MATHÉMATIQUES

22

Algèbres de Boole

BTS | Mathématiques | Groupements B2, B3

Objectifs du chapitre

- Maîtriser les opérations et axiomes de l'algèbre de Boole.
- Construire et simplifier des fonctions booléennes (méthode algébrique et tableaux de Karnaugh).
- Identifier et utiliser les portes logiques élémentaires.
- Analyser des circuits combinatoires courants (additionneur, multiplexeur, décodeur).
- Comprendre le fonctionnement des bascules RS, JK, D, T dans les circuits séquentiels.

Situation professionnelle — Logique combinatoire dans un automatisme industriel

Un technicien en automatismes programme un automate programmable industriel (API) pour commander un convoyeur de tri de pièces. La sortie « moteur en marche » doit être activée lorsque : le capteur de présence pièce est actif **ET** la sécurité porte n'est pas déclenchée, **OU** lorsqu'un mode test forcé est sélectionné par l'opérateur. Formaliser ce cahier des charges en équation logique, simplifier l'expression et la câbler sur un circuit numérique sont des compétences directement issues de l'algèbre de Boole. Ce chapitre fournit toutes les bases nécessaires à ces tâches.

1. Définition d'une algèbre de Boole

DÉFINITION — ALGÈBRE DE BOOLE

Une **algèbre de Boole** est un ensemble $B = \{0, 1\}$ muni de trois opérations :

- **Addition logique** (OU, noté $+$ ou $\vee$) : $a + b$
- **Multiplication logique** (ET, noté $\cdot$ ou $\wedge$) : $a \cdot b$
- **Complément** (NON, noté $\bar{a}$ ou $\neg a$)

Ces opérations satisfont un ensemble d'axiomes garantissant la cohérence du calcul logique. En électronique numérique, 0 correspond à un niveau bas (environ 0 V) et 1 à un niveau haut (environ 3,3 V ou 5 V selon la technologie).

EXEMPLE — TABLES DES OPÉRATIONS DE BASE

OU $(+)$

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

ET $(\cdot)$

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

NON $(\bar{a})$

a	$\bar{a}$
0	1
1	0

La priorité des opérations est : NON > ET > OU (comme en arithmétique classique : négation > produit > somme).

2. Axiomes et théorèmes fondamentaux

PROPRIÉTÉS — AXIOMES DE L'ALGÈBRE DE BOOLE

Pour tout $a, b, c \in \{0, 1\}$:

Nom	ET $(\cdot)$	OU $(+)$
Éléments neutres	$a \cdot 1 = a$	$a + 0 = a$
Éléments absorbants	$a \cdot 0 = 0$	$a + 1 = 1$
Idempotence	$a \cdot a = a$	$a + a = a$
Complémentarité	$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$
Involution	$\bar{\bar{a}} = a$	
Commutativité	$a \cdot b = b \cdot a$	$a + b = b + a$
Associativité	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$	$(a + b) + c = a + (b + c)$
Distributivité	$a \cdot (b + c) = ab + ac$	$a + bc = (a + b)(a + c)$
Absorption	$a(a + b) = a$	$a + ab = a$

THÉORÈMES DE DE MORGAN

Les théorèmes de De Morgan permettent de distribuer une négation sur une opération logique :

$$\overline{a \cdot b} = \bar{a} + \bar{b} \qquad (\text{NON d'un ET} = \text{OU des NON})$$

$$\overline{a + b} = \bar{a} \cdot \bar{b} \qquad (\text{NON d'un OU} = \text{ET des NON})$$

Généralisation à n variables :

$$\overline{a_1 \cdot a_2 \cdots a_n} = \bar{a}_1 + \bar{a}_2 + \cdots + \bar{a}_n$$

$$\overline{a_1 + a_2 + \cdots + a_n} = \bar{a}_1 \cdot \bar{a}_2 \cdots \bar{a}_n$$

EXEMPLE — APPLICATION DE DE MORGAN

Simplifier $S = \overline{(A + B) \cdot C}$.

$$S = \overline{(A + B) \cdot C} = \overline{A + B} + \bar{C} = \bar{A}\bar{B} + \bar{C}$$

Puis vérifier les cas limites : si $A = 0$, $B = 0$, $C = 1$ alors $S = 1 \cdot 1 + 0 = 1$ ✓

ATTENTION — ERREURS FRÉQUENTES

- $\overline{A + B} \neq \bar{A} + \bar{B}$: il faut bien changer l'opérateur (De Morgan).
- La priorité d'opération : $AB + C$ se lit $(A \cdot B) + C$, pas $A \cdot (B + C)$.
- L'involution s'applique une variable à la fois : $\overline{\overline{A + B}} = A + B$, pas $\bar{A} + \bar{B}$.

3. Fonctions booléennes

DÉFINITION — FONCTION BOOLÉENNE

Une **fonction booléenne** de n variables est une application $f : \{0, 1\}^n \to \{0, 1\}$.

Pour n variables, la table de vérité comporte 2^n lignes.

Elle peut être représentée par :

- Une **table de vérité** exhaustive.
- Une **expression algébrique** (non unique).
- Ses **formes canoniques** : somme de mintermes (SOM/FND) ou produit de maxtermes (POM/FNC).

MÉTHODE — FORMES CANONIQUES (SOM ET POM)

Somme de mintermes — FND (Forme Normale Disjonctive)

Le *minterme* m_i est le produit de toutes les variables, chacune complémentée ou non selon le rang binaire i. On écrit la fonction comme somme des mintermes où $f = 1$.

Numérotation : le rang i est la valeur décimale de la combinaison des variables. Ex. $(A, B, C) = (1, 0, 1)$ donne $i = 5$, minterme $m_5 = A\bar{B}C$.

Produit de maxtermes — FNC (Forme Normale Conjonctive)

Le *maxterme* M_i est la somme de toutes les variables. On écrit la fonction comme produit des maxtermes où $f = 0$.

Ex. $(A, B, C) = (0, 1, 0)$ donne $i = 2$, maxterme $M_2 = A + \bar{B} + C$.

EXEMPLE — TABLE DE VÉRITÉ ET FORMES CANONIQUES (3 VARIABLES)

Soit $f(A, B, C)$ définie par la table :

N°	A	B	C	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

FND : $f = m_0 + m_2 + m_4 + m_5 + m_7 = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + ABC$

FNC : $f = M_1 \cdot M_3 \cdot M_6$

4. Simplification des fonctions booléennes

4.1 Simplification algébrique

MÉTHODE — SIMPLIFICATION ALGÉBRIQUE

1. Développer si nécessaire (distributivité).
2. Regrouper les termes partageant un sous-produit commun.
3. Appliquer absorption, idempotence, complémentarité.
4. Vérifier avec quelques lignes de la table de vérité.

EXEMPLE 1 — FACTORISATION ET ABSORPTION

Simplifier $f = AB + A\bar{B} + \bar{A}B$.

$$f = A(B + \bar{B}) + \bar{A}B = A \cdot 1 + \bar{A}B = A + \bar{A}B$$
$$= (A + \bar{A})(A + B) = 1 \cdot (A + B) = A + B$$

EXEMPLE 2 — UTILISATION DE DE MORGAN

Simplifier $g = \overline{A\bar{B}} \cdot \overline{\bar{A}B}$.

$$g = (\bar{A} + B)(A + \bar{B})$$
$$= \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B} = 0 + \bar{A}\bar{B} + AB + 0 = \bar{A}\bar{B} + AB$$

C'est la fonction XNOR : $g = \overline{A \oplus B}$.

4.2 Tableaux de Karnaugh (K-map)

DÉFINITION — TABLEAU DE KARNAUGH

Le **tableau de Karnaugh** est une grille représentant la table de vérité, où les cases voisines ne diffèrent que d'**un seul bit** (code Gray). Il est cyclique : les bords opposés sont adjacents.

Les groupes (implicants) de 2^k cases à 1 adjacentes correspondent à des termes simplifiés. Plus le groupe est grand, moins il contient de variables.

MÉTHODE — TABLEAU DE KARNAUGH À 2 VARIABLES

	$\bar{B}$ (B=0)	B (B=1)
$\bar{A}$ (A=0)	0	1
A (A=1)	1	1

Groupe de 2 : $\{(0,1),(1,1)\} \Rightarrow B$ | Groupe de 2 : $\{(1,0),(1,1)\} \Rightarrow A$

Résultat : $f = A + B$

MÉTHODE — TABLEAU DE KARNAUGH À 3 VARIABLES

On dispose A en ligne, B et C en colonnes en code Gray : `00, 01, 11, 10`.

A \ BC	00	01	11	10
$A = 0$	1	0	0	1
$A = 1$	1	1	1	1

Groupe de 4 (ligne bas + coins gauches/droits) : A | Groupe de 4 (colonne 00 + 10) : $\bar{C}$

Résultat : $f = A + \bar{C}$

MÉTHODE — TABLEAU DE KARNAUGH À 4 VARIABLES

Variables AB en lignes, CD en colonnes, chacun en code Gray.

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	1	0	0
10	1	1	0	1

Groupe de 4 (colonne 01 entière) : $\bar{C}D$ | Groupe de 4 (coins) : $\bar{B}\bar{D}$

Résultat : $f = \bar{C}D + \bar{B}\bar{D}$

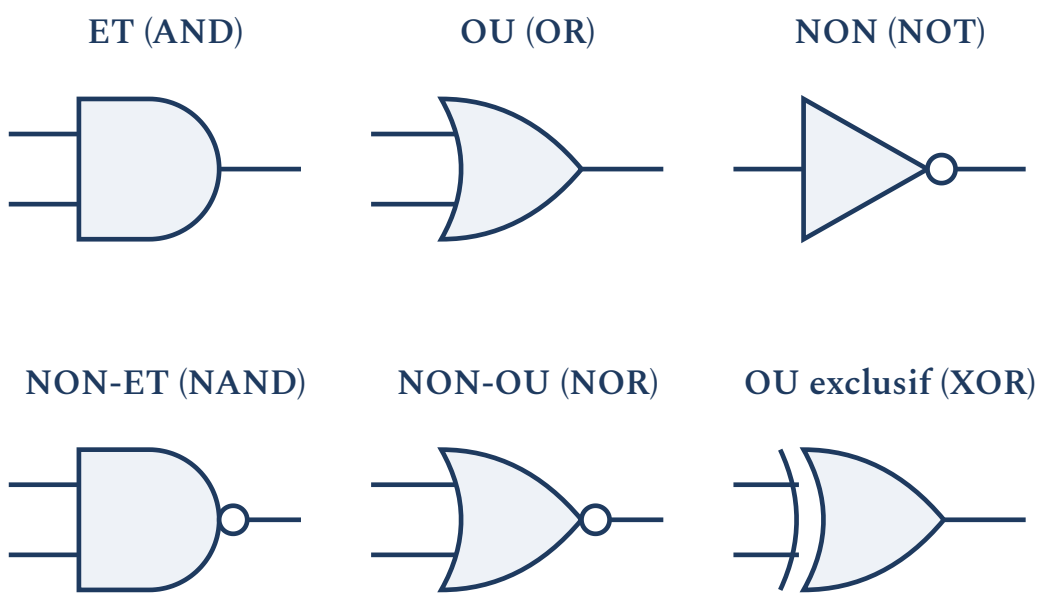
ATTENTION — RÈGLES DU K-MAP À RESPECTER

- Le tableau est **cyclique** : les quatre coins forment un groupe adjacent.
- Toujours chercher les groupes les **plus grands** d'abord.
- Un terme peut appartenir à **plusieurs groupes**.
- Chaque 1 doit être couvert par au moins un groupe.
- Les cases « indifférentes » (don't care, notées X ou –) peuvent être utilisées ou non selon ce qui agrandit le groupe.

5. Portes logiques

DÉFINITION — PORTE LOGIQUE

Une **porte logique** est un composant électronique réalisant une fonction booléenne élémentaire sur un ou plusieurs bits. Elle est caractérisée par son symbole normalisé (norme CEI 60617 ou norme américaine MIL) et sa table de vérité.



Symboles américains (norme MIL) des portes logiques. Le petit cercle en sortie marque la négation (NAND, NOR, NOT).

AND (ET)

$S = A \cdot B$

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

OR (OU)

$S = A + B$

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

NOT (NON)

$S = \bar{A}$

A	S
0	1
1	0

NAND (NON-ET)

$S = \overline{A \cdot B}$

NOR (NON-OU)

$S = \overline{A + B}$

XOR (OU exclusif)

$S = A \oplus B = A\bar{B} + \bar{A}B$

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

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

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

À retenir — Universalité des portes NAND et NOR

Les portes NAND et NOR sont dites **universelles** : toute fonction logique peut être réalisée en n'utilisant que l'un ou l'autre type.

Fonction	Avec NAND	Avec NOR
NOT A	$\overline{A \cdot A}$	$\overline{A + A}$
AND AB	$\overline{\overline{A \cdot B}}$	$\overline{\overline{A + A} + \overline{B + B}}$
OR $A + B$	$\overline{\bar{A} \cdot \bar{B}}$	$\overline{\overline{A + B}}$

6. Circuits combinatoires

DÉFINITION — CIRCUIT COMBINATOIRE

Un **circuit combinatoire** est un circuit logique dont les sorties dépendent uniquement des valeurs présentes des entrées, sans mémoire d'états passés. Il est entièrement caractérisé par sa table de vérité.

6.1 Demi-additionneur (Half-Adder)

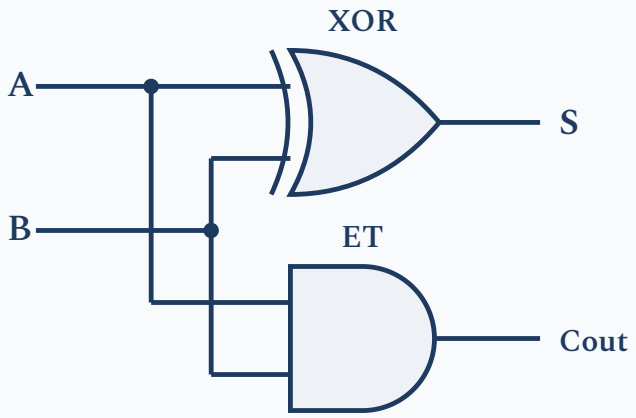
EXEMPLE — DEMI-ADDITIONNEUR

Additionne deux bits A et B. Sorties : somme S et retenue sortante C_{out}.

$$S = A \oplus B \qquad C_{out} = A \cdot B$$

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

Réalisation : 1 porte XOR + 1 porte AND.



Câblage du demi-additionneur : $S = A \oplus B$ (porte XOR), $C_{out} = A \cdot B$ (porte ET).

6.2 Additionneur complet (Full-Adder)

EXEMPLE — ADDITIONNEUR COMPLET

Additionne trois bits : données A, B et retenue entrante C_{in}.

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B) \cdot C_{in}$$

On peut cascader des additionneurs complets pour additionner des mots de n bits (additionneur ripple-carry).

6.3 Comparateur 1 bit

EXEMPLE — COMPARATEUR 1 BIT

Compare deux bits A et B. Trois sorties :

$$S_{=} \ : \ A = B \quad \Rightarrow \quad S_{=} = \overline{A \oplus B} = AB + \bar{A}\bar{B}$$

$$S_{>} \ : \ A > B \quad \Rightarrow \quad S_{>} = A\bar{B}$$

$$S_{<} \ : \ A < B \quad \Rightarrow \quad S_{<} = \bar{A}B$$

6.4 Multiplexeur (MUX)

DÉFINITION — MULTIPLEXEUR

Un **multiplexeur 2^n-vers-1** (MUX) aiguille une des 2^n entrées de données vers la sortie unique, selon le code présent sur les n entrées de sélection.

MUX 2-vers-1 : $S = \overline{SEL} \cdot I_0 + SEL \cdot I_1$

MUX 4-vers-1 : $S = \bar{S}_1\bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1\bar{S}_0 I_2 + S_1 S_0 I_3$

Applications : aiguillage de bus, conversion parallèle-série, réalisation de fonctions logiques quelconques.

6.5 Décodeur

DÉFINITION — DÉCODEUR

Un **décodeur n-vers-2^n** active exactement une sortie parmi 2^n selon le code binaire de n bits en entrée.

Décodeur 2-vers-4 (toutes sorties actives à 1) :

$$Y_0 = \bar{A}\bar{B}, \quad Y_1 = \bar{A}B, \quad Y_2 = A\bar{B}, \quad Y_3 = AB$$

Applications : sélection de mémoire, afficheur 7 segments, démultiplexage.

7. Circuits séquentiels — Bascules

DÉFINITION — CIRCUIT SÉQUENTIEL

Un **circuit séquentiel** possède une mémoire : ses sorties dépendent des entrées actuelles *et* de l'état passé. La **bascule** (flip-flop) est l'élément de mémorisation 1 bit fondamental.

On distingue :

- Circuits séquentiels **asynchrones** : les changements d'état se produisent dès que les entrées changent.
- Circuits séquentiels **synchrones** : les changements d'état sont cadencés par un signal d'horloge (CLK).

7.1 Bascule RS

Bascule RS (Set-Reset)

Entrées : S (Set = forcer $Q = 1$), R (Reset = forcer $Q = 0$). Sorties : Q et $\bar{Q}$.

S	R	Q_{n+1}	Commentaire
0	0	Q_n	Mémorisation (pas de changement)
0	1	0	Reset : mise à 0
1	0	1	Set : mise à 1
1	1	Interdit	Contradiction : Q = Q_bar = 1

> **État $S = R = 1$ interdit** : les deux sorties seraient à 1 simultanément, ce qui contredit $Q \neq \bar{Q}$. À proscrire absolument dans la conception.

7.2 Bascule JK

Bascule JK

Amélioration de la RS : l'état $J = K = 1$ est redéfini comme **basculation** (toggle) au lieu d'être interdit.

J	K	Q_{n+1}	Commentaire
0	0	Q_n	Mémorisation
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}_n$	Basculement (toggle)

Équation caractéristique : $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

7.3 Bascule D

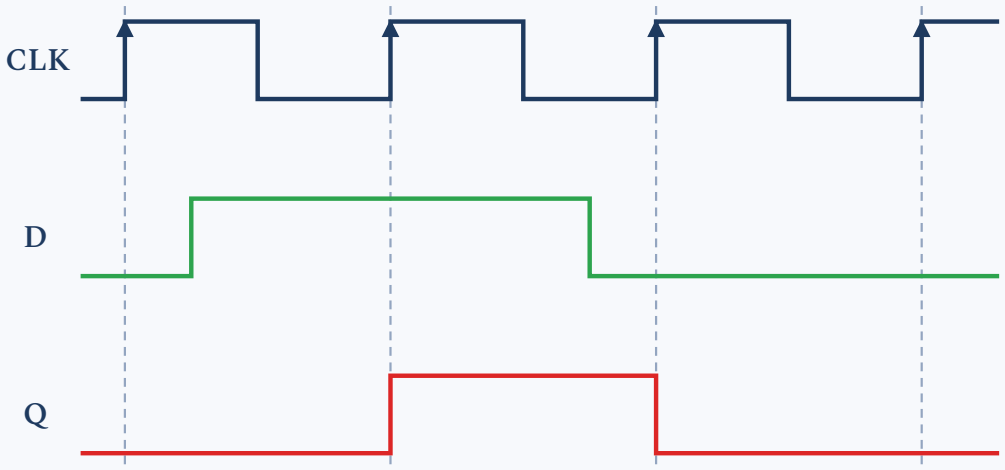
Bascule D (Data / Delay)

Une seule entrée de donnée D. Sur le front actif de l'horloge (généralement montant ↑), la sortie copie l'entrée.

$$Q_{n+1} = D$$

D	CLK	Q_{n+1}
0	↑	0
1	↑	1
X	bas ou haut (pas de front)	Q_n (mémorisation)

Applications : registres à décalage, mémoires vives SRAM, verrous.



Les flèches indiquent les fronts montants de CLK : Q recopie alors D.

7.4 Bascule T

Bascule T (Toggle)

Si $T = 1$, la sortie bascule à chaque front d'horloge actif. Si $T = 0$, mémorisation.

$$Q_{n+1} = T \oplus Q_n$$

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Applications : compteurs binaires, diviseurs de fréquence (une bascule T avec $T = 1$ divise la fréquence par 2).

À retenir — Récapitulatif des bascules

Bascule	Entrées	Équation	Particularité
RS	S, R	Set ou Reset de Q	S=R=1 interdit
JK	J, K	$Q_{n+1} = J\bar{Q} + \bar{K}Q$	J=K=1 → toggle
D	D	$Q_{n+1} = D$	Transparente/synchrone
T	T	$Q_{n+1} = T \oplus Q$	Diviseur de fréquence

Synthèse du chapitre 22

- **Opérations de base** : OU ($+$), ET ($\cdot$), NON ($\bar{}$) sur $\{0, 1\}$.
- **Axiomes clés** : idempotence $a \cdot a = a$, absorption $a + ab = a$, complémentarité $a + \bar{a} = 1$.
- **De Morgan** : $\overline{AB} = \bar{A} + \bar{B}$; $\overline{A + B} = \bar{A}\bar{B}$ — indispensable pour transformer NAND/NOR.
- **Karnaugh** : groupes de 2^k cases, tableau cyclique, don't care utilisables.
- **NAND et NOR** sont universels.
- **Circuits combinatoires** : additionneur, comparateur, MUX, décodeur — sorties = f(entrées actuelles).
- **Circuits séquentiels** : bascules RS, JK, D, T — sorties = f(entrées + état mémorisé).

MATHÉMATIQUES

22

Algèbres de Boole

Algèbres de Boole | BTS | Mathématiques

Rappels : notations $a+b$ (OU), $a\cdot b$ (ET), $\bar{a}$ (NON). Règles : $a+a=a$, $a\cdot a=a$, $a+1=1$, $a\cdot 0=0$, $a+\bar{a}=1$, $a\cdot\bar{a}=0$. Absorption : $a+a\cdot b=a$. De Morgan : $\overline{a\cdot b}=\bar{a}+\bar{b}$ et $\overline{a+b}=\bar{a}\cdot\bar{b}$.

Exercice 1 — Table de vérité

Dresse la table de vérité de $f=a\cdot b+\bar{a}$ (variables a,b).

Exercice 2 — Simplifications

Simplifie : **1.** $a+a\cdot b$ **2.** $a\cdot(a+b)$ **3.** $a+\bar{a}\cdot b$.

Exercice 3 — Lois de De Morgan

Exprime sans barre globale : **1.** $\overline{a\cdot b}$ **2.** $\overline{a+b}$.

Exercice 4 — Portes logiques

Donne la sortie d'une porte **NAND** (NON-ET) pour les entrées (a,b) = (1,1), (1,0), (0,0).

Exercice 5 — Circuit combinatoire (type BTS)

Un système d'alarme se déclenche ($S = 1$) si le contact a est ouvert OU si le détecteur b ET le détecteur c sont actifs.

1. Écris l'expression booléenne de S.

2. Donne S pour $a = 0, b = 1, c = 1$.

MATHÉMATIQUES

22

Algèbres de Boole

Algèbres de Boole | BTS | Mathématiques

🕐 **Durée :** 1 heure 🧮 **Calculatrice :** autorisée ✍️ **Barème :** 20 points

📄 **Documents :** non autorisés

Exercice 1 — Simplifications *(8 points)*

Simplifie : **1.** $b + b \cdot c$ *(2)* **2.** $x \cdot (x + y)$ *(2)* **3.** $a \cdot \bar{a} + b$ *(2)* **4.** $\overline{x + y}$ (De Morgan) *(2)*

Exercice 2 — Table de vérité *(6 points)*

Dresse la table de vérité de $f = \bar{a} \cdot b + a$ (variables a, b).

Exercice 3 — Circuit logique *(6 points)*

Un moteur démarre ($S = 1$) si le bouton marche m est appuyé ET que l'arrêt d'urgence u n'est PAS actionné.

1. Écris S en fonction de m et u. *(3 pts)*

2. Donne S pour $m = 1, u = 1$, puis $m = 1, u = 0$. *(3 pts)*