ELECINF102
Processeurs et Architectures Numériques

Contrôle de connaissances

30 juin 2016

Document autorisé : une feuille recto-verso

Durée: 1h30 minutes

Ce contrôle comporte 3 parties **indépendantes** :

1. Logique CMOS
2. Décodeur vidéo
3. NanoProcesseur et interruptions

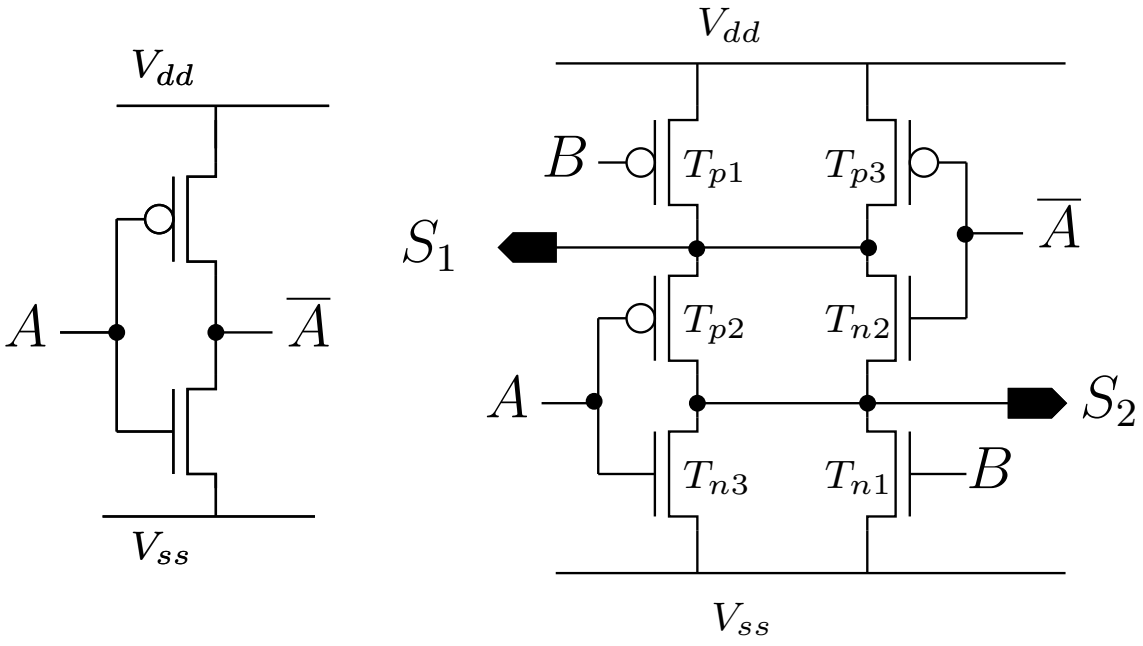
Consignes importantes : Si des **schémas** sont demandés dans les différents exercices, ils doivent être impérativement clairs, lisibles et sans ambiguïté. Les dimensions des bus doivent être indiquées. Si nécessaire le sens des signaux doit être précisé. Pour la logique synchrone, les signaux d'horloge et d'initialisation asynchrone (`reset_n`) ne seront pas représentés dans ces schémas.

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

Bon courage !

1 Logique CMOS

Soit le circuit suivant à base de transistors MOS. Les entrées correspondent aux signaux A et B, les sorties correspondent au signaux S_1 et S_2.



Question 1.1 : Déterminez l'état (passant ou bloqué) des différents transistors de la structure générant S_1 et S_2 en fonction des signaux A et B (faites un tableau avec les signaux d'entrée et les états des transistors).

Question 1.2 : En déduire les équation logiques des fonctions S_1 et S_2.

Question 1.3 : Quel est l'intéret d'une telle structure par rapport à un assemblage de portes simples de type

- INV : inverseur
- NAND2 : porte non-et à 2 entrées
- NOR2 : porte non-ou à 2 entrées

2 Décodeur vidéo de Canal+

Peu avant les années 2000, une célèbre chaîne de télévision cryptée, excédée par le nombre de décodeurs pirates en circulation, décidait de changer radicalement son processus de cryptage. Malheureusement, les FPGA étaient déjà inventés... L'objectif de cet exercice est de réaliser la partie vidéo de ces nouveaux décodeurs pirates.

En télévision les pixels sont transmis les uns après les autres, ligne par ligne. L'algorithme de chiffrement (un peu simplifié) découpait l'image en blocs de 32 lignes. Dans chaque bloc de 32 lignes, les lignes étaient ensuite mélangées selon un ordre déterminé par une clef secrète. Mais la première ligne de chaque bloc était toujours au bon endroit : transmise en premier. L'attaque consistait à faire de la force brute : acquérir les 32 lignes d'un bloc et les ré-ordonner en considérant que deux lignes consécutives ont une corrélation forte, en commençant par la première ligne du bloc.

Chaque ligne est une suite de 722 pixels, chaque pixel représentant une luminosité sur 8 bits non signés (on travaille ici en noir et blanc, la version couleur n'étant pas plus complexe). `Un nouveau pixel arrive au décodeur à chaque cycle d'horloge`. Les lignes sont transmises les unes à la suite des autres sans pause. On considère dans la suite de l'exercice qu'on ne traite qu'un seul bloc de 32 lignes.

Question 2.1 : Créez le schéma d'un système permettant de stocker une ligne de pixels. `On pourra utiliser des "..."` pour éviter d'avoir à tracer des schémas trop larges.

Question 2.2 : Créez le schéma d'un système basé sur le précédent permettant de stocker un bloc de 32 lignes complet.

En appelant $P_i(n)$ le n-ième pixel de la ligne i, la corrélation entre deux lignes i et j est donnée par

$$\sum_{n=0}^{721} |P_i(n) - P_j(n)|$$

Question 2.3 : Créez un schéma, à partir de toutes les portes logiques vues en cours, de l'opérateur "valeur absolue".
Question 2.4 : Créez le schéma d'un système calculant au fur et à mesure de l'arrivée des pixels d'une ligne la corrélation entre cette ligne et une ligne arbitraire déjà stockée. Précisez le nombre de bits sur lequel doit être stockée cette corrélation.

On considère maintenant que toutes les lignes sont arrivées, stockées dans le système de la question2, et que pour chacune d'entre elle, on dispose de sa corrélation avec les 31 autres grâce au système de la question 4.

Question 2.5 : Créez le schéma d'un système séquentiel prenant en entrée 31 nombres non signés et produisant au bout de 31 cycles d'horloge le plus grand d'entre eux.

Question 2.6 : Modifiez votre schéma précédent pour qu'il prenne en entrée un ensemble de 31 valeurs de corrélation (appelées C_0 à C_{30}) et qu'il produise, toujours au bout de 31 cycles, le numéro de la corrélation la plus grande.

Question 2.7 BONUS : Sachant que la première ligne est à la bonne place, expliquez en mots simples comment vous envisagez la suite du système pour sortir les 31 autres lignes de façon ré-ordonnée.

3 NanoProcesseur et interruptions

Les microprocesseurs courants disposent de la possibilité de réagir à des "interruptions" en provenance du monde extérieur. Il s'agit en général :

- de détecter une demande d'interruption via un signal spécifique (dit signal d'interruption)
- de **quitter le programme en cours** pour exécuter un code spécifique (dit code d'interruption)
- de conclure l'exécution de ce code spécifique en **retournant dans le programme en cours**.

Nous désirons modifier le **NanoProcesseur** pour prendre en charge des interruptions.

Nous rappelons que le microprocesseur est piloté par un automate en 3 cycles nommés `IF`, `AF` et `EX` pour "Instruction Fetch", "Address Fetch" et "Execute".

3.1 Détection de l'interruption

- Nous ajoutons un signal entrant nommé `IRQ` : pour "Interrupt Request".
- Si ce signal change d'état alors une interruption est demandée.

Nous désirons générer un nouveau signal `IRQ_event` dont les caractéristiques sont les suivantes :

- Au repos, le signal `IRQ_event` est égal à `1'b0` ;
- A chaque changement d'état du signal `IRQ` le signal `IRQ_event` passe à l'état `1'b1` pendant exactement 3 cycles de l'horloge du système puis repasse à 1'b0.

Question 3.1 Définissez un code SystemVerilog ou un shéma d'une structure générant le signal `IRQ_event`.

Question 3.2 Quelles sont les raisons, d'après vous, pour lesquelles :

- d'un part nous transformons le signal `IRQ` en une impulsion.
- d'autre part la durée de l'impulsion est de 3 cycles.

3.2 Traitement de l'interruption

- Nous réservons l'adresse **240** de la mémoire pour stocker le début du code d'interruption.
- Si une interruption est détectée, alors le NanoProcesseur doit :
  - terminer l'exécution de l'instruction en cours ;
  - sauvegarder l'adresse de la prochaine instruction prévue dans le programme en cours ;
  - sauter de manière inconditionnelle à l'adresse `240`.
- Nous ajoutons une instruction `RTI` (ReTurn from Interrupt) à la liste des intructions.
- L'exécution de l'instruction `RTI` doit forcer le microprocesseur à reprendre le programme en cours.

Vous trouverez en fin de sujet, un schéma du NanoProcesseur ainsi que les codes du compteur de programme et du contrôleur de base.

Question 3.3 : Pendant quel cycle de l'automate du NanoProcesseur la gestion des interruptions doit elle être prise en compte (justifiez).

Question 3.4 : Proposez une modification des codes et/ou une modification du schéma permettant d'implémenter la gestion des interruptions.

Question 3.5 : L'adresse du code d'interruption est fixe. Quelles nouvelles évolutions faudrait-il mettre en œuvre pour que cette adresse puisse être définie par le programme en cours d'exécution (nous attendons une simple explication des principes).

Code du PC "de base" :

```
...
always @(posedge clk or negedge reset_n)
  if(!reset_n)
    PC <= 0 ;
  else
    if(Load_PC)
       PC <= Q ;
    else
       PC <= PC + Inc_PC ;
...
```

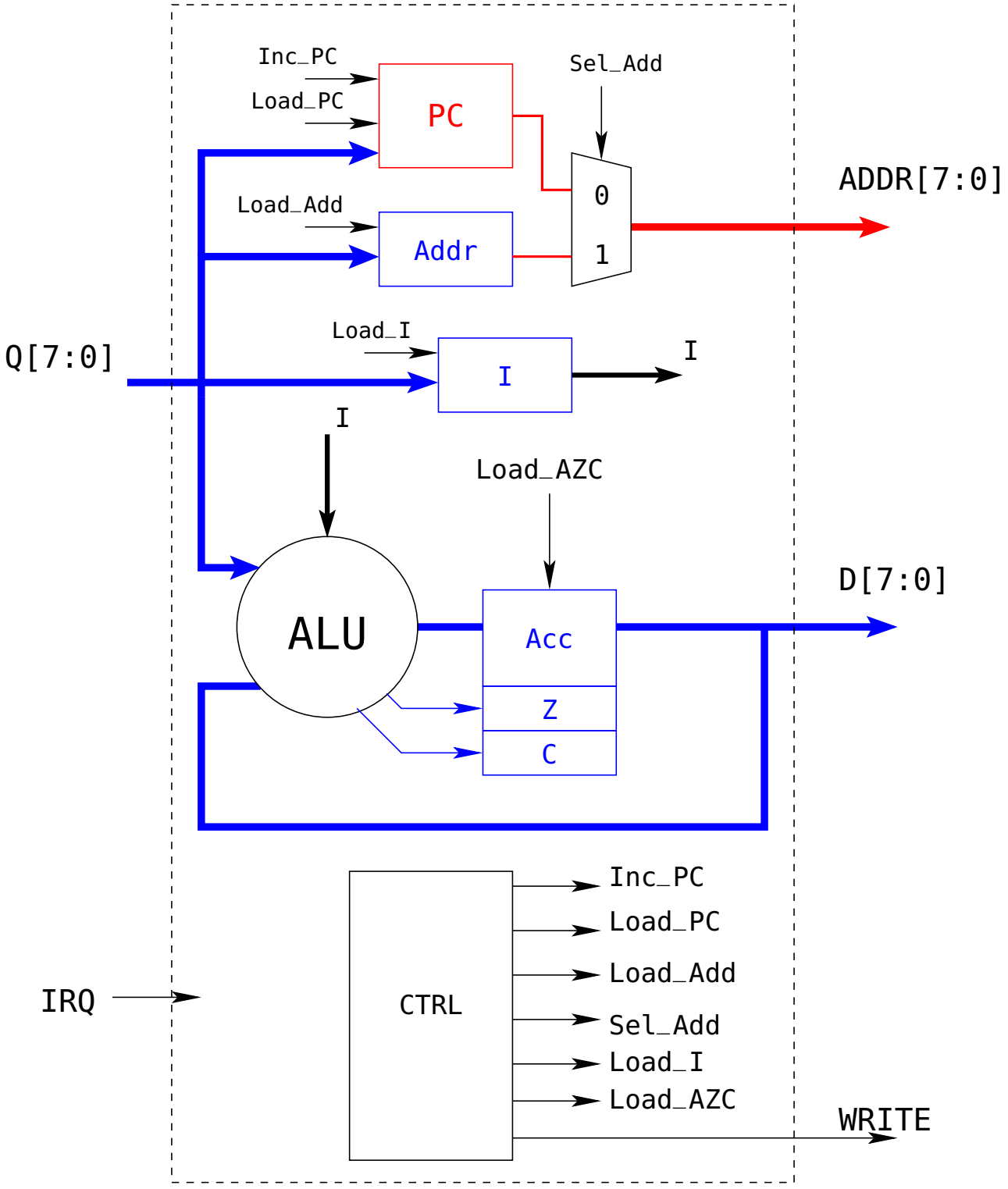
Code du Contrôleur "de base" : Seuls les codes utiles pour les questions posées sont indiqués.

```
...
always @(*)
begin
   Inc_PC    <= (Etat == IF) || (Etat == AF) ;
   Load_PC   <= (Etat == AF) && ((I == JMP || (I==JNC && !C) || (I==JNZ && !Z)) ;
   Load_Add  <= (Etat == AF) ;
   Sel_Add   <= (Etat == EX) ;
   Load_I    <= (Etat == IF) ;
   Load_AZC  <= ...
   WRITE     <= (Etat == EX) && (I ==  STA) ;
end
...
```

Schéma du NanoProcesseur :

Vous pouvez, si vous le voulez, inclure ce schéma "modifié par vos soins" dans votre copie. N'oubliez pas d'indiquer vos noms et prénoms sur le shéma.

- **Nom :**
- **Prénom :**
- **Casier :**



4 QCM

1. Quelle fonction est implémentée par ce module System Verilog :

```
module func(input logic signed [31:0] x,
            output logic signed [32:0] y);

always @(*)
  y <= 33'h0FFFFFFFF & ((-(!x[31]) & x) | (-x[31] & (~x+1'b1)));

endmodule
```

   (a) $y = x$
   (b) $y = 2x$
   (c) $y = x^2$
   (d) $y = |x|$
   (e) Ce code ne compile pas.

2. Lequel des chronogrammes (Fig. 1) décrit un comportement correspondant au code suivant ?

```
module test(input  logic clk,
            input  logic sw,
            output logic led);

logic val;

always @(posedge clk) begin
       if(sw ) val <= 1'b1;
       if(val) led <= 1'b1;
end

endmodule
```

   (a) (i)
   (b) (ii)
   (c) (iii)
   (d) (iv)
   (e) Ce code ne compile pas.

3. Considerez le module suivant. Si la valeur de -32 est présentée à l'entrée de ce module (i.e. $x = -32$), quelle sera la valeur de la sortie ?

```
module func(input logic signed [31:0] x,
            output logic signed [31:0] y);

always @(*)
  y <= x/2 - (x >> 1);
endmodule
```

   (a) 0
   (b) 1
   (c) 2^{31}
   (d) -2^{31}

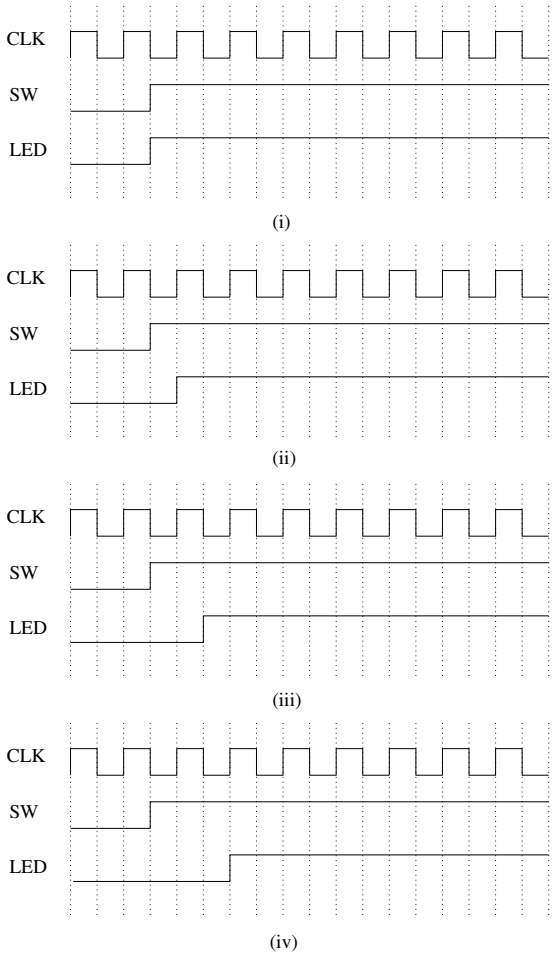


Figure 1 – *Figure pour problème 2.*

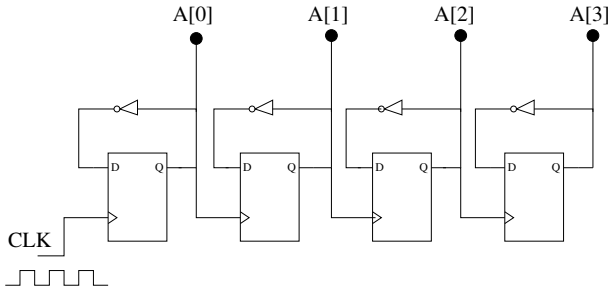


Figure 2 – *Figure pour problème 4.*

(e) Ce code ne compile pas.

4. Dans le circuit de la figure 2, quelle est la valeur de la sortie A[3 :0] apres 12 cycles de CLK. Supposons que tous les bascules D sont idéales.

   (a) 3

   (b) 5

   (c) 7

   (d) 10

(e) 12

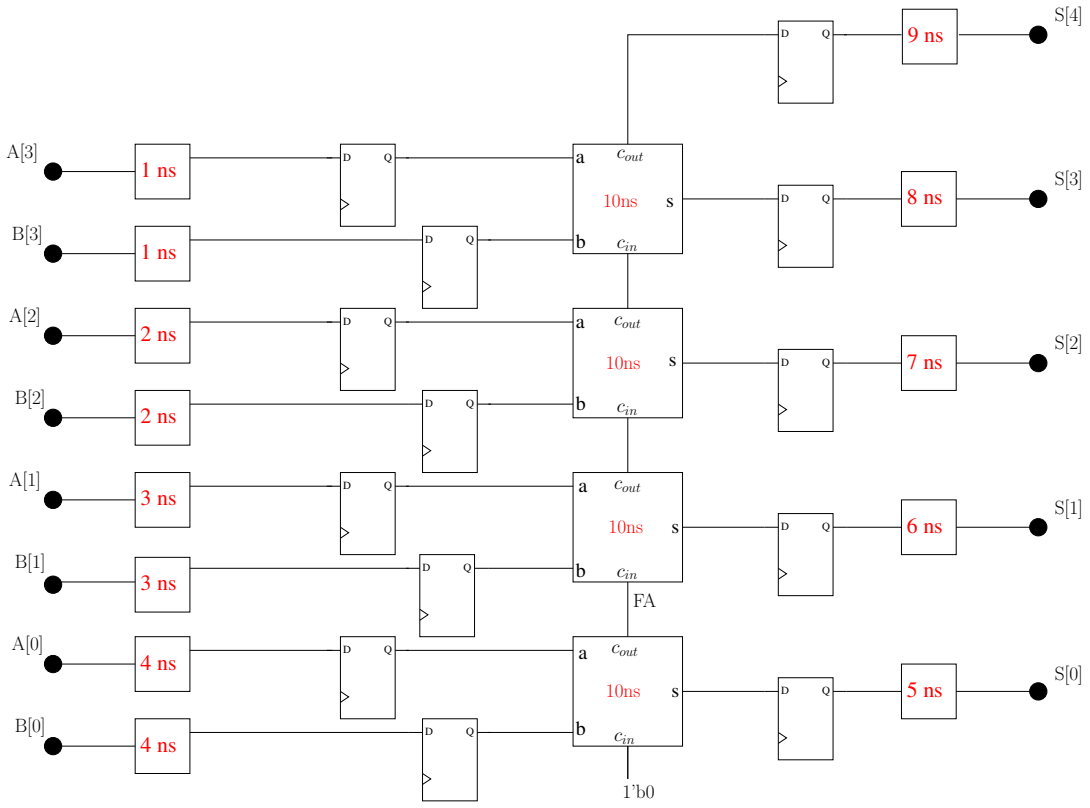


Figure 3 – *Figure pour problème 4.*

5. Considerez le circuit de la figure 3. C'est un additioneur synchrone 4 bit à propagation de retenu. Les entrées A et B sont présentées à une cadence de $f_{max} = 25$ MHz. Les délais des blocs et des connections sont marqués en rouge. La fonction de transfert de ce circuit est

$$S(n+2) = A(n) + B(n) \tag{1}$$

ou n est le temps discret, en unité de cycle d'horloge. La fréquence maximale d'opération de ce circuit est 25 MHz, à cause du chemin critique de l'additioneur. Nous voulons pipeliner cet additionneur pour qu'il fonctionne à 50 MHz. Combien de bascules D (minimum) devons-nous ajouter dans ce circuit pour avoir un débit d'au moins 50 MHz et un fonctionnement correct à part la latence introduit par le pipeline ? C'est-à-dire

$$S(n+2+L) = A(n) + B(n) \tag{2}$$

où L est la latence, en unité de cycle d'horloge.

(a) 2
(b) 4
(c) 7
(d) 8
(e) 16

6. Dans le circuit de la figure 4, les bascules D sont initialisées avec une valeur x aléatoire, i.e.

$$A(n=0) = x$$

Quelle est la valeur après 150 cycles d'horloge ?

$$A(n = 150) = ??$$

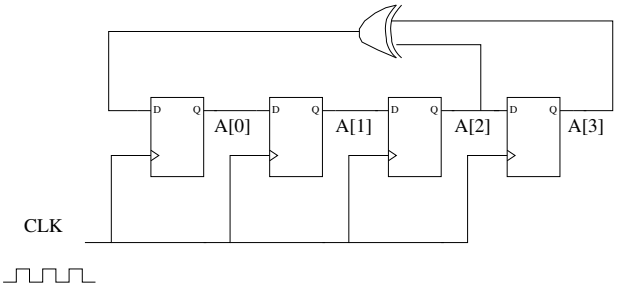


Figure 4 – *Figure pour problème 5.*

(a) x
(b) $x + 1$
(c) $x + 2$
(d) $x + 3$
(e) $x + 4$