Evaluation FORMATIVE

Session S4 GEGI - Unité 2

Génie électrique
Génie informatique

GEN430 Circuits logiques séquentiels
GEN 420 Mathématiques des circuits logiques séquentiels

Faculté de génie

Université de Sherbrooke

2024

QUESTION 1 (GEN420)

Une voiture est dotée d'essuie-glaces et d'une pompe permettant de projeter du liquide lave-glace sur le pare-brise. Cet exercice s'intéresse à la conception d'une machine à états finis (MEF) de type Moore pour contrôler uniquement la séquence de lavage d'un pare-brise commandée par le conducteur sans tenir compte des autres commandes des essuie-glaces. (Ne pas tenir compte si les essuie-glaces étaient déjà en marche, on fait comme s'ils étaient au repos)

Pour déclencher la séquence de lavage, le conducteur tire vers lui la manette des essuie-glaces. Tant et aussi longtemps que la manette *A* est tirée, la pompe *P* est actionnée et donc le liquide lave-glace est projeté sur le pare-brise. Tant et aussi longtemps que la manette *A* est tirée, le moteur *M* est actionné et les essuie-glaces font des cycles de balayage sur le pare-brise.

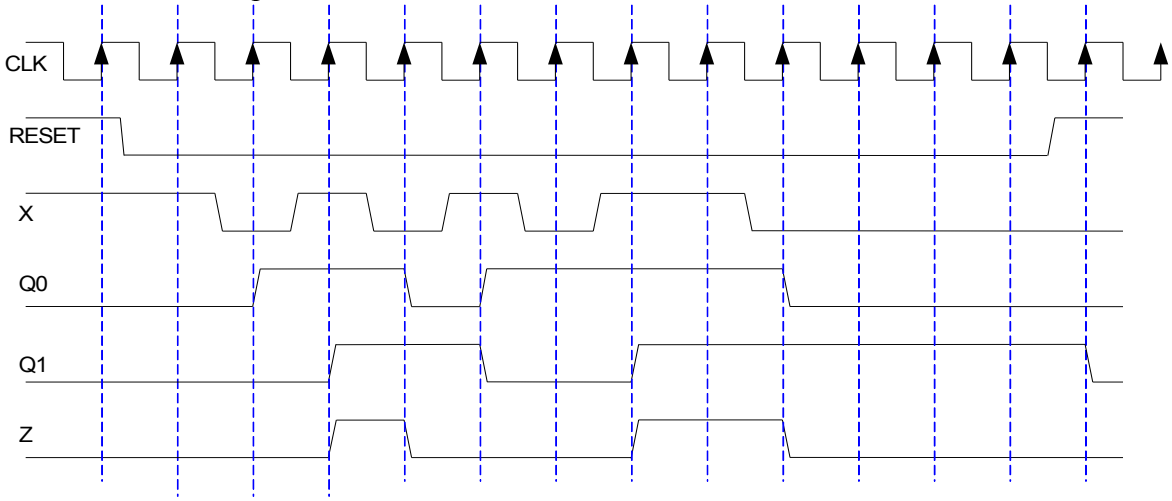
Un capteur *C* détermine la fin d'un aller-retour des essuie-glaces. Un niveau logique haut apparaît sur ce capteur lorsque les essuie-glaces passent vis-à-vis la position repos. Le capteur C est conditionné pour générer un pulse d'une longueur de 1 cycle d'horloge. À la détection de ce pulse (*strobe*), la MEF décide en fonction des spécifications ci-dessous d'arrêter le moteur *M* = '0' ou de le laisser continuer *M* = '1'.

Lorsque la manette *A* est relâchée par le conducteur alors la pompe *P* cesse de fonctionner. En revanche, pour évacuer le liquide sur le pare-brise, les essuie-glaces continuent d'effectuer des cycles jusqu'à ce que le capteur *C* soit touché à deux reprises. À la deuxième détection du capteur, le moteur *M* s'arrête; ce qui immobilise les essuie-glaces en position repos. À tout moment, le conducteur peut tirer à nouveau la manette ce qui a pour effet de recommencer à zéro tout le processus.

1. En suivant une bonne démarche, concevez un diagramme d'états pour une machine à états finis de type Moore capable de contrôler *M* et *P*. Utilisez les noms abrégés présentés dans l'énoncé pour nommer les entrées et les sorties.
2. Proposez une table d'états-transitions encodée en binaire Gray.

QUESTION 2

1. (GEN420 et GEN430) Concevoir un circuit séquentiel qui a comme entrée X et comme sortie Z. Le circuit doit réaliser la fonction décrite par le chronogramme de la figure suivante. Q0 et Q1 sont les sorties des bascules de la machine à états finis. On demande de tracer le diagramme d'état de la machine à états finis et de respecter les étapes d'une démarche rigoureuse lors de la résolution du problème. Le *Reset* n'est pas à considérer comme une entrée.
2. (GEN430) Déterminer la fréquence maximum de l'horloge pour le circuit. Justifier très clairement votre réponse.



Les paramètres des bascules sont équivalents à ceux donnés pour la série 74HCT74 par la table donnée ci-après et les temps de propagation dans les portes (t_p) sont de 5 ns (min et max égaux).

Paramètres de 74HCT74:

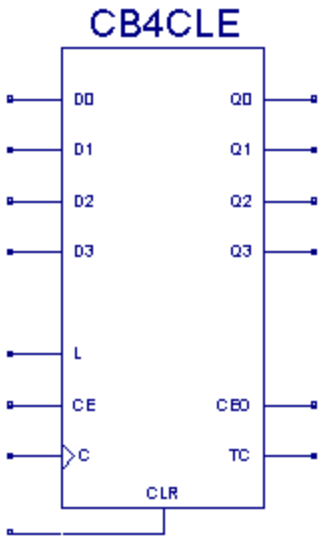
- t_{setup} = 15 ns
- t_{hold} = 3 ns
- t_{ffpd_min} = 35 ns et t_{ffpd_max} = 44 ns
- t_w (L ou H) = 23 ns

Note t_{ffpd_max} correspond à t_{pmax} dans la notation du manuel Digital Design de Roth

QUESTION 3 (GEN430)

En utilisant un compteur standard de type CB4CLE et de la logique combinatoire, réaliser un compteur particulier qui compte de 3 à 12 inclusivement. D0, D1, D2, D3 sont les entrées avec D0 étant le bit le moins significatif. Q0, Q1, Q2, Q3 sont les sorties correspondantes.

Le cycle initial peut débuter à 0.



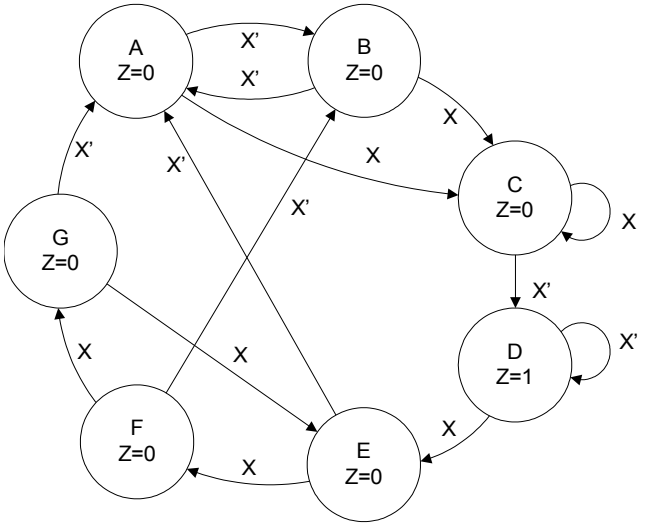
Extrait de la fiche technique :
CB2CLE, CB4CLE, CB8CLE, and CB16CLE are, respectively, 2-, 4-, 8-, and 16-bit (stage) synchronously loadable, asynchronously clearable, cascadable binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The data on the D inputs is loaded into the counter when the load enable input (L) is High during the Low-to-High clock transition, independent of the state of clock enable (CE). The Q outputs increment when CE is High during the Low-to-High clock transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High. (…)

QUESTION 4 (GEN420)

Considérant le diagramme ci-après :

1- Déterminer la table des états et sorties;
2- Faire l'assignation binaire des états et déterminer la table de transitions et sorties;
3- Déterminer les équations d'excitations pour des bascules D avec cette assignation;
4- Faire l'assignation « One Hot » des états et déterminer la table des états et sorties;
5- Déterminer les équations d'excitations pour des bascules D avec cette assignation.

X = Entrée
Z = Sortie
A état initial
(reset)



QUESTION 5 (GEN420)

On considère un circuit séquentiel à deux entrées X et Y et une sortie Z, synchronisé sur une horloge. Élaborer le diagramme d'états d'une machine de Mealy qui produit à la sortie un 1 quand, au pulse précédent de l'horloge, X est différent de Y. Décrire la situation que chacun des états représente.

QUESTION 6

Questions diverses

1. (GEN430) Quelle différence existe-t-il entre une réinitialisation (reset) synchrone et asynchrone d'un compteur.

2. (GEN430) Quelle différence existe-t-il entre un circuit logique synchrone et asynchrone ?

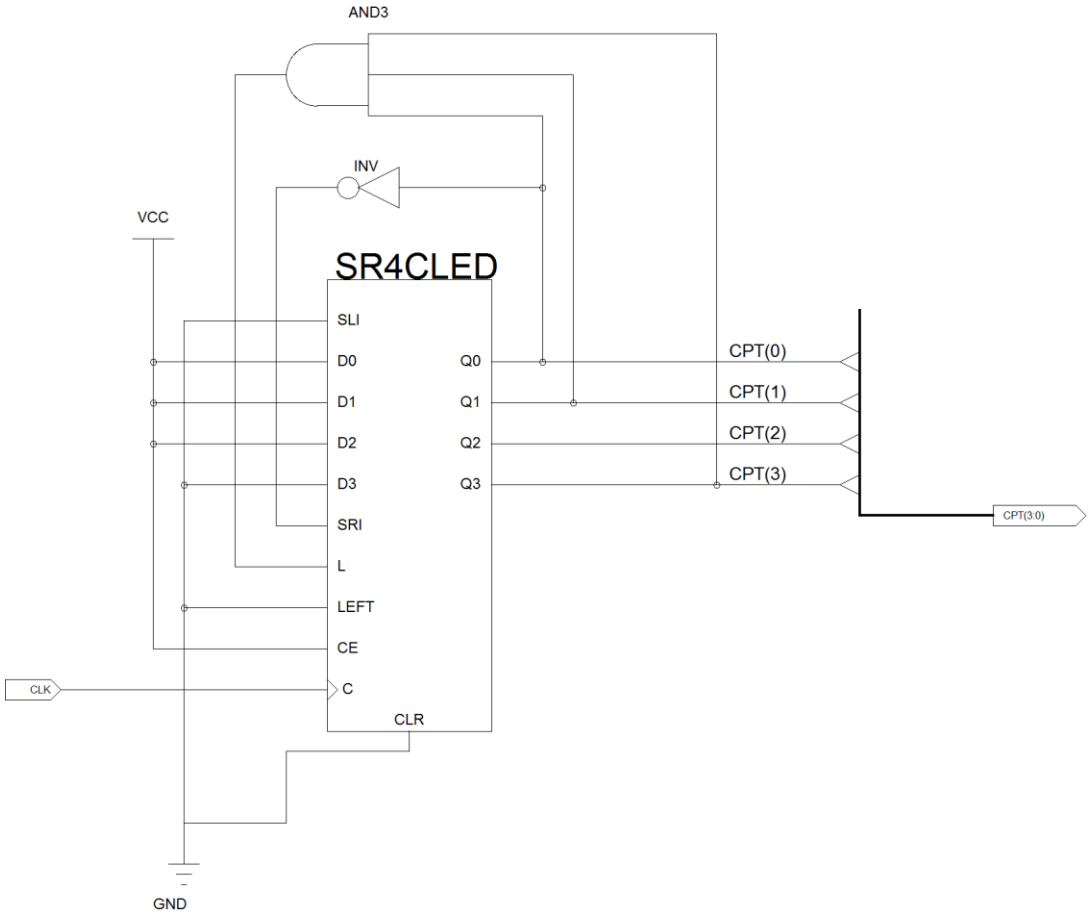
3. (GEN420) Quels sont les attributs d'une table de transition qui correspondent aux cercles d'un diagramme d'état.

QUESTION 7 (GEN430 et GEN420)

Montrer que le circuit suivant est un « self-starting counter ». Dans ce circuit, un décalage vers la droite se produit lorsque L = 0 et LEFT = 0 et un chargement intervient seulement quand L = 1. À la mise sous tension, vous ne pouvez pas supposer l'état initial. La fiche technique sur le SR4CLED est à la page suivante

Pour vous aider à résoudre ce problème, commencez par analyser le rôle de chaque entrée :

- *Quelles entrées correspondent aux données?*
- *Quels sont les signaux de contrôle et quels comportements en résulte?*
- *Quelle est la configuration actuelle?*
- *Posez les valeurs Q3-Q0. Que se passe-t-il?*



INFORMATION SUR LE SR4CLED

Macro: 4-Bit Shift Register with Clock Enable and Asynchronous Clear

Introduction

This design element is a shift register with shift-left (SLI) and shift-right (SRI) serial inputs, parallel inputs (D), parallel outputs (Q), and four control inputs: clock enable (CE), load enable (L), shift left/right (LEFT), and asynchronous clear (CLR). The register ignores clock transitions when (CE) and (L) are Low. The asynchronous clear, when High, overrides all other inputs and resets the data outputs (Qn) Low.
When (L) is High and (CLR) is Low, the data on the (D) inputs is loaded into the corresponding (Q) bits of the register. When (CE) is High and (L) and (CLR) are Low, data is shifted right or left, depending on the state of the LEFT input. If LEFT is High, data on the SLI is loaded into (Q0) during the Low-to-High clock transition and shifted left (for example, to Q1 or Q2) during subsequent clock transitions. If LEFT is Low, data on the SRI is loaded into the last (Q) output during the Low-to-High clock transition and shifted right during subsequent clock transitions. The logic tables indicate the state of the (Q) outputs under all input conditions.
This register is asynchronously cleared, outputs Low, when power is applied. For FPGA devices, power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the appropriate STARTUP_*architecture* symbol.

Logic Table

Inputs								**Outputs**		
CLR	**L**	**CE**	**LEFT**	**SLI**	**SRI**	**D3 : D0**	**C**	**Q0**	**Q3**	**Q2 : Q1**
1	X	X	X	X	X	X	X	0	0	0
0	1	X	X	X	X	D3– D0	↑	D0	D3	Dn
0	0	0	X	X	X	X	X	No Change	No Change	No Change
0	0	1	1	SLI	X	X	↑	SLI	q2	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1
qn-1 and qn+1 = state of referenced output one setup time prior to active clock transition.										

QUESTION 8 (GEN430 et GEN420)

On considère le code VHDL suivant. Dessiner un diagramme de machine à états finis correspondant.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY DetecteurMEF IS
   PORT (CLK,x,RESET: IN std_logic;
         y: OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF DetecteurMEF IS
   TYPE type_sreg IS (S0,S1,S2);
   SIGNAL sreg, next_sreg : type_sreg;

BEGIN

   PROCESS (CLK, RESET, next_sreg)
   BEGIN
      IF ( RESET='1' ) THEN
         sreg <= S0;
      ELSIF rising-edge(CLK) THEN
         sreg <= next_sreg;
      END IF;
   END PROCESS;

mef:  PROCESS (sreg, x)
   BEGIN
   CASE sreg IS
      WHEN S0 =>
         IF ( x='1' ) THEN
           next_sreg<=S1;
          ELSE
            next_sreg<=S0;
         END IF;
     WHEN S1 =>
        IF ( x='1' ) THEN
           next_sreg<=S2;
         ELSE
           next_sreg<=S0;
         END IF;
     WHEN S2 =>
         IF ( x='1' ) THEN
           next_sreg<=S2;
         ELSE
           next_sreg <=S0;
         END IF;
     WHEN OTHERS =>
           next_sreg <=S0;
   END CASE;
   END PROCESS;


--                                                                    (suite page suivante)


sortie: PROCESS (sreg)
   BEGIN
```

```
      CASE sreg IS
         WHEN S0 =>
               y <= '0';
         WHEN S1 =>
               y <= '1';
         WHEN S2 =>
               y <= '0';
         WHEN OTHERS =>
      END CASE;
   END PROCESS;

END BEHAVIOR;
```

QUESTION 9 (GEN430 et GEN420)

Circuit séquentiel (partiel) pour réaliser un multiplieur

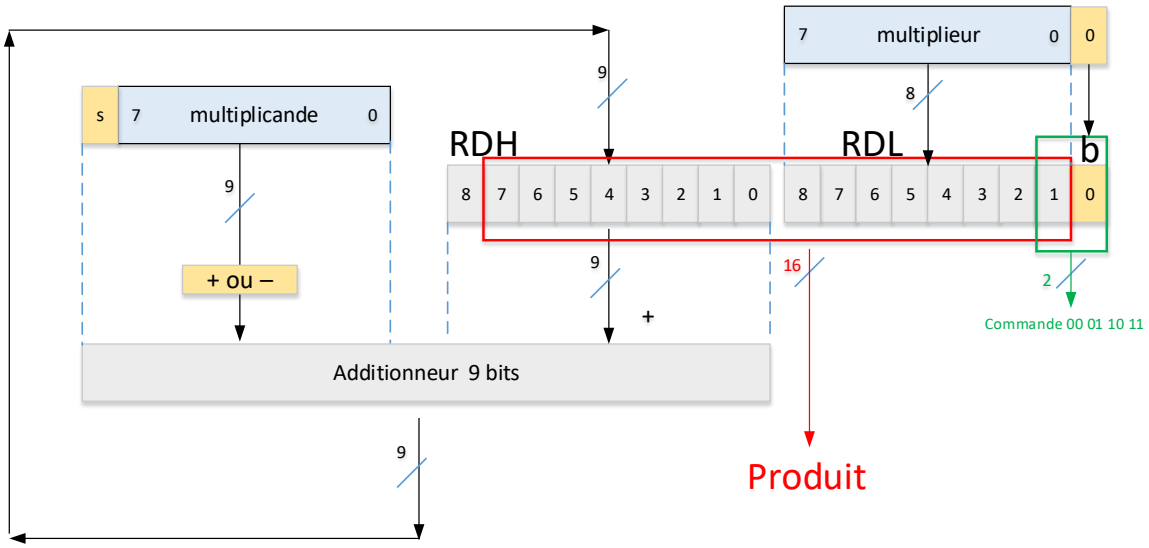
On désire réaliser un circuit en VHDL pour un multiplieur séquentiel pour opérandes de 8 bits en complément à 2 par l'algorithme de Booth

```
entity booth_mult is
   port(
     clk          : in   std_logic;
     reset        : in   std_logic;
     i_debut      : in   std_logic;
     i_mpcand     : in   std_logic_vector(7 downto 0);
     i_mpieur     : in   std_logic_vector(7 downto 0);
     o_produit    : out  std_logic_vector(15 downto 0);
     o_occupe     : out  std_logic;
     o_fin        : out  std_logic
);
end booth_mult;
```

Fonction

Synchronisé sur l'horloge CLK, le multiplieur commence l'opération de multiplication à l'occurrence d'une impulsion sur i_*debut*. Pendant l'exécution la sortie *o_occupe* est maintenue à '1', et quand le résultat est disponible sur *o_ produit*, le module génère une impulsion synchrone sur une seule période *o_fin*.

La figure suivante illustre les unités requises pour le fonctionnement qui seront contrôlés par une MEF :



Un bit initialement à zéro est placé à droite des bits du multiplieur. Le registre à décalage complet constitué des parties RDH, RDL et b contient 18 bits.

Noter qu'un 9ième bit est ajouté à gauche au multiplicande ainsi qu'au registre RDH. Ainsi les opérandes de l'additionneur ont 9 bits pour éviter les débordements arithmétiques qui peuvent se produire avec seulement 8 bits. Le bit « s » ajouté au multiplicande est une copie de son bit de signe dans la représentation du complément à 2.
L'additionneur est combinatoire.

Vous devez réaliser **le code VHDL de la machine à état fini** qui contrôle les circuits nécessaires à l'algorithme de Booth. On ne cherche pas ici à implémenter directement l'algorithme de Booth lui-même, **seulement la MEF.**

Questions :

1. Définir et décrire en VHDL une machine à états finis MEF pour **commander les unités séquentielles de ce circuit** et pour générer les signaux *o_occupe* et *o_fin*. Les commandes (signaux de sortie de la MEF) à générer sont :
    - Initialiser le registre et le compteur
    - Décaler
    - Compter
    - Générer les impulsions *o_occupe* et *o_fin*

Commencer par dessiner le schéma-bloc et le diagramme d'état de votre MEF avant d'écrire le VHDL. Quelles sont les étapes de l'alagorithme de Booth? Est-ce que certaines peuvent se faire en parallèle – donc dans le même état?
ATTENTION! La MEF contrôle les étapes séquentielles seulement. La logique combinatoire sera faite séparément.

2. Déterminer un « process » **qui génère l'opérande issu du multiplicande** avec son signe adéquat pour l'entrée de l'additionneur.

Ici on ne cherche pas à faire l'addition ou la soustraction directement. On veut trouver l'opérande à mettre à l'entrée de l'additionneur pour effectuer l'opération visée par la commande de 2 bits. Quelle modification faut-il apporter au multiplicande pour que l'additionneur exécute une addition? Une soustraction? Rien?

Algorithme (pas nécessaire à la solution)
L'algorithme de Booth pour la multiplication tire parti du fait que lorsque le multiplieur contient une série continue de bits de valeur 1, il est possible de réduire le nombre d'additions par rapport à l'algorithme classique.
Ceci est illustré par l'exemple suivant :

00111100 = 01000000 – 00000100

ce qui permet dans cet exemple de faire la multiplication avec des opérations de décalage accompagnées d'une soustraction et d'une addition au lieu des opérations de décalages accompagnées de quatre additions.

Un intérêt de l'algorithme de Booth est de s'appliquer aux opérandes signés dans la représentation du complément à 2. La question est posée pour deux opérandes signés de 8 bits en représentation complément à deux. Le produit sera défini sur 16 bits en complément à deux. L'algorithme fait intervenir un registre à décalage avec commande de chargement composé de 18 bits vus comme 3 parties (description ci-dessous), un additionneur complet sur 9 bits, un compteur sur 3 bits, et un module de contrôle qui peut activer le compteur, faire un décalage à droite du registre complet avec son signe préservé.

L'algorithme de Booth se décrit ainsi :

Initialiser le registre a décalage : RDH avec tous ses bits à '0', RDL avec le multiplieur et '0'.
Exécuter la séquence suivante en comptant de 0 à 7 :
À chaque étape :

Examiner les deux bits à droite du registre à décalage et selon leur valeur faire exécuter l'opération suivante par l'additionneur/soustracteur :

« 01 » : faire une addition
« 10 » : faire une soustraction
« 11 » : ne rien faire
« 00 » : ne rien faire

Actionner le compteur d'étape
Faire un décalage à droite du registre à décalage (18 bits combinés) en préservant le signe.