

Du transistor au processeur : Partie 1

Cet atelier a pour objectif de vous faire comprendre comment est conçu un processeur à partir de transistors. Il est probable que de terminer celui-ci vous demande environ 2 à 3h de travail en hors-présentiel (exercices 8 et 9). Finir ces exercices n'est pas obligatoire pour la suite du cours, mais réussir à finir est relativement satisfaisant puisque vous aurez réussi à concevoir un processeur 8 bits complet relativement avancé, ce qui vous permettra de mieux appréhender les notions déjà vus et à venir.

Exercice 1 : De transistor aux portes logiques (~ 40mn)

On suppose que nous sommes sur une île déserte et que nous devons reconstruire des ordinateurs à partir de zéro. Heureusement, nous avons un ami physicien sur l'île. Cet ami est capable d'assembler des circuits complexes construits à partir de transistors, l'élément de base de la plupart des circuits électroniques. Pour reconstruire un ordinateur, on considère des transistors de type MOS (Metal-Oxide-Semiconductor) car ce sont ceux aujourd'hui les plus couramment utilisés pour construire les micro-processeurs. Il existe deux types de transistors MOS : les transistors nMOS et les transistors pMOS. Dans les processeurs actuels, on trouve les deux types de transistors, on parle donc de technologie CMOS (Complementary Metal-Oxide-Semiconductor).

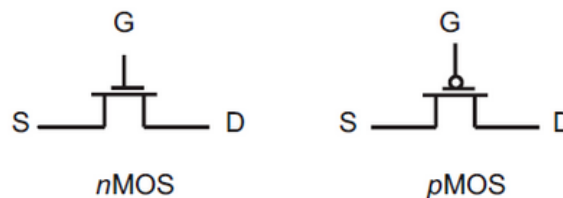


Figure 1 – Transistors nMOS et pMOS.

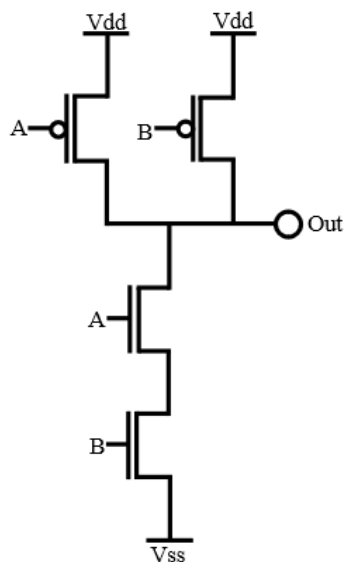
Comme présenté sur la figure 1, un transistor de type MOS possède trois connexions :

- la source, nommée S,
- la grille, nommée G.
- le drain, nommé D,

Un transistor MOS se comporte comme un interrupteur électrique. Pour un nMOS, lorsque la grille est alimentée, la source est connectée au drain, alors que pour un transistor pMOS, c'est l'inverse. Pour illustrer ce fonctionnement, si on imagine une rivière coulant de la source vers le drain, la grille agit comme une sorte de barrage empêchant l'eau de passer. Dans le cas d'un nMOS, lorsque la grille est alimentée, c'est comme si le barrage était ouvert : le courant passe de la source au drain. Dans le cas d'un pMOS, c'est l'inverse : lorsque la grille est alimentée, le barrage est fermé et le courant ne peut pas passer.

Partie 1 : Du transistor à la porte NAND

Les transistors permettent de construire des portes logiques qui sont à la base de l'électronique. À cette question, nous étudions notre première porte logique : la porte NAND (pour Not AND). L'avantage de la porte NAND, c'est que c'est une porte logique universelle : elle permet de construire des portes logiques NOT, OR, AND ou XOR comme vous le verrez dans l'exercice suivant.




Porte NAND		
		
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Figure 2.a – La porte logique NAND (credit wikipedia). Figure 2.b – Table de vérité de la porte logique NAND.

La construction de la porte logique NAND à partir de portes nMOS et pMOS est donnée en figure 2.a, alors que la table de vérité de cette porte est donnée en figure 2.b. Dans la figure 2.a, Vdd est le courant haut qui correspond, par convention, à la valeur vrai. Vss est le courant bas qui correspond, par convention, à la valeur faux. Les deux transistors du haut (avec un petit rond sur la grille) sont des pMOS qui empêchent le courant de passer entre la source et le drain lorsque la grille est alimentée, alors que les deux transistors du bas (sans petit rond sur la grille) sont des nMOS qui laissent donc le courant passer entre la source et le drain lorsque la grille est alimentée.

Question 1.a : validation

Vérifiez que le circuit présenté en figure 2.a met bien en œuvre une porte NAND.

Partie 2 : Prise en main du simulateur de circuit

Question 1.b : Connexion

Maintenant que nous avons notre première porte logique, nous pouvons l'utiliser pour construire les autres portes logiques. Avant de nous lancer dans cette mise en œuvre, nous prenons en main l'environnement de développement que nous utiliserons pour créer des circuits.

Dans un navigateur Web, connectez-vous au site <https://circuitverse.org/simulator>. Ce site permet de construire des circuits logiques de façon conviviale. De façon à pouvoir enregistrer vos travaux, il faut créer un compte. Pour créer un compte, cliquez sur Sign In. Ensuite, cliquez sur New user? Sign up. À partir de ce point, vous pouvez soit créer un nouveau compte, soit, si vous en avez un, utiliser votre compte google ou votre compte facebook pour vous authentifier.

Question 1.c : Configuration d'un projet

Maintenant que vous avez un compte, vous pouvez entrer dans le simulator. Cliquez sur le menu Simulator qui se trouve en haut de la fenêtre. Vous pouvez commencer à configurer votre projet. Donnez-lui un nom, par exemple YnovAsm, en modifiant la partie Project Properties se trouvant en bas à gauche de la fenêtre.

Question 1.d : Les entrées

Avant d'aller plus loin, nous allons placer un premier élément dans notre circuit. Allez dans l'onglet Input se trouvant à gauche de la fenêtre et sélectionnez la première icône (lorsque vous passez dessus

avec la souris, vous devriez voir `Input`). Déplacez votre souris sur l'espace de travail (la grille se trouvant à droite de la fenêtre), et cliquez quelque part. Félicitations, vous avez construit une `entrée binaire` ! Vous devriez constater que, lorsque vous cliquez sur votre entrée binaire, vous pouvez la faire passer de 0 à 1 et inversement.

Question 1.e : Enregistrement

Avant d'aller plus loin, nous allons vérifier que nous pouvons bien sauver notre travail. Dans le menu `Project`, cliquez sur `Save Online`, ce qui va créer une sauvegarde de votre projet hébergée sur le serveur (vous pouvez aussi enregistrer votre travail hors-ligne avec `Save Offline`). Pour vérifier que tout s'est bien passé, allez dans la configuration de votre compte représenté par votre nom d'utilisateur en haut à droite de la fenêtre, et cliquez sur `My circuits`. Vérifiez que vous retrouvez bien votre projet et cliquez sur `Launch` pour relancer le simulateur.

Question 1.f : Déplacer un élément

Avant de se lancer dans des conceptions plus intéressantes, nous allons juste apprendre à créer un circuit simple constitué d'une unique porte logique NAND. Vous trouverez cet élément dans l'onglet `Gates` à gauche de la fenêtre. Dans un premier temps, ajoutez une seconde entrée et une porte logique NAND. Vous devriez remarquer que, lorsque vous passez sur un élément dans l'onglet `Gates`, son nom s'affiche. Vous devriez aussi remarquer que, lorsque vous cliquez sur un élément placé dans l'espace de travail, vous pouvez le déplacer.

Remarque : Pour déplacer un élément, il faut cliquer à l'intérieur de l'élément et non sur les petits points verts se trouvant sur le bord et représentant les points de connexions.

Question 1.g : Constructions de fils (1/2)

Nous allons maintenant apprendre à connecter les deux entrées à la porte logique. Pour connecter deux éléments, il faut créer un fil électrique entre ces éléments. Les points de connexions sont représentés par les points verts se trouvant sur le bord d'un élément. Lorsque vous cliquez sur un point de connexion et que vous déplacez votre souris, vous commencez à créer un fil. Connectez les entrées à la porte NAND.

Question 1.h : Sorties

Pour finir, nous allons créer une sortie permettant de visualiser le résultat du calcul effectué par notre porte NAND. Dans l'onglet `Output` à gauche de l'écran, sélectionnez le premier élément (nommé `Output`). Cet élément, que nous appellerons une sortie binaire, permet d'afficher l'état d'un fil électrique (0 ou 1). Ajoutez une sortie binaire et connectez la sortie de la porte NAND à cette sortie. Vous devriez obtenir le circuit présenté en figure 3. Vérifiez, en modifiant les entrées, que votre circuit respecte bien la table de vérité de la porte NAND. Vous devriez remarquer qu'un fil alimenté (valeur 1) apparaît en vert clair alors qu'un fil non alimenté (valeur 0) apparaît en vert foncé.



Figure 3 – Un circuit NAND.

Question 1.i : Constructions de fils (2/2)

Pour continuer cette prise en main de l'outil, nous allons apprendre à mettre un circuit en court-circuit, c'est-à-dire que nous allons à la fois mettre les valeurs 0 et 1 sur un fil. Il ne faut bien sûr jamais mettre un circuit en court-circuit (comme déjà dit en cours), cette question n'a que pour but de vous montrer ce qui se passe, ce qui va vous permettre d'identifier les erreurs dans vos circuits.

Pour réaliser ce court-circuit, cliquer quelque part sur le fil reliant une des entrées à la porte NAND, ce qui permet de démarrer un second fil à partir de ce point. Construisez un fil qui contourne entièrement la porte (par le bas si vous partez du bas ou par le haut sinon). Pour cela, il faut construire plusieurs fils. Vous devriez obtenir le résultat présenté en figure 4. Vous devriez voir des messages s'afficher en bas de la fenêtre vous indiquant qu'il y a un problème. Vous devriez aussi voir que le fil reliant la porte à la sortie est à moitié vert foncé et à moitié vert clair, et que le simulator vous a ajouté de petits rond aux points de connexions qui permettent de remonter à l'origine du problème (sur le haut de la figure dans le cas de la figure 4).

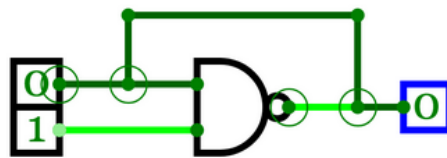


Figure 4 – Un circuit NAND en court-circuit.

Question 1.j : Suppression d'éléments

Nous allons réparer notre circuit en détruisant le court-circuit. Pour cela, cliquez sur un fil et appuyez sur la touche `backspace` (retour arrière se trouvant en haut à droite de votre clavier), ce qui détruit un élément. Supprimez tous les fils réalisant le court-circuit.

Question 1.k : Déplacer plusieurs éléments

Pour finir notre prise en main, nous apprenons à déplacer plusieurs éléments. Pour cela, il suffit d'effectuer une sélection en appuyant sur la touche `Shift` (remarquez que si vous n'appuyez pas sur la touche `Shift` alors que vous cliquez sur une zone sans éléments avant de déplacer la souris, vous déplacez simplement la grille). Ensuite, vous pouvez, en cliquant sur l'un des éléments sélectionné, déplacer l'ensemble de ces éléments.

Partie 3 : De la porte NAND aux autres portes logiques

Question 1.l :

Le but de cet exercice est de construire les portes logiques manquantes à partir de la porte logique NAND. Les tables de vérités et les icônes associées aux portes que vous devez construire sont présentées en figure 5.







Porte NOT			Porte AND			Porte OR			Porte NOR			Porte XOR			Porte XNOR		
																	
a	Out		a	b	Out	a	b	Out	a	b	Out	a	b	Out	a	b	Out
0	1		0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
1	0		0	1	0	0	1	1	0	1	0	0	1	1	0	1	0
			1	0	0	1	0	1	1	0	0	1	0	0	1	0	0
			1	1	1	1	1	1	1	1	0	1	1	0	1	1	1

Figure 5 – Tables de vérités des principales portes.

Commencez par nettoyer votre espace de travail en supprimant le circuit créé à l'exercice précédent. Pour cela, sélectionnez tout votre circuit et utilisez `backspace` pour le supprimer (vous pouvez aussi aller dans le menu `Project->Clear Project`).

Ensuite, en considérant que vous n'avez que la porte logique NAND, mettez en œuvre les autres portes. Au fur et à mesure que vous construisez de nouvelles portes, vous pouvez bien sûr les utiliser pour construire les suivantes.

Conseils : construire les portes dans cet ordre et reprendre vos notes de cours rappelées ci-dessous

- *NOT* - Remarquez que $NAND(A,A) = NOT(A)$.
- *AND* - Remarquez que $NOT(NAND(A,B)) = AND(A,B)$.
- *OR* - Remarquez que $OR(A,B) = NOT(AND(NOT(A),NOT(B))) = NAND(NOT(A),NOT(B))$.
- *NOR* - Remarquez que $NOR(A,B) = NOT(OR(A,B))$.
- *XOR* - Remarquez que $XOR(A,B)$ se comporte un peu comme $OR(A,B)$, sauf quand A et B valent 1 : dans ce cas, on veut avoir 0 comme sortie. On peut donc facilement voir que $XOR(A,B) = AND(A \text{ et } B \text{ différent de } 1, OR(A,B))$. Ensuite une porte logique mettant en œuvre " A et B différent de 1 donne 1" est simplement la porte *NAND*. Donc, $XOR(A,B) = AND(NAND(A,B), OR(A,B))$.
- *XNOR* - Remarquez que $XNOR(A,B) = NOT(XOR(A,B))$.

Félicitations, vous avez maintenant construit les éléments de base permettant de concevoir des ordinateurs à partir de transistors !

Exercice 2 : L'additionneur (~ 80mn)

À cette question, nous réalisons un circuit permettant d'additionner deux nombres encodés sur 8 bits.

Partie 1 : L'additionneur 1 bit

Question 2.a : Le demi-additionneur 1 bit

Comme première étape, nous allons réaliser un circuit permettant d'additionner deux bits (un bit est une valeur binaire, pouvant donc prendre les valeurs 0 ou 1). Exactement comme on le fait en décimal, il peut y avoir une retenue (carry en anglais), donc votre circuit doit produire un résultat sur deux bits :

- 0 + 0 donne 0 et je retiens 0,
- 0 + 1 donne 1 et je retiens 0,
- 1 + 0 donne 1 et je retiens 0,
- 1 + 1 donne 0 et je retiens 1 (comme 5 + 5 donne 0 et je retiens 1 en décimal),

a	b	s	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 6 – Table de vérité d'un demi-additionneur.

La figure vous donne la table de vérité de l'additionneur. Les entrées `a` et `b` sont les nombres à additionner alors que `s` est la somme et `cout` la retenue.

Commencer par supprimez tous vos anciens circuits. À partir de cette étape, ne détruisez plus jamais aucun des circuits que nous aurons réalisés : ils vont tous nous ressortir à un moment où un autre.

Ensuite, réalisez le circuit permettant d'effectuer une addition binaire.

Vous pouvez remarquer que :

- la somme `sum` réalise exactement un XOR entre `a` et `b`,
- on a retenue si et seulement si `a` et `b` valent 1.

Question 2.b : Réutilisation d'un circuit

De façon à additionner plusieurs bits, nous aurons besoin de réaliser plusieurs fois le circuit d'addition d'un unique bit. Notre logiciel de simulation permet de réutiliser des circuits qui ont déjà été mise en œuvre.

Pour rendre votre circuit réutilisable, commencez par donner un nom à vos entrées et à vos sorties. Pour cela, sélectionnez une des entrées. Dans la partie gauche de la fenêtre, vous devriez voir apparaître une zone `Properties`. À la ligne `Label`, donnez un nom à vos deux entrées (`a` et `b`). De la même façon, donnez des noms à vos deux sorties (`s` pour la somme et `cout` pour la retenue).

Ensuite, vous pouvez donner un nom à votre circuit. Pour cela, cliquez n'importe où sur la grille en dehors de tout élément. Ensuite, dans `Project Properties`, remplacez le nom actuel du circuit (ligne `Circuit` indiquant `Main`) en `half-add`. Le nom que vous voyez au-dessus de la grille devrait changer de façon adéquate. Enfin, cliquez sur `Edit layout` en bas à gauche de la fenêtre, ce qui vous permet de modifier l'apparence de votre circuit quand vous le réutilisez. Il est conseillé de laisser les entrées à gauche et les sorties à droite. Il est aussi conseillé de mettre `a` en haut et `b` en dessous, et `s` en haut et `cout` en haut bas. Enfin, nous vous conseillons de laisser deux carreaux entre les entrées ou entre les sorties. Après avoir donné l'apparence qui vous plaît à votre circuit, appuyez sur `Save` pour revenir sur le schéma du circuit. Pensez à sauvegarder votre travail (menu `Project->Save Online`).

Question 2.c : L'additionneur 1 bit

Comme on le fait en décimal, quand on effectue un calcul sur plusieurs digits, il faut non seulement additionner les deux digits courants, mais il faut aussi additionner ce résultat avec la retenue de l'étape précédente. Nous réalisons donc ce circuit à cette étape. Créez un nouveau circuit (menu `Circuit->New circuit`) que vous nommerez `1-add` (pour additionneur 1 bit).

Ajoutez 3 entrées nommées `a`, `b` et `cin` (retenue d'entrée provenant de l'étape précédente).

Ajoutez aussi 2 sorties nommées `s` pour la somme et `cout` pour la retenue de sortie.

Enfin, utilisez des circuits `half-add` (menu `Circuit->Insert SubCircuit`) pour que `s` soit égal à la somme de `a`, `b` et `cin`, et que `cout` soit égal à la retenue de cette somme.

Aide : Pour la somme, il suffit d'enchaîner deux `half-add`. Pour la retenue, il suffit que l'une des deux additions génère une retenue pour que le `1-add` génère une retenue. Le `cout` du circuit `1-add` est donc égal à un OU entre les sorties `cout` des deux `half-add`.

Partie 2 : Du bit à l'entier

Question 2.d : L'entrée 8 bits

De façon à pouvoir créer un additionneur capable d'effectuer des additions sur des nombres encodés sur 8 bits, nous commençons par créer une entrée 8 bits. Créez un nouveau circuit nommé `8-add`. Dans ce circuit, ajoutez une entrée. Si vous sélectionnez cette entrée, vous devriez voir apparaître ses propriétés. Donnez la valeur 8 à la propriété `BitWidth` de façon à ce que l'entrée soit une entrée avec 8 bits. Donnez le nom `a` à votre entrée.

Remarque : Si vous n'êtes pas à l'aise avec la représentation des nombres en binaire et en hexadécimal, cliquez sur l'aide.

On commence par étudier les nombres sur 4 bits. Ce sont des nombres écrits en base 2 qui possèdent 4 digits. La table de correspondance de la figure 7 vous présente la correspondance entre les 16 premiers nombres binaires, décimaux et hexadécimaux. L'hexadécimal est simplement une représentation des nombres en base 16 dans laquelle les nombres décimaux de 10 à 15 sont représentés par les lettres de A à F. Dans la table, les nombres suffixés par `b` sont des nombres représentés en binaire, alors que les nombres préfixés par `0x` sont des nombres représentés en hexadécimal.

Binaire	Décimal	Hexadécimal
0000b	0	0x0
0001b	1	0x1
0010b	2	0x2
0011b	3	0x3
0100b	4	0x4
0101b	5	0x5
0110b	6	0x6
0111b	7	0x7
1000b	8	0x8
1001b	9	0x9
1010b	10	0xA
1011b	11	0xB
1100b	12	0xC
1101b	13	0xD
1110b	14	0xE
1111b	15	0xF

Figure 7 – Correspondance binaire/décimale/hexadécimale.

On voit qu'on peut représenter un nombre sur 8 bits avec deux digits hexadécimaux. Par exemple, 10010011b est le nombre $1001b * 1000b + 0011 = 0x9 * 0x10 + 0x3 = 0x93$.

Question 2.e : Affichage d'un entier 8 bits

Le simulateur `circuitverse` permet d'afficher un nombre 4 bits facilement. Pour utiliser cette fonctionnalité, nous avons donc besoin de séparer notre nombre 8 bits en deux nombres sur 4 bits, l'un avec les 4 bits de poids fort, et l'autre avec les 4 bits de poids faibles. Dans l'onglet `Misc`, sélectionnez un `Splitter` (deuxième icône). Quand `circuitverse` vous demande la taille du nombre, mettez 8 puisque votre nombre fait 8 bits. Ensuite, il faut donner la décomposition de notre nombre. Nous voulons deux groupes de 4 bits, donc saisissez 4 4, ce qui construit 2 sorties de 4 bits. Connectez le `Splitter` à a. Ensuite, dans l'onglet `Output`, sélectionnez `HexDisplay` (5^{ème} icône) et placez deux `HexDisplay` à côté de votre `Splitter`. Connectez les 4 premiers bits du `Splitter` au second `HexDisplay` et les 4 derniers bits au premier `HexDisplay`. Vous devriez obtenir le circuit présenté en figure 8.

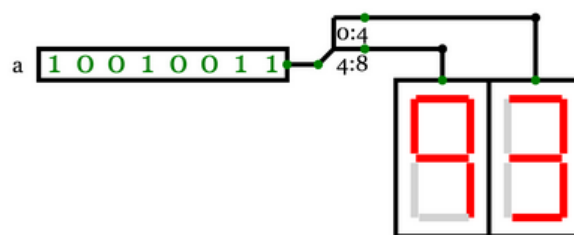


Figure 8 – Décomposition et affichage d'un nombre 8 bits.

Partie 3 : L'additionneur 8 bits

Question 2.f : Vous pouvez maintenant créer l'additionneur 8 bits. Pour cela, vous avez besoin des entrées suivantes :

- a sur 8 bits que vous avez déjà créé à la question précédente,
- b sur 8 bits que vous devez ajouter. Pensez à ajouter des `HexDisplay` et un `Splitter` pour voir la valeur de b,
- cin sur 1 bit que vous devez ajouter. Cette entrée donne la retenue initiale, qui est bien sûr égale à 0 pour une addition, mais vous verrez dans les questions suivantes comment exploiter cette entrée pour faire des soustractions.

Ensuite, utilisez des `Splitter` 8 vers 1 1 1 1 1 1 1 1 pour retrouver les bits qui composent a et b. Ajoutez aussi 8 circuits 1-add et connectez les de façon adéquate pour créer un additionneur 8 bits.

Pour construire la sortie, ajoutez un `Splitter` 8 vers 1 1 1 1 1 1 1 1 et changez sa direction vers `LEFT` dans ses propriétés. Connectez les sorties des additionneurs 1 bits au `Splitter`, et connectez de l'autre côté du `Splitter` une sortie nommée s dans laquelle vous spécifierez que `BitWidth` est égal à 8. Ajoutez aussi des `HexDisplay` et un `Splitter` 8 vers 4 4 pour voir le résultat. Enfin, ajoutez une sortie `cout` à votre circuit.

Remarque : Si gérer autant de fils vous semble trop difficile, n'hésitez pas à créer d'abord un circuit 4-add pour additionner des entiers 4 bits et à utiliser ce circuit pour construire le 8-add.

Aide : Pour les retenus, il faut les connecter en cascade. Le cin du premier 1-add reçoit la retenue d'entrée du circuit, et chaque cin suivant reçoit le cout de l'étape précédente.

Félicitations, vous venez de créer votre premier composant électronique complexe !

Exercice 3 : Le multiplexeur (~ 30mn)

Dans un circuit, il arrive souvent d'avoir besoin de sélectionner une entrée parmi N en fonction d'une valeur. Ce mécanisme peut par exemple être utilisé pour sélectionner les registres qui participent à une instruction. Dans cet exercice, nous mettons un œuvre des multiplexeurs dont le rôle est justement de sélectionner une entrée parmi N.

Question 3.a : 2x1-mux

On commence par construire un multiplexeur permettant de sélectionner une entrée parmi deux et dans lequel chaque entrée est un bit. Créez un nouveau circuit nommé 2x1-mux. Ce circuit doit posséder deux entrées binaires nommées x0 et x1, et un sélecteur nommé sel. Le circuit doit aussi posséder une sortie binaire nommée out.

Quand sel vaut 0, le circuit doit renvoyer x0 sur out, et quand sel vaut 1, le circuit doit renvoyer x1 sur out.

Aide : Le circuit renvoie 1 si et seulement si :

- x0 vaut vrai et sel vaut 0,
- ou x1 vaut vrai et sel vaut 1.

La formule logique du circuit est $OR(AND(x0, NOT(sel)), AND(x1, sel))$.

Question 3.b : 2x8-mux

Sur le modèle de l'additionneur 8 bits, construisez un nouveau circuit nommé 2x8-mux prenant en entrée 2 valeurs sur 8 bits nommées x0 et x1, et un sélecteur sur un bit. Le circuit doit renvoyer x0 si sel vaut 0 et x1 sinon.

Question 3.c : 2x3-mux

Sur le modèle de 2x8-mux, construisez un nouveau circuit nommé 2x3-mux prenant en entrée 2 valeurs sur 3 bits nommées x0 et x1, et un sélecteur sur un bit. Le circuit doit renvoyer x0 si sel vaut 0 et x1 sinon.

Remarque : Travailler sur des entiers encodés sur 3 bits peut vous sembler inattendu. Il faut comprendre qu'avec 3 bits, on peut encoder $2^3 = 8$ valeurs. Comme à terme, notre processeur possédera 8 registres, nous aurons donc besoin de manipuler des numéros de registre encodés sur 3 bits. Le 2x3-mux permettra donc de sélectionner des numéros de registres.

Question 3.d : 8x8-mux

À partir de 2x8-mux, construisez un circuit 8x8-mux permettant de sélectionner une entrée parmi 8, chacune des entrées faisant 8 bits.

Aide : De façon à pouvoir sélectionner une entrée parmi $8 = 2^3$, le sélectionneur doit faire 3 bits. Il faut utiliser un total de 7 2x8-mux. Les 4 premiers sont connectés directement aux entrées et le premier bit du sélecteur sert à sélectionner une des deux valeurs. Les deux suivants sont connectés aux quatre premiers 2x8-mux et le second bit du sélecteur permet de sélectionner une des deux valeurs. Le dernier est connecté aux deux 2x8-mux précédent et le dernier bit du sélecteur permet de sélectionner une des deux valeurs.

Bravo pour tous ceux qui sont arrivés à ce point. La suite de l'atelier vous sera transmise par mail.