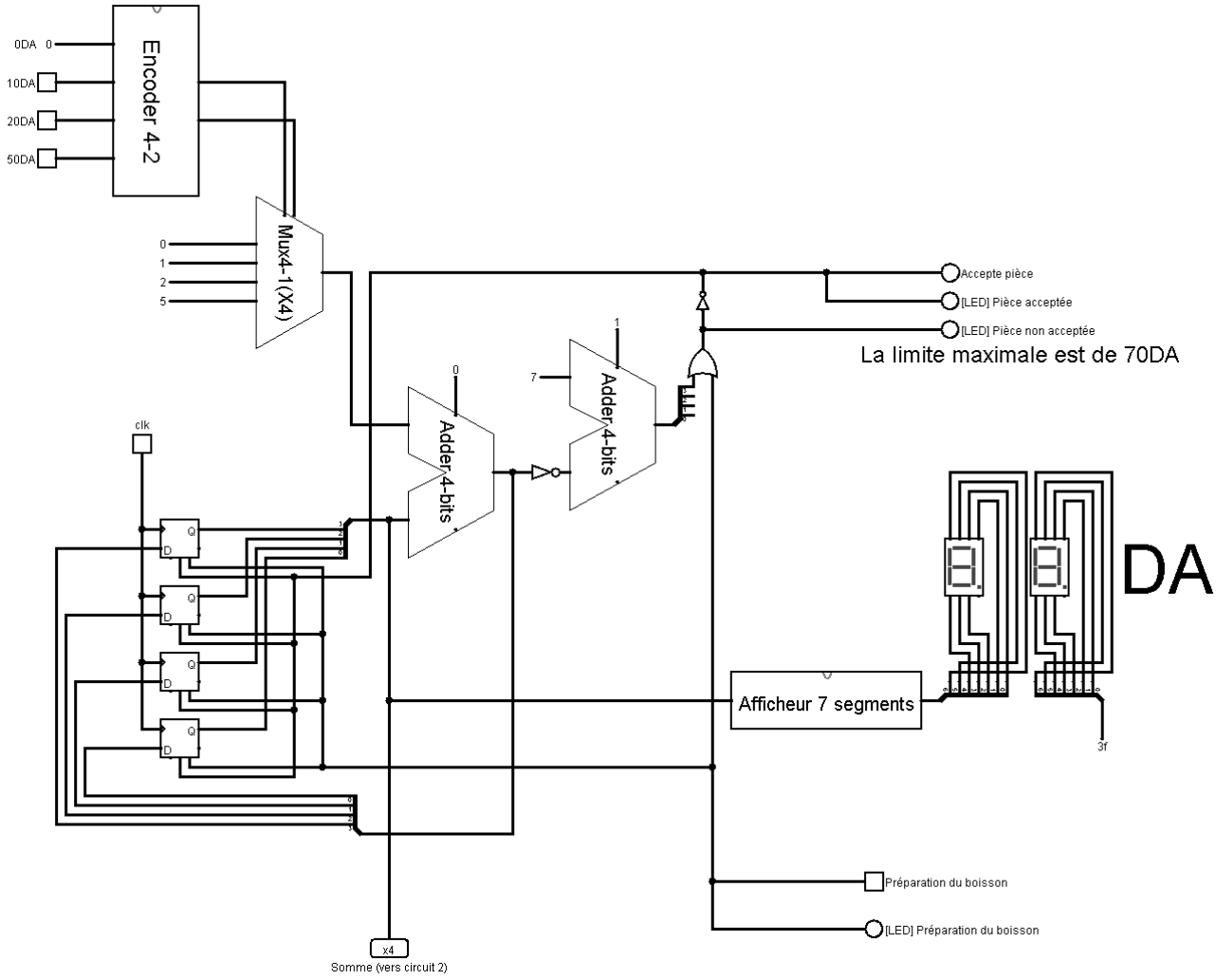
Université Sétif 1
Faculté des Sciences
Tronc Commun Mathématique Informatique

1-ière année (année2020/21)
Module Structures Machines 2
Enseignant Kara Abdelaziz

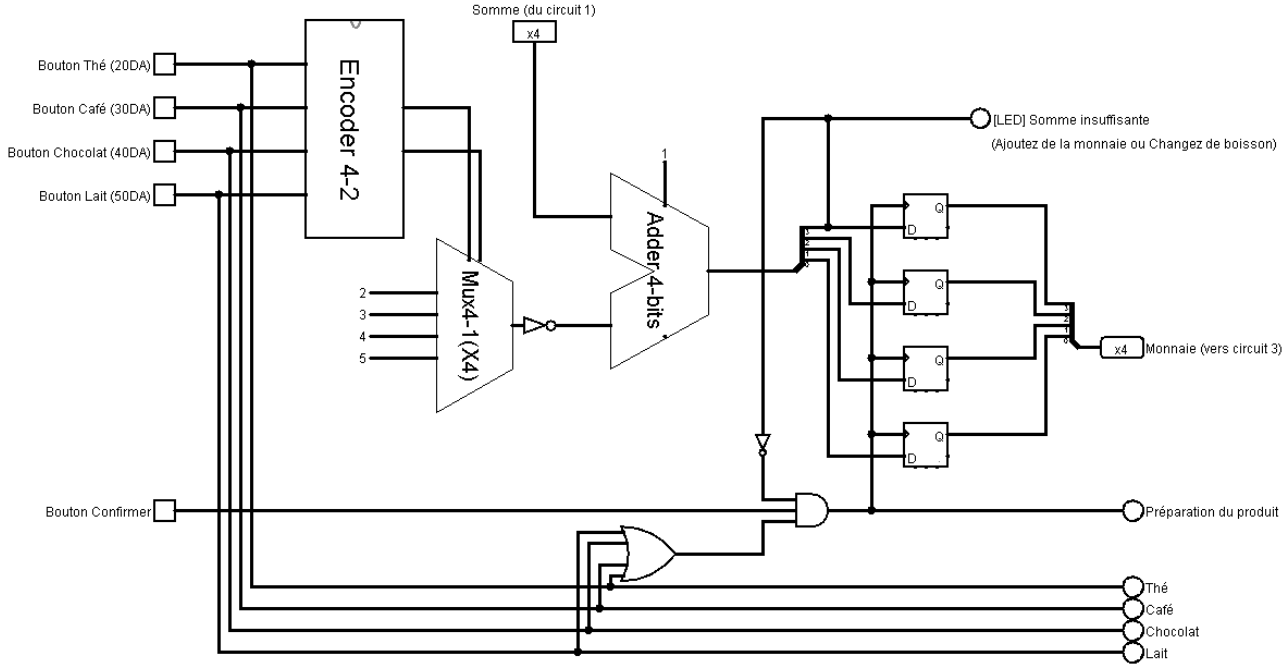
Solution du mini-projet : Machine de Distribution Automatique de Boisson

1)

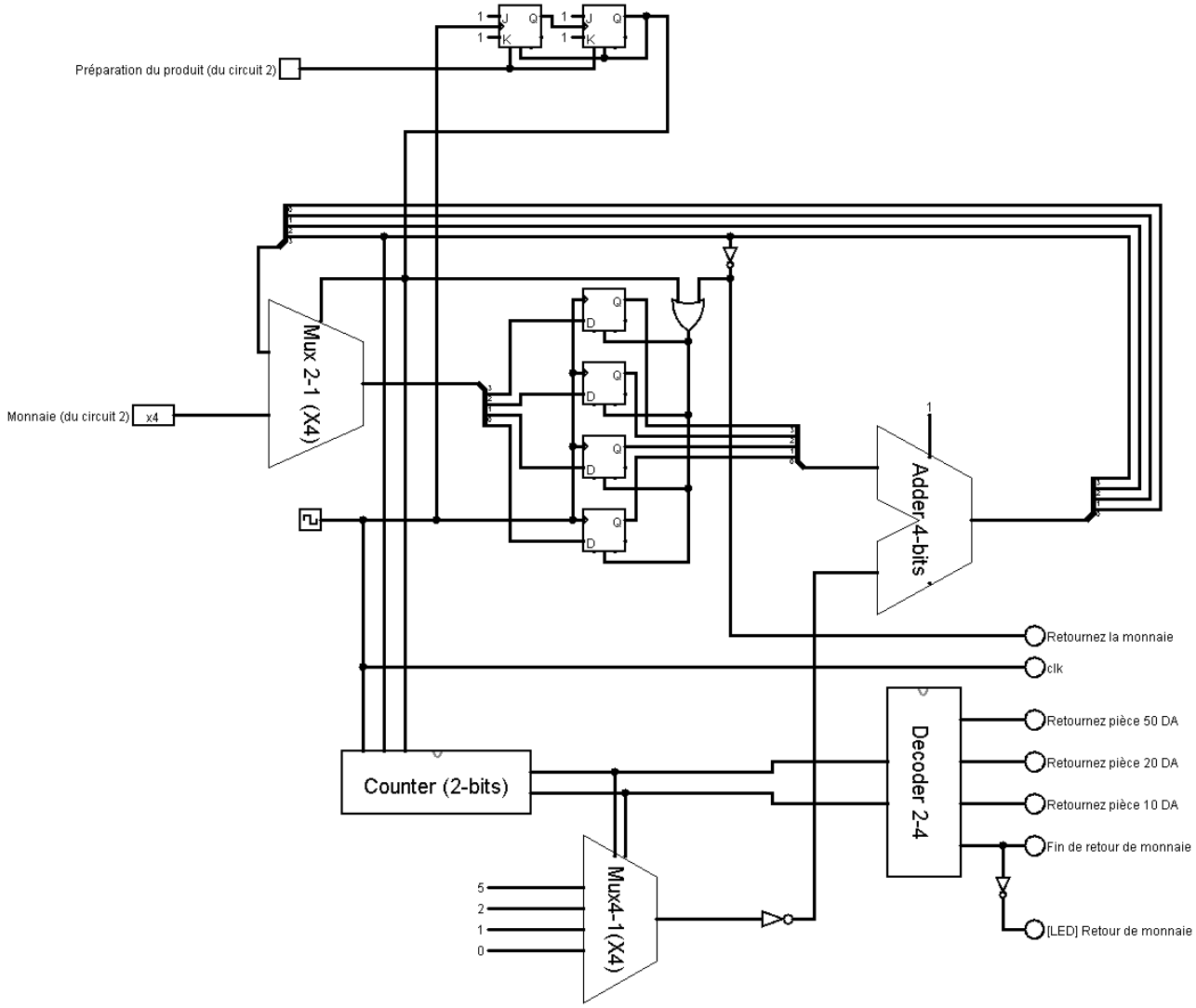
Circuit 1 :



Circuit 2 :



Circuit 3 :



Remarque 1: Cette manière de concevoir des circuits numériques est différente de ce qu'on faisait jusque là, c'est la conception en RTL (Registre Transfert Level), c'est beaucoup plus approprié pour les circuits plus complexes.

Remarque 2: Dans la méthode RTL les éléments de construction ce n'est pas les portes logiques mais plutôt des Circuits Combinatoires et des Circuits Séquentiels, plus des Registres. Les Registres sont utilisés pour garder l'information traitées par les circuits.

Remarque 3: Les Circuits Combinatoires dans RTL peuvent être des circuits manuellement développés ou bien des circuit connus préfabriqués, comme les Multiplexeurs /Démultiplexeurs, les Encodeurs/Décodeurs, les Additionneurs...etc.

Remarque 4: Même chose pour les Circuits Séquentiels, ça peut être des circuits développés, ou circuits préfabriqués comme les Shifteurs, les Compteurs, les Timers...etc.

2) Pour augmenter la limite de la somme d'argent il faut augmenter le nombre de bits, principalement dans les Registres et les Additionneurs et les Multiplexeur. Par exemple l'utilisation des Registres, des Additionneurs et des Multiplexeur à 8 bits permettrait d'augmenter la limite à 1270 DA.

3) Pour pouvoir gérer plus de monnaie, il faut augmenter la capacité des Décodeurs, des Multiplexeurs 4-1, et du Compteur et de l'Encodeur. Si par exemple on prendrais des Décodeurs 8-3, des Multiplexeur 8-1, un Encodeur 8-3 et un Compteur 3-bits, une gestion de 7 pièces de monnaie serait possible.

4) Pour pouvoir gérer plus de boisson, il faut augmenter la capacité de l'Encodeur et de son Multiplexeur, par exemple un Encoder 16-4 et un Multiplexeur 16-1 permettrait de gérer 16 boisons différentes.

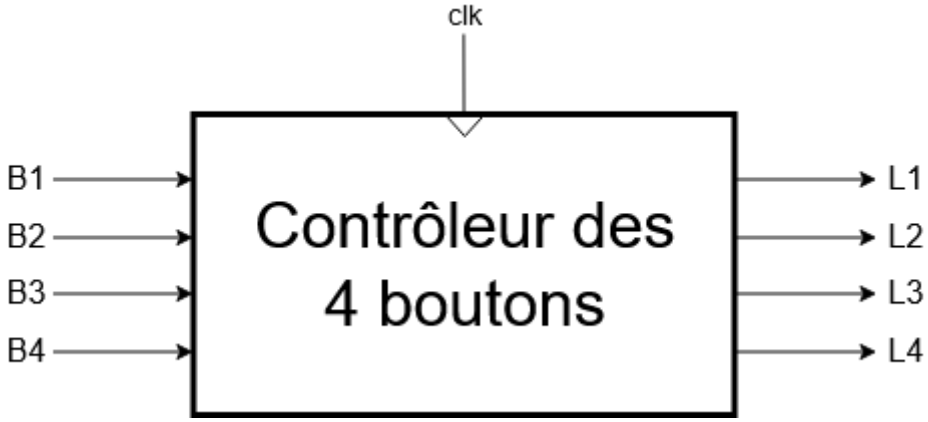
5) Pour que la machine puisse rendre l'argent, on doit créer un choix de boisson avec 0 DA, de cette manière si l'utilisateur le choisit il aura comme monnaie le retour de tous son argent. Ou même mieux, en peut ajouter un bouton Annuler stimulant l'appui sur les 2 boutons, le bouton choix 0 DA et le bouton accepter en même temps.

6) La construction du Circuit Séquentiel pour simuler les 4 boutons mécaniques en suivant la Machine de Moore :

La solution normal de faire la méthode à 7 étapes est valide, mais pour ce cas ce n'est pas la plus optimale, parce que si en analyse le circuit on aurais pratiquement 5 états pour l'automate (4 pour les 4 choix et 1 pour l'état activé/désactivé du choix) donc il en faut 3 cellules mémoires, plus 4 entrées de bouton, ça ferait 7 variables pour le circuit de l'État Suivant. Ça ferait beaucoup de combinaisons pour l'analyse du Circuit Combinatoire.

C'est pour cette raison qu'une autre méthode plus intuitive à base d'analyse serait plus appropriée pour ce genre de situation :

1. **Schéma global**



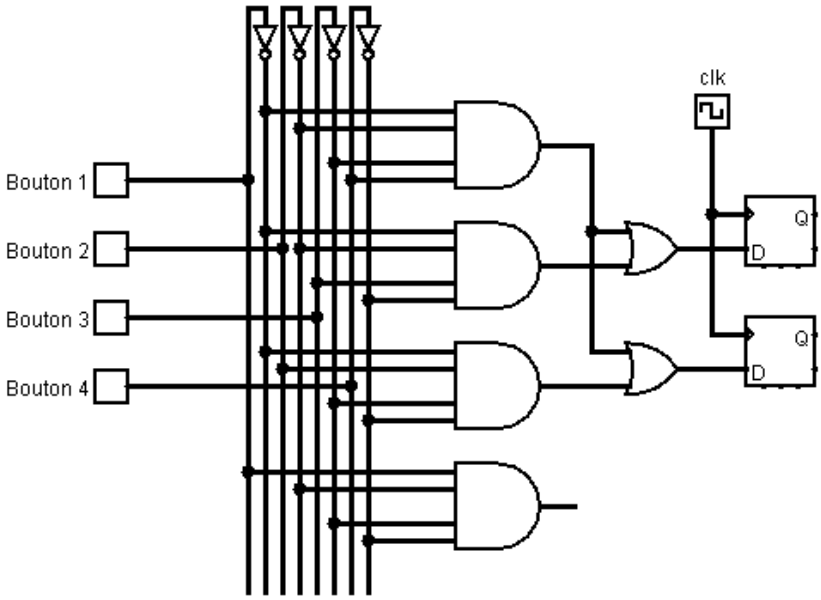
B1..B4 : Bouton1 jusqu'au Bouton 4
L1..L4 : LED1 jusqu'au LED4

Remarque : Les sorties sont des sortis vers les LED, mais qui sont aussi le même temps les signaux du choix de la boisson.

2. On peut prévoir 2 cellules mémoires dans notre circuit destinée à sauvegarder le choix de l'utilisateur, 00 pour le premier choix, 01 pour le deuxième,... et ainsi de suite. On pourrais imaginer une troisième cellule pour l'état activé ou désactivé du choix.

Remarque : On rappel juste qu'on a besoin de l'état activé/désactivé en rapport avec le fonctionnement des 4 boutons mécaniques, si l'utilisateur appuie 2 fois sur le même bouton il le désactive.

3. On peut imaginer intuitivement un Circuit Combinatoire pour nos 4 boutons d'entrées qui génère pour chaque bouton une valeur sur 2 bits. Le Bouton 1 génère la valeur 00, le Bouton 2 la valeur 01, et ainsi de suite. Ces valeurs serons à sauvegarder dans les 2 cellules mémoires.



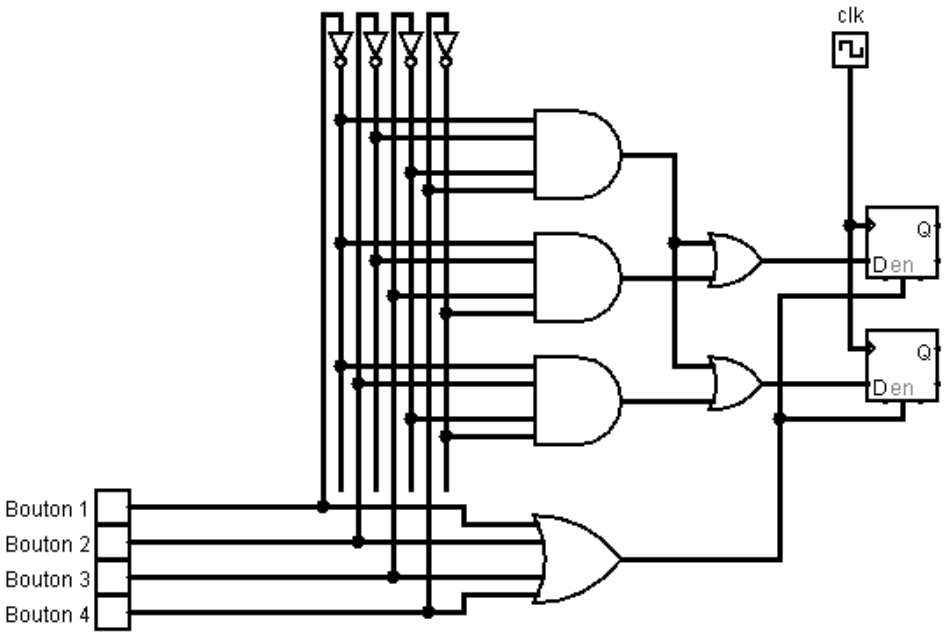
Remarque 1: Chaque porte AND s'active pour un bouton donnée, par exemple la première AND pour le bouton 1, ça formule est $B1 \cdot \overline{B2} \cdot \overline{B3} \cdot \overline{B4}$, la deuxième pour le bouton 2, ça formule est $\overline{B1} \cdot B2 \cdot \overline{B3} \cdot \overline{B4}$ et ainsi de suite.

Remarque 2: Les portes OR sont utilisées pour la formulation des 2 bits de la valeur du bouton. La porte OR en bas désigne le premier bit de la valeur, donc elle ne peut être à 1 que pour le bouton 2 (01) et le bouton 4 (11) comme sur le schéma et la porte en haut pour de deuxième bit, donc du bouton 3 (10) et du bouton 4 (11).

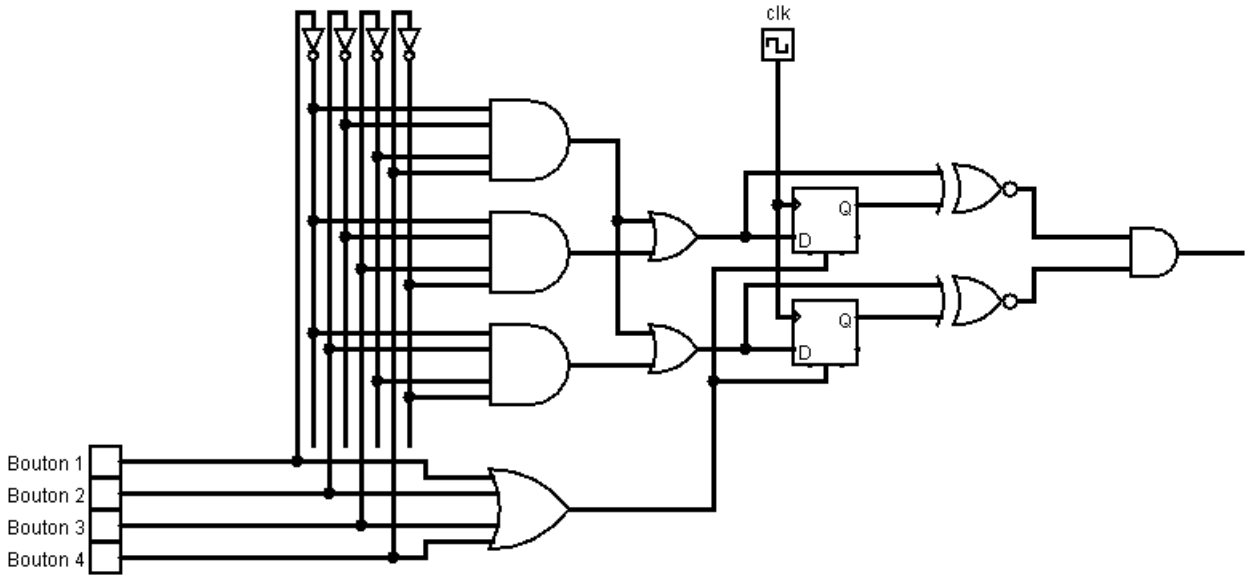
Remarque 3: On pourrait s'affranchir des portes AND et n'utiliser que les portes OR, la formule du bit 0 deviendrait B2+B4 et le bit 1 serait B3+B4, mais dans ce cas l'appui sur 2 boutons ou plus en même temps est considéré comme valide, alors qu'on réalité il ne l'est pas.

Remarque 4: L'appui sur le bouton 1 remet la valeur à 00, donc il n'a pas à être utilisé dans les portes OR et sont utilisation est inutile, il sera enlevé dans le future du logigramme.

4. Le circuit jusque là qui fait normalement sauvegarder le choix de l'utilisateur n'est pas fonctionnel, son défaut est qu'il perd la valeur sauvegarder lors du prochain front montant de l'horloge lorsque l'utilisateur relâche le bouton et n'appuie sur aucun bouton, pour remédier à ce problème il faut modifier le circuit de tel-sort que la modification à l'intérieur des cellules mémoires ne s'effectue que lorsque l'utilisateur appuie sur l'un des boutons, sinon la valeur reste inchangée. Pour cela on peut utiliser l'entrée de commande *enable* des D-FlipFlop à fin de limiter la modification que lors de l'appui d'un bouton. On peut voir sur le schéma en bas cette implémentation avec la porte OR sur les 4 boutons.

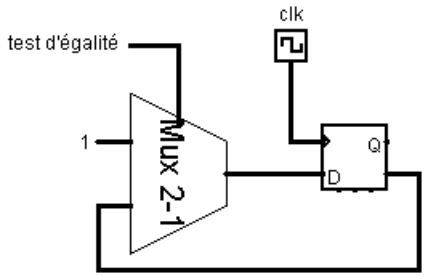


5. L'étape suivante est d'implémenter le mécanisme des boutons mécaniques que lorsque l'utilisateur appuie une deuxième fois sur le même choix ça désactive le choix. Pour cela on a besoin d'un Circuit Combinatoire qui compare la nouvelle valeur entrée de l'utilisateur avec la valeur dans les cellules mémoires. L'opération d'égalité logique (=) est toujours implémentée bit par bit en utilisant une porte XNOR, sachant que le XNOR retourne 1 pour les cas 00 et 11. On a besoin d'une porte AND pour tester des valeurs sur 2 bits comme on peut le voir sur le schéma en bas. Le circuit retourne 1 si la nouvelle entrée de l'utilisateur est égale à la valeur dans les cellules, sinon ça retourne 0.



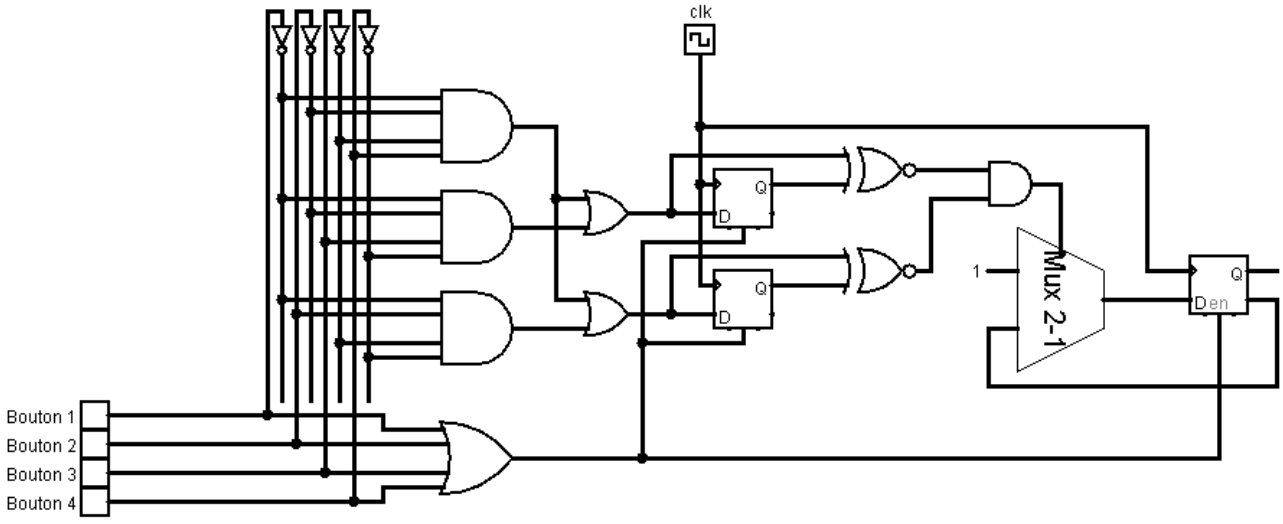
6. Après, on devrait implémenter la cellule mémoire qui suit si le choix de l'utilisateur est active ou pas. 2 cas de figures sont possible pour l'entrée à cette cellule mémoire, si la

valeur entrée par l'utilisateur est différente de celle sauvegarder, donc la cellule est mis-à-1, le choix est active, sinon la nouvelle valeur entrée est égale à celle sauvegarder, donc on doit inverser la valeur de la cellule, l'activer si elle est désactivée et la désactiver si elle est activée, on doit prendre la valeur $\overline{Q}$ de la cellule. Le signal du test d'égalité est fournie par le circuit précédent, en va l'utiliser dans un Multiplexeur 2-1 pour faire le choix entre les 2 cas expliqués précédemment. La figure en bas donne un aperçu sur le circuit.



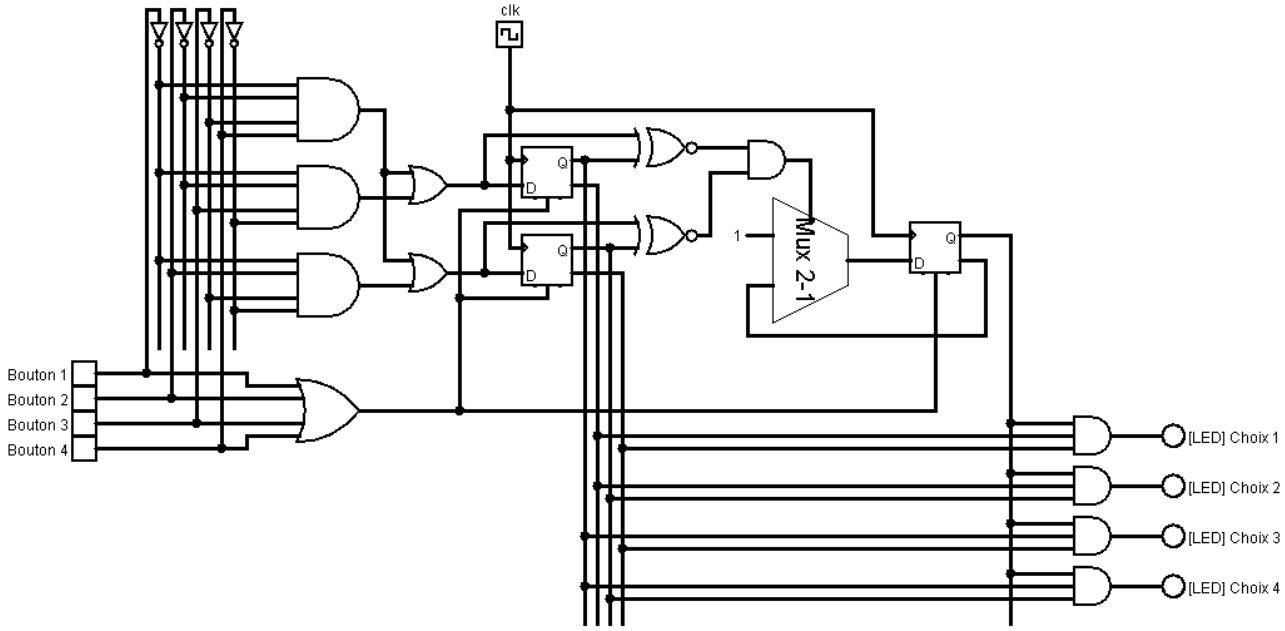
Remarque : Il est bien connu dans le domaine de la Conception Hardware que les opérations de tests ou de choix équivalentes au *Si* ou le *Case* dans la programmation sont implémentés par les Multiplexeurs. Le Mux de notre circuit est le choix entre les 2 cas.

7. Même précautions doivent êtres prises pour cette cellule que celles prises pour des 2 cellules du choix de l'utilisateur, la modification à l'intérieur de la cellule mémoire ne doit s'effectuer que lorsque l'utilisateur appuie sur l'un des boutons, sinon la valeur reste inchangée. Donc on doit user de l'entrée *enable* de la D-FlipFlop comme sur la figure en bas.



8. La dernière étape est de construire le Circuit Combinatoire de Sortie, c'est un circuit relativement facile à implémenter, les 4 sorties du circuit sont les 4 choix de l'utilisateur et un seul choix à un moment donné doit être active, le Circuit Combinatoire produit ses sorties en lisant le choix de l'utilisateur à partir des 2 cellule mémoires qui le mémorise,

ainsi une porte AND est utilisée pour chaque sortie, par exemple la sortie du choix 1 est formulée par l'expression $\overline{Q}_1 \cdot \overline{Q}_0$ des 2 cellules, si la valeur dans les 2 cellules est 00 le choix 1 est active. Les sorties des choix doivent aussi prendre en considération la cellule du choix activé/désactivé toujours en utilisant la porte AND, cette réalisation est démontré sur le schéma final en bas du logigramme de notre Circuit Séquentiel.



Remarque 1: Le bon fonctionnement de ce circuit suppose que l'utilisateur reste appuyé le long d'une période incluant un unique front montant, en réalité on ne peut pas savoir exactement combien de temps l'utilisateur doit rester appuyé, c'est variable d'un appui à l'autre, en plus il est probable que lors de l'appui des micro-coupure dans le signal surgissent appelés en Anglais *bounce* (rebond en Français) et qui peuvent poser problème. Ils existes des solutions concrètes pour gérer le signal d'entrée de l'appui d'un utilisateur qui ne serrons pas abordées ici.

Remarque 2: Cette manière de construire le Circuit Séquentiel est plus efficace et plus optimal que son équivalent en suivant la méthode à 7 étapes, mais ça demande une bonne analyse du circuit pour pouvoir le décomposer en plusieurs petits circuits, sans oublier que tous les circuits ne sont pas décomposables.