EPFL

Institute of Electrical and Microengineering

Biomedical and neuromorphic microelectronic systems

Systèmes logiques 2023, MT-BA3, SEL-BA3. EE110-EE207-2023_TP02S_v2.7.0.fm

v1.0	T. Kluter	EPFL 2016	v2.4.4	A. Schmid	2020.03.18	v2.5.0	A. Schmid	2021.02.26
v2.5.1	A. Schmid	2021.09.13	v2.6.0	A. Schmid	2022,01.20	v2.6.1	A. Schmid	2022,06.07
v2.7.0	A. Schmid	2022,12.05						

SYSTEMES LOGIQUES
TRAVAIL PRATIQUE TP02 - SOLUTIONS

2. TP02: DÉCODEUR 7-SEGMENTS

Le travail pratique TP02 voit la synthèse et l'optimisation de circuits combinatoires simples. Un décodeur 7-segments est développé. La simplification repose sur l'étude des tables de Karnaugh. Le décodeur est utilisé au contrôle d'affichages de nombres au formats décimal et hexadécimal.

2.1 DÉCODEUR 7-SEGMENTS

Un affichage 7-segments comprend sept (à huit) LEDs qui forment l'image d'un nombre au format hexadécimal. L'activation d'un segment au niveau logic-1 correspond au passage de courant dans la LED qui s'allume. Chaque segment doit être activé individuellement (Fig. Figure 2.1).

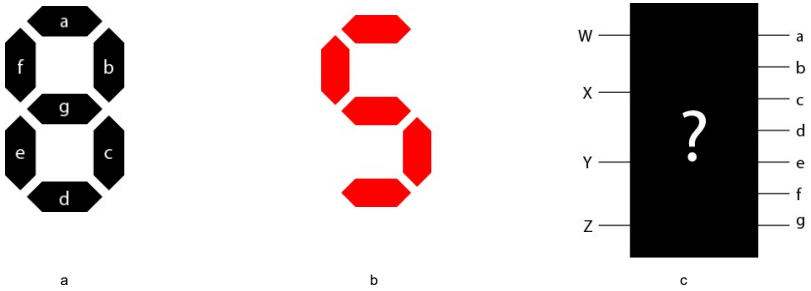


Figure 2.1: (a) Principe de l'affichage par segments individuels marqués a à g, (b) exemple de l'affichage du chiffre cinq, et (c) boîte noire représentant un décodeur transformant un nombre présenté dans un format binaire dans un contrôle d'affichage par segment.

Les nombres binaires, décimaux ou hexadécimaux ne sont pas représentés dans un format qui permette l'activation des segments. Un décodeur doit être développé qui fasse correspondre à chaque nombre un code qui active plusieurs segments. A chaque segment correspond donc une fonction logique qui décrit quels nombres activent ce segment, lorsque cette fonction est vraie. Un décodeur réalise l'ensemble des fonctions.

A ce stade, le décodeur est représenté par une boîte noire, dont les sous-fonctions seront réalisées suivant la table en Figure 2.2. W, X, Y, Z représentent un nombre présenté en entrée au format binaire ou hexadécimal, et a à g représentent chacun un signal binaire de contrôle de chaque segment respectif (voire Fig 2.1).

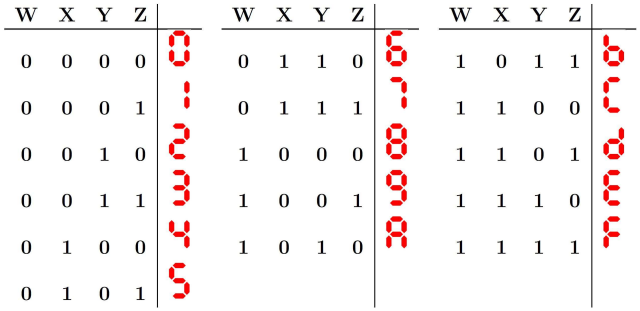


Figure 2.2: Table de décodage

Les tâches suivantes sont proposées.

- Dresser la table de vérité du décodeur (boîte noire) en Figure 2.3,

W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

Figure 2.3: Tabe de vérité du décodeur.

- Simplifier les fonctions logiques de contrôle de chaque segment au moyen de la méthode basée sur les tables de Karnaugh. Pour la synthèse des fonctions a, c, f et g, il est recommandé d'utiliser les maxterms. Pour la synthèse des fonctions b, d, et e, il est conseillé d'utiliser les minterms.
  - Premièrement, de la table de vérité, dérivez les expressions algébriques de chaque segment, puis

$$a=\overline{\overline{W}\cdot\overline{X}\cdot\overline{Y}\cdot Z+\overline{W}\cdot X\cdot\overline{Y}\cdot\overline{Z}+W\cdot\overline{X}\cdot Y\cdot Z+W\cdot X\cdot\overline{Y}\cdot Z}$$

$$b=\overline{W}\cdot\overline{X}+\overline{X}\cdot\overline{Z}+\overline{W}\cdot\overline{Y}\cdot\overline{Z}+\overline{W}\cdot Y\cdot Z+W\cdot\overline{Y}\cdot Z$$

$$c=\overline{W\cdot X\cdot\overline{Z}+W\cdot X\cdot Y+\overline{W}\cdot\overline{X}\cdot Y\cdot\overline{Z}}$$

$$d=W\cdot\overline{Y}+X\cdot Y\cdot\overline{Z}+\overline{X}\cdot Y\cdot Z+\overline{W}\cdot\overline{X}\cdot\overline{Z}+X\cdot\overline{Y}\cdot Z$$

$$e=W\cdot X+W\cdot Y+\overline{X}\cdot\overline{Z}+Y\cdot\overline{Z}$$

$$f=\overline{W\cdot X\cdot\overline{Y}\cdot Z+\overline{W}\cdot Y\cdot Z+\overline{W}\cdot\overline{X}\cdot Y+\overline{W}\cdot\overline{X}\cdot Z}$$

$$g=\overline{\overline{W}\cdot\overline{X}\cdot\overline{Y}+W\cdot X\cdot\overline{Y}\cdot\overline{Z}+\overline{W}\cdot X\cdot Y\cdot Z}$$

  - deuxièmement, représentez chacune des expressions dans une table de Karnaugh en Figure 2.4, et effectuez leur simplification afin d'obtenir une expression algébrique simplifiée.

- Développer le circuit logique du contrôleur en utilisant logisim-evolution, et vérifier la fonctionnalité. Dans ce cas, n'utilisez que des portes logiques AND, OR et NOT. Développez chaque fonction de façon individuelle, puis dans un circuit de niveau hiérarchique supérieur, connectez toutes les fonctions à leur entrées (DIP Switch) et sorties (7-segment).

  Les solutions de cette question ainsi que le solutions schématiques des questions suivantes sont présentées dans le fichier logisim-evolution TP02.circ disponible sur le site Moodle.
- Réaliser la fonction a en n'utilisant que des portes logiques NOR. A cette fin, modifiez l'expression algébrique de la fonction, puis développez et vérifiez le circuit dans logisim-evolution.

  Manipulation algébrique:

$$a=\overline{\overline{W}\cdot\overline{X}\cdot\overline{Y}\cdot Z+\overline{W}\cdot X\cdot\overline{Y}\cdot\overline{Z}+W\cdot\overline{X}\cdot Y\cdot Z+W\cdot X\cdot\overline{Y}\cdot Z}$$

$$=\overline{\overline{\overline{W}\cdot\overline{X}\cdot\overline{Y}\cdot Z}+\overline{\overline{W}\cdot X\cdot\overline{Y}\cdot\overline{Z}}+\overline{W\cdot\overline{X}\cdot Y\cdot Z}+\overline{W\cdot X\cdot\overline{Y}\cdot Z}}$$

$$=\overline{\overline{W+X+Y+\overline{Z}}+\overline{W+\overline{X}+Y+Z}+\overline{\overline{W}+X+\overline{Y}+\overline{Z}}+\overline{\overline{W}+\overline{X}+Y+\overline{Z}}}$$

  Implémentation de la négation:

$$\overline{A+A}\Leftrightarrow\overline{A+0}\Leftrightarrow\overline{A}$$

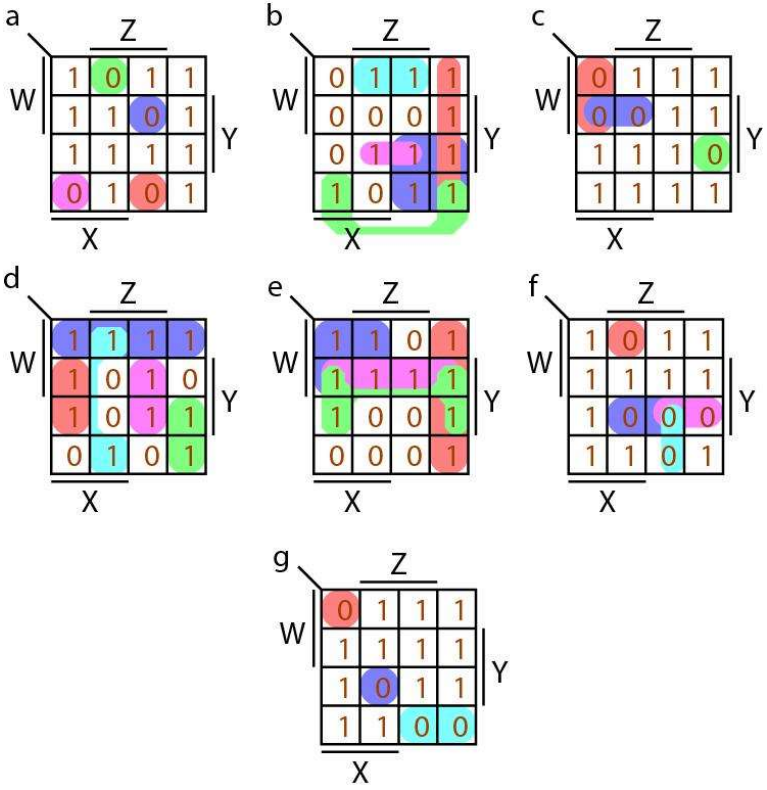
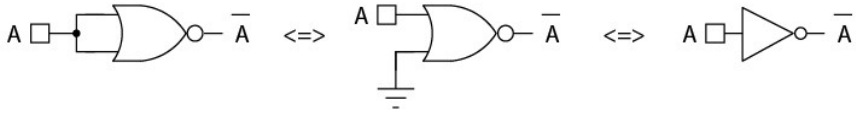


Figure 2.4: Tables de Karnaugh.



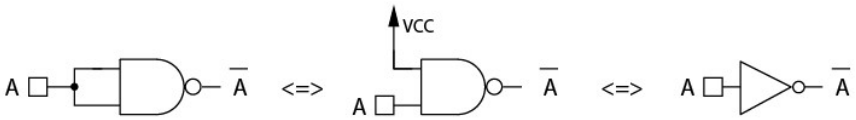
- Réaliser la fonction b en n'utilisant que des portes logiques NAND. A cette fin, modifiez l'expression algébrique de la fonction, puis développez et vérifiez le circuit dans logisim-evolution.

Manipulation algébrique:

$$\begin{aligned} b &= \overline{W} \cdot \overline{X} + \overline{X} \cdot \overline{Z} + \overline{W} \cdot \overline{Y} \cdot \overline{Z} + \overline{W} \cdot Y \cdot Z + W \cdot \overline{Y} \cdot Z \\ &= \overline{\overline{\overline{W} \cdot \overline{X} + \overline{X} \cdot \overline{Z} + \overline{W} \cdot \overline{Y} \cdot \overline{Z} + \overline{W} \cdot Y \cdot Z + W \cdot \overline{Y} \cdot Z}} \\ &= \overline{\overline{\overline{W} \cdot \overline{X}} \cdot \overline{\overline{X} \cdot \overline{Z}} \cdot \overline{\overline{W} \cdot \overline{Y} \cdot \overline{Z}} \cdot \overline{\overline{W} \cdot Y \cdot Z} \cdot \overline{W \cdot \overline{Y} \cdot Z}} \end{aligned}$$

Implémentation de la négation:

$$\overline{A \cdot A} \Leftrightarrow \overline{A \cdot 1} \Leftrightarrow \overline{A}$$



- Réaliser la fonction c en n'utilisant que des portes logiques OR et XOR. A cette fin, modifiez l'expression algébrique de la fonction, puis développez et vérifiez le circuit dans logisim-evolution.

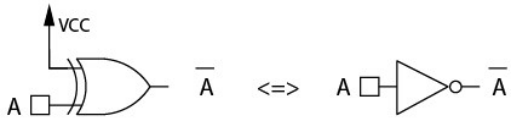
Manipulation algébrique:

$$\begin{aligned} c &= \overline{W \cdot X \cdot \overline{Z} + W \cdot X \cdot Y + \overline{W} \cdot \overline{X} \cdot Y \cdot \overline{Z}} \\ &= \overline{\overline{\overline{W \cdot X \cdot \overline{Z}} + \overline{\overline{W \cdot X \cdot Y}} + \overline{\overline{\overline{W} \cdot \overline{X} \cdot Y \cdot \overline{Z}}}}} \\ &= \overline{\overline{\overline{W} + \overline{X} + Z} + \overline{\overline{W} + \overline{X} + \overline{Y}} + \overline{W + X + \overline{Y} + Z}} \end{aligned}$$

Le XOR doit être réservé à l'implémentation de la négation. Il ne fait pas partie des quatre formulation algébriques équivalentes (sommes des produits, produits de sommes, produits, sommes des termes).

Implémentation de la négation:

$$A \oplus 1 \Leftrightarrow \overline{A}$$



- Réaliser la fonction e dans sa forme canonique conjonctive (produit des sommes). A cette fin, modifiez l'expression algébrique de la fonction, puis développez et vérifiez le circuit dans logisim-evolution.

Manipulation algébrique:

$$\begin{aligned} e &= W \cdot X + W \cdot Y + \overline{X} \cdot \overline{Z} + Y \cdot \overline{Z} \\ &= \overline{\overline{W \cdot X + W \cdot Y + \overline{X} \cdot \overline{Z} + Y \cdot \overline{Z}}} \\ &= \overline{(\overline{W} + \overline{X}) \cdot (\overline{W} + \overline{Y}) \cdot (X + Z) \cdot (\overline{Y} + Z)} \end{aligned}$$

2.2 CARTE DE10-LITE

Portez le contrôleur sur le système-cible et confirmer le contrôle correct d'un des affichages 7-segments. Si un DIP Switch est utilisé dans logisim-evolution de taille inférieure à la taille du DIP Switch disponible, alors le synthetiseur permettra de choisir les affectation de chaque switch individuel.

2.3 SYNTHÈSE AUTOMATIQUE

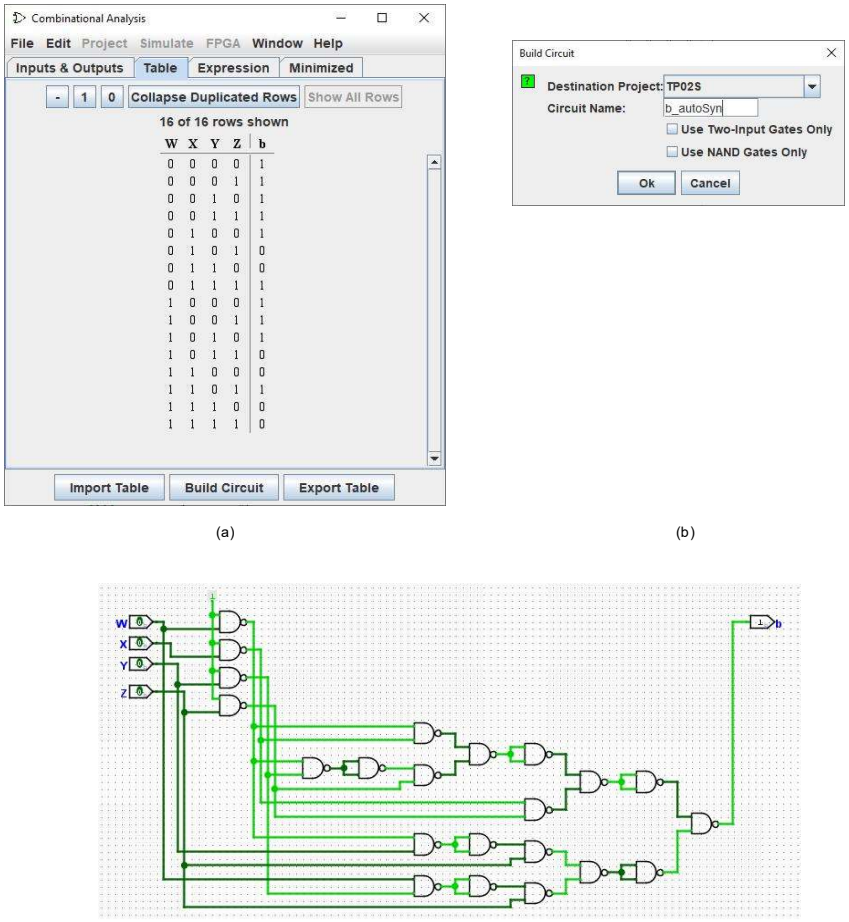
Il est indispensable de maîtriser le développement des circuits sous différentes formes à partir des expressions algébriques, les techniques de synthèse et de simplification au moyen de tables de Karnaugh, et la manipulation des équations permettant d'en modifier la réalisation.

Toutes ces techniques ont un aspect systématique qui en permet l'intégration algorithmique dans des softwares. logisim-evolution permet la synthèse automatique, sur la base d'expressions ou de table de Karnaugh. Utilisez la version **v3.7.2-all** de logisim-evolution. Dans le menu Window, le sous-menu Combinational Analysis permet d'accéder à la fonction utilisée dans la suite. Utiliser le tutoriel accessible par Help→Tutorial à la Section Combinational Analysis.

Les tâches suivantes sont proposées

- Regénérer les circuits implémentant les fonctions de contrôle des segments au moyen du module d'analyse combinatoire de logisim-evolution, par exemple

- Réaliser la fonction b en n'utilisant que des portes logiques NAND. A cette fin, entrez la table de vérité de la fonction, Figure 2.5(a), et faîtes effectuer la synthèse logique (Build Circuit) , Figure 2.5(b); a cette étape, indiquez la contrainte du choix de porte logique , Figure 2.5(c),.

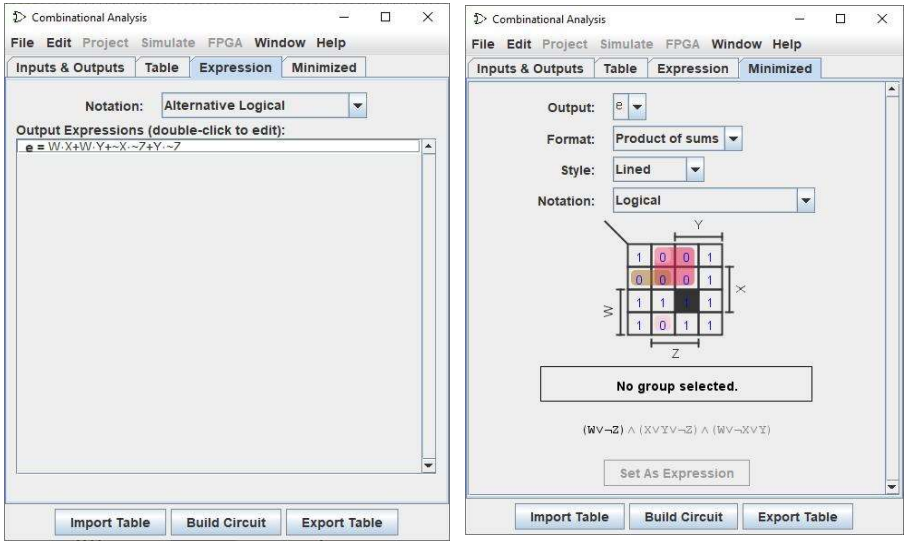


(a) (b)

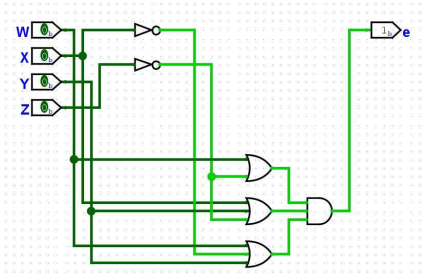
(c)

Figure 2.5: Etapes de synthèse automatique par table de vérité.

- Réaliser la fonction e dans sa forme canonique conjonctive (produit des sommes). A cette fin, entrez l'expression algébrique (non-simplifiée) de la fonction, Figure 2.6(a), puis sélectionnez la représentation en Product of Sums et faîtes effectuer la synthèse logique.



(a) (b)



(c)

Figure 2.6: Etapes de synthèse automatique par expression algébrique,forme conjonctive.

2.4 REMARQUE FINALE

Les valeurs de WXYZ situées dans l'intervalle entre 0000 et 1001 permettent de contrôler l'affichage 7-segments pour toutes les valeurs numériques dans l'intervalle 0 à 9. Ainsi, les valeurs décimale (base 10) peuvent être affichées.

Le code nommé Binary-Coded Decimal (BCD) bénéficie de cette propriété et est principalement utilisé pour le contrôle de plusieurs affichages 7-segments. Chaque chiffre décimal est codé par un nombre binaire, et est affiché sur un affichage 7-segment. Un décodeur est nécessaire pour chaque chiffre et donc pour chaque affichage. Par exemple, des affichage d'horloges numériques sont réalisées par ce moyen.