TP 3 : Calculatrice

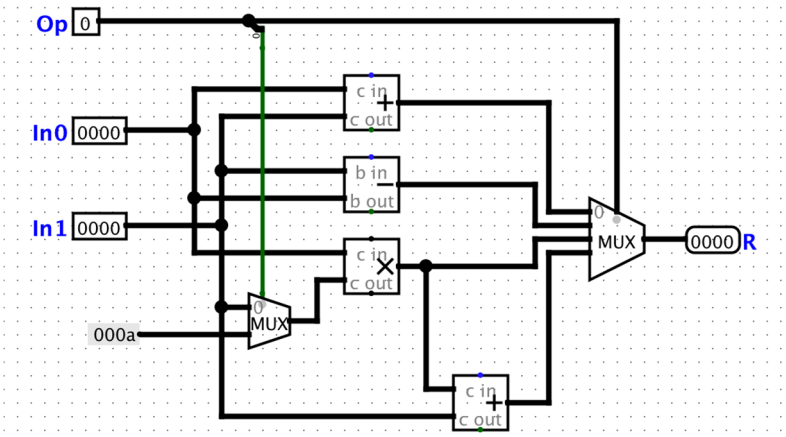
L'objectif de ce TP est de vous faire réaliser une calculatrice qui fonctionne sur le même principe que les calculatrices du commerce. Elle utilisera pour cela le clavier numérique de 4x4 et affichera le résultat courant sur les afficheurs 7 segments de la carte DE2. Afin de combler l'absence de touche +,-,= et *, vous utiliserez la correspondance définie ci-dessous (la touche F réalisera l'addition, la touche E la soustraction, etc.).

1	2	3	+/F
4	5	6	-/E
7	8	9	*/D
CE	0		=/C

La conception complète d'une calculatrice à partir de zéro est trop complexe pour 2h de TP, vous partirez donc d'une base de circuit dont il faudra comprendre le fonctionnement afin de pouvoir le compléter. Les questions qui suivent ont pour but de vous aider à appréhender le fonctionnement du circuit qui vous sert de point de départ.

Partie 1 : Analyse du composant UAL

La calculatrice que vous allez mettre en œuvre utilise une UAL (Unité Arithmétique et Logique) différente de celle réalisée au cours du TP1. Le circuit de cette UAL est représenté ci-dessous et est disponible dans le fichier UAL_CAL.circ.

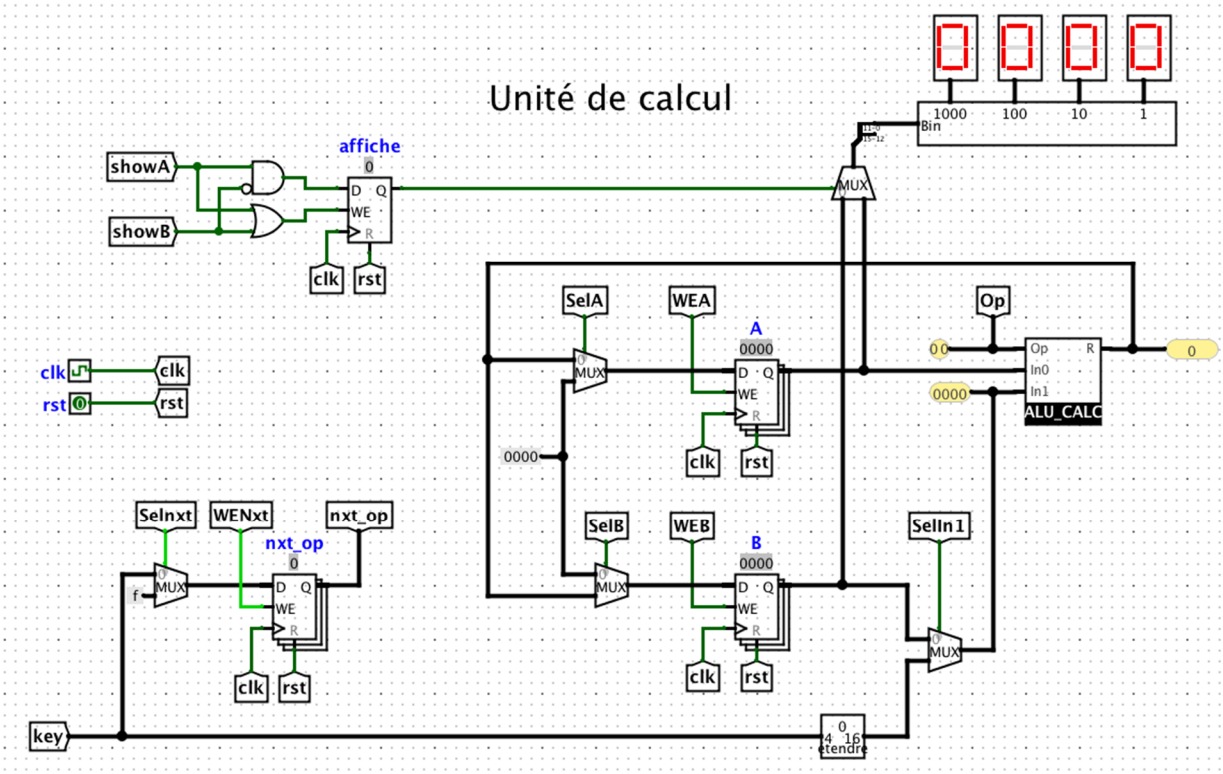


Question 1 : en vous basant sur le schéma de l'UAL, complétez sa table de vérité ci-dessous

OP	**R**	Description du calcul
00		
01		
10		
11		

Partie 2 : Unité de calcul

Le cœur de calcul du circuit, disponible dans le dans le fichier TP-Calculatrice.circ, est représenté ci-dessous



Ce cœur de calcul est construit autour de trois registres (nommés **A**, **B** et **op_nxt**). Ces registres disposent de commandes de chargement (commandes **WEA**, **WEB** et **WENxt**) et stockent des données sur 16 bits pour **A** et **B** et sur 4 bits pour **nxt_op**. Les valeurs pouvant être chargées dans ces registres sont issues de composant multiplexeurs commandés par les signaux **SelA**, **SelB**, **SelNxt**.

Dans ce cœur de calcul, le composant UAL étudié à la question précédente est utilisé pour réaliser des opérations arithmétiques sur le contenu des registres **A**, **B** ou **nxt_op**. La sélection de l'opération à effectuer se fait au travers du signal **Op**.

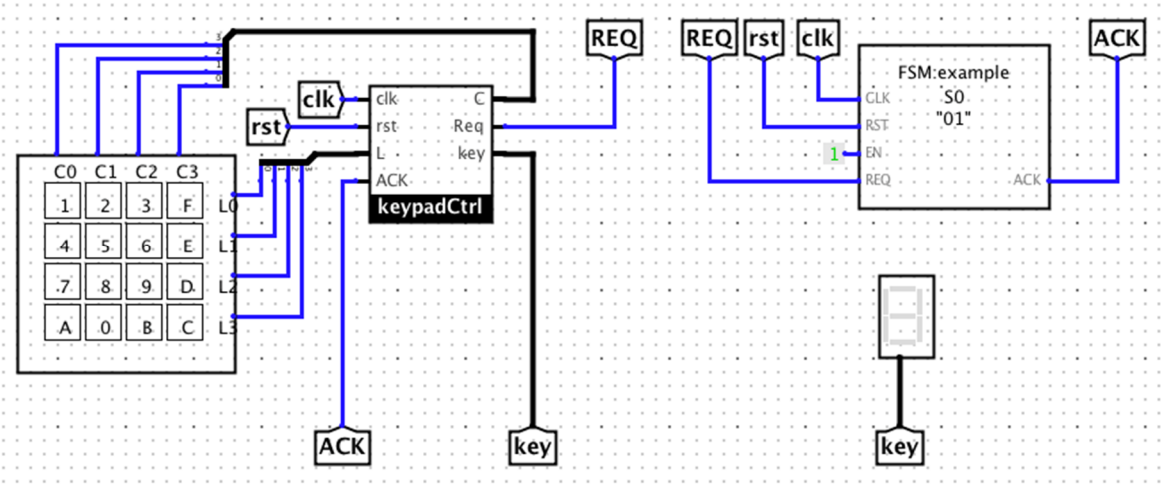
Le cœur de calcul intègre également un affichage contrôlé par une bascule D (nommée affiche) qui mémorise lequel des registres **A** et **B** doit être affiché. Le contrôle de cette bascule se fait par les commandes **showA** (pour basculer vers l'affichage de **A**) et **showB** (pour basculer vers l'affichage de B).

Question 2 : en vous basant sur les explications fournies ci-dessus, complétez le tableau ci-après avec les valeurs (0, 1 ou ϕ) à assigner aux différents signaux de commandes pour réaliser le traitement indiqué dans la première colonne.

Traitement	WEA	WEB	WENxt	SelA	SelB	SelNxt	Op	SelIn1	showA	showB
A = 0;										
B = 0;										
A = A*10+key										
B = A+B										
B = B-A										
affiche(A)										
affiche(B);										
nxt_op=f ;										
nxt_op=key										

Partie 3 : interfaçage avec le contrôleur keypad

La récupération d'un caractère depuis le contrôleur de keypad se fait au travers d'un protocole REQ/ACK. Dans la version du contrôleur utilisée ici, c'est le clavier qui signale la disponibilité d'un nouveau caractère en plaçant le signa REQ à 1.



Du point de vue du composant calculatrice, la séquence d'opérations à effectuer pour récupérer un caractère est donc la suivante :

1) Placer **ACK**= 0 et attendre jusqu'à ce que **REQ** passe à 1
2) Lire la valeur sur le port **key** du contrôleur de keypad
3) Placer la commande ACK à 1 et attendre que REQ repasse à 0
4) Replacer ACK à 0

Question 3 : ouvrez le fichier question3.circ, et proposez un diagramme d'état pour la machine à état *question_3* permettant de lire les données issues du clavier et les afficher sur l'afficheur hexadécimal

4 : saisie et affichage d'un nombre décimal

Vous allez maintenant rouvrir le fichier TP-Calculatrice.circ pour travailler sur le circuit complet qui contient à la fois l'interface keypad, et le cœur de calcul décrit dans la partie 2 du TP. L'ensemble de ces composants est piloté par une machine à état dont il faudra complétez le diagramme d'état afin de faire fonctionner la calculatrice.

On rappelle que les chiffres reçus du keypad sont codées en BCD. L'unité de calcul fonctionnant sur des représentations en binaire, il est nécessaire de réaliser une conversion. Plutôt que de réaliser cette conversion à l'aide d'un circuit combinatoire, vous allez réaliser l'opération en plusieurs étapes, selon l'algorithme ci-dessous :

```
affiche(A);
while(true) {
   key= keypad();
   if (key<10)
      A=A*10+key;
   else
      A=0 ;
}
```

1) Basculer l'affichage vers le registre A
2) Lire un caractère sur le keypad (utilisation du protocole REQ/ACK)
3) Si le caractère est un chiffre décimal (0-9) calculer A= A*10+key sinon forcer A=0
4) Recommencer en 2)

Question 4 : en utilisant les résultats des question 2 et 3, complétez le diagramme d'état de manière à permettre la saisie et l'affichage d'un nombre saisi au keypad.

Partie 4 : mise en œuvre de la calculatrice « complète »

L'algorithme complet décrivant le fonctionnement de la calculatrice est fourni ci-dessous.

```
#define PLUS  0xF
#define MINUS 0xE
#define MUL   0xD
#define EQUAL 0xC

int A=0, B=0, next_op =PLUS, current_key;

while(true) {
      current_key = keypad();
      while(current_key<10) {
            A=A*10+ current_key;
            current_key = keypad(), display(A);
      }
      switch(next_op) {
            case PLUS :
                  B= A+B ; A=0; display(B);
                  break;
            case MINUS :
                  B= B-A ; A=0; display(B);
                  break;
            case MUL :
                  B= B*A ; A=0; display(B);
                  break;
            case EQUAL :
                  A=0; display(B);
                  break;
      }
      next_op = current_key;
} while(1);
```

Afin de vous approprier son fonctionnement, vous allez « simuler » son exécution à la main, en complétant le tableau ci-dessous. Chaque ligne du tableau indique les valeurs des variables du programme au point d'exécution (1) indiqué dans le code, dans le cas ou l'entrée clavier correspond à la première colonne du tableau.

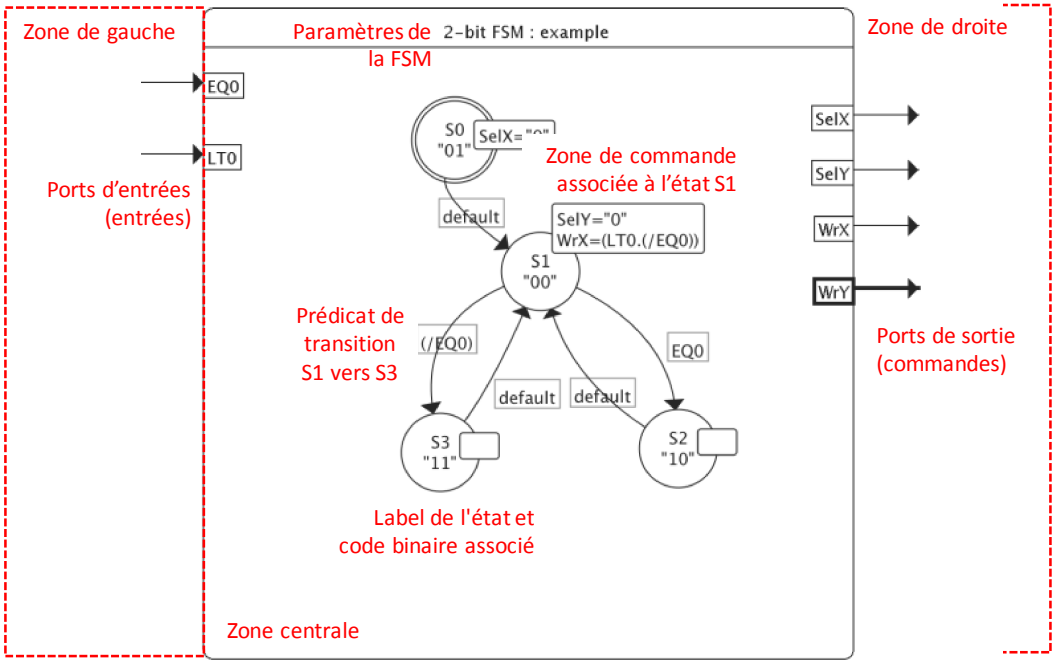
key	A	B	op	affichage
1				
3				
+				
4				
5				
-				
2				
=				

Question 4 : complétez le diagramme d'état de manière à pour obtenir une calculatrice entièrement fonctionnelle. Vous validerez ensuite son fonctionnement sur logisim puis (enfin !) sur la carte.

Annexe A : éditeur de machine à état

La fenêtre de l'éditeur de machine à état se décompose en plusieurs zones, représentées sur la copie d'écran annotée fournie ci-dessous.

Pour ouvrir cette fenêtre d'édition, il est nécessaire de « pointer » sur le composant **`Finite State Machine`** instancié dans un schéma et de cliquer dans la rubrique « *click to edit* » de la fenêtre *Properties*. (à gauche en bas)



Les opérations courantes sont résumées dans la table ci-dessous :

Quoi faire	Comment le faire
Nommer la machine d'états et modifier sa taille (nombre max d'états)	Placez-vous sur le bandeau supérieur de la fenêtre « Paramètres de » dans la figure ci-dessus et cliquez 2 fois. Une fenêtre « State Configuration » apparaît. Vous devez nommer la FSM et indiquer le nombre de bits nécessaires « *code width* » pour coder les états. Par exemple, si le nombre est de 4, le nombre maximum d'états sera de 16.
Ajouter un état	Placez-vous sur la zone centrale et faites apparaître le menu contextuel (clic droit). Sélectionnez l'entrée *add state*, puis déplacez-vous pour placer le nouvel état à l'endroit qui vous convient. Validez avec un clic gauche.
Supprimer un état	Sélectionnez un état (clic gauche), puis faites apparaître le menu contextuel (clic droit). Sélectionnez l'entrée *delete*. L'état et toutes les transitions entrantes sortantes seront supprimés du diagramme.
Ajouter une transition	Sélectionnez un état (clic gauche), puis faites apparaître le menu contextuel (clic droit). Sélectionnez l'entrée *add transition*. Une transition partant de l'état sélectionné doit apparaître, sélectionner l'état destination en déplaçant le pointeur de souris vers l'état concerné, puis en validant par un clic gauche.
Ajouter un port d'entrée	Placez-vous sur la zone de gauche et faites apparaître le menu contextuel (clic droit). Sélectionnez l'entrée *add input port*, puis déplacez-vous sur la bordure gauche du diagramme pour placer le nouveau port, validez avec un clic gauche.
Ajouter un port de sortie (commande)	Placez-vous sur la zone de droite et faites apparaître le menu contextuel (clic droit). Sélectionnez l'entrée *add output port*, puis déplacez-vous sur la bordure droite du diagramme pour placer le nouveau port, validez avec un clic gauche.

Modifier le prédicat associé à une transition/commande	Double-cliquez sur le prédicat associé à la transition/commande qui vous intéresse, une boite de dialogue s'ouvrira alors qui vous permettra de spécifier sous forme textuelle un nouveau prédicat/nouvelle commande. Si le texte saisi n'est pas syntaxiquement correct (voir syntaxe plus loin), un message d'erreur apparaît.

Les valeurs assignées aux ports de sortie sont exprimées à l'aide d'expressions dont le résultat est un mot binaire. La syntaxe de ces expressions (en notation BNF) est décrite ci-dessous. Les assignations sont séparées entre elles par un point-virgule.

ident = expr	Assignation d'une valeur *expr* à la sortie *ident* (**`Y="0"`**)
exp := expr . expr	Et logique sur les opérandes (exemple **`A.B`**)
exp := expr + expr	Ou logique (exemple **`A+B`**)
exp := / expr	Négation logique (exemple **`/X`**)
exp := expr == expr	Egalité (exemple **`A[2:1]==B[3:2]`**)
exp := expr != expr	Différence (exemple **`A!="010"`**)
expr := expr [val : val]	Extraction (pour mot de *n* bits) (exemple **`X[3:2]`**)
expr := '{' expr, expr '}'	Concaténation (exemple **`{A[1],"0"}`**)
expr := '"' {0,1} '"'*	Constantes binaire (ex **`"010010"`**)
expr := ident	Identificateur (nom d'un port d'entrée)

Les prédicats de transition suivent ce même principe, mais ceux-ci doivent forcement s'évaluer comme un résultat booléen (0 ou 1). Il est par ailleurs également possible d'utiliser le mot **`default`** qui est évalué à vrai lorsque toutes les autres transitions issues du même état sont fausses.