LISV Laboratoire d'ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Architecture des systèmes numériques et informatiques
TP Logisim : De la logique combinatoire à l'UAL

Halim Djerroud

5 novembre 2025

Préparation

— Lancer le logiciel Logisim (ou Logisim-evolution)
— Créer un nouveau projet
— Se familiariser avec l'interface : bibliothèque de composants, zone de travail, outil de câblage

PARTIE 1 : Système de Vote Électronique (45 minutes)

Contexte

Vous devez concevoir un système de vote électronique pour un comité de 3 personnes (A, B, C). Une décision est adoptée si au moins 2 personnes votent OUI.

Question 1.1 : Spécifications

— **Entrées** : A, B, C (1 = vote OUI, 0 = vote NON)
— **Sortie** : S (1 = décision adoptée, 0 = décision rejetée)
Établir la table de vérité complète du système.

A	B	C	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Question 1.2 : Simplification par table de Karnaugh

Remplir la table de Karnaugh à 3 variables et simplifier l'expression de S.

A \ BC	00	01	11	10
0				
1				

Donner l'équation simplifiée sous forme de somme de produits : $S = ?$

Question 1.3 : Implémentation dans Logisim

1. Créer un nouveau circuit nommé "SystemeVote"
2. Placer 3 entrées (pins d'entrée) et les nommer A, B, C
3. Implémenter l'équation simplifiée en utilisant :
   — Des portes AND

— Des portes OR
— Des inverseurs (NOT) si nécessaire

4. Placer une sortie (pin de sortie) nommée S
5. Câbler l'ensemble

Conseil : Menu `Simulate -> Simulation Enabled` pour activer la simulation.

Question 1.4 : Test et validation

Tester votre circuit avec toutes les combinaisons possibles et vérifier qu'il correspond bien à la table de vérité.
Créer une table de vérité automatique : `Projet -> Analyze Circuit`

Question 1.5 : Extension (bonus)

Modifier le circuit pour un système à 4 votants où la décision nécessite une majorité stricte (au moins 3 votes OUI).

PARTIE 2 : Circuits Arithmétiques (45 minutes)

Exercice 2.1 : Demi-additionneur (10 min)

Un demi-additionneur additionne deux bits A et B et produit une somme S et une retenue C (Carry).

Question 2.1.1 : Conception

Établir la table de vérité :

A	B	S (Somme)	C (Retenue)
0	0		
0	1		
1	0		
1	1		

Déterminer les équations logiques :
— $S = ?$
— $C = ?$

Question 2.1.2 : Implémentation

Créer un sous-circuit "DemiAdd" dans Logisim avec 2 entrées (A, B) et 2 sorties (S, C).

Exercice 2.2 : Additionneur complet 1 bit (15 min)

Un additionneur complet prend en compte une retenue entrante C_{in}.

Question 2.2.1 : Table de vérité

Compléter la table de vérité :

A	B	C_{in}	S	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Question 2.2.2 : Simplification

Utiliser deux tables de Karnaugh (une pour S, une pour C_{out}) et simplifier.
Équations :
— $S = ?$
— $C_{out} = ?$

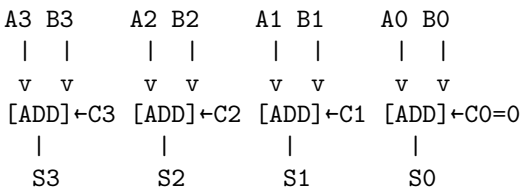
Question 2.2.3 : Construction avec demi-additionneurs

Un additionneur complet peut être construit avec 2 demi-additionneurs et 1 porte OR.
Créer le sous-circuit "AddComplet" en utilisant des instances de "DemiAdd".

Exercice 2.3 : Additionneur 4 bits (20 min)

Question 2.3.1 : Architecture

Concevoir un additionneur 4 bits en cascade qui additionne deux nombres A[3 :0] et B[3 :0].
Schéma de principe :



Question 2.3.2 : Implémentation

1. Créer un circuit "Add4bits"
2. Utiliser 4 instances de "AddComplet"
3. Utiliser des **Splitters** pour décomposer les bus 4 bits
4. Entrées : deux bus 4 bits A[3 :0] et B[3 :0]
5. Sorties : un bus 4 bits S[3 :0] et le bit de retenue C_{out}

Astuce Logisim : Utiliser `Wiring -> Splitter` pour diviser un bus en bits individuels.

Question 2.3.3 : Test

Tester votre additionneur avec :
— $5 + 3 = 8$ (0101 + 0011 = 1000)
— $15 + 1 = 16$ (1111 + 0001 = 10000, avec débordement)
— $12 + 7 = 19$ (1100 + 0111 = 10011)

PARTIE 3 : Unité Arithmétique et Logique - UAL (60 minutes)

Introduction

L'UAL (ALU en anglais) est le cœur d'un processeur. Elle effectue des opérations arithmétiques et logiques sur des données.
Vous allez concevoir une UAL 4 bits capable d'effectuer 8 opérations différentes.

Question 3.1 : Spécifications de l'UAL (5 min)

Entrées :
— A[3 :0] : premier opérande (4 bits)
— B[3 :0] : deuxième opérande (4 bits)
— OP[2 :0] : code d'opération (3 bits, permet 8 opérations)
Sorties :
— S[3 :0] : résultat (4 bits)
— Z : flag Zero (1 si résultat = 0)

— C : flag Carry (retenue pour additions)
— N : flag Négatif (bit de signe du résultat)

Table des opérations :

OP[2 :0]	Mnémonique	Opération
000	AND	$S = A \wedge B$
001	OR	$S = A \vee B$
010	XOR	$S = A \oplus B$
011	NOT	$S = \overline{A}$
100	ADD	$S = A + B$
101	SUB	$S = A - B$
110	INC	$S = A + 1$
111	SHL	$S = A << 1$ (décalage gauche)

Question 3.2 : Bloc des opérations logiques (15 min)

Créer un sous-circuit "UnitéLogique" qui implémente les 4 opérations logiques (AND, OR, XOR, NOT).

Question 3.2.1 : Architecture

Utiliser un multiplexeur 4->1 pour sélectionner l'opération avec les 2 bits de poids faible de OP.
Structure pour un bit :

```
A_i --+---> AND ---+
      |            |
B_i --+---> OR ----+---> MUX 4:1 ---> S_i
      |            |       ^
      +---> XOR ---+       |
      |            |    OP[1:0]
      +---> NOT ---+
         (ignoré)
```

Question 3.2.2 : Implémentation

1. Créer 4 instances de chaque porte logique (pour les 4 bits)
2. Utiliser un multiplexeur 4->1 de 4 bits (ou 4 multiplexeurs 1 bit)
3. Les entrées de sélection sont OP[1 :0]

Composant Logisim : `Plexers -> Multiplexer`

Question 3.3 : Bloc des opérations arithmétiques (20 min)

Question 3.3.1 : Addition (ADD)

Réutiliser votre additionneur 4 bits de la partie 2.

Question 3.3.2 : Soustraction (SUB)

La soustraction $A - B$ peut être réalisée en complément à 2 : $A + (\overline{B} + 1)$
Méthode :

1. Inverser tous les bits de B
2. Ajouter 1 en mettant $C_{in} = 1$ dans l'additionneur
3. Additionner avec A

Question 3.3.3 : Incrémentation (INC)

$A + 1$: utiliser l'additionneur avec B = 0000 et $C_{in} = 1$.

Question 3.3.4 : Décalage gauche (SHL)

Décaler A d'une position vers la gauche revient à :
— $S_3 = A_2$
— $S_2 = A_1$
— $S_1 = A_0$
— $S_0 = 0$
Utiliser un **Bit Shifter** ou câbler manuellement.

Question 3.3.5 : Multiplexage

Créer un sous-circuit "UnitéArithmétique" avec un multiplexeur pour sélectionner entre ADD, SUB, INC, SHL.

Question 3.4 : Assemblage de l'UAL complète (15 min)

Question 3.4.1 : Sélection finale

Utiliser le bit OP[2] pour sélectionner entre :
— OP[2] = 0 : sortie de l'unité logique
— OP[2] = 1 : sortie de l'unité arithmétique

Question 3.4.2 : Génération des flags

Implémenter les flags de sortie :
— **Flag Z (Zero)** : $Z = 1$ si $S = 0000$
  — Utiliser une porte NOR à 4 entrées
  — Ou : $Z = \overline{S_3 + S_2 + S_1 + S_0}$
— **Flag N (Négatif)** : $N = S_3$ (bit de poids fort)
— **Flag C (Carry)** : provient du C_{out} de l'additionneur (uniquement pour ADD/SUB)

Question 3.4.3 : Circuit principal

Créer le circuit "UAL_4bits" qui assemble :

1. L'unité logique
2. L'unité arithmétique
3. Le multiplexeur de sélection final (commandé par OP[2])
4. Les circuits de génération des flags

Question 3.5 : Tests et validation (5 min)

Tester votre UAL avec les cas suivants :

Test	A	B	OP	S attendu	Z	C
AND	1100	1010	000	1000	0	-
OR	1100	1010	001	1110	0	-
XOR	1111	1111	010	0000	1	-
NOT	1010	XXXX	011	0101	0	-
ADD	0101	0011	100	1000	0	0
ADD	1111	0001	100	0000	1	1
SUB	1000	0011	101	0101	0	-
INC	0110	XXXX	110	0111	0	0
SHL	0110	XXXX	111	1100	0	-

Question 3.6 : Améliorations (Bonus)

Si vous avez terminé en avance, proposez et implémentez :

1. Une opération de décalage à droite (SHR)
2. Une opération de comparaison (CMP) qui active des flags sans modifier S

3. Un flag V (oVerflow) pour détecter les débordements signés
4. Étendre l'UAL à 8 bits ou 16 bits