ELECINF102
Processeurs et Architectures Numériques

Contrôle de connaissances

14 juin 2018 à 08h30

Document autorisé : une feuille recto-verso

Durée : 1h30

Ce contrôle comporte 3 parties **indépendantes** pouvant donc être traitées dans n'importe quel ordre...

1. Réalisation d'un digicode.
2. Analyse et mise au goût du jour d'un vieux schéma.
3. Détection des touches du clavier d'un digicode.

Consignes importantes : Si des **schémas** sont réalisés, ils doivent être impérativement clairs et sans ambiguïté. Les dimensions des bus doivent être indiquées. Si nécessaire le sens des signaux doit être précisé. Pour la logique synchrone, les signaux d'horloge et d'initialisation asynchrone (reset) ne seront pas représentés dans ces schémas, mais l'état des bascules D à l'initialisation sera indiqué.

Si des **codes SystemVerilog** sont écrits, tous les signaux utilisés doivent être correctement déclarés, leur taille (nombre de bit) doit être définie. Les processus synchrones, ou combinatoires doivent être clairement distingués.

N'oubliez pas d'inscrire nom, prénom, et numéro de casier sur votre copie.

Bon courage !

1 Réalisation d'un Digicode

On désire réaliser un digicode. L'objectif de l'exercice est de concevoir un système permettant de détecter l'entrée du bon code (**246A**) sur un clavier et qui, dans ce cas, passse une sortie **`S`** à 1 pendant un cycle d'horloge permettant ainsi d'ouvrir la serrure.

Le système dispose d'une horloge **`clk`** à 10MHz et d'un signal d'initialisation **`reset_n`**, actif à l'état bas. Ces deux signaux seront implicites sur les schémas, donc non représentés. On se contentera d'indiquer la valeur d'initialisation des éventuelles bascules D. Enfin l'appui d'une touche est sensée durer plusieur cycles d'horloge.

Vous disposez d'un clavier tel que décrit en figure 1. Ce clavier dispose en sortie d'un bus **synchrone** sur 4 bits, **`C[3:0]`**, indiquant la touche appuyée :

- Si on appuie sur touche, **`C`** prend comme valeur le numéro de la touche : 0 pour la touche 0, 1 pour la touche 1, . . . , 9 pour la touche 9, 10 (0xA) pour la touche A, 11 (0xB) pour la touche B.
- Si on n'appuie sur aucune touche, ou si on appuie sur deux touches en même temps, **`C`** prend comme valeur 15 (0xF en hexadécimal).

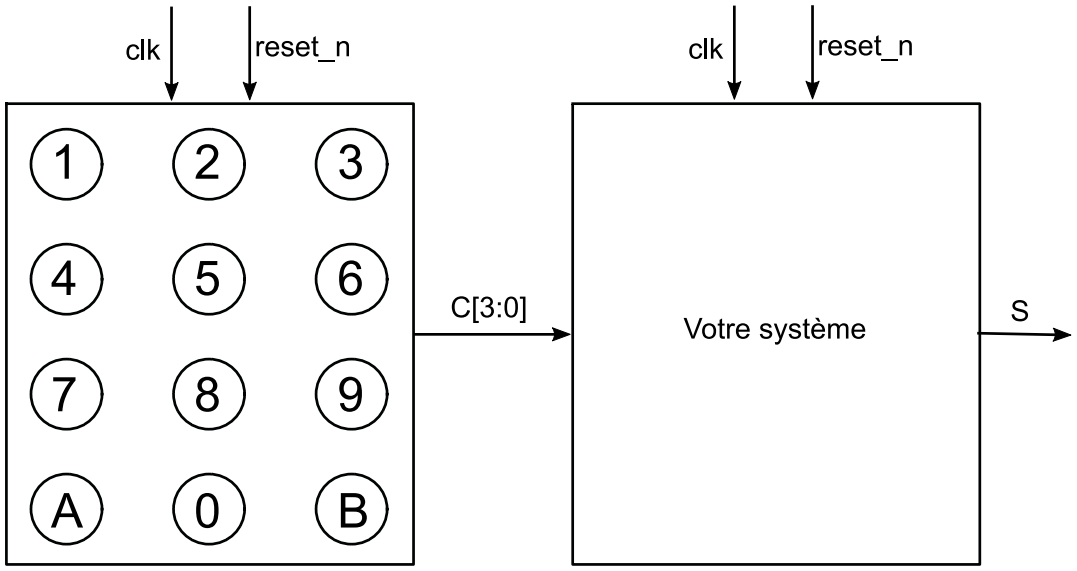

Figure 1 – *Digicode*

Question 1 Détection de l'appui sur une touche

Faites le schéma d'un dispositif qui produit un signal **`enable`** passant à 1 pendant un seul cycle d'horloge lorsqu'une touche est pressée, quelle que soit la durée de l'appui sur la touche.

Question 2 Détection du code

Faites le schéma d'un dispositif qui produit un signal **`OK`** qui passe à 1 uniquement lorsque l'utilisateur a entré le bon code (246A). Ce signal peut rester à 1 aussi longtemps qu'on le souhaite. Si l'utilisateur a entré une séquence de touches ne correspondant pas au bon code, ce signal doit passer à 0.

Question 3 Ouverture de la porte

Faites le schéma d'un dispositif prenant en entrée le signal **`OK`** produit à la question 2 et produisant, lorsque cette entrée passe à 1, un signal de sortie **`S`** qui ne vaudra 1 que pendant un seul cycle d'horloge.

2 Analyse et mise au goût du jour d'un vieux vieux schéma

Votre chef a retrouvé un vieux schéma montré en figure 2. Il désire en réaliser une nouvelle version. Pour cela vous devez utiliser le langage SystemVerilog. Les bascules du schéma sont supposées être mises à **0** au moment de la réinitialisation.

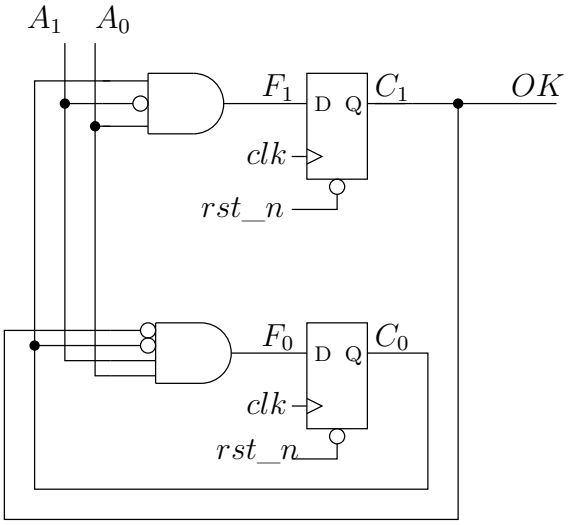

Figure 2 – *Vieux schéma*

Question 1 : Ecrivez (sans astuces...) le code SystemVerilog correspondant au schéma. Le code doit être complet de l'entête de module, à la fin de module.

Question 2 : Montrez que le couple de signaux C_1C_0 ne peut jamais atteindre la valeur **11**.

Vous montrez votre code à votre chef, il n'est pas content, il n'y comprend rien. Il vous propose alors la méthode suivante :

- Considérer les entrées A_1 et A_0 comme le codage binaire sur 2 bits d'un nombre nommé **A**
- Considérer que les signaux C_1 et C_0 comme le codage binaire d'un état **C** dont les valeurs symboliques seront nommées **S0** pour le code **00**, **S1** pour **01**, **S2** pour **10**.

Question 3 : Quand **C** est dans l'état **S0**, pour quelle(s) valeur(s) de **A** peut on changer d'état, et pour aller vers quel(s) état(s) futur(s) ?

Question 4 : Quand **C** est dans l'état **S1**, pour quelle(s) valeur(s) de **A** peut on changer d'état, et pour aller vers quel(s) état(s) futur(s) ?

Question 5 : Quand **C** est dans l'état **S2**, pour quelles(s) valeur(s) de **A** peut on changer d'état, et pour aller vers quel(s) état(s) futur(s) ?

Question 6 : En déduire un graphe d'états correspondant au circuit (sans oublier d'indiquer la valeur du signal **OK** dans chacun des états).

Question 7 : En déduire un code SystemVerilog équivalent au schéma, plus "compréhensible" pour un être humain et ne faisant pas usage d'équations booléennes.

3 Détection des touches d'un digicode

L'objectif de l'exercice est de concevoir un système permettant de détecter quelle touche du clavier d'un digicode est appuyée.

Le système dispose d'une horloge **`clk`** à 10MHz et d'un signal d'initialisation **`reset_n`**, actif à l'état bas.

Vous disposez d'un clavier, dit "à balayage" comme décrit en figure 3. Le clavier dispose de deux sorties, **`data`** et **`scanning`** chacune sur un bit. Le clavier balaye en permanence les lignes et colonnes du clavier suivant la séquence suivante :

- Il balaye les 3 colonnes une par une, de la gauche vers la droite, et passe **`data`** à 1 quand il détecte une touche appuyée sur la colonne courante.
- Puis balaye les 4 lignes une par une en commençant par le haut, et passe **`data`** à 1 quand il détecte une touche appuyée sur la ligne courante.
- Le signal **`scanning`** passe à 1 pendant toute la durée du balayage.
- Entre deux balayages, **`scanning`** repasse à 0 pendant exactement un cycle d'horloge.

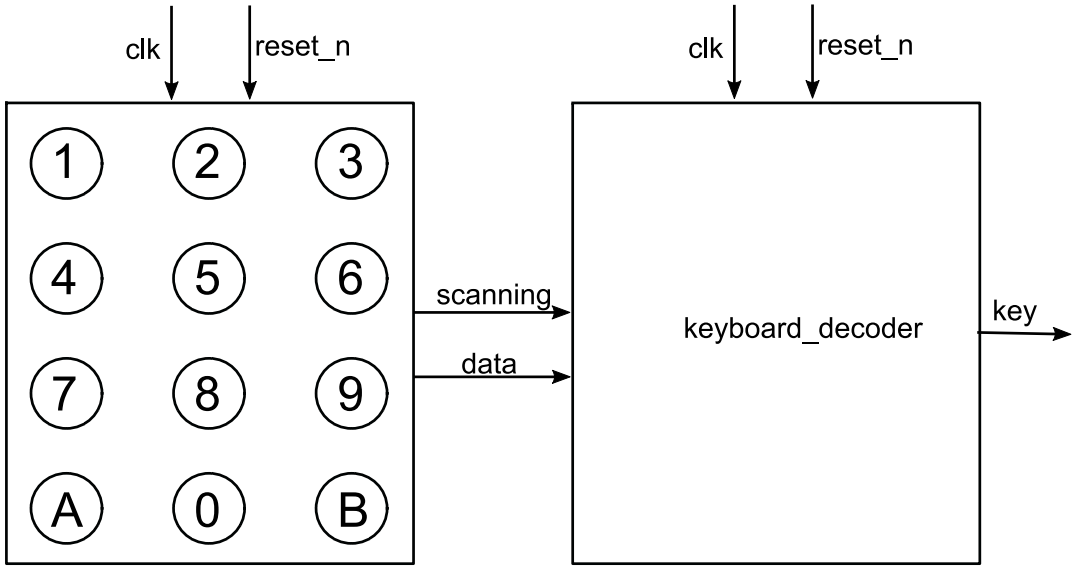

Figure 3 – *Digicode*

Les figures suivantes montrent quelques cas de figure d'utilisation du clavier. Dans ces chronogrammes, la ligne "Étape" indique ce que le clavier est en train de balayer.

- Si on n'appuie sur aucune touche, on obtiendra le chronogramme de la figure 4.
- Si on appuie sur la touche 2 (colonne 1, ligne 0), on obtiendra le chronogramme de la figure 5.
- Si on appuie sur la touche B (colonne 2, ligne 3), on obtiendra le chronogramme de la figure 6.

Question 1 Interprétation des chronogrammes

Examinez le chronogramme de la figure 7. Qu'a fait l'utilisateur du clavier ?

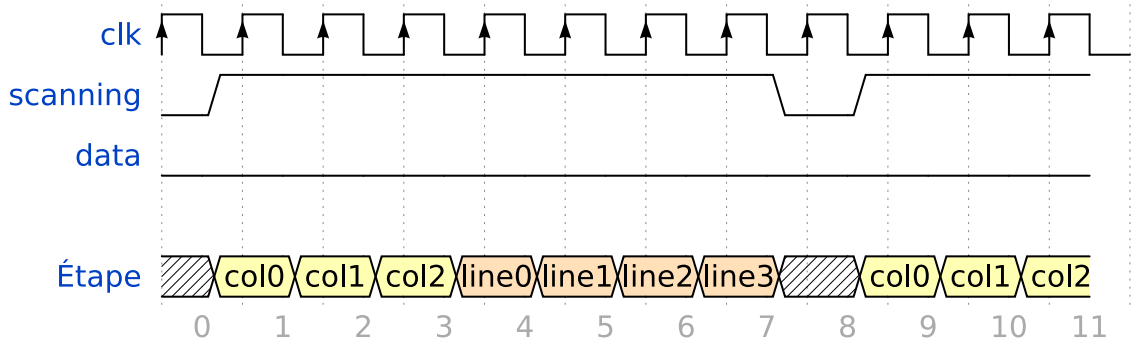

Figure 4 – *Aucune touche*

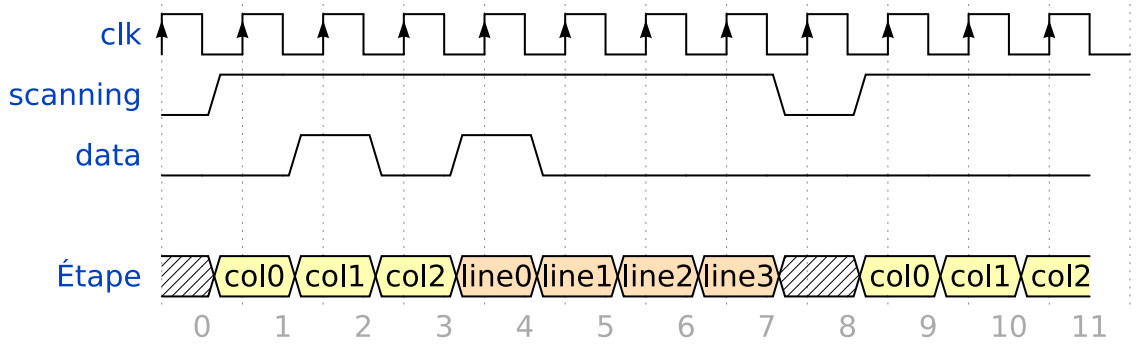

Figure 5 – *Touche 2*

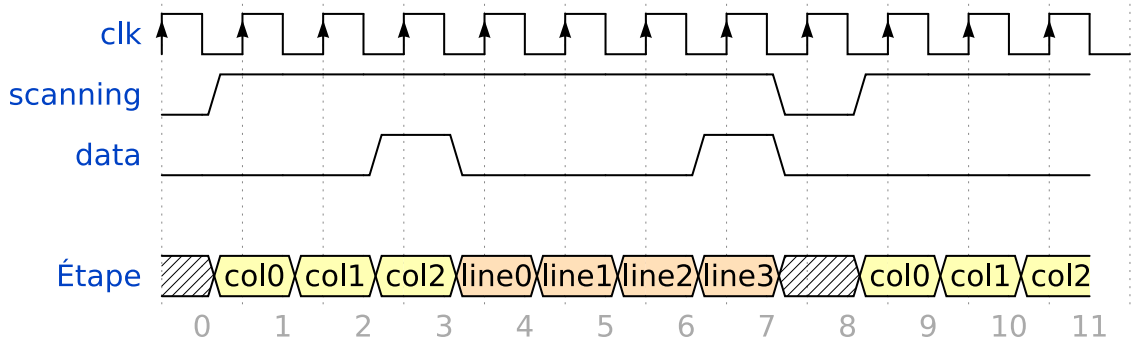

Figure 6 – *Touche B*

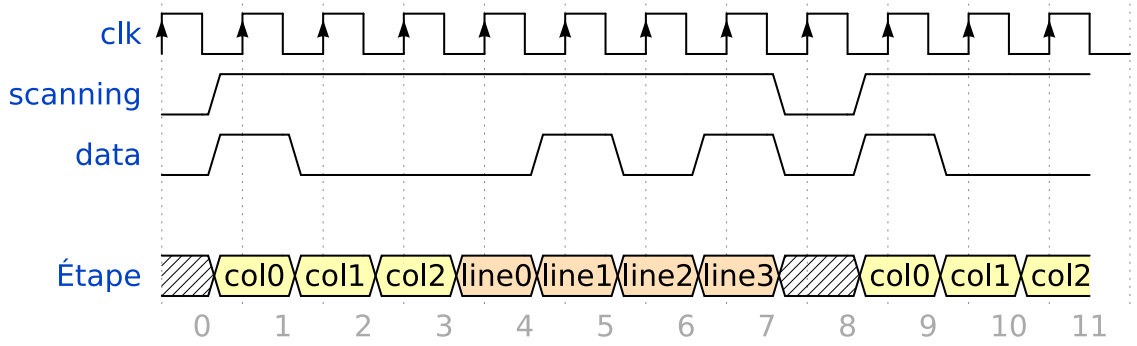

Figure 7 – *Que fait l'utilisateur ?*

L'objectif de la suite est d'écrire le code SystemVerilog d'un système faisant en sorte que ce clavier ait le comportement suivant :

- Quand une touche est pressée, on sort son numéro sur le signal **key**.
- Quand aucun touche n'est pressée, ou quand plusieurs touches sont pressées à la fois, on sort 0xF.

Dans la suite vous allez construire le code bloc par bloc. **Chaque bloc est simple, sans piège, et fait moins de 8 lignes**.
On partira du squelette de module suivant :

```
module keyboard_decoder(input logic clk,
                        input logic reset_n,
                        input logic scanning,
                        input logic data,
                        output logic[3:0] key
                        );

   // Votre code ici !

endmodule
```

Question 2 Génération des étapes
On veut un "compteur d'étape" **cpt** qui vaut 0 par défaut, est incrémenté à la fin de chaque cycle où **scanning** vaut 1, et remis à zéro à la fin d'un cycle où **scanning** vaut 0.

- Jusqu'à combien doit pouvoir compter ce compteur ?
- Écrivez le code SystemVerilog d'un bloc séquentiel qui génère **cpt**.

Question 3 Stockage de la colonne
On veut stocker dans un signal **col** le numéro de la colonne de la touche pressée. Si aucune touche n'est appuyée, ou si plusieurs touches sont appuyées, on y stockera n'importe quoi.

- Sur combien de bits doit être codé **col** ?
- En remarquant que pour générer **col** il suffit de regarder **data** pendant que **cpt** est compris entre 0 et 2, donnez le code SystemVerilog d'un bloc séquentiel qui génère **col**.

Question 4 Stockage de la ligne
On veut stocker dans un signal **line** le numéro de la ligne touche pressée. Si aucune touche n'est appuyée, ou si plusieurs touches sont appuyées, on y stockera n'importe quoi.

- Sur combien de bits doit être codé **line** ?
- En remarquant que pour générer **line** il suffit de regarder **data** pendant que **cpt** est compris entre 3 et 6, donnez le code SystemVerilog d'un bloc séquentiel qui génère **line**.

Question 5 Détection d'un appui valide

Pendant que **scanning** est haut, combien de fois **data** est il a **1** pendant le balayage si :

- l'utilisateur n'appuie sur aucune touche,
- l'utilisateur appuie sur une seule touche,
- l'utilisateur appuie sur plusieurs touches ?

Donnez le code SystemVerilog d'un bloc séquentiel qui génère un **n_data** indiquant le nombre de fois où **data** est passé à 1 pendant que **scanning** était haut.

Question 6 Génération de la sortie key

À partir des signaux générés prédécemment, on propose le code suivant pour générer la sortie **key**.

```
// Lorsque qu'on a finit le scan, on sort le numéro de la touche appuyée.
// Pour cela, il faut que n_data ==...  Sinon, ça veut dire qu'on n'a
// appuyé sur aucune touche ou qu'on a commencé à appuyer pendant que
// clavier scannait ou qu'on a appuyé sur deux touches en même temps.
always @(posedge clk or negedge reset_n)
  if(!reset_n)
    // Au reset, on sort le code "aucune touche" :  0xF.
    key <= 0xF;
  // Lorsque le compteur indique qu'on a finit le scan,
  else if (cpt==8)
    // si on a bien eu l'appui d'une seule touche, on génère son code sur key.
    if (n_data == ...)
         // Pour les touches de 1 à A compris, c'est simple.
         key <= col + (3*line);
         // Pour 0 et B, c'est un cas particulier.
         if ((line == 3) & (col == 1))
             key <= 0;
         if ((line == 3) & (col == 2))
             key <= 0xB;
    else
      // sinon (n_data est invalide), on sort le code 0xF.
      key <= 4'hF;
```

Corrigez ce code.