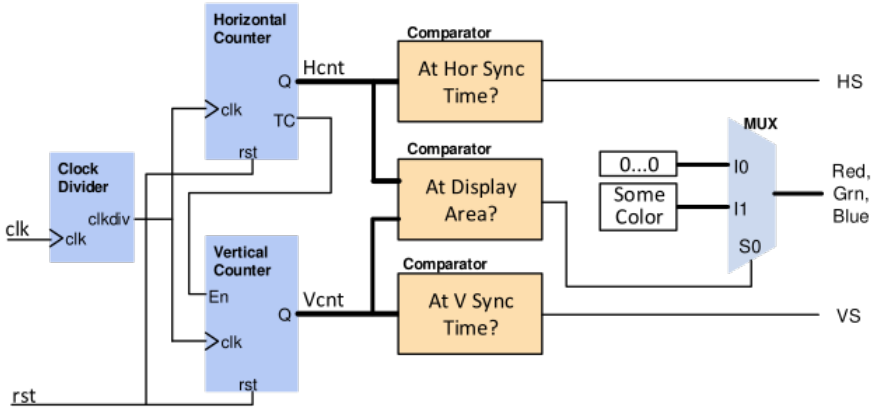


FIGURE 5.6 – Schéma fonctionnel du controleur VGA

5.6 Travail à réaliser

1. Réaliser le controleur VGA 800x600@72Hz.
2. En utilisant 12 switchs d'entrées, tester sur la carte Basys3 le changement de couleur sur le moniteur (image unie).
3. Créer un nouveau circuit qui :
   (a) sera constitué d'une mémoire dans laquel vous rajouterez le fichier "image.txt" disponible sur Moodle.

(b) viendra lire les pixels contenus dans la mémoire et les transfèrera correctement au controleur VGA .

4. Quelle image s'affiche ?

Chapitre 6

Annexes

MC14001UB, MC14011UB

UB-Suffix Series CMOS Gates

The UB Series logic gates are constructed with P and N channel enhancement mode devices in a single monolithic structure (Complementary MOS). Their primary use is where low power dissipation and/or high noise immunity is desired. The UB set of CMOS gates are inverting non–buffered functions.

- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Linear and Oscillator Applications
- Capable of Driving Two Low–power TTL Loads or One Low–power Schottky TTL Load Over the Rated Temperature Range
- Double Diode Protection on All Inputs
- Pin–for–Pin Replacements for Corresponding CD4000 Series UB Suffix Devices

MAXIMUM RATINGS (Voltages Referenced to V_{SS}) (Note 1.)

Symbol	Parameter	Value	Unit
V_{DD}	DC Supply Voltage Range	–0.5 to +18.0	V
V_{in}, V_{out}	Input or Output Voltage Range (DC or Transient)	–0.5 to V_{DD} + 0.5	V
I_{in}, I_{out}	Input or Output Current (DC or Transient) per Pin	±10	mA
P_D	Power Dissipation, per Package (Note 2.)	500	mW
T_A	Ambient Temperature Range	–55 to +125	°C
T_{stg}	Storage Temperature Range	–65 to +150	°C
T_L	Lead Temperature (8–Second Soldering)	260	°C

1. Maximum Ratings are those values beyond which damage to the device may occur.
2. Temperature Derating:
   Plastic “P and D/DW” Packages: – 7.0 mW/°C From 65°C To 125°C

This device contains protection circuitry to guard against damage due to high static voltages or electric fields. However, precautions must be taken to avoid applications of any voltage higher than maximum rated voltages to this high–impedance circuit. For proper operation, V_{in} and V_{out} should be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$.

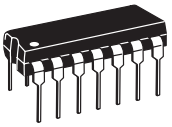
Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}). Unused outputs must be left open.

ON Semiconductor

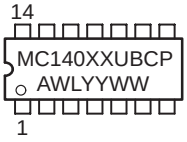
http://onsemi.com

MC14001UB
Quad 2–Input NOR Gate
MC14011UB
Quad 2–Input NAND Gate

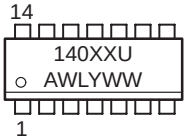
MARKING DIAGRAMS



PDIP–14
P SUFFIX
CASE 646



SOIC–14
D SUFFIX
CASE 751A

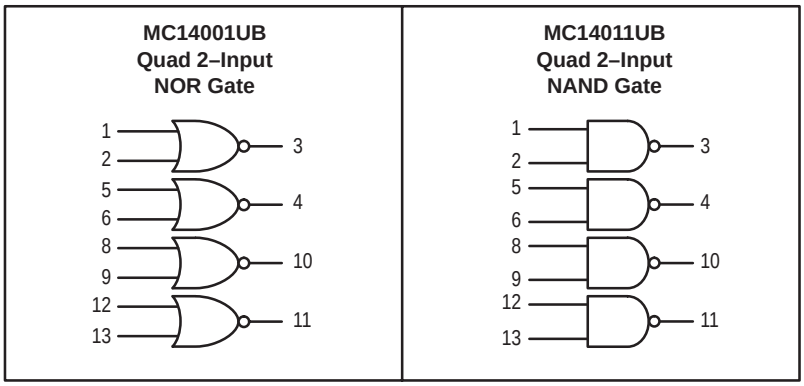


XX = Specific Device Code
A = Assembly Location
WL or L = Wafer Lot
YY or Y = Year
WW or W = Work Week

ORDERING INFORMATION

Device	Package	Shipping
MC14001UBCP	PDIP–14	2000/Box
MC14001UBD	SOIC–14	55/Rail
MC14001UBDR2	SOIC–14	2500/Tape & Reel
MC14011UBCP	PDIP–14	2000/Box
MC14011UBD	SOIC–14	55/Rail
MC14011UBDR2	SOIC–14	2500/Tape & Reel

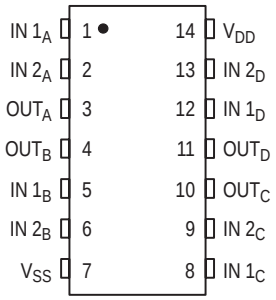
LOGIC DIAGRAMS



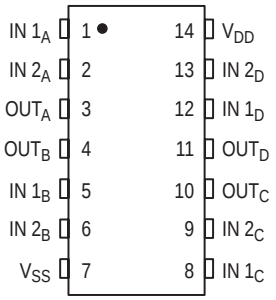
V_{DD} = PIN 14
V_{SS} = PIN 7
FOR ALL DEVICES

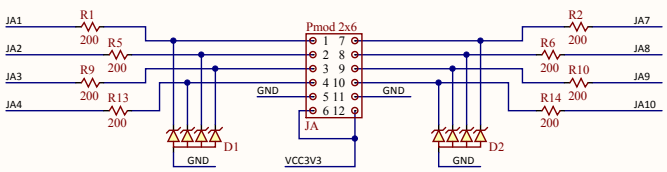
PIN ASSIGNMENTS

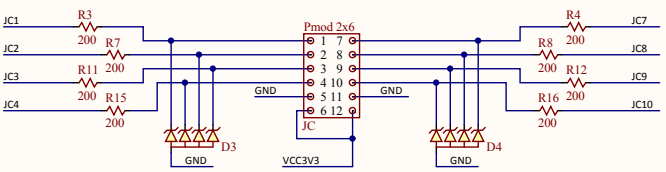
MC14001UB
Quad 2–Input NOR Gate

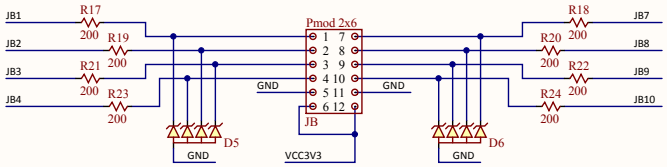


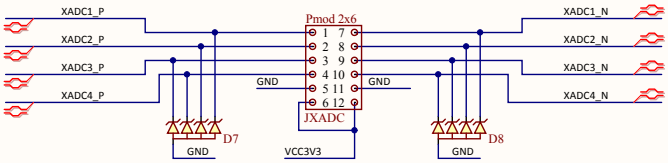
MC14011UB
Quad 2–Input NAND Gate

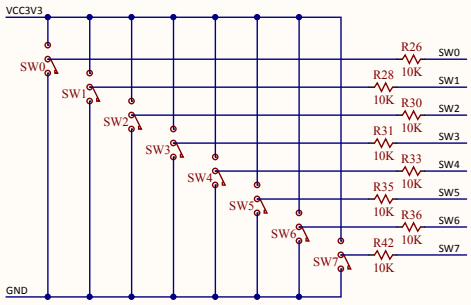


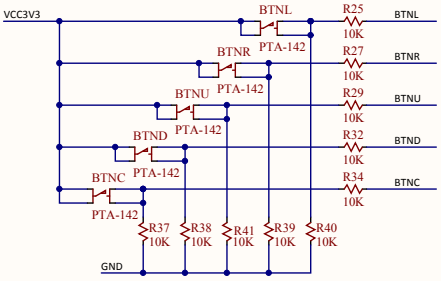


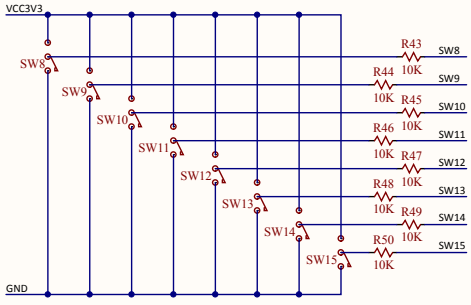












F1 Foot
F2 Foot
F3 Foot
F4 Foot

Digilent Inc. | CE | ROHS | Chinese ROHS | Basys 3 | XUP | Linear Tech

Title	Basys 3
Rev	C.0
Copyright	2014
Circuit	PMOD, I/O
Doc#	500-183
Engineer	MTA
Author	GMA
Date	5/21/2014
Sheet#	1 out of 8

DIGILENT BEYOND THEORY