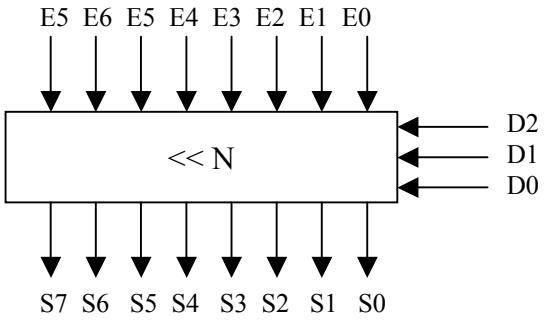
MI41 – FINAL

Documents autorisés : poly de cours et TD
Durée 2h

1. Circuit de Décalage

1.1. Décaleur simple

On souhaite réaliser un circuit dit décaleur 8 bits. Le principe consiste à décaler la valeur d'un mot de 8 bits de N rangs vers la gauche ou vers la droite.
N est compris entre 0 et 7. On se limitera au décalage à gauche (poids faible vers poids fort). Dans ce type de circuit, lors des décalages à gauche, les bits de poids faibles prennent pour nouvelle valeur 0.
Exemple :



Décaleur N bits

Les entrées E sont les entrées de données, les entrées D correspondent au nombre de rangs de décalage (N). S correspond aux sorties décalées.
Ainsi si D = 011 (N = 3) S est égal à E décalé de 3 rangs vers la gauche :

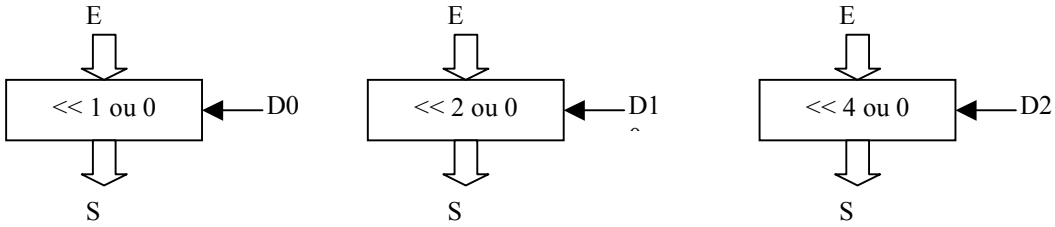
S7 = E4 ; S6 = E3 ; S5 = E2 ; S4 = E1 ; S3 = E0 ; S2 = 0 ; S1 = 0 ; S0 = 0

1.1.1. Description VHDL

1. Donnez la description comportementale (pas d'équations logiques) VHDL de ce circuit décaleur N bits (entity/architecture)

1.1.2. Réalisation à l'aide de portes logiques

Le principe du décalage est basé sur 3 circuits de décalage. Chacun de ces circuits de décalage permet d'effectuer, en fonction d'une entrée de commande D, les décalages suivants :
Décalage à gauche de 0 bit si D0 = 0 ou décalage à gauche de 1 bit si D0 = 1.
Décalage à gauche de 0 bit si D1 = 0 ou décalage à gauche de 2 bits si D1 = 1.
Décalage à gauche de 0 bit si D2 = 0 ou décalage à gauche de 4 bits si D2 = 1.



Décaleurs 1, 2 et 4 bits

Exemple du décaleur 1 bit

Si D0 = 0, [S7 S6 S5 S4 S3 S2 S1 S0] = [E7 E6 E5 E4 E3 E2 E1 E0]
Si D0 = 1, [S7 S6 S5 S4 S3 S2 S1 S0] = [E6 E5 E4 E3 E2 E1 E0 0]

1. Donner les équations logiques correspondant à la sortie S7 du décaleur 2 bits

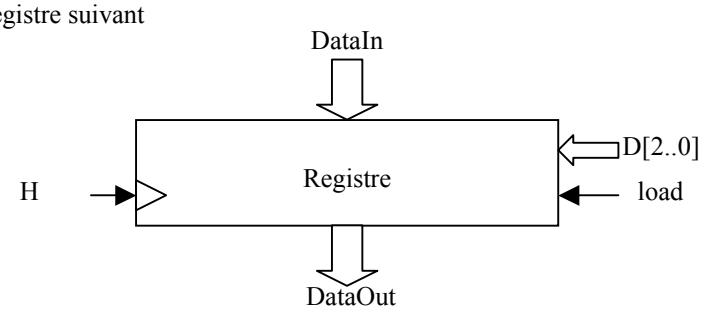
2. Donner le logigramme correspondant à la sortie S7 du décaleur 2 bits (on n'utilisera que des portes logiques à 2 entrées)

3. Donner la structure du décaleur N bits en utilisant les blocs fonctionnels, donnés ci-dessus, des décaleurs 1, 2 et 4 bits.

4. Si les portes logiques utilisées ont un temps de traversée de 1ns, quel est le temps de traversé du décaleur N bits ?

1.2. Registre décaleur

On souhaite réaliser le registre suivant



Avec :

- ***DataOut*** correspond à la valeur (un octet) mémorisée dans le registre
- Lorsque ***load*** est actif (i.e. lorsque ***load*** = 1) la donnée, sur un octet, présente sur l'entrée ***DataIn*** est chargée dans le registre sur un front montant de l'horloge ***H***.
- Lorsque la valeur présente sur l'entrée ***D[2..0]*** est différente de "000" et que ***load*** est inactif la donnée dans le registre est décalée sur un front montant de l'horloge ***H*** de N rang vers la gauche, où N correspond au nombre binaire ***D[2..0]***.

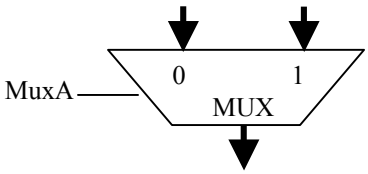
1.2.1. Description VHDL

On souhaite, pour réaliser le registre décaleur, réutiliser le circuit décaleur simple. On suppose que ce dernier est disponible sous forme de *component* et peut donc être utilisé dans cette nouvelle description à l'aide de l'instruction ***port map***.

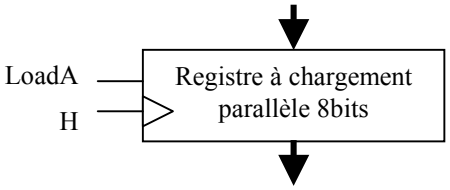
1. donner la description comportementale en VHDL de ce registre décaleur (entity/architecture)

1.2.2. Réalisation câblée

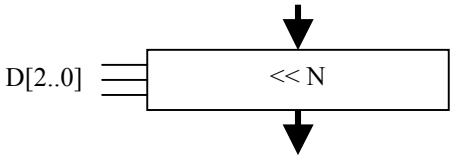
On dispose des composants suivant :



Multiplexeur 2 voies vers 1, les bus sont sur 8 bits



Registre à chargement parallèle synchrone : ***LoadA*** lorsqu'il est actif autorise le chargement du registre sur front montant d'horloge H. Dans le cas contraire les données contenues dans le registre A sont inchangées



Décaleur 8 bits

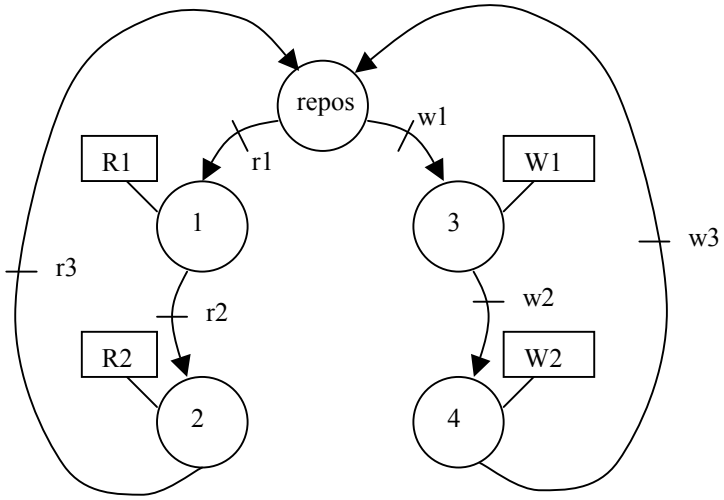
1. Donnez, en utilisant les composants ci dessus et en commentant votre réponse, la structure du registre décaleur sous forme de schéma.

2. Quel est l'intérêt d'une telle structure par rapport à un simple registre à décalage ?

3. Si votre registre était implanté en « dur » dans un microprocesseur. Quelle fréquence d'horloge maximale autoriserait-il ? (En considérant toujours des portes logiques à 2 entrées ayant un temps de traversée égal à 1 ns).

2. Séquenceur de commande

Un processus de lecture écriture d'une interface périphérique (échange de données) est donné par le graphe ci-dessous :



On souhaite réaliser un séquenceur de commande câblé à l'aide de bascules D.

1. Expliquer comment réaliser ce séquenceur câblé en vous appuyant sur un schéma.

2. Donner les équations des entrées *Di* des bascules du séquenceur câblé (on ne simplifiera pas les équations)

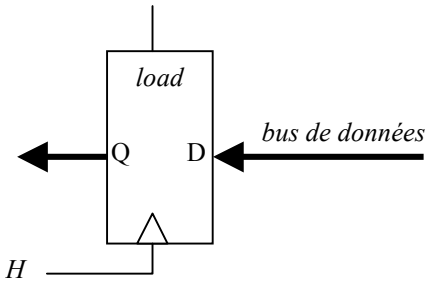
On souhaite que le passage de l'état "repos" à l'état "1" s'effectue avec la condition suivante :

- *nStrobe* = 0
- *nRead* = 0

et que le passage de l'état "repos" à l'état "3" s'effectue avec la condition suivante :

- *nStrobe* = 0
- *nRead* = 1

Par ailleurs, lors de la transition entre l'état repos et l'état 1 la valeur présente sur un bus de donnée doit être mémorisée dans un registre à chargement parallèle synchrone :



load : commande synchrone de chargement
H : horloge du séquenceur

3. Donner l'équation de la condition d'évolution *r1*

4. Donner l'équation de *load*

MI41 - Final

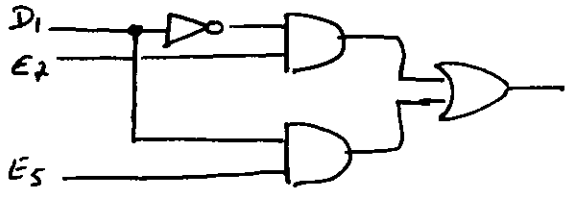
Description VHDL

```
library ieee;
use ieee.std_logic_1164.all

entity décaleur is
port (
        E : in std_logic_vector (7 downto 0);
        D : in std_logic_vector (2 downto 0);
        S : out std_logic_vector (7 downto 0)
);
end décaleur

Architecture a of decaleur is
Begin
    with D select

        S <= E                          when "000",
             E(7 downto 1) & '0'        when "001",
             E(7 downto 2) & "00"       when "010",
             E(7 downto 3) & "000"      when "011",
             E(7 downto 4) & "0000"     when "100",
             E(7 downto 5) & "00000"    when "101",
             E(7 downto 6) & "000000"   when "110",
             E(7) & "0000000"           when others;

    end a;
```

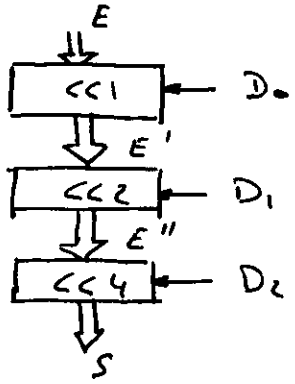
Logique câblée

1. $S_7 = E_7 \overline{D_1} + E_5 D_1$

2.



3.



4. Les logigrammes sont identiques pour les 3 décalem.

Le temps de traverse est déterminé pour le cas le plus défavorable à savoir 3 portes logiques par décalem soit au total.

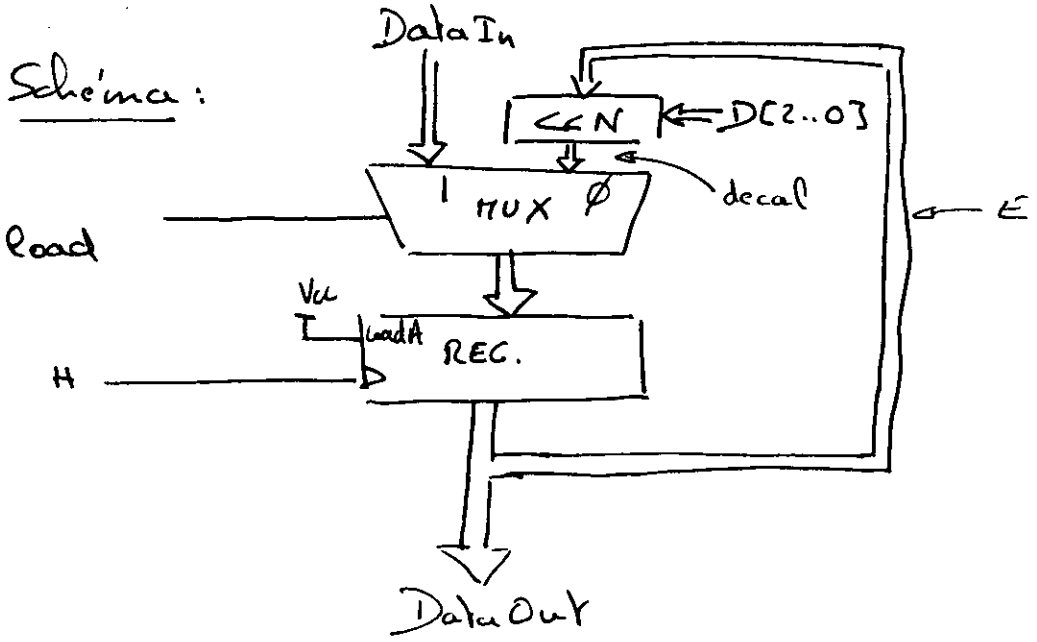
$$3 \times 3 \times 1ns = 9ns$$

1.2 - Registre décalem

Description VHDL

```
library ieee;
use ieee.std_logic_1164.all
library work;
use work.decalem.all;

entity reg_dec is
port(
      H, load : in std_logic
      DataIn  : in std_logic_vector(7 downto 0);
      D       : in std_logic_vector(2 downto 0);
      DataOut : out std_logic_vector(7 downto 0);
    );
architecture a of reg_dec is

  signal E, decal : std_logic_vector(7 downto 0);
begin

  U0 : decalem port map (E, D, decal);

  process (H)
  variable mem : std_logic_vector(7 downto 0);
  begin
    If (H'event and H='1') then
      If load='1' then
        mem := DataIn;
      else
        mem := decal;
      end if;
    end if;
    DataOut <= mem;
    E       <= mem;
  end process;
```

Schéma :

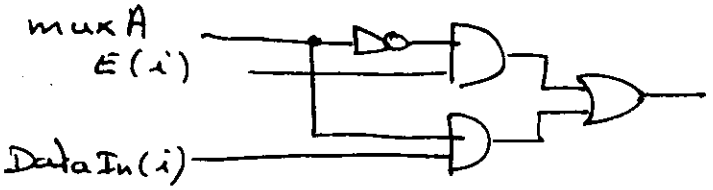


2- Intérêt : décalage de N bit en 1 période d'horloge au lieu d'un seul.

3- fréquence d'horloge maximale :

dépend du nombre de portes traversée dans le cas le plus défavorable

Structure d'un MUX :



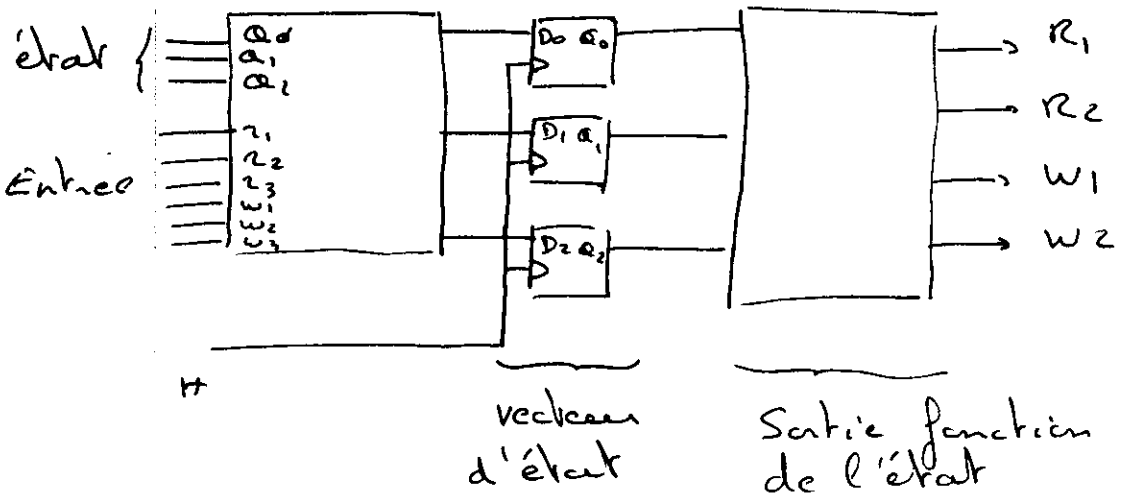
=> 3 portes logiques

finalement on obtient dans le cas le + défavorable

9 ns pour le décaleur + 3ns pour le MUX

soit donc $\boxed{12\,ns}$ donc $F_{max} = \frac{1}{12 \cdot 10^{-9}} = 83\,MHz$

2- Séquenceur



Le vecteur d'état (registre) permet de mémoriser l'état du système et d'après cet état :

→ calcul de l'état suivant fonction des conditions d'évolution

→ mise à jour des sorties.

$$D_0 = X_0 r_1 + X_1 \bar{r}_2 + X_0 w_1 + X_3 \bar{w}_2 = X_0(r_1 + w_1) + X_1 \bar{r}_2 + X_3 \bar{w}_2$$

$$D_1 = X_1 r_2 + X_2 \bar{r}_3 + X_0 w_1 + X_3 \bar{w}_2$$

$$D_2 = X_3 w_2 + X_4 \bar{w}_3$$

où

$$\begin{cases} X_0 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 \\ X_1 = Q_2 \bar{Q}_1 Q_0 \\ X_2 = \bar{Q}_2 Q_1 \bar{Q}_0 \\ X_3 = Q_2 Q_1 Q_0 \\ X_4 = Q_2 \bar{Q}_1 \bar{Q}_0 \end{cases}$$

3- $r_1 = \overline{nStrobe} \cdot \overline{nRead}$

4. $load = r_1 \cdot X_0$

→ chargement si $r_1 = 1$ et que l'on est en X_0

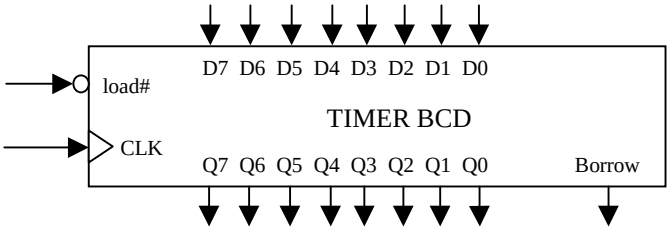
⇒ passage en X_1 et chargement sur le front.

MI41 – Examen final - 2h

Les documents sont autorisés excepté livres et photocopies de livres. Le prêt de documents entre étudiants n'est pas autorisé.

1. Description VHDL (7pts)

Donnez la description (entity et architecture) du timer BCD (décimal codé binaire) décrit ci-dessous.
Descriptif fonctionnel :

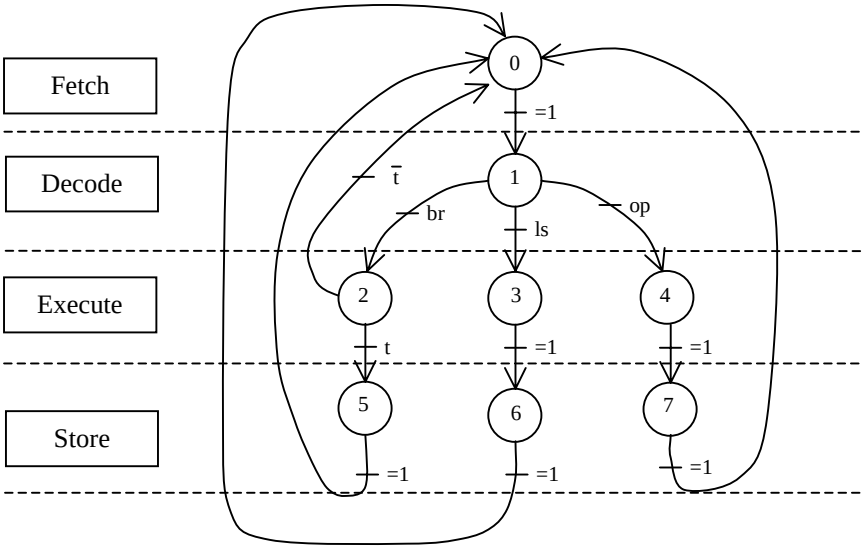


Ce timer BCD est un décompteur BCD qui ne décompte qu'une seule fois et qui génère un signal en fin de décomptage :

- la commande load# est une commande synchrone active niveau bas
- lorsque load# est actif la valeur présente sur les entrées D est chargée
- lorsque load# est inactif, la valeur courante est décrémentée d'une unité BCD
- lorsque le décompteur atteind la valeur Zéro la sortie borrow est activée. Cette sortie reste active tant qu'une nouvelle valeur n'est pas chargée.

2. Séquenceur de machine RISC (6 pts)

Il s'agit dans cet exercice de réaliser un séquenceur câblé simplifié pour une machine de type RISC



L'exécution des instructions est décomposée en 4 étapes qui sont pour une opération entière : la recherche en mémoire de l'instruction, le décodage de l'instruction, l'éxécution et le rangement des résultats.
Les instructions sont de 3 types :

- **op** : opération entière (registre à registre),
- **ls** : opération load/store,
- **br** : branchement conditionnel.

Description des différentes étapes

N° place	description de l'opération	Opération réalisée en sortie de place (franchissement transition)
0	recherche instruction	mémorisation dans registre d'instruction
1	décodage instruction	mémorisation dans registre de micro-commandes
2	calcul des conditions de branchement	résultat test (registre d'état) → t = vrai ou faux
3	calcul adresse load/store	mémorisation dans registre d'adresse
4	exécution du calcul (opération UAL)	mémorisation dans registre de résultat
5	calcul adresse du saut	mémorisation dans compteur de programme
6	donnée placée sur le bus (store) attente donnée présente (load)	lecture ou écriture effective
7	déplacement résultat dans registre de destination	résultat opération sauvegardé dans registre destination

Questions : Il s'agit de réaliser le séquenceur câblé de ce microprocesseur qui implémente le graphe à états donnés précédemment, à l'aide de bascules D actives sur front montant.

1. Expliquez le principe d'un séquenceur câblé et donnez la structure du séquenceur demandé

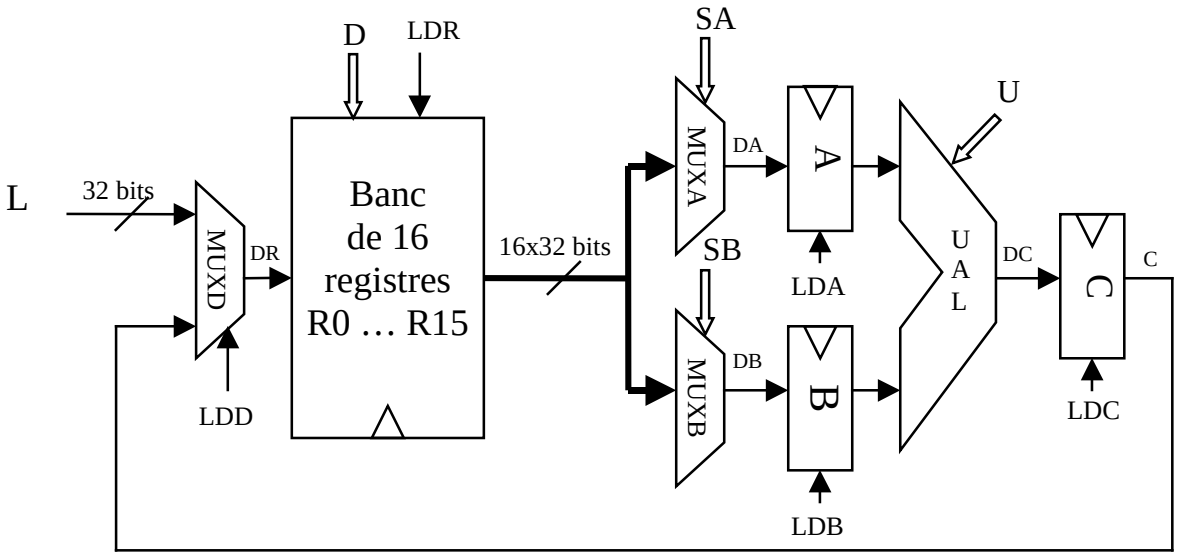
2. Donnez les équations d'entrée de chaque bascule en expliquant votre démarche

Notes importantes :

- on gardera le codage des états proposés
- on ne demande pas de dessiner le logigramme complet

3. Etude d'une unité d'exécution entière (7 pts)

On considère l'unité d'exécution entière d'un processeur RISC suivante :



3.1. Description

Les données traitées sont toutes de longueur 32 bits
A,B,C et R0 à R15 sont des registres à commande de chargement synchrone
Les multiplexeurs et l'UAL sont purement combinatoires (uniquement des portes logiques, pas de bascules).

Fonctionnement des registres

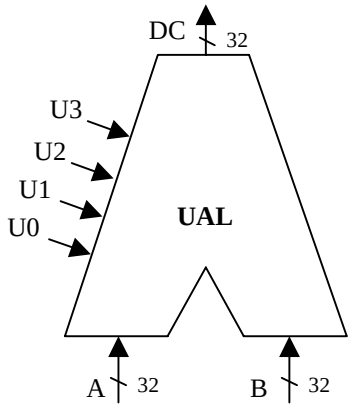
- Tous les registres ont un fonctionnement synchrone. Pour que la valeur présente en entrée d'un registre soit chargé, il faut que la commande de chargement (LDA, LDB, LDC, LDR) du registre concerné soit active (égale à '1') et qu'il y ait un front montant d'horloge.
- Le banc de registres comporte 16 registres indépendants. D est une commande de sélection de registre sur 4 bits (D3 D2 D1 D0) permettant de désigner le registre qui va être chargé. Ainsi pour sauvegarder une valeur dans un des registres il faut qu'au moment du front, LDR soit actif et que le numéro du registre soit spécifié sur D. Exemple si D = "0110" et que LDR est actif, au moment du front, le registre 6 est chargé avec la valeur présente sur DR.

Fonctionnement des multiplexeurs :

- Le multiplexeur MUXD est un multiplexeur 2 voies de 32 bits vers 1. Il permet d'aiguiller une donnée provenant du bus de données (L) ou de la sortie du registre C selon la valeur de LDD vers les entrées des 16 registres du banc de
  LDD = 0 aiguillage de L (i.e. DR = L)
  LDD = 1 aiguillage de C (i.e. DR = C)
- Les multiplexeurs MUXA et MUXB sont des multiplexeurs 16 voies de 32 bits vers 1. Ils permettent respectivement d'aiguiller la sortie d'un des 16 registres du banc de registres vers les entrées des registres A et B.
  SA (resp. SB) est une commande 4 bits SA3 SA2 SA1 SA0 (resp SB3 SB2 SB1 SB0) permettant d'aiguiller la sortie du registre correspondant au numéro de sélection. Ainsi si SA = "0010" (donc 2) c'est la valeur de R2 qui se trouve présente en sortie du multiplexeur.

Fonctionnement de l'UAL

L'UAL utilisée permet en fonction du mot de commande U d'effectuer les opérations arithmétiques et logiques ci-contre :



U	Opération DC
0 0 0 0	A
0 0 0 1	B
0 0 1 0	non A
0 0 1 1	non B
0 1 0 0	A plus B
0 1 0 1	non (A plus B) plus 1
0 1 1 0	A moins B
0 1 1 1	non (A) plus 1
1 0 0 0	B moins A
1 0 0 1	non (A) plus 1
1 0 1 0	A+B
1 0 1 1	non (A+B)
1 1 0 0	A · B
1 1 0 1	non (A.B)
1 1 1 0	A ⊕ B
1 1 1 1	non (A ⊕ B)

Fonctionnement du système

Le système est commandé par un séquenceur câblé fournissant après chaque front d'horloge une commande constituée de 21 bits.

LDD | LDR LDA LDB LDC | D3 D2 D1 D0 | SA3 SA2 SA1 SA0 | SB3 SB2 SB1 SB0 | U3 U2 U1 U0

Exemple de commande :

0 9 5 A 7 B h (0 1001 0101 1010 0111 1011 b) :
- en sortie du MUXD DR = L
- commande de chargement du registre R5 active
- commande de chargement du registre C active
- en sortie du MUXA on a DA = R10
- en sortie du MUXB on a DB = R7
- opération non (A + B) sélectionnée au niveau de l'UAL

Au prochain front d'horloge on aura :
- la valeur présente sur L qui sera chargée dans R5
- la valeur en sortie de l'UAL (soit l'opération logique non(A + B)) qui sera chargée dans le registre C
- la présence d'une nouvelle commande envoyée par le séquenceur pour réaliser une autre opération

3.2. Questions

1. On souhaite transférer la valeur d'un registre vers un autre registre. Avec l'architecture proposée, la valeur transférée doit nécessairement transiter par l'UAL.

1.1. Expliquez les opérations nécessaires pour faire transiter une valeur du registre R3 au registre R7 (copie de R3 dans R7).

1.2. Combien de périodes d'horloge sont nécessaires pour réaliser cette opération ?

2. Donnez la séquence des commandes (en héxadécimal) que doit envoyer le séquenceur pour réaliser l'opération donnée ci-après. Vous détaillerez chaque opération.

ADD R2, R1, R0 : addition du contenu des registres R0 et R1 et sauvegarde du résultat dans R2 (R2 ← R0 plus R1).

3. On considère le programme suivant :

```
ADD R4, R1, R0
ADD R6, R2, R3
ADD R9, R7, R8
ADD R5, R4, R2
```

Donnez la séquence minimale des commandes pour réaliser ce programme. On utilisera au maximum les possibilités de recouvrement des opérations ("pipelining").

4. Si les opérations sont exécutées en recouvrement on parle de "pipeline". De combien d'étages est constitué ce pipeline ?

5. On considère le programme suivant :

```
1  load R0, L        chargement d'une donnée fournie par le bus de données (L) dans le registre R0
2  load R1, L        chargement d'une donnée fournie par le bus de données (L) dans le registre R1
3  ADD R4, R1, R0
4  OR R5, R4, R2     OU logique bit à bit de R2 et R4, le résultat est sauvegardé dans R5 (R5 ← R4 OU R2)
5  ADD R6, R2, R3
6  ADD R7, R4, R2
7  OR R11, R3, R1
```

5.1. Donnez, en justifiant votre réponse, le nombre de cycles nécessaires pour exécuter ce programme sachant qu'il utilise au maximum les possibilités de recouvrement (pipeline).

5.2. Proposez une solution (logicielle) pour réduire la durée de traitement de ce programme.

Examen final MI 41 – Automne 1999

1. Séquenceur programmé (10 pts.)

On souhaite réaliser un séquenceur d'un automatisme séquentiel à l'aide d'une mémoire de 16 octets (ROM 16x8). Le graphe de description de l'automatisme et la structure du séquenceur sont fournis en annexes.

Description :

- Lorsqu'une adresse est fournie en entrée de la mémoire, le contenu de la cellule mémoire (8 bits) associée est disponible en sortie.
- L'adresse de l'octet mémoire est fournie par la sortie du compteur.
- La sortie du multiplexeur MUX est reliée au $\overline{load}$ du compteur.
- Compteur préchargeable synchrone :
  - la valeur courante du compteur est constamment disponible en sortie
  - lorsque load est mis à 0 le mot placé en entrée est chargé sur un front actif d'horloge et le compteur prend alors la nouvelle valeur après le front
  - lorsque load est à 1, la valeur courante du compteur s'incrémente d'une unité à chaque front d'horloge
  - Lorsque $\overline{clear}$ est à 0, la valeur du compteur est forcée à 0000 sur le front actif d'horloge.
- X et Y sont les sorties de commande du système.

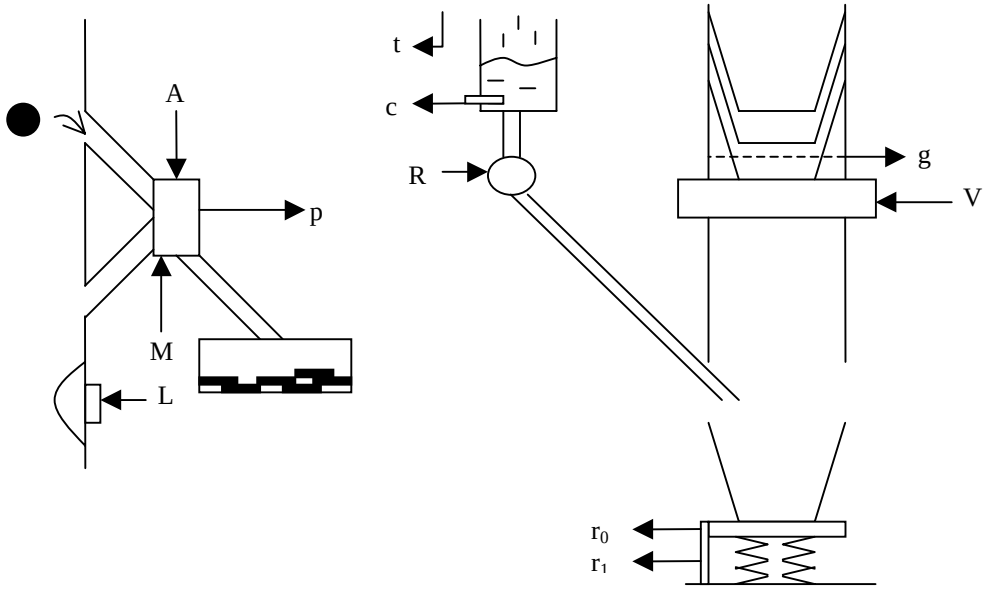
- Expliquez la structure du mot mémoire.
- En justifiant la démarche adoptée, donner le contenu de la mémoire pour obtenir le fonctionnement désiré. Vous rendrez avec votre copie l'annexe complétée en n'oubliant pas d'y inscrire votre nom.
- Expliquez comment vous réaliseriez ce séquenceur en logique câblée (de manière synchrone). Vous appuierez vos explications en donnant un schéma de principe sous forme de blocs fonctionnels où chacun des blocs sera clairement explicité.

  Note : On ne demande ni de résoudre les équations logiques mises en jeu ni de dessiner le logigramme complet.

2. Séquenceur de machine à café (5 pts)

A la mise sous tension de la machine, la lampe est éteinte, aucun gobelet ne peut tomber, le café ne peut couler, la machine rend tout jeton glissé dans la fente prévue à cet effet. La machine ne sera disponible que lorsque la variable binaire t sera à 1 pour signifier la fin du chauffage initial du café. Ensuite, le fonctionnement de la machine est le suivant :

- Tant qu'il manque soit du café, soit des gobelets, la lampe est éteinte et le jeton est rendu à l'utilisateur
- S'il ne manque ni café, ni gobelet, la lampe est allumée, le système attend un jeton
- Lorsque le jeton est détecté, alors
  - la lampe s'éteint
  - puis le gobelet tombe
  - puis le café.
  - Lorsque le gobelet est plein, la pièce est avalée.
  - Un nouveau café ne peut être servi que si l'utilisateur à retiré son gobelet



Signification des variables binaires :

- p = 1 : jeton en attente
- g = 1 : gobelet présent dans réserve
- c = 1 : café présent dans réservoir
- r0 = 1 gobelet présent sous tuyau
- r1 = 1 gobelet plein présent
- R = 1 : ouverture vanne café
- V = 1 : ouverture clapet gobelet
- M = 1 : jeton rendu à l'utilisateur
- A = 1 : jeton avalé
- L = 1 : lampe allumée
- t = 1 : café chaud

- Etablir le graphe d'état correspondant au cahier des charges
- On suppose que pour permettre le chauffage initial du café, il est nécessaire d'attendre une minute.
  - Comment réaliseriez vous la variable binaire t indiquant la fin du chauffage initial.
  - Justifiez la fréquence d'horloge que vous choisiriez pour réaliser le séquenceur de la machine.

3. Assembleur 68HC11 (5 pts)

Dans cette partie, il s'agit d'écrire des sous routines en assembleur 68HC11. Pour chacune des sous routines afin qu'elles soient portables, si besoin, on utilisera uniquement des variables locales situées dans la pile. De même le passage de paramètres s'effectuera par la pile et le résultat retourné par la fonction s'effectuera dans le registre A pour les données 8 bits et dans le registre D pour les données 16 bits.

Note importante:

- pour les paramètres stockés dans la pile, l'ordre croissant des adresses correspond aux paramètres lus de gauche à droite dans la fonction prototype en C.
- La pile sera vidée des variables temporaires passées en paramètres avant de retourner au programme appelant.

3.1. Addition BCD

- Donner le code assembleur d'une routine prenant deux nombres BCD en paramètre et retournant le résultat BCD de l'addition de ces 2 nombres.

Prototype C

unsigned char AdditionBCD (unsigned char val1, unsigned char val2)

- Ajouter à votre fonction une gestion d'erreur : si le résultat est trop grand, le nombre retourné sera constitué que de '1'.

3.2. Permutation dans un tableau

- Donner le code assembleur d'une routine réalisant une permutation entre 2 données d'un tableau. C'est à dire la valeur contenue dans une case i d'un tableau est mise dans une case j et la valeur de la case j est mise dans la case i.

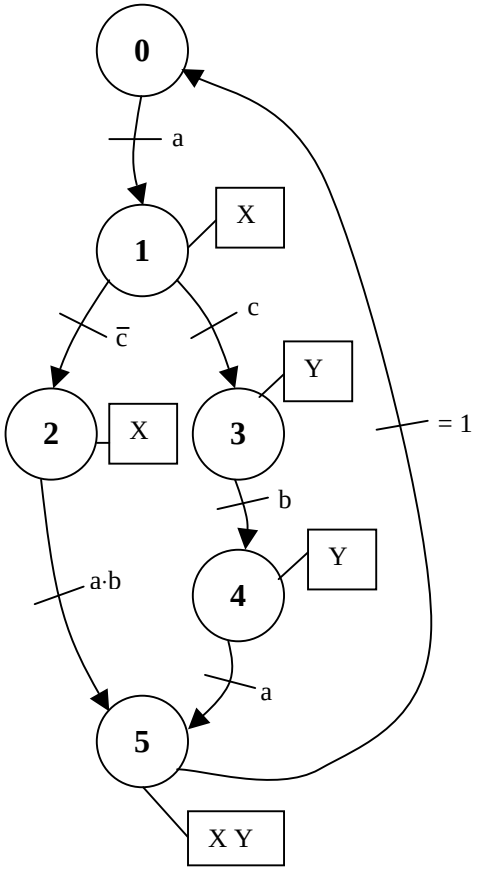
Prototype C :

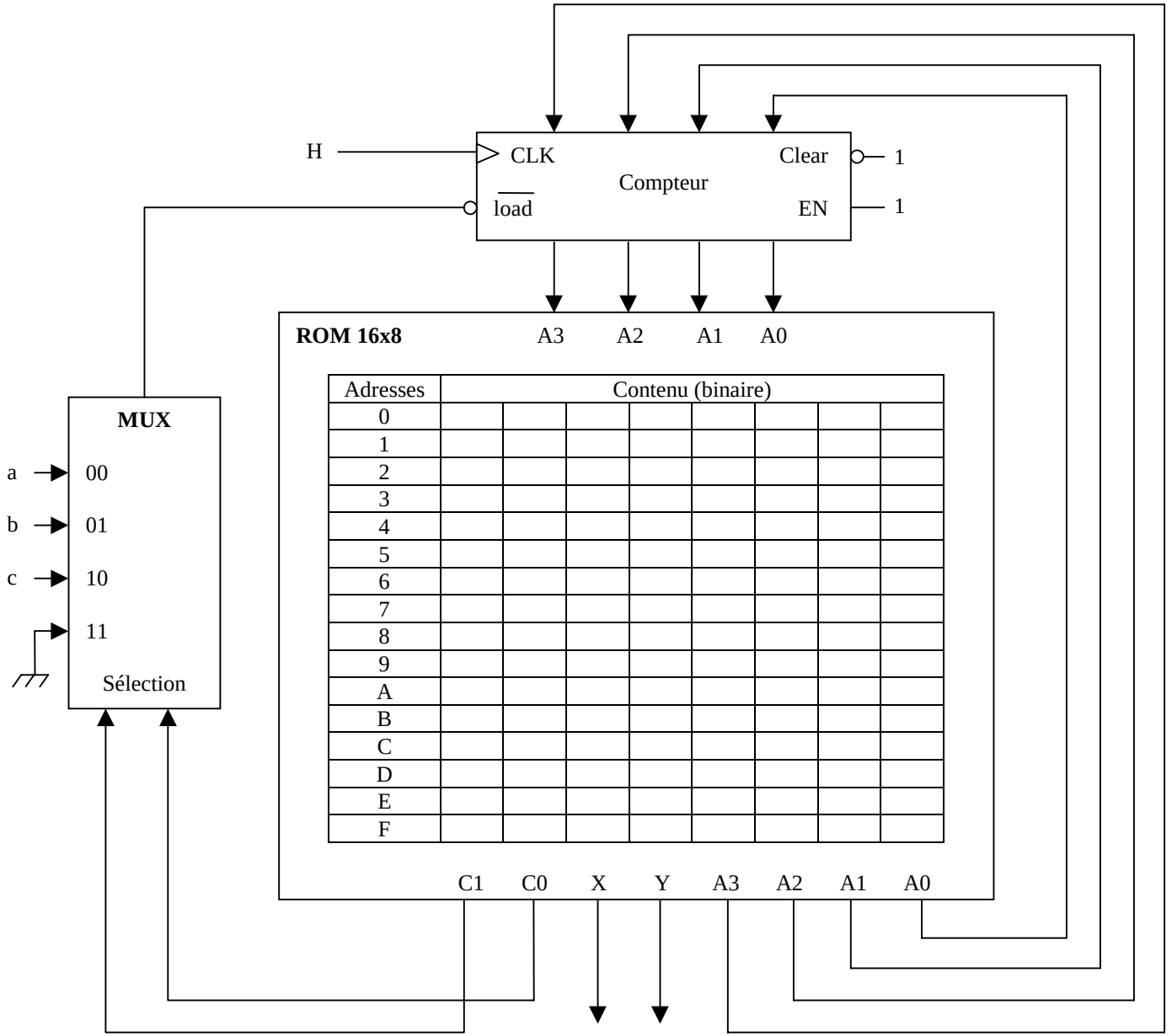
```
void EchangeDonnées(char * val1, char * val2)
```

- Quelle modification faudrait-il apporter à votre code si les données pointées étaient du type "short int" (entiers sur 16 bits);

Rappel sur les pointeurs : un pointeur correspond à une adresse où se trouve la variable ainsi **pointeur = valeur* signifie que *pointeur* est une adresse et que le contenu de la ou des case(s) mémoire correspondante(s) est *valeur*.

ANNEXES





NOM : **PRENOM :**

Médian MI41

Durée : 2h. Documents autorisés : polycopiés de cours et TD uniquement. Le barème est donné à titre indicatif uniquement.

1. Conversions (2pts)

On adopte une représentation signée sur 12 bits. La convention de représentation est la représentation en complément à 2. Donnez les représentations binaires et hexadécimales des nombres décimaux suivants :

1. +132

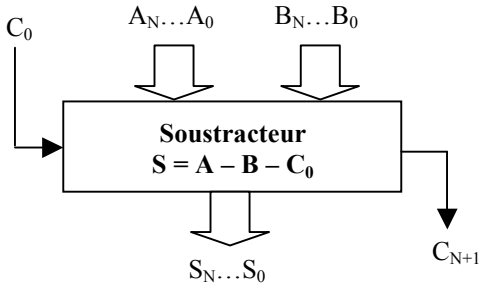
2. - 507

3. + 2048

4. -3020

2. Soustracteur binaire (13pts)

On souhaite réaliser un soustracteur binaire qui effectue la soustraction binaire d'un nombre **A** sur N bit par un nombre **B** également sur N bits et retourne le résultat **S** (S = A – B). Ce soustracteur devra être complet : retenue entrante C_0 et retenue sortante C_{N+1}.



2.1. N = 1 (4pts)

En expliquant clairement votre démarche, déterminez les équations logiques vérifiées par S_0 et C_1 d'un soustracteur complet 1 bit (N = 1).
Tracez (proprement à la main) le logigramme

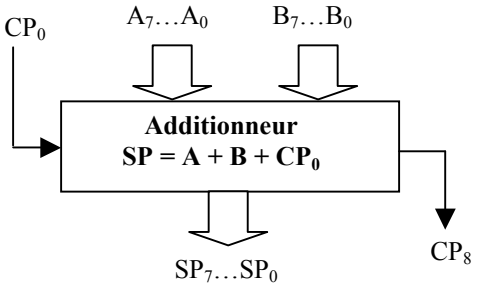
2.2. N = 4 (1pt)

Donnez la structure sous forme de bloc du soustracteur complet pour N = 4.

2.3. Additionneur/soustracteur

2.3.1. Additionneur/Soustracteur sans retenue 8 bits (5 pts)

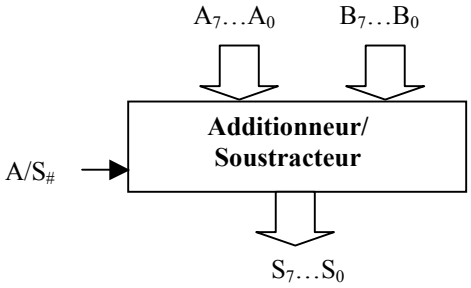
On souhaite réaliser un soustracteur 8 bits (sans retenue entrante ni retenue sortante)à l'aide d'un additionneur complet 8 bits,.



1. Rappelezr comment faire la soustraction de deux mots de 8 bits A – B à l'aide de l'opérateur d'addition

2. Donnez le schéma de principe du soustracteur sans retenue utilisant le circuit additionneur donné ci-dessus.

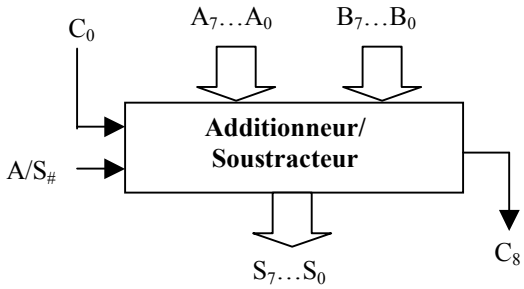
3. Expliquez comment faire un additionneur/soustracteur commandé sans retenue à l'aide du circuit additionneur :



$A/S_{\#}$ commande du circuit :
- Si $A/S_{\#} = 1$ alors S = A + B
- Si $A/S_{\#} = 0$ alors S = A - B

2.3.2. Additionneur/soustracteur avec retenue 8 bits (3pts)

On souhaite maintenant d'une part tenir compte de la retenue entrante C_0 et d'autre part générer une retenue C_8 pour un circuit additionneur/soustracteur de poids supérieurs.



$A/S_{\#}$ commande du circuit :
- Si $A/S_{\#} = 1$ alors $S = A + B + C_0$
- Si $A/S_{\#} = 0$ alors $S = A - B - C_0$

On reprend le circuit d'additionneur complet pour réaliser ce circuit

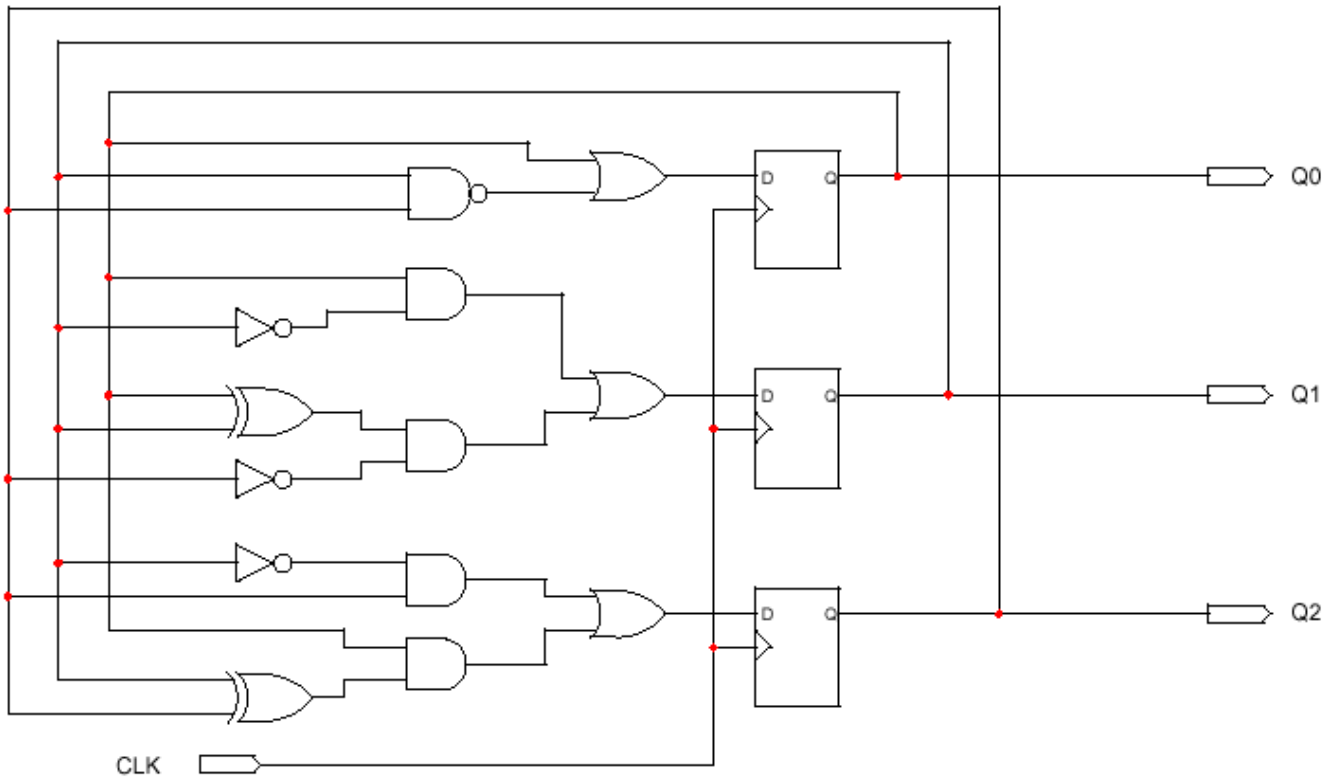
1. Expliquer comment tenir compte de la retenue entrante C_0

2. Expliquer comment générer la retenue sortante C_8

3. Donnez le schéma de principe de l'additionneur/soustracteur complet.

3. Système synchrone (5pts)

On considère le système ci dessous, où CLK est un signal d'horloge.



Les sortie des bascules sont initialement à zéro Q2=Q1=Q0= 0 ;
En détaillant votre démarche, donnez le cycle décrit par Q2 Q1 Q0

Médian MI41 Automne 2001

1. Conversions (représentation 12 bits signée)

1. +132 0000 1000 0100 soit 084h.
2. -507 $C\ell_2(0001\ 1111\ 1011) = 1110\ 0000\ 0101$ soit E05h
3. +2048 Non représentable.
4. -3020 Non représentable.

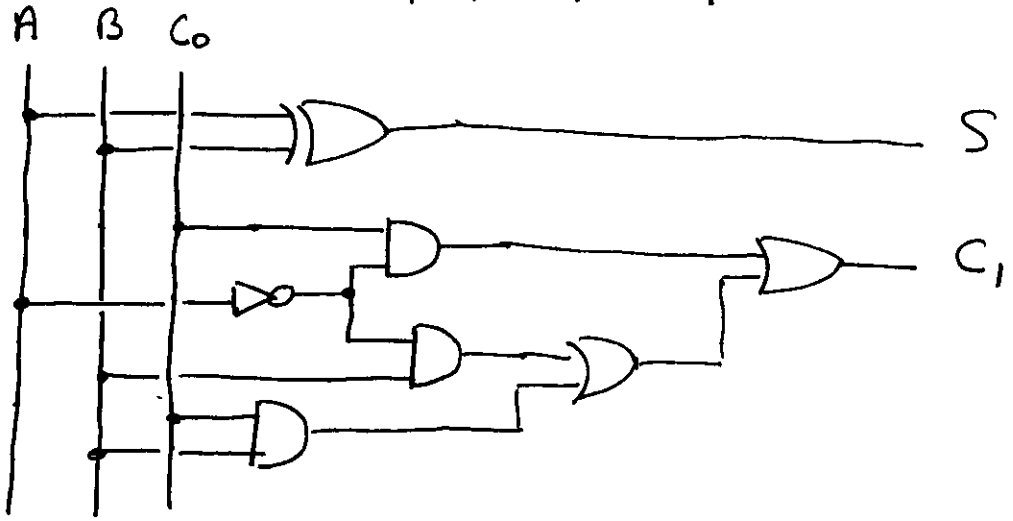
2. Soustracteur binaire

2.1. N = 1

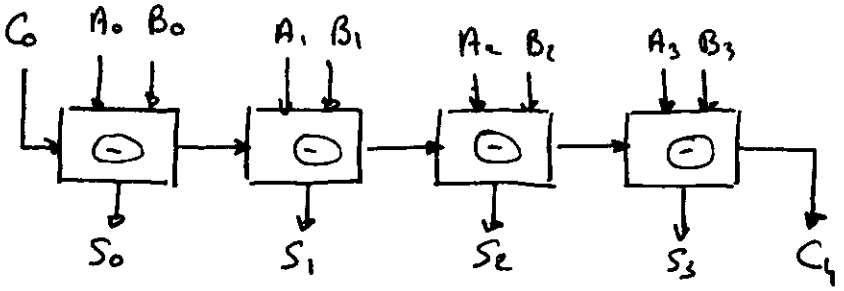
C_ϕ	A	B	S = A-B	C_1
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

$S = A \oplus B \oplus C_\phi$

$C_1 = C_0\bar{A} + \bar{A}B + C_0B$



2.2. Structure pour N = 4



2.3. Additionneur/soustracteur.

1. $A - B = A + CP2(B)$ où $CP2(B) = \bar{B} + 1$

$\bar{B} = (\bar{B}_{N-1}\ \bar{B}_{N-2} \dots \bar{B}_0)$ inversion bit à bit

2. Pour réaliser un additionneur soustracteur sans retenue entrante il suffit

* Addition . $CP\emptyset = \emptyset$
  . A : première entrée
  . B : seconde entrée

* Soustraction . $CP\emptyset = 1$
  . $\overline{A}$: première entrée
  . $\overline{B}$: seconde entrée.

Ainsi . $CP\emptyset = \overline{A/S\#}$

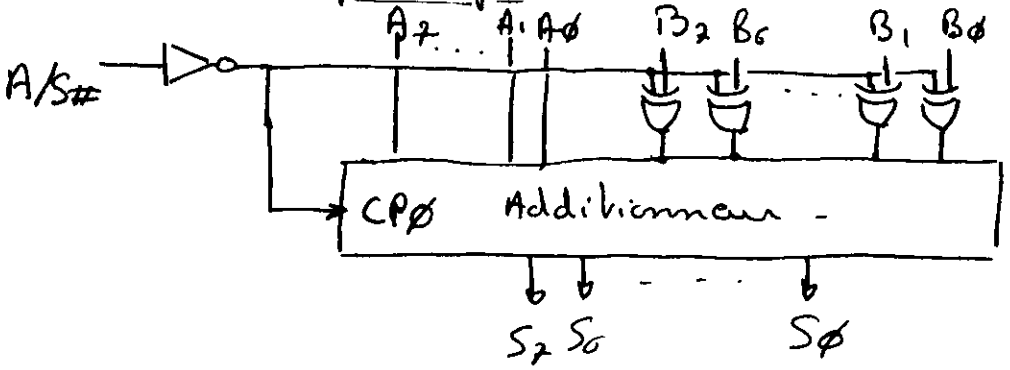
. Entrée 1 = A

Entrée 2 = $B\, A/S\# + \overline{B}\ \overline{A/S\#}$

$= B \oplus \overline{A/S\#}$

→ OU exclusif bit à bit

Schéma de principe.



2.3.2. Additionneur soustracteur avec retenue

1 retenue entrante.

a) Addition.
- $A/S\# = 1$
- entrée 1 = A
- entrée 2 = B
- $CP\emptyset = C_\emptyset$

b) Soustraction
- $A/S\# = \emptyset$
- entrée 1 = $\overline{A}$
- entrée 2 = $\overline{B}$
- $CP\emptyset = C_0$ *

* en effet A-B+1 donné par $A + \overline{B} + 1 - C_\emptyset$ or
$1 - C_\emptyset$ → 1 si $C_\emptyset = \emptyset$
→ $\emptyset$ si $C_\emptyset = 1$ donc $1 - C_\emptyset = \overline{C_0} = CP\emptyset$

2. retenue sortante

a) addition
$C_8 = CP_8$

b) Soustraction
$C_8 = \overline{CP_8}$

en effet la soustraction est donnée par $A + CP2(B) = A + \underbrace{2^N - 1 - B}_{CP1(B)} + 1$
soit $A - B + 2^N$

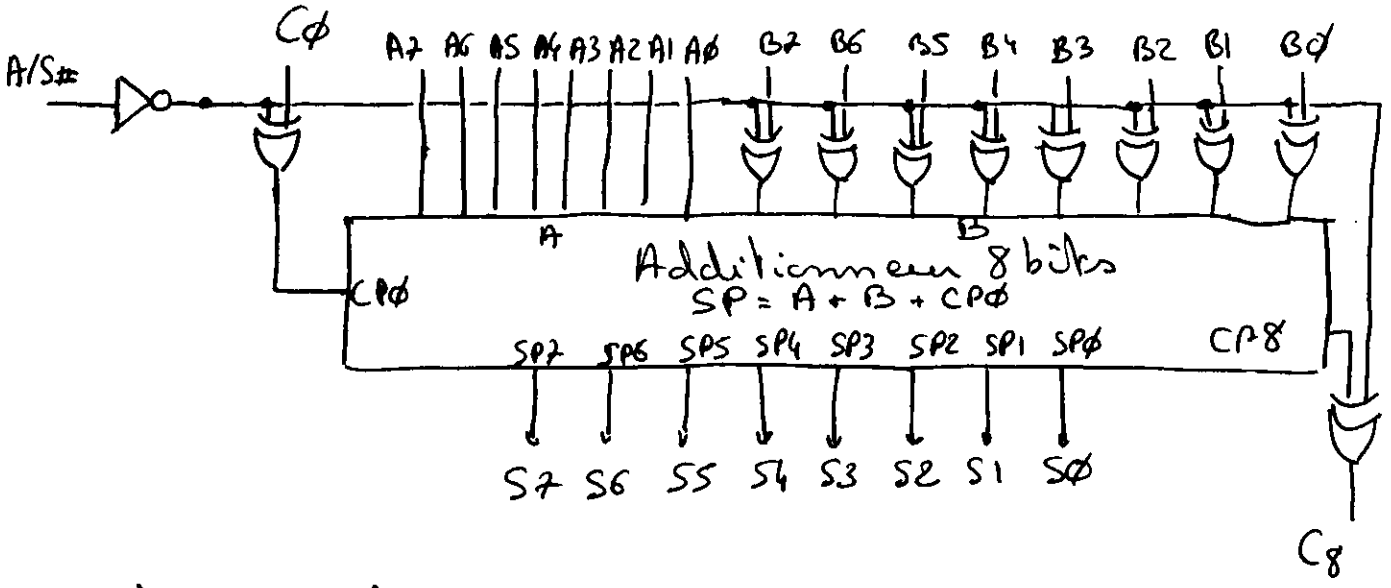
• Si $A-B > \emptyset$ $\quad A-B+2^N > 2^N$ Il y a donc apparition d'une retenue au rang N $\quad CP8 = 1$

• Si $A-B < \emptyset$ $\quad A-B+2^N < 2^N$ Il n'y a donc pas de retenue $\quad CP8 = \emptyset$

finalement $\quad C_8 = \overline{CP8}$

finalement le schéma général est le suivant :

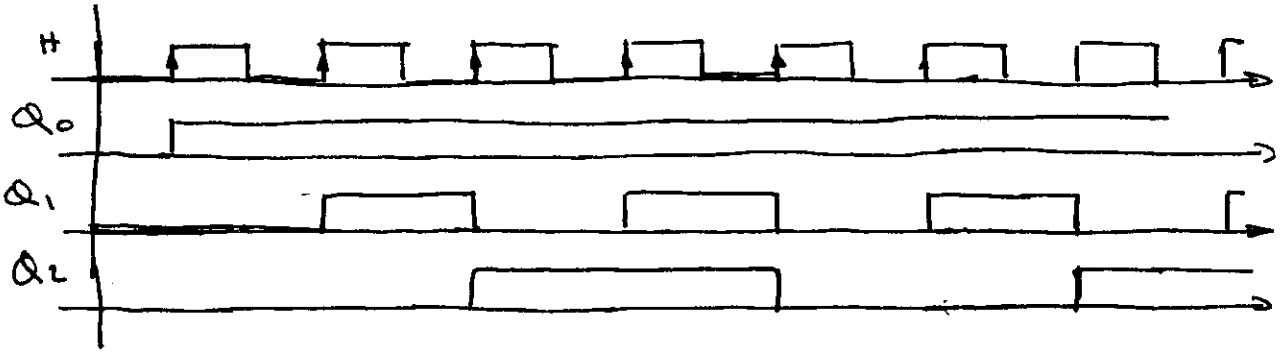
Additionneur / Soustracteur complet.



3- Système asynchrone.

$D_0 = Q_0 + \overline{Q_1 Q_2}$; $D_1 = Q_0 \overline{Q_1} + (Q_0 \oplus Q_1)\overline{Q_2}$;

$D_2 = \overline{Q_1} Q_2 + (Q_1 \oplus Q_2) Q_0$



MI41 - Médian

Durée : 2h.
Documents autorisés exceptés livres et photocopies de livres.
Lisez bien l'énoncé avant de commencer.

1. Représentation binaire des nombres

1.1. Représentations signées complément à 2 sur 5 bits

1. Donnez les valeurs décimales des représentations numériques suivantes :
01110 ; 10101 ; 10000

2. Donnez les représentations numériques des nombres décimaux suivants :
13 ; -11 ; 29

1.1.2. Représentation flottante normalisée IEEE 754 32 bits

Soit le nombre flottant suivant :

1100 0100 1000 0000 0000 0010 1001 0000

On souhaite convertir ce nombre en un nombre entier. Les nombres entiers sont également représentés sur 32 bits suivant une représentation signée complément à 2. Le nombre flottant donné n'étant par forcément entier la conversion devra tronquer le nombre à sa partie entière.

Donnez la représentation binaire du nombre entier correspondant

2. Transcodeur binaire réfléchi/binaire naturel

Le code binaire réfléchi permet de coder les nombres de manière à ce qu'entre deux nombres consécutifs un seul bit change. L'intérêt d'un tel code et de réduire les conséquences d'une erreur de détection d'un bit. Ainsi si on fait une erreur d'un bit sur un mot binaire codé en binaire réfléchit, alors le nombre ne sera faux que d'une unité.

Dans cet exercice, le but est d'établir une relation entre le code binaire réfléchi et le code binaire naturel.

1. Mots de 2 bits :
A (A1, A0) est le mot codé en binaire réfléchi et B le mot codé en binaire naturel. Donnez l'équation la plus simple possible de B0 en fonction de A1 et A0

A1	A0	B1	B0
0	0	0	0
0	1	0	1
1	1	1	0
1	0	1	1

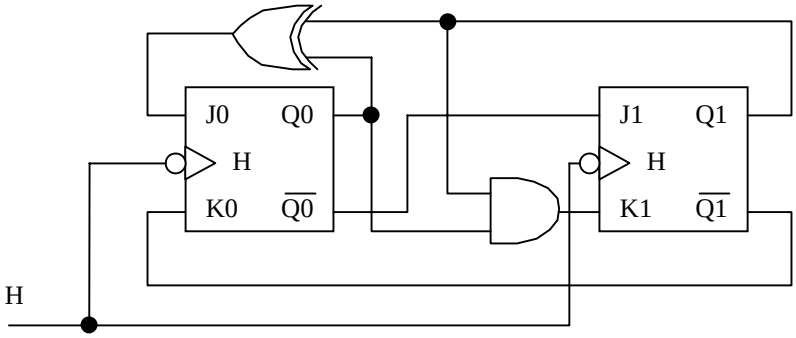
2. Mots de 3 bits :
Déterminez les équations de B2, B1 et B0 en fonction de A2, A1, A0

A2	A1	A0	B2	B1	B0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

3. Mots de N bits :
Etablissez une relation permettant de déterminer un bit Bi du mot B (binaire naturel) en fonction des bits Aj du mot A (binaire réfléchi)

3. Chronogrammes

Soit le schéma suivant :

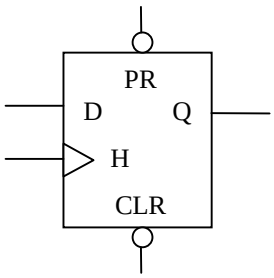


Le système est initialement dans l'état Q1 Q0 = 0 0. Tracez, en les justifiant, les chronogrammes d'évolution de Q1 et Q0 en fonction de l'horloge H.

4. Compteur asynchrone

On dispose de bascules D actives sur front montant avec Preset et Clear asynchrone ainsi que de portes logiques élémentaires..

1. Réalisez un compteur asynchrone par 5 (5 états complets) réalisant le cycle suivant : 0, 1, 2, 3, 4, 0 …



2. On souhaite modifier le cycle un fois sur 2. Une fois sur 2 l'état correspondant à la valeur 4 est sauté ce qui donne pour le cycle : 0, 1, 2, 3, 4, 0, 1, 2, 3, 0 …
Donnez le schéma du système réalisant le nouveau cycle.

Vendredi 26 novembre 1999

Médian MI41 – 2h

1. Conversions (4 pts.)

On considère des nombres représentés sur 6 bits. Compléter le tableau suivant lorsque cela est possible (mettre une croix sinon).

Décimal	Binaire naturel	Hexadécimal associé	Binaire signé (Complément à 2)	Hexadécimal associé
	X	**X**	**111010**	
	X	**X**	**010011**	
17				
-23	**X**	**X**		
55				
73				
-32	**X**	**X**		

2. Logique combinatoire (4 pts.)

Soit la table de vérité suivante :

abcd	**S**
0000	0
0001	1
0010	1
0011	X
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	X
1101	1
1110	1
1111	X

1. Donner l'équation la plus simple possible de cette fonction logique.
2. Donner le logigramme correspondant.

3. Description VHDL (7 pts.)

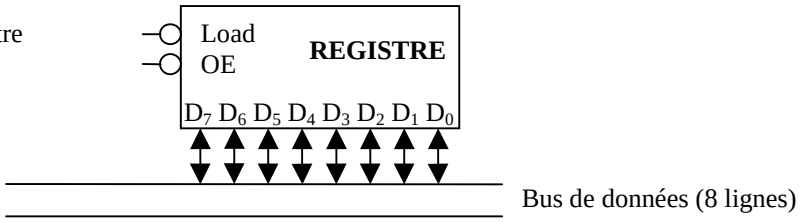
Ecrire l'***entity*** et l'***architecture*** d'un registre 8 bits 3 états, possédant:

2 lignes de commandes : load et OE
8 lignes de données

Fonctionnement :

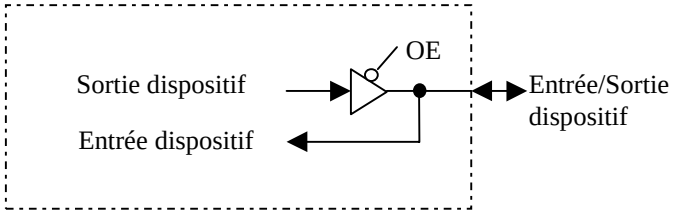
- Lorsque load est à '0' la valeur présente sur le bus de données est acquise
- Lorsque load repasse à '1' la valeur acquise est mémorisée à l'intérieur du registre
- Lorsque OE est à '0' la valeur mémorisée à l'intérieur du registre est mise en communication avec le bus de données
- Lorsque OE est à '1' les sorties du registre sont à l'état haute impédance ('Z' en VHDL)

Note : On ne gérera pas de priorité entre load et OE (les 2 pouvant être simultanément actif).

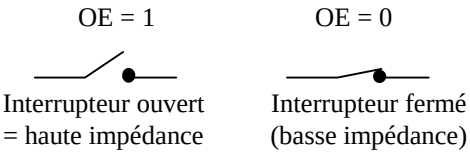


⚠ **Votre description devra être commentée et justifiée…**

Rappel sur la haute impédance

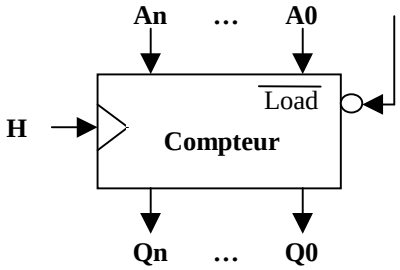


Fonctionnement équivalent à un interrupteur monodirectionnel :



4. Compteur synchrone (7 pts.)

On souhaite réaliser un compteur synchrone par 7 avec des bascules D actives sur front montant. Ce compteur compte suivant la fréquence d'horloge H de 0 à 6. On souhaite pouvoir charger une nouvelle valeur à tout moment et de manière synchrone. Le chargement s'effectue en plaçant la broche $\overline{load}$ à un niveau bas. Lorsque $\overline{load}$ est à 0, sur un front montant de H, la valeur présente sur An … A0 est chargée dans le compteur.



1. Réaliser ce compteur synchrone à l'aide de bascules D actives sur front montant et de portes logiques élémentaires.
2. Ajouter une broche d'inhibition EN à votre schéma. Lorsque EN vaut '1' le dispositif fonctionne comme décrit précédemment, lorsque EN vaut '0' la valeur courante du compteur est conservée.