Conception Et Simulation De L’Unité De Traitement Du Microprocesseur Z-80 En Technologie SOI

Promoteurs:
Prof. Dr. J. D. Legat
Prof. Dr. J. P. Colinge

Mémoire présenté par
Ayadi Kamel

En vue de l’obtention de grade de
Master of Applied Sciences
Spécialisation en circuits intégrés

Septembre 1994
Je tiens à remercier Prof. J. D. Legat et Prof. J. P. Colinge pour les conseils qu’ils m’ont fourni dans l’élaboration de ce travail.

Je suis également reconnaissant envers Monsieur P. Lenoble pour les nombreuses informations qu’il m’a fourni et qui sont avérées bien utile.

**ABSTRACT**

We present here a design for the Data-Path of the microprocessor Z-80 using the SOI technology of CMOS transistors 3-µm of the “Laboratory of Microelectronics”. The Data-Path is full-custom designed. We have used CADENCE, “DFWII” version, for the design and layout, and Hspice for the simulation. All are mounted and run on Unix. The circuits and the combinational blocs are designed cell-by-cell and simulated with a 25 MHz clock frequency. Big part of the Data-Path is realized and tested.

**ABRÈGE**

Le sujet de ce travail est la conception de l’unité de traitement du Microprocesseur Z-80 en utilisant la technologie SOI des transistors CMOS 3-µm du “Laboratoire de Microélectroniques”. L’unité de traitement est conçue avec la méthode “Full-custom design“. Nous avons utilisé CADENCE, version “DFWII”, pour la conception et le dessin de layout, et Hspice pour la simulation. Tout est monté et turne sur Unix. Les circuits et les blocs combinatoires ont été conçus cellule par cellule et validés par simulation à une fréquence de 25 MHz. Une grande partie de l’unité de traitement est réalisée et testée.
# TABLE DES MATIÈRES

**INTRODUCTION** ................................................................................................................. 1  
  a) Transistor CMOS en SOI .......................................................................................... 2  
  b) L’unité de traitement du µP Z80 ............................................................. 5  
  c) Outils et méthode de conception suivie .................................................. 7  
  d) Conclusion ...................................................................................................... 7  

**CHAPITRE 1** ...................................................................................................................... 8  
**ARCHITECTURE** .................................................................................................................. 9  
  1.1) MNÉMONIQUES ................................................................................................. 9  
     1.1-a) Les instructions ..................................................................................... 9  
     1.1-b) Architecture ........................................................................................ 10  
  1.2) Bloc 1 .............................................................................................................. 10  
     1.2-a) Niveau 2 .............................................................................................. 10  
     1.2-b) Niveau 1 .............................................................................................. 11  
  1.3) Bloc 2 .............................................................................................................. 14  
     1.3-a) Niveau 2 Organisation des registres ............................................ 14  
     1.3-b) Niveau 1 .............................................................................................. 14  
  1.4) Bloc 3 .............................................................................................................. 15  
  1.5) Transistors de passage ............................................................................... 16  
  1.6) Conclusion .................................................................................................. 17  

**CHAPITRE 2** ...................................................................................................................... 18  
**LES CELLULES** .................................................................................................................. 19  
  2.1) LE REGISTRE ................................................................................................. 19  
     2.1-a) Description ......................................................................................... 19  
     2.1-b) Choix des portes logiques ............................................................ 19  
     2.1-c) Layout .................................................................................................. 20  
     2.1-d) Simulation .......................................................................................... 22  
     2.1-e) Conclusion ........................................................................................ 24  
  2.2) L'INCREMENTEUR ...................................................................................... 26  
     2.2-a) Programme en ASM ......................................................................... 26  
     2.2-b) Impliquant premiers ......................................................................... 27  
     2.2-c) Choix des portes logiques ............................................................. 27  
     2.2-d) Layout .................................................................................................. 29  
     2.2-e) Simulation .......................................................................................... 30  
     2.2-f) Conclusion .......................................................................................... 37  
  2.3) L'ALU .............................................................................................................. 37  
     2.3-a) Cellule NOT ........................................................................................ 37  
     2.3-b) FULL-ADDER ........................................................................................ 39  
     2.3-c) Cellule OR ............................................................................................ 47  
     2.3-d) Cellule AND ........................................................................................ 49  
     2.3-e) Cellule XOR ........................................................................................ 51  
     2.3-f) Sous-bloc ROTATE-SHIFT ................................................................. 53  
     2.3-g) Cellule SRT .......................................................................................... 61  
     2.3-h) Conclusion .......................................................................................... 63
CHAPITRE 3 ................................................................................................................................. 64

LES BLOCS ..................................................................................................................................... 65

3.1) BLOC 1 ..................................................................................................................................... 65
   3.1-a) Les registres ..................................................................................................................... 65
   3.1-b) L’ALU ............................................................................................................................. 65
   3.1-c) Dessin de layout du bloc 1 ............................................................................................. 66
   3.1-d) Simulation ...................................................................................................................... 69
   3.1-e) Résultat .......................................................................................................................... 71

3.2) BLOC 2 ..................................................................................................................................... 71
   3.2-a) Layout ............................................................................................................................ 72
   3.2-b) Simulation ...................................................................................................................... 74
   3.2-c) Résultat .......................................................................................................................... 76

3.3) BLOC 3 ..................................................................................................................................... 76
   3.3-a) Layout ............................................................................................................................ 76
   3.3-b) Simulation ...................................................................................................................... 78
   3.3-c) Résultat .......................................................................................................................... 82

3.4) "MACRO-CELL" de toute l’unité ............................................................................................. 82
   3.4-a) Description .................................................................................................................... 83
   3.4-b) Layout ............................................................................................................................ 84
   3.4-c) Simulation ...................................................................................................................... 84
   3.4-d) Résultat .......................................................................................................................... 84

CONCLUSION .................................................................................................................................. 87

OUVRAGES DE REFERENCE ........................................................................................................ 88

ANNEXES
   ANNEXE 1: Tables des instructions .......................................................................................... 89
   ANNEXE 2: Règles de conception ............................................................................................. 102
INTRODUCTION
INTRODUCTION

Le microprocesseur Z-80 a été réalisé la première fois avec la technologie des années 70 qui est seulement basée sur le transistor enrichit de type N, autrement dit le NMOS. De nos jours une telle technologie n'est plus utilisée car il y a d'autres plus avancées qui apparaissent et offrent des avantages au niveau physique du transistor.

Et d'ici vient l'idée de concevoir l'unité de traitement (Data-Path) de ce microprocesseur (µP) en CMOS et en utilisant la technologie SOI (3-µm) du laboratoire. Cette technologie est basée sur l'isolation des couches actives du substrat, et par suite la minimisation des effets qui dégradent le fonctionnement du transistor MOS.

a) Transistor CMOS en SOI

La figure 1.1 est un schéma simple des transistors NMOS et PMOS en SOI. L'une des avantages qu'on peut constater est l'apparition d'une capacité d'oxyde en série, $C_{ox2}$, de faible valeur et qui s'ajoute aux deux capacités habituelles du transistor Bulk.

La figure 1.2 montre le placement des différentes capacités des transistors NMOS Bulk et SOI, ainsi le placement de la capacité $C_{ox2}$ dont la valeur est très faible vue l'épaisseur d'oxyde ($\approx 4000$ Å). Les valeurs de ces capacités sont déduites de:

$$C_{ox} = \varepsilon_{ox}/t_{ox}$$
soit: \( C_{ox2} = 86,32 \, \mu \text{F/m}^2 \)
\( C_{ox1} = 645,42 \, \mu \text{F/m}^2 \)
\( C_{si1} = 1295 \, \mu \text{F/m}^2 \)

donc \( C_{ox2} \ll C_{ox1} \)

e t \( C_{ox2} \ll C_{si1} \)

La capacité arrière \( C_{si2} \) du Substrat P- dépend des tensions appliquées de dessus et de dessous, et elle est pratiquement dans le même ordre de grandeur que \( C_{ox2} \).

Soit \( C_d \cong C_{ox2}/2 \)

e t donc l'effet des capacités \( C_{si1} \) et \( C_{ox1} \) sera réduit.

![Diagram](image)

Figure 1.2: (a) transistor bulk NMOS; (b) transistor SOI NMOS.

en SOI \( \lambda \cong 1 + C_{ox1}/C_{ox2} \)

e t en bulk \( \lambda \cong X_{dmax} \)

avec

\[
X_{dmax} = \sqrt{\frac{4 \varepsilon_{Si} \Phi_f}{qN_a}}
\]

soit, en SOI \( \lambda = 1,08 \) est plus proche de l'unité qu'en bulk qui est de 1,15

Le simulateur Hspice demande la profondeur de déplétion \( x_j \) de la jonction métallurgique pour le transistor bulk (sous le drain), une estimation de \( x_j = 60 \, \text{nm} \) est faite pour le transistor SOI.

Les paramètres électroniques du modèle du transistor utilisé dans la conception des différents blocs logiques sont:
<table>
<thead>
<tr>
<th>LEVEL=3</th>
<th>NMOS</th>
<th>PMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>vt0</td>
<td>0,7</td>
<td>-0,357</td>
</tr>
<tr>
<td>phi</td>
<td>26,38</td>
<td>13,95</td>
</tr>
<tr>
<td>gamma</td>
<td>0,89</td>
<td>1,05</td>
</tr>
<tr>
<td>nsub</td>
<td>2,98e+18</td>
<td>5,31e+17</td>
</tr>
<tr>
<td>nfs</td>
<td>1,5e+12</td>
<td>8e+11</td>
</tr>
<tr>
<td>u0</td>
<td>614,22</td>
<td>228,34</td>
</tr>
<tr>
<td>vmax</td>
<td>199000</td>
<td>275500</td>
</tr>
<tr>
<td>xj</td>
<td>6e-08</td>
<td>8e-08</td>
</tr>
<tr>
<td>ld</td>
<td>4e-07</td>
<td>4e-07</td>
</tr>
<tr>
<td>theta</td>
<td>0,0224</td>
<td>0,107</td>
</tr>
<tr>
<td>tox</td>
<td>5,35e-08</td>
<td>5,35e-08</td>
</tr>
<tr>
<td>wd</td>
<td>3e-07</td>
<td>3e-07</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>js</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>jsw</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cj</td>
<td>4,3e-05</td>
<td>4,3e-05</td>
</tr>
<tr>
<td>cjsw</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mj</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mjsw</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rsh</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>delta</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kappa</td>
<td>0,001</td>
<td>7,5</td>
</tr>
<tr>
<td>eta</td>
<td>0,451</td>
<td>0,156</td>
</tr>
<tr>
<td>acm</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
b) L'unité de traitement du µP Z-80

Présente:

• un bus de données 8 bits et un bus d'adresses de 16 bits.
• une ALU de 8 bits qui effectue les opérations de décalage, de rotation, et des opérations logiques (AND, XOR, OR, NOT) et l'addition de deux bytes.
• ACT et TMP: deux registres qui injectent les deux vecteurs de 8 bits à l'ALU. Le TMP peut-être utiliser comme registre temporaire.

Le registre I: sert à contenir l'adresse indirecte concernant une interruption.

Le registre R: est pour le rafraîchissement des mémoires dynamiques.

• 4 registres d'adresses 16 bits: contiennent des adresses et qui sont reliées directement au bus d'adresses:

- PC: est un registre de 16 bits qui contient l'adresse de la prochaine instruction à exécuter.
- SP: est un registre de 16 bits qui contient l'adresse mémoire du sommet de la pile.
- IX et IY: sont utilisés lorsqu'il s'agit d'axée à un mot situé dans un bloc de données.
- Inc/Dec: incrémente ou décrémente l'adresse de 16 bits se trouvant dans le registre PC.
- Buffeur d'adresses: sert à injecter le vecteur d'adresses à l'entrée de l'incrémenteur.

La figure 3 englobe toutes les parties qui forment l'unité de traitement, sans tenir compte du registre IR (8 bits) qui sert à stoker l'instruction que la MPU vient de chercher de la mémoire pour que le µP l'exécute; aussi sans tenir compte du registre P qui est connecter au bus d'adresses externe.

L'unité de traitement est formée donc par plusieurs parties qui peuvent être conçues de plusieurs manières: suivent l'architecture, l'emplacement des différentes parties, et le choix des formes logiques qui réalisent les fonctions.
c) Outils et méthode de conception suivie

La conception des blocs combinatoires est faite en utilisant l'outil de conception CADENCE (version DFWII4.2.1a). La simulation et la dimension des transistors sont faite à partir de l'outil de simulation HSPICE (version H92).

La conception des blocs combinatoires suit la démarche suivante:

• description fonctionnelle du bloc, et s'il s'agit d'une combinatoire on décrit son fonctionnement en langage C qui facilite la séquence des étapes, on finit par la saisie définitive du bloc combinatoire et son emplacement par rapport aux autres blocs.

• ensuite, en descend au niveau des portes logiques qui réalisent la fonction que doit faire.

• l'avant dernière étape consiste à minimiser le nombre de transistors de chaque porte logique, car en CMOS on peut arriver à réaliser la fonction par plusieurs manières, et à chaque cas on choisit la forme la plus conforme.

• l'étape finale est le dessin de layout tout on minimisant la surface de silicium occupée par les transistors (densité d'intégration la plus élevée), et puis la simulation du dessin de l'extraction qui tient compte des formes géométriques des pistes de métal, de poly-siliciums utilisées...(résistances et capacités parasites induites entre différents niveaux: métal/substrat, métal/poly-silicium, poly-silicium/substrat...).

d) Conclusion

L'utilisation d'une technologie SOI ressente, un choix de l'architecture, et la dimension des transistors, convergent toutes à la réalisation d'une unité de traitement de données la plus rapide que possible.
CHAPITRE 1
ARCHITECTURE

La conception des différents blocs de l'unité de traitements exige une architecture bien définie qui répond au fonctionnement du microprocesseur.

1.1) MNÉMONIQUES

Des instructions de l'unité de traitement exigent la manière dont on doit répartir les blocs logiques (par exemple: on ne peut pas mettre le registre PC loin de l'incrémenteur d'adresses, aussi les registres ACT et TMP de l'ALU).

1.1-a) Les instructions

LD r, r': charger registre source r' dans registre destination r avec r et r' peuvent être A, B, C, D, E, H ou L. Ces registres seront placés les uns à côté des autres. Ils peuvent effectuer des opérations par pairs: (BC) (DE) et (HL), et ceci est justifié par des instructions comme:

- EX DE, HL: échanger le contenu de (HL) avec celui de (DE).
- EXX: échanger les contenus de (BC) (DE) et (HL) avec leurs auxiliaires(B'C') (D'E') et (H'L'). C'est l'adresse de retour à sauvegarder quand le µP sort du programme pour exécuter une routine ou un sous-programme.

Des opérations peuvent être effectuer avec l'ALU, l'instruction suivante fait l'addition entre un ou pair de registres avec l'accumulateur, exemple:

- ADD A, r
- ADD A,(HL)

Au niveau de l'incrémenteur deux types d'opérations peuvent s'effectuer:

- INC r: incrémente le registre r.
- DEC m: décrémente l'un des registres r ou la paire (HL), (IX+r) ou (IY+r).

L'un ou l'autre des instructions peut être effectuer après un nombre de cycles machines qui peut dépasser les 20 cycles, suivant la complexité.
Des tables complètes récapitulant toutes les instructions, que le microprocesseur effectue, sont en annexe 1.

1.1-b) Architecture

L'architecture proposée sépare l'unité de traitement en trois parties (blocs fonctionnels) qui se communiquent à travers des transistors de passage. La figure 6 est une vision de dessus de la position occupée par les 3 blocs.

![Diagramme de la répartition des différents blocs constituant l'unité de traitement](image)

La descente au niveau de chacun de ces blocs se fait en deux niveaux: le niveau 2 est celui de l'organisation des différents sous blocs fonctionnels, le niveau 1 est celui le plus bas où on organise les deux types de transistors au sein du sous bloc.

1.2) Bloc 1

Organisation de l'ALU, TMP et ACT:

1.2-a) Niveau 2

L'ALU est de 8 bits, constituée par les sous blocs suivants:

FULL-ADDER, AND, OR, XOR, NOT, SRT, ROTATE et CHIFT; de 8 bits chacun. Ils sont placés les uns à côté des autres de telle sorte que lorsque les 2 vecteurs d'entrées de 8 bits se présentent à l'entrée de l'ALU, l'un de ces sous blocs effectue l'opération et les autres se trouvent en mode transparent.

Le vecteur résultat sera injecté au bus de données par l'intermédiaire des transistors de passage CMOS.

Le registre TMP et l'accumulateur ACT sont des registres 8 bits, chaque cellule (de un bit) du registre est constituée par:
• un point mémoire.
• commande "rst" pour sa remise à zéro, utile quand le µP redémarre.
• et de plus pour l'ACT, une commande "set" pour sa remise à 1.

Un sous bloc SRT est ajouté à l'ALU qui est capable de mettre un bit de l'ACT à 1, à 0; ou le testé.

La figure 1.7 est une configuration du bloc 1 avec ces constituants et leurs emplacements.

1.2-b) Niveau 1

Étant donné que le bloc 1 est propre au bus de données 8 bits, les pistes de métal constituant le bus de données aussi bien l'alimentation et la masse sont placées horizontalement et perpendiculairement aux pistes de poly-siliciums. Ces derniers constituent les commandes, qui viennent de l'unité de commandes, pour activer et ordonner le fonctionnement des transistors par leurs grilles.

Dans une technologie de plusieurs couches de métal, l'alimentation et la masse occupent de préférence les couches les plus élevées. Alors que les signaux de hautes fréquences seront assignées aux couches les plus bas de faible effet capacitif.

Ce choix est basé sur le fait que, dans les procédures de fabrication des transistors, les lignes de métal se trouvent au plus haut niveau séparé du niveau des lignes de poly-silicium et des couches actives par des couches isolantes d'oxyde. Le contact entre les différents niveaux de piste se fait à travers des trous remplis par du tungstène par une technique CVD. La technologie utilisée ici est d'une seule couche de poly-silicium et d'une seule couche de métal.

La figure 1.8 met en évidence ces pistes aussi bien le placement des 2 types de transistors.


En général les transistors PMOS présentent leurs sources du côté de l'alimentation, car le PMOS transmet nettement la tension 5-V de la source au drain, alors que le NMOS ne transmet que

\[ v_{\text{dsat}} = \frac{v_{\text{gs}} - v_t}{\lambda} \]

qui est de 3,98 V pour le modèle SOI ainsi utilisé. Cette tension est à laquelle le transistor se sature. En contrepartie le NMOS transmet nettement la masse 0-V de la source au drain, alors que le PMOS ne fait passer que 0,65 V du drain à la source.

La figure 1.9 est une résumée des tensions de seuils du même modèle pour la conception.

Le calcul du courant fourni par chaque transistor est donné par:

\[ I_d = \frac{1}{2} \mu^* c_{\text{ox}} \frac{w}{l} \left\{(v_{\text{g}} - v_{\text{st}}) + (v_{\text{g}} - v_{\text{dt}})\right\} (v_d - v_s) \]

L'expression de la tension de seuil est limitée aux termes du premier ordre

\[ v_{\text{t}}(\text{ch}) = v_{\text{t}0} + \lambda v_{\text{ch}} \]
et à partir du diagramme

\[ I_d = \mu^* c_{ox} \left( \frac{W}{L} \right) S \]

avec S la surface hachurée visible dans la figure 1.9.

Une duplication 8 fois suivant la verticale de la figure 1.8 forme le bus de 8 bits, on plaçant la piste de masse de la cellule d'en bas sur celle de la cellule du haut, comme c'est indiqué dans la figure 1.10 où on a dupliqué le dessin de la figure 1.8 deux fois.

Figure 1.9: diagramme de tension de seuil explique le domaine de fonctionnement du modèle des transistors utilisés.

Figure 1.10: duplication deux fois la même cellule, le vss du milieu est commun.
1.3) Bloc 2

1.3-a) Niveau 2: Organisation des registres

On a des registres 16 bits IX, IY et des registres 8 bits (A,B,C,D,E,H,L) et leurs auxiliaires. L'idée est de concevoir tous en un seul modèle de registre de 16 bits, puisqu'on a des registres qui effectuent des opérations par pair.

Par exemple: le registre B occupe les 8 bits des plus forts poids et le registre C occupe les 8 bits des plus faibles poids, tout en laissant la souplesse d'accéder à chacun deux. Ceci nous amène à gagner sur la surface du silicium.

La communication du bloc 2 avec le bloc 1 se fait par l'intermédiaire de transistors de passage CMOS comme c'est indiqué par la figure 1.6. Ces registres sont placés en serials et se communiquent directement en bus d'adresses.

Figure 1.11 est une organisation de ces registres et aussi des transistors de passage qui relit le bus de données au bus d'adresses.

1.3-b) Niveau 1

Le bus d'adresses doit être de 16 bits, un placement des pistes de métal est un peut différent de la figure 1.8. On ajoute une piste de métal pour doubler le nombre de lignes du bus. Figure 1.12 est une copie de la figure 1.8 modifier par une piste de métal en bas, de plus.

La duplication 8 fois de la figure 1.12 suivant la verticale nous donne le bus de d'adresses 16 bits.

Figure 1.13 est une duplication 2 fois d'une même cellule.
Figure 1.12: placement des pistes pour une cellule liée au bus d'adresses 16 bits.

Figure 1.13: duplication deux fois de la figure 12. Le nombre de bus est doublé par rapport à la figure 10.

1.4) Bloc 3

Il est constitué par:

- deux blocs combinatoires de 8 bits chacun, qui effectuent ensemble l'incrémentation et la décrémentation d'un vecteur d'entrée 16 bits (suivant une commande 1 ou 0 de l'unité de commande).

- d'un registre PC 16 bits et d'un bufeur d'adresses 16 bits qui sert à passer l'adresses à l'entrée de l'incrémenteur et vers l'extrémité du bus d'adresses. Après que le calcul de la nouvelle adresse est fini, l'incrémenteur injecte l'adresse effectuée sur le
bus vers le registre PC.

Une organisation de ce bloc est résumée dans la figure 1.14.

Figure 1.14: bloc 3 où se trouve la rupture du bus au niveau des buffeurs et continue suivant les flèches.

1.5) Transistors de passage

Le choix des dimensions des transistors de passage, qui jouent un rôle important dans toute cette architecture, est fait par HSPICE. L'idée est de mettre un inverseur de dimension fixe qui va générer un signal sur le bus, en passant par les transistors de passage CMOS, et charge une capacité du bus de 0,1 pF. Cette valeur est une estimation de la capacité de la ligne en parallèle avec une capacité de grille d'un transistor se trouvant à l'autre extrémité de la ligne.

Le schéma logique du modèle est dans la figure 1.15.

Figure 1.15: schéma logique pour les dimensions des transistors de passage avec \(L_p=L_n=4 \, \mu m\).

On a pris trois différentes dimensions des largeurs de grilles des transistors de passage pour la simulation. Le but est de voir l'effet de la largeur de grille des transistors de passage sur le signal de sortie (au niveau de la capacité).

Les trois courbes OUT1, OUT2 et OUT3, obtenues avec des dimensions \(W_p/W_n\) des
transistors de passage CMOS 11/4, 30/12 et 40/15, sont générées par Hspice. Ces courbes justifient le fait que on ne doit pas prendre des transistors de passage CMOS de dimension inférieure à celle de l'inverseur (30/12) à fin de ne pas étrangler le signal que cet inverseur génère. Une augmentation au-delà de 30/12 ne fait qu'augmenter la résistance du canal. OUT2 est le signal le plus rapide à la montée qu'à la descente, donc les dimensions de ces transistors de passage seront prises identiques à celle de l'inverseur qui génère OUT2.

1.6) Conclusion

Le choix de cette architecture offre la souplesse de concevoir des différentes parties, constituant l'unité de traitement, en passant par le dessin d'une cellule d'un bit, et la duplication 8 fois suivant le vertical pour former un bloc combinatoire de 8 bits. La cellule est dupliquée 16 fois quant on a à former un bloc combinatoire de 16 bits.

De plus, il y a la possibilité de découper cette unité de traitement en différent blocs combinatoires qui peuvent effectuer des opérations séparément et en parallèle. Le contrôle et la circulation des données et des adresses sur les bus sont assurés par des transistors CMOS de passage.
CHAPITRE 2
LES CELLULES

2.1) LE REGISTRE

2.1-a) Description

Le registre est un élément important dans toute l'unité de traitement, il se trouve dans les trois blocs:

- bloc 1: a les registres 8 bits TMP et ACT.
- bloc 2: 12 registres 16 bits.
- bloc 3: registre PC 16 bits et buffeur d'adresses 16 bits.

Dans chacun de ces trois blocs, le point mémoire du registre reste le même. En fait, ce point mémoire est constitué par deux inverseurs qui se suivent et de dimensions croissantes, et d'un transistor NMOS pour le refroidissement par la commande "load" à fin de ne pas perdre l'information qui ne peut survivre que quelques ms. Ce temps est largement suffisant pour que ce point mémoire se refroidisse plusieurs fois; étant donné qu'on travaille dans le domaine des ns.

D'autres parties sont ajoutées au point mémoire suivant l'utilité:

- un NMOS de passage (Wn = 12 µm) commandé par "wr" pour écrire sur le registre à partir du bus.
- un NMOS (Wn = 12 µm) commandé par "rst" pour la remise à zéro du contenu du registre.
- deux transistors CMOS de passage (Wp = 30 µm et Wn = 12 µm) commandés par "rd" et "rd-bar" pour lire du registre vers le bus.
- un NMOS (Wn = 12 µm) commandé par "set" pour la remise à 1 du registre.

2.1-b) Choix des portes logiques

La figure 2.1 présente tous les différents registres de l'unité; le registre PC est identique aux registres du bloc 2.
Figure 2.1: schéma du registre PC et du buffeur d'adresses, 16 bits. Les dimensions des inverseurs et des transistors de passage sont des Wp et Wn; avec toujours Lp = Ln = 3 µm.

Figure 2.2: schéma des registres ACT et TMP, 8 bits. Les dimensions sont à la Figure 2.1.

Les dimensions des transistors de lecture et d'écriture sont basées sur la simulation faite au chapitre précédent.

Étant donnée que le registre peut avoir des données d'un autre registre, ces transistors NMOS et PMOS seront de même grandeur que l'inverseur du point mémoire (Wp = 30µm, Wn = 12µm), car cet inverseur génère le signal de sortie du registre.

2.1-c) Layout

Le dessin de layout du registre 16 bits est fait pour une seule cellule à deux pistes du bus, de sorte qu'on peut la dupliquer huit fois sans problème, tout en respectant les règles de conception imposées par le fabriquant. Les règles de conception sont dans annexe 2.

Pour compacter le dessin il faut serrer suivant l'axe des x, car ∆y reste constant pour toutes les cellules. Les dimensions géométriques pour une seule cellule sont: ∆x = 244,5 µm, ∆y = 106 µm
figure 2.3 : Layout d'une cellule du registre 16 bits
2.1-d) Simulation

A partir de CADENCE, la simulation du dessin de l'extraction est faite par HSPICE. En fait après le dessin et la vérification (DRC) des distances critiques imposées par le fabricant, CADENCE extrait le layout et génère un "netlist" qui met en évidence l'existence des transistors, des pistes de métal et de leurs dimensions, et des capacités parasites. On ajoute dans le fichier netlist les signaux de commandes et différents données qui semblent provenir de l'extérieur de l'unité de traitement. HSPICE utilise le "netlist" pour simulée le circuit, et génère les signaux de sorties.

Deux simulations ont été faites pour le registre avec un clock de fréquence \( f = 25 \text{ MHz} \) (soit une période de \( T = 40 \text{ ns} \)). Dans le fichier netlist, une capacité de \( c = 0.1 \text{ pf} \) est ajoutée au piste du bus

*Simulation N°1: écriture sur le registre*

On a initialisé le registre à zéro \( (a_0 = 0 \text{ V et } b_0 = 0 \text{ V}) \).

Le bus est à 1 \( (\text{bus} = 5 \text{ V}) \), à \( t = 0 \text{ s} \) on écrit sur le registre on activant "wr", jusqu'au temps \( \text{set} \) qui est de(20 ns).

Au deuxième cycle d'horloge on active "load", on rafraîchit son point mémoire, et au troisième cycle d'horloge on active "rst" pour la remise à zéro du contenu du registre.
Résultat: en 4 ns le registre change d'état (b0 = 5 V et a0 = 3,98 V).

*Simulation N°2:* écriture sur le bus

On a initialisé le bus à zéro (bus = 0 V). Le registre est à 1 (a0 = 3,98 V tension fixée dans le diagramme de tension de seuil) à t = 0 s on lit du registre en activant "rd" et "rd-bar".

Résultat: en 3 ns le registre charge le bus (bus = 5 V).
2.1-e) Conclusion

Avec une fréquence de \( f = 25 \text{ MHz} \) le registre fonctionne correctement.

Ce registre est typique car on peut l'utiliser comme registre ACT, on ajoutant un transistor NMOS pour la remise à 1. De plus il a un point mémoire conçu par deux inverseurs de dimensions croissantes pour amplifier le signal vers le bus sans risque d'être atténuer.

Théoriquement cet inverseur (\( W_p = 30 \mu \text{m}, W_n = 12 \mu \text{m} \)) est capable de charger une capacité de \( c = 0,1 \text{ pF} \) en \( \Delta t \):

\[
\Delta t = \frac{c \Delta V}{\lambda}
\]

avec

\[
I = \mu C_{ox} \frac{w (V_g - V_t)^2}{2\lambda}
\]
donc \[ \Delta t = 0,5 \text{ ns} \]
alors que le résultat de la simulation est de 3 ns:

![Diagram](image)

Figure 2.5: choix des signaux de commande qui sont définis par \( t_{\text{monté}} \) et \( t_{\text{descente}} \).

Le décalage, entre la valeur théorique \( (t = 0,5 \text{ ns}) \) et la valeur de la simulation \( (t = 3 \text{ ns}) \), est dû au temps de monter et de descendre du signal de commande "wr" et "rd" par la grille du transistor. Ce signal n'est pas brusque, et est estimé de 2 ns.

Cette estimation est faite en supposant que le signal, semble provenir de l'unité de commande, est généré par des inverseurs de même dimensions que les inverseurs conçus dans l'unité de traitement.

De ce fait, on injecte un signal parfait "in" (monté et descente verticale) à une suite de quatre inverseurs. Le signal de sortie "out" se stabilise à un temps de monter et de descente de \( t = 2 \text{ ns} \); il l'est déjà à partie du troisième inverseur d'après notre simulation.
2.2) L'INCREMENTEUR

Il s'agit d'un bloc combinatoire qui doit-être capable d'incrémenter ou de décrémenter suivant une commande "inc" de l'unité de commande.

Chaque cellule reçoit le bit d'entrée "in", la commande "inc" pour l'incrémentation ou la décrémentation, et une entrée "Cin" carry de la cellule précédente; et génère la sortie "out" résultat de l'opération, et son carry "Cout" pour la cellule suivante.

![Diagram of the incrementer](image)

Figure 2.6: entrées et sorties des signaux de la boîte noire de l'incrémenteur.

2.2-a) Programme en ASM

```
N1: if(inc) goto N2;
    else goto N3;
N2: if(cin) goto N4;
    else goto N5;
N3: if(cin) goto N6;
    else goto N5;
N4: out = inc; Cout = in;
N5: out = in; Cout = 0;
N6: out = in-bar;
```
2.2-b) Impliquant premiers

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\begin{align*}
\text{Cout} &= \text{Cin} \quad (\text{inc} \oplus \text{in}) \\
\text{out} &= \text{in} \quad \oplus \quad \text{Cin}
\end{align*}

Figure 2.7: impliquant premiers de l'incrementeur.

2.2-c) Choix des portes logiques

On a trois sortes de portes logiques XOR, XNOR et AND. Pour avoir le minimum de transistors par porte, on a utilisé le XOR \((\text{Cin}, \text{in})\) à 6 transistors, voir figure (2.8).

\[
\begin{align*}
\text{XOR} \quad (\text{Cin} \quad \text{in}) & \quad \text{XNOR} \quad (\text{inc} \quad \text{in}) \\
\text{AND} \quad (\text{x} \quad \text{Cin}) &
\end{align*}
\]

Figure 2.8: portes logiques de l'incrémenteur qui ne sont que des transistors de passage mais ils nécessitent des précautions pour que les signaux "Cout" et "out" ne s'atténuent pas.
La porte XOR\((\text{in}, \text{Cin})\) est souple point de vue utilisation. En faite, si on change la place de l'entrée "\text{in}" avec son complément, la porte devient XNOR\((\text{in}, \text{Cin})\).

Cependant, elle ne génère aucun signal, vu qu'elle est formée par des portes de passage NMOS et PMOS. Point de vue électrique, il y a un risque d'atténuation des signaux causer par des filtres RC qui règnent dans les canaux des transistors et les pistes de poly-silicium de grille.

Des précautions ont été prises tel que: les signaux de sorties des portes doivent être générer par des inverseurs de sorte que le signal "\text{out}" soit amplifié avant d'être mis sur le bus; aussi le signal "\text{Cout-bar}" doit être générer avant qu'il soit injecté à la cellule suivante sous forme de "\text{Cin-bar}". Ce signal "\text{Cin-bar}" est à son tour inversé pour entrer à la cellule sous forme de "\text{Cin}"; comme c'est indiqué sur la figure 2.9.

![Diagramme de la cellule XOR](image1)

Figure 2.9: succession des cellules; les inverseurs des sorties génèrent les signaux \text{out}_0 et \text{out}_1 et empêchent les états intermédiaires le long du temps de calcul.

La structure complète de la boîte et les dimensions des différents transistors sont tel que: à l'intérieur du bloc combinatoire, les transistors prennent les dimensions les plus petites \((\text{Wn}/\text{Ln} = 4\mu\text{m}/3\mu\text{m} \text{ et } \text{Wp}/\text{Lp} = 11\mu\text{m}/3\mu\text{m})\); avec toujours \(\text{Wp}/\text{Wn} \cong 2,7\)

La valeur du rapport \((\text{Wp}/\text{Wn})\) est liée aux équations des courants des transistors de
type N et P:

\[ I_n = \mu_n \text{Cox} \ (W/L)_n \ S \]
\[ I_p = \mu_p \text{Cox} \ (W/L)_p \ S \]

Une égalisation des deux équations, pour avoir le même courant qui coule dans les deux types de transistors, donne:

\[ \mu_n \ (W/L)_n = \mu_p \ (W/L)_p \]

donc \[ \mu_n/\mu_p = (W/L)_p/(W/L)_n \]

La mobilité est donnée par: \[ \mu_n(y) = \mu_{\text{max}} \left[ \frac{E_c}{E_{\text{eff}}(y)} \right]^c \]

avec \[ E_{\text{eff}}(y) > E_c \]

\( \mu_{\text{max}} \), \( E_c \) et \( c \) sont des paramètres dépendants des procédures de fabrication de l’oxyde de grilles et des diverses propriétés des transistors SOI.

\[ E_{\text{eff}} = E_{s1}(y) - \frac{Q_{\text{inv}1}(y)}{2\varepsilon_{\text{si}}} \]

avec

\[ E_{s1}(y) = \frac{Q_{s1}(y) - Q_{s2}(y)}{t_{\text{si}}} + \frac{qN_a t_{\text{si}}}{2\varepsilon_{\text{si}}} \]

De ces équations on peut tirer la conclusion: que la mobilité est liée à la concentration des impuretés \( N_a \), de l’épaisseur du film de silicium \( t_{\text{si}} \), et des phénomènes physiques et électriques (charges devant et arrière, dépletions partielles ou complètement). La valeur numérique du rapport des mobilités est \( \mu_n/\mu_p = 2,7 \)

Les dimensions des transistors PMOS et NMOS sont basées sur cette valeur, soit:

\( (W/L)_p/(W/L)_n \in [2,5 ; 3] \)

**2.2-d) Layout**

Le circuit logique de l’incrémenteur est à la figure 2.12. Le layout est à la figure 2.13. Les dimensions géométriques sont: \( \Delta x = 132,5 \mu m \), \( \Delta y = 106 \mu m \).
Figure 2.12: schéma logique de l'incrémenteur. Niveau transistors (choix définitif de la cellule de l'incrémenteur). Les sorties "out" et "Cout" sont générées par des inverseurs. Les grandeurs sont celle de Wp et Wn car on utilise Lp = Ln = 3 µm.

2.2-e) Simulation

La simulation d'une cellule est faite en prenant toutes les possibilités que peuvent avoir les trois entrées: in, Cin, inc; rien que pour garantir le fonctionnement de cette cellule. Les signaux doivent reproduire la table de vérité suivante:

<table>
<thead>
<tr>
<th>inc</th>
<th>in</th>
<th>Cin</th>
<th>out</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
figure 2:13 : cellule de l'incrémenteur
La vraie simulation est de tout le sous-bloc 16 bits de l'incrémenteur, car il s'agit d'une combinatoire dont les cellules sont misent en série: le carry se propage, et chaque cellule dépend du calcul fait par la précédente.

Le Cino qui entre à la première cellule est mit à 1; donc en relie le Cin0-bar à la masse dans le dessin de layout.

Si "inc" = 1 on fait l'incrémentation du vecteur d'entrée.

Si "inc" = 0 on fait la décrémentation

Deux simulations ont été faites pour l'incrémenteur de 16 bits:

Simulation N°1: décrémentation
donnée:
inc = 0
vecteur d'entrée = 1000000000000000

Résultat: à t = 20 ns les courbes "out" changent de valeurs.
Vecteur de sortie = 0111111111111111
Cout = 0

Simulation N°2: incrémentation
donnée:
inc = 1
vecteur d'entrée = 0111111111111111

Résultat: à t = 20 ns les courbes "out" changent de valeurs.
vecteur de sortie = 1000000000000000
Cout = 0

• à t = 0 s on injecte le vecteur d'adresses, à t = 20 ns on relie la sortie de l'incrémenteur au bus.
• la décrémentation et l'incrémentation sont faites après 20 ns, temps suffisant pour faire les calculs. L'incrémenteur prend 4 ns pour charger le bus (on a ajouté une capacité de C = 0,1 pF dans le netlist pour chaque ligne du bus).
layout de l'incrementeur 16 bits
2.2-f) Conclusion

L'incrémenteur peut fonctionner avec la fréquence de \( f = 25 \) MHz. Il est capable de générer le signal de sortie vers le PC par un inverseur de (\( W_p/W_n = 18/6 \)) sans aucun risque de s'atténuer.

2.3) L'ALU

L'ALU est conçue par sept différents sous-blocs de 8 bits, comme c'est présenté dans figure 1.7: NOT, FULL-ADDER, OR, AND, XOR, ROTATE, et SRT.

Chaque sous-bloc doit effectuer l'opération de deux vecteurs de 8 bits des registres TMP et ACT. Il doit-être capable de se comporter comme transparent pour l'ACT et TMP, afin que ces derniers trouvent un chemin vers les autres sous-blocs qui suivent; d'une manière à ce qu'un seul sous-bloc de l'ALU effectue l'opération et les six autres restent en mode transparent.

2.3-a) Cellule NOT

Le sous bloc NOT est formé par 8 cellules identiques, chacun effectue l'inversion d'un bit du registre TMP.

La figure 2.14 est un schéma logique qui présente deux entrées "tmp" et "not". En faite, si not = 0, alors out = tmp (mode transparence); et si not = 1, out = tmp-bar.

\[
\begin{array}{c}
\text{tmp} \\
\text{not} \\
\end{array}
\begin{array}{c}
11 \\
6 \\
\end{array}
\begin{array}{c}
11 \\
6 \\
\end{array}
\begin{array}{c}
\text{not} \\
\text{tmp} \\
\text{tmp-out} \\
\end{array}
\begin{array}{c}
\text{tmp} \\
\text{not} \\
\text{tmp-out} \\
\end{array}
\]

\[
\text{XOR (tmp , not)}
\]

Figure 2.14 : porte logique de la cellule NOT; si not = 0 alors out = tmp c'est le mode transparent de la cellule.

La figure 2.15 est le dessin de layout.

Les dimensions géométriques sont: \( \Delta x = 61,5 \) µm, \( \Delta y = 106 \) µm
Table de vérité NOT.

<table>
<thead>
<tr>
<th>tmp</th>
<th>not</th>
<th>tmp-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Dans la simulation, on a pris toutes les possibilités ainsi indiquées dans table de vérité. La cellule reproduit intégralement les valeurs de la table avec les temps critiques de charge et de décharge: $t_{montée} = 5 \text{ ns}$, $t_{descendue} = 7 \text{ ns}$. La fréquence de la cellule "not", signal le plus rapide dans la table de vérité, est de 25 MHz.
2.3-b) **FULL-ADDER**

Une cellule FULL-ADDER, d'un bit, est capable d'effectuer l'addition d'un bit de l'ACT avec celui du TMP et le carry "Cin" provenant de la cellule précédente. Elle génère le signal "out" sur le bus de données, résultat de l'opération, et le carry "Cout" pour la cellule suivante. Elle se comporte comme transparente pour \( \text{tmp} = \text{Cin} = 0 \), dans ce cas (\( \text{out} = \text{act} \)). La soustraction est faite en deux étapes:

- on complimente le TMP par le sous-bloc NOT,
- on l'additionne avec l'ACT et on met le (Cin0 = 1) de la première cellule.

Ainsi on a effectué l'opération logique: \( \text{ACT} + \text{NOT(TMP)} + 1 = \text{TMP} - \text{ACT} \).

La figure 2.17 est le schéma logique d'une cellule du FULL-ADDER.
Le dessin de layout du FULL-ADDER est à la figure 2.18. Les dimensions géométriques sont: $\Delta x = 215 \, \mu\text{m}$, $\Delta y = 106 \, \mu\text{m}$.

Deux simulations ont été faites pour la cellule du FULL-ADDER:

Simulation N°1:

Avec une fréquence de 25 MHz on veut reproduire la table de vérité du FULL-ADDER.

On ajoute au fichier "netlist", généré par CADENCE, les signaux de "cin", "tmp" et de "act" aussi bien les capacités de lignes (de l'ordre de 0,1pF).

En faite, Hspice utilise un autre fichier "si.inp" qui est un regroupement des données du fichier "netlist", des signaux de commandes et le modèle du transistor.
Résultat:

Pas de redondance, le "cout" se propage correctement, reste à ajuster les dimensions des transistors pour que le signal "out-act" puisse charger la ligne (simulation N°2).
Table de vérité du FULL-ADDER.

<table>
<thead>
<tr>
<th>act</th>
<th>tmp</th>
<th>Cin</th>
<th>out-act</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Simulation N°2:

L'idée est de voir le temps de monter et de descendre du signal généré par la cellule, et la propagation du signal à travers la ligne qui présente une capacité de C = 0,1 pF.

si cin = 0, tmp = 1, et act = 0 alors out-act = 1

On initialise "la ligne out-tmp" à 1 (voir dessin de layout)
• à t = 0 "add" = 0 et "add-bar" = 1 (signaux de commande des transistors de passage: transparence avec add = 0)
  out-act = act la ligne se décharge à 0.
• à partir de t = 18 ns "add" passe à 1 et "add-bar" à 0
  out-act = 1 la ligne se charge à 1.

Résultat:

La courbe out-act est celle de la charge et la décharge du bus par l'incrémenteur.

Le temps de montée et de descente est: t_{descente} = 2,5 ns et t_{montée} = 12 ns.
figure de Layout du FULL-ADDER 8bits
Une simulation est faite pour tout le sous-bloc combinatoire 8 bits du FULL-ADDER.

L'idée est d'effectué l'addition du TMP avec l'ACT: on injecte un vecteur d'entrée du registre ACT et un autre vecteur du registre TMP vers le FULL-ADDER.

Soit \( \text{ACT} = 11101101, \text{TMP} = 01110111, \text{Cin} = 0 \)

à \( t = 40 \text{ ns} \) (deuxième cycle d'horloge), ces deux vecteurs passent dans le FULL-ADDER; on active en même temps les signaux de commandes des transistors de passage pour relier la sortie du FULL-ADDER au bus qui présente une capacité de 0,1 pf.

Figure 2.18_bis: représentation du FULL-ADDER dans l'ALU.

Résultat:

Le vecteur de sortie est \( \text{out} = 01100100 \) et \( \text{cout} = 1 \)

Le FULL-ADDER effectue l'opération de deux vecteurs de 8 bits.

Les courbes "out0" "out1"..."out7" et de "Cout" sont générées après un temps de:

\( t\text{montée} = 10 \text{ ns}, t\text{descente} = 10 \text{ ns}. \)
2.3-c) Cellule OR

La cellule effectue l'opération logique OR du registre ACT et TMP. La figure 2.19 est le schéma logique de la cellule. Le sous-bloc est une duplication 8 fois de cette cellule. Donc le temps mis par chaque cellule est aussi celui de tout le sous-bloc.

Figure 2.19: schéma logique de la cellule OR avec \( \text{Wp}=11, \text{Wn}=4, \text{Lp}=\text{Ln} = 3 \mu \text{m} \).

![Diagram of the OR cell](image)
Si or = 0 alors out = act, la cellule est transparente, et si or = 1 on effectue l'opération OR (act, tmp). Le dessin de layout est à la figure 2.20. Les dimensions géométriques de la cellule sont tel que: ∆x = 64 µm, ∆y = 106 µm.

La simulation de la cellule OR est faite de manière à ce qu'elle produise la table de vérité propre à elle.

Table de vérité OR.

<table>
<thead>
<tr>
<th>or</th>
<th>tmp</th>
<th>act</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Résultat:

La cellule reprend toutes les valeurs "out" de la table de vérité avec un temps de montée et de descente de: t\text{montée} = 5 \, \text{ns}, t\text{descente} = 7 \, \text{ns}.
2.3-d) Cellule AND

Opération logique qui effectue la multiplication bit par bit tout en laissant la souplesse de mettre la cellule en mode transparent, comme c'est indiqué à la figure 2.22.

Le dessin de layout est à la figure 2.23, avec $\Delta x = 54 \mu m$, $\Delta y = 106 \mu m$.

![Diagram](image)

Figure 2.22: schéma logique de la cellule AND. si $(\overline{\text{and}} = 1)$ alors $(\text{out} = \text{act})$. Les dimensions des transistors sont $W_p = 11$ et $W_n = 4$, $L_p = L_n = 3 \mu m$. 
Figure 2.23: Layout de la cellule AND

Table de vérité AND.

<table>
<thead>
<tr>
<th>and-bar</th>
<th>tmp</th>
<th>act</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Résultat: La cellule reproduit la table de vérité avec des temps critiques (de la courbe "out") de: \( t_{montée} = 7 \text{ ns}, t_{descente} = 5 \text{ ns}. \)

2.3-e) Cellule XOR

Cette cellule effectue l'opération logique qui met la sortie à 1 quand un seulement des deux entrées "tmp" et "act" est à 1. Le mode transparent de la cellule est lorsque la commande 'xor' est à 0, voir figure 2.25.

![Diagrama XOR](image)

Figure 2.25: schéma logique de la cellule XOR: (a) niveau transistors, (b) niveau portes. si (xor = 0) alors (out = act) transparence.
Le layout est à la figure 2.26. Les dimensions géométriques sont: \( \Delta x = 67,5 \, \mu m, \Delta y = 106 \, \mu m. \)

Table de vérité XOR.

<table>
<thead>
<tr>
<th>xor</th>
<th>tmp</th>
<th>act</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
La simulation de la cellule XOR est réalisée à une fréquence de 25 MHz.

D’après la table de vérité, le signal "act" est le plus rapide. A cette fréquence la cellule fonctionne correctement.

Le temps de montée et de descente est $t_{montée} = 6 \text{ ns}$, $t_{descente} = 3,5 \text{ ns}$.

2.3-f) Sous-bloc \textit{ROTATE-SHIFT}

Il est formé par des transistors de passage NMOS uniquement, de sorte qu’il peut effectuer toutes les opérations de rotations d’un registre de 8 bits aussi bien la conversion en BCD. Le nombre de transistors de passage varie d’une cellule à l’autre, comme c’est indiqué à la figure 2.27.

Une fois que le signal passe à travers le transistor de passage NMOS ($W_n = 4 \mu m$), il sera amplifiée vers l’extérieur de la cellule par une paire d’inverseurs de tailles croissantes ($W_p/W_n = 11/4$ et 18/6).

Les signaux de commandes sont bien placés de manière à ce que le sous bloc
répond aux différents types d'opérations suivantes:

- **RLA**: faire la rotation à gauche du registre ACT en tenant compte du carry qui se trouve dans le registre STATUT. Ce registre n'est pas conçu ici, car il présente une structure non régulière. Sa conception sera faite à l'extérieur de cette architecture.

- **RLD**: faire la rotation à gauche digital. Il s'agit d'une rotation que le processeur l'utilise au cours de l'exécution d'un programme.

![Diagram](image)

L'opération de Rotation s'effectue en deux cycles machine: on exécute (1) puis (2).

**A**: registre de 8 bits du bloc2.

**(HL)**: registre pointeur de 16 bits du bloc2.

On charge le registre TMP avec le contenu de l'emplacement mémoire dans l'adresse se trouve dans le registre (HL).

On charge le registre ACT avec le contenu du registre A.

Pour effectuer la rotation de la moitié du byte de plus faible poids de ACT dans celui de TMP, et en même temps ce dernier dans le plus fort poids du même registre TMP; on active les commandes "stmp3" "stmp2" "stmp1" "stmp0" et "sact3" "sact2" "sact1" "sact0" du sous-bloc "ROTATE".

Avec ces commandes on a chargé le bus de données comme suit:

- **<7,4>** par la moitié du byte de plus faible poids du registre TMP.
- **<3,0>** par la moitié du byte de plus faible poids du registre ACT.

Ce bus de données est connecté au bus de données externe jusqu'au mémoire, et il ne reste qu'à placer la nouvelle donnée dans l'emplacement mémoire dont l'adresse se trouvant déjà dans le pointeur (HL).

Au deuxième cycle mémoire, on va charger la moitié byte de plus faible poids du registre ACT par la moitié byte de plus fort poids du registre TMP.

On active alors les commandes "sact7" "sact6" "sact5" "sact4" et "stmp7" "stmp7"
"stmp6" "stmp5" "stmp4" du sous-bloc "ROTATE".

Avec ces commandes on a chargé le bus de données comme suit:

<7,4> par la moitié du byte de plus fort poids du registre ACT.

<3,0> par la moitié du byte de plus fort poids du registre ACT.

On charge alors le registre A se trouvant dans le bloc 2 par cette nouvelle donnée.

D'autres instructions de rotation du même type que RLC sont possibles, tel que la rotation gauche sans carry, rotation droite avec et sans carry ... Chaque fois on active les bonnes commandes signalées dans le schéma logique (figure 2.27).

Le dessin de layout présentant tout le sous-bloc 8 bits est à la figure 2.28. Les dimensions géométriques qu'il occupe sont: $\Delta x = 334 \, \mu m$, $\Delta y = 820 \, \mu m$. 
(la suite est à la page suivante)
Figure 2.27: schéma logique du sous-bloc ROTATE formé par 8 cellules différentes. Les transistors de passages sont de dimension \((W_n = 4 \, \mu m, \, L_n = 3 \, \mu m)\). Les inverseurs forment un buffeur de dimension croissant \((W_p/W_n = 11/4)\) et \((W_p/W_n = 18/6)\). 

57
figure 2:28: Layout du sous-bloc combinatoire 16 bits ROTATE
La simulation:

L'idée est d'effectué la rotation gauche avec carry (RLA).

On donne le vecteur d'entrée semble provenir du registre ACT: la valeur choisie est $ACT = 01000110$

On met le carry "$cy" à 1.

On active les commandes "sact6" "sact543" "scy0" à $t = 40$ ns.

Le carry passe dans le bit n°0, alors que ce bit n°0 passe dans le bit n°1...Le bit n°6 passe dans le bit n°7 et celui du n°7 passe dans le bit du carry se trouvant dans le registre STATUT.

Résultat:

On récupère à la sortie le vecteur de sortie $out<7,0> = 10001101$.

Les courbes "out0"...."out7" sont les sorties du sous-bloc ROTATE. De cette simulation on peut déduire le temps mis pour charger et décharger le bus, soit: $tmontée = 4$ ns, $tdescente = 4$ ns.
SIMULATION DU SOUS-BLOC "ROTATE" (RALLIE AVEC L'ENTREE-01000110 ET CY-1)

94/09/14 23:42:34
2.3-g) Cellule SRT

Cette cellule met à 1, à 0 ou effectue le test d'un bit du registre ACT. La figure 2.30 est le schéma logique de cette cellule, et le sous-bloc 8 bits est une duplication de cette cellule. On ne peut en aucun cas appliquer 1 à la fois aux commandes "set" et "rst". Le mode transparent est lorsqu'on les met à 0. Le layout est donné à figure 2.31. Les démontions géométriques sont $\Delta x = 102 \, \mu m$, $\Delta y = 106 \, \mu m$.

figure 2.30: schéma logique de la cellule SRT. Interdit de mettre "set" et "rst" à 1 à la fois. Les dimensions des transistors sont $W_p = 11$, $W_n = 4$, $L_p = 3$, $L_n = 3 \mu m$. 

![Layout de la cellule SRT](image)
La courbe "data" reproduit intégralement les valeurs de la table de vérité.

En effet, on a activé les transistors de passage qui relit l'ALU au bus de données, de manière à ce que la cellule SRT se trouvant à l'extrémité de l'ALU puisse injecter sa sortie au bus.

Le temps critique de monté et de descente est de: $t_{montée} = 15 \text{ ns}$, $t_{descente} = 6 \text{ ns}$,
2.3-h) Conclusion

Chaque fois qu'on simule une cellule de l'ALU on essaye de mettre la fréquence de 25 MHz au signal de commande le plus rapide.

Toutes les combinaisons sont testées par les cellules et des exemples sont choisis dans le test des sous blocs ROTATE et FULL-ADDER. Les signaux générés prennent des temps de retard de montée et de descente qui ne dépassent pas les 15 ns.

La table suivante est le résumé de ces temps critiques:

<table>
<thead>
<tr>
<th>Cellule</th>
<th>$t_{montée}$(ns)</th>
<th>$t_{descente}$(ns)</th>
<th>$\Delta x$ (µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc-déc</td>
<td>4</td>
<td>4</td>
<td>132,5</td>
</tr>
<tr>
<td>registre</td>
<td>4</td>
<td>4</td>
<td>244,5</td>
</tr>
<tr>
<td>NOT</td>
<td>5</td>
<td>7</td>
<td>61,5</td>
</tr>
<tr>
<td>full-adder</td>
<td>10</td>
<td>10</td>
<td>215</td>
</tr>
<tr>
<td>OR</td>
<td>5</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>AND</td>
<td>7</td>
<td>5</td>
<td>54</td>
</tr>
<tr>
<td>XOR</td>
<td>6</td>
<td>3,5</td>
<td>67,5</td>
</tr>
<tr>
<td>rotate</td>
<td>4</td>
<td>4</td>
<td>334</td>
</tr>
<tr>
<td>SRT</td>
<td>15</td>
<td>6</td>
<td>102</td>
</tr>
</tbody>
</table>

Remarque: La cellule SRT a été simulée avec les transistors de passages reliant l'ALU au bus de données, pour cette raison elle prend un temps important. Pour les sous-blocs FULL-ADDER et l'incrémenteur, il s'agit du temps de charge du bus après une période d'horloge de calcul.
CHAPITRE 3
LES BLOCS

Une fois que les cellules et les sous-blocs combinatoires sont dessinés et simulés, on passe à former les trois blocs à partir des cellules. Ensuite, on colle ces blocs par des transistors de passage CMOS commandés par l'unité de commande à fin de permettre à chaque bloc la liberté de fonctionner indépendamment des autres.

3.1) BLOC 1

Description:

Bloc 1 est formé par le registre ACT, le registre TMP, et l'ALU. Ce bloc entre en communication en bus de données qui est de 8 bits. Donc les sous-blocs, qui constituent ce bloc 1, sont formés chacun de 8 cellules identiques.

3.1-a) Les registres

Les registres TMP et ACT, présentés dans Figure 2.2, sont conçus à partir du même point mémoire, d'un transistor de passage NMOS pour l'écriture sur le bus de données, d'un transistor NMOS pour la remise à zéro (registre reset), de transistors CMOS pour la lecture, et particulièrement pour le registre ACT un transistor NMOS pour sa remise à 1.

3.1-b) L'ALU

Elle est constituée par les sous blocs NOT, FULL-ADDER, OR, AND, XOR, ROTATE, SRT qui sont places dans cet ordre. Les vecteurs d'entrées, provenant des deux registres, doivent passer par tous ces sous-blocs, de manière que l'unité de contrôle active le sous-bloc combinatoire de l'ALU qui va effectuer l'opération et met les autres en mode transparent.

Bloc 1 contient des transistors de passage CMOS, pour l'entrée des données provenant du bus externe, et des autres identiques pour mettre le résultat de calcul de l'ALU sur le bus de données interne.
Figure 3.1: schéma du bloc 1 et sa communication en bus de données.

3.1-c) Dessin de layout du bloc 1

Le dessin de Layout de tout le bloc 1, ses dimensions géométriques sont: $\Delta x = 1219 \mu m$, $\Delta y = 822 \mu m$

Bloc1 englobe:

- 397 transistors PMOS et 494 transistors NMOS
- 632 S-D capacité ($\cong 20 \text{ ff}$) et 648 P capacité ($\cong 9$ à $25 \text{ ff}$)

En effet, ces capacités sont des capacités parasites:

- S-D capacité: c'est une capacité entre la source et le substrat, et entre le drain et le substrat d'un transistor. Le transistor MOS est symétrique.
- P capacité: est une capacité de recouvrement de pistes de métal et de pistes de poly-silicium qui sont de niveau différent.

On trouve aussi dans le fichier d'extraction d'autres types de capacités que peuvent exister; l'extraction ne tient compte que celles dans la valeur $\geq 1 \text{ fF}$. Ces capacités sont calculées à partir de la formule: $C_{ox} = \varepsilon_{ox}/t_{ox}$

Le fichier technologie, MODEL FILE, a donc une valeur de la capacité d'oxyde par carré, et il ne reste qu'à multiplier cette valeur par la surface de recouvrement entre pistes de niveaux différents ou entre pistes et substrat (dans ce cas $t_{ox} = \sum t_{ox}$) etc ...

On doit tenir compte de la déplétion arrière qui minimise la capacité parasite à la moitié. Dans la technologie utilisée, on a les épaisseurs d'oxyde comme dans le schéma suivant:
Figure 3.2: coupe transversale où on a introduit les épaisseurs d’oxyde des différents niveaux.
3.1-d) Simulation

L'idée est d'effectuer la soustraction de deux bytes avec une fréquence d'horloge "clock" de 25 MHz, période de 40 ns.

On fait entrer une donnée se trouvant au bus extérieur vers le bus de données "data", on activant la commande "in" dès la première période d'horloge. Cette donnée charge le registre ACT en activant "wr-act". On a pris $ACT = 10111001$. Au deuxième période, la donnée est changeée, on charge alors le registre TMP en activant "wr-tmp". On a pris $TMP = 10101011$.

Les données partent vers l'ALU dès l'activation des "wr". Pour cela, on active en même temps que "wr-tmp" les commandes suivantes:

- "not" du sous-bloc NOT: pour effectuer l'opération NOT(tmp).
- mettre le "cin" à 1 du sous-bloc FULL-ADDER pour effectuer la soustraction; en faite $ACT - TMP = ACT + NOT(TMP) +1$.
- la commande "add" du sous-bloc FULL-ADDER: pour permettre aux résultats de l'addition de passer vers les autres sous-blocs qui suivent et qui sont en mode transparent.
- la commande "to-data" pour permettre aux résultats parcourus tous les sous-blocs de l'ALU de se mettre sur le bus "data".

Les précautions prises sont:

- désactiver les commandes "wr" de chaque registre dès qu'il sera chargé, ceci pour ne pas perdre la donnée reçue par ce registre quand le bus extérieur change de valeurs au cours du temps. A cette fréquence de 25 MHz, on les active pour une demi-période (temps set est largement suffisante).
- activer les commandes "not", "add", "to-data", dès la charge du registre TMP, et ceci jusqu'à l'obtention du résultat sur le bus et "cin".
- on ajoute des capacités de 50 fF sur les pistes de métal constituant le bus de données dans le fichier netlist. C'est une estimation de ce qu'il peut y avoir de capacités de sources et de drains des transistors de passage des autres blocs quant on regroupe tous les trois blocs formant toute l'unité de traitement.
3.1-e) Résultat

Le bloc 1 peut supporter la fréquence de 25 MHz, car on arrive à écrire sur les registres à la moitié d'une période (temps \(set\)). La commande des différents sous-blocs et des transistors de passage est réussie.

D'après les résultats de la simulation, lorsqu'on injecte les deux vecteurs d'entrées à l'ALU, on attend une période d'horloge qui est le temps nécessaire pour effectuer l'opération. Ensuite on récupère le résultat déjà sur le bus dans un registre des autres blocs, et non pas dans le TMP ou ACT à fin de ne pas faire une boucle fermée ALU-TMP ou ALU-ACT. En effet TMP at ACT ne sont pas des registres Maître-esclave. Autre possibilité est d'injecté le résultat directement sur le bus externe. Pour notre cas, nous avons activé le "wr-tmp" à \(t = 40\) ns jusqu'à \(t = 60\) ns.

Le résultat du calcul est prêt sur le bus "data" à partir de \(t = 80\) ns. Donc on peut activer le "wr" du registre qui va récupérer ce résultat à partir de \(t = 80\) ns jusqu'à \(t = 100\) ns (temps \(set\) du clock).

3.2) BLOC 2

Description:

Le bloc 2 est formé par 12 registres 16 bits, comme dans la figure 3.4, et des registres de 8 bits. On dispose dans notre conception qu'un seul modèle de registre.

Figure 3.4: on présente ici 1 bit du poids fort et un bit du poids faible. Une duplication 8 fois de ce schéma fait former le registre 16 bits.
Le bloc 2 a aussi des transistors de passage pour lier le bus de données 8 bits (côté bloc 1) au bus d'adresses 16 bits. Ce bloc entre en communication avec le bus d'adresses, donc les sous-blocs qui le constituent sont formés chacun par 16 cellules identiques.

3.2-a) Layout
Le dessin de layout ses dimensions géométriques sont:
\[ \Delta x = 3016 \, \mu m, \quad \Delta y = 822 \, \mu m \]
Bloc2 englobe:
99 transistors PMOS et 171 transistors NMOS
119 S-D\_capacités (\( \approx 1,5 \text{ à } 300 \, \text{fF} \)) et 163 P\_capacités (\( \approx 1,2 \text{ à } 40 \, \text{fF} \))
3.2-b) Simulation

La simulation de ce bloc 2 consiste essentiellement à voir le temps mis pour qu'une information, se trouvant au premier registre, passe dans le dernier registre à travers le bus de données.

On premier lieu, on fait entrer une donnée semble provenir du bloc 1 dans le registre n°1. Ensuite, de ce registre on l'a fait passer dans le registre n°12.

La fréquence d'horloge utilisée pour cette simulation est de 25 MHz.

On met sur le bus de données "data" le premier byte.

Au premier tope d'horloge, on active la commande "data-to-Hadd" des transistors de passage qui raccordent le bus "data" au bus d'adresses <15,7>. On fait une écriture sur le byte du poids fort du registre n°1 on activant la commande "wr-Hr1" jusqu'à la fin du temps set de ce premier cycle d'horloge.

Au deuxième cycle, la donnée sur le "data" est changé, on active "data-to-Ladd" et on active la commande "wr-Lr1" jusqu'à la fin du temps set. On a effectué une écriture du byte du plus faible poids.

Au troisième cycle d'horloge, on fait le transfère du registre n°1 au registre n°12 on activant les commandes "rd-Hr1" "rd-Lr1" (pour la lecture du registre n°1) et "wr-Hr12" "wr-Lr12" (pour l'écriture sur le registre n°12).

Au quatrième cycle d'horloge, on active la commande "load" pour le rafraîchissement des points mémoires des registres.

Les courbes obtenues sont celles des commandes et des contenus de chaque point mémoire d'une cellule du poids fort (a1-H b1-H et a12-H b12-H) et d'une cellule du poids faible (a1-L b1-L et a12-L b12-L) des registres n°1 et n°12

Les précautions prises de plus que celles énoncées pour les registres ACT et TMP, sont: on ajoute le fait que lorsqu'on va faire le rafraîchissement et avant d'activer "load", on doit inhiber "wr" et "rd" pour ne pas perturber la valeur du point mémoire du registre aussi bien le bus. On ajoute dans le fichier netlist des capacités de 50 fF des pistes de métal constituant le bus de données.
3.2-c) Résultat

La lecture et l’écriture sur le registre ce fait dans le temps set (40 ns - 60 ns) de la période d’horloge, où on a activé la commande. Le transfère de données du registre, se trouvant à l’extrémité du bus vers un autre registre se trouvant à l’autre extrémité, se fait dans un temps critique de: \( t_{\text{montée}} = 5 \text{ ns} \), \( t_{\text{descente}} = 4 \text{ ns} \)

Sur les courbes, ce transfert est effectué au cours du temps set (80 ns - 100 ns).

Le rafraîchissement est fait à partir de \( t = 120 \text{ ns} \), et il a un effet correcteur de la valeur qui se décharge au cours du temps à cause des courants de fuite des transistors.

3.3) BLOC 3

Ce bloc contient le registre PC, l'incrémenteur, le buffer d'adresses, et les amplificateurs utiles pour les plots de sortie. Pour libérer le bus d'adresses à celui de l'extérieur reliant l'unité de traitement à la mémoire du µP, on a ajouté des transistors de passage CMOS entre ces amplificateurs et le reste du bloc 3.

![Schéma du bloc 3](image)

Figure 3.6: schéma du bloc 3. On plus des commandes visibles on a les commandes "load" "wr" "rd" se trouvant à l'intérieur des sous-blocs PC, INC, et BUFF.

3.3-a) Layout

Les dimensions géométriques du layout sont \( \Delta x = 1023,5 \mu \text{m} \), \( \Delta y = 822 \mu \text{m} \).

Bloc3 englobe:

339 transistors PMOS et 435 transistors NMOS

523 \( S_D \text{ capacities} \) (≈ 3 à 22 fF) et 703 \( P \text{ capacities} \) (≈ 5 à 35 fF)
figure 3:7: Layout du bloc 3
**3.3-b) Simulation**

L'idée est d'incrémenter l'adresse du registre PC:

On injecte une adresse de 16 bits semble provenir du bloc 2. Cette adresse s'écrit dans le registre PC, ensuite en fait une lecture vers le buffeur du PC. Automatiquement cette adresse passe à l'incrémenteur et à l'extrémité du bus du côté des amplificateurs. Aussi on l'a fait passer aux plots, et on attend à ce que l'incrémenteur calcule la nouvelle adresse, et donc on injecte la nouvelle dans le registre PC.

On a ainsi incrémenté le PC: (PC = PC + 1).

La fréquence de simulation est de 25 MHz et la valeur choisie de départ est bus = 1001100111111111.

Au premier cycle d'horloge, on active la commande "in" des transistors de passage pour l'injecter à l'intérieur du bloc 3, et on même temps "wr-pc" pour l'écriture sur le registre PC.

Au second cycle d'horloge et pour incrémenter le contenu du registre PC, on active "rd-pc" et "wr-buff". L'adresse est déjà à l'intérieur de l'incrémenteur et à l'extrémité du bus. On active "to-ampli" pour l'injecter aux plots de sorties en passant par les amplificateurs. On active la commande "inc" en même temps pour préciser le fonctionnement de l'incrémenteur (on a choisi "inc" = 1 : incrémentation de l'adresse). Ainsi on a l'adresse précédente sur les plots de sorties et une nouvelle à la sortie de l'incrémenteur.

Au troisième cycle d'horloge (temps nécessaire pour le calcul de cette nouvelle adresse), on active la commande "out-inc" et "wr-pc".

La nouvelle adresse est reçue par le registre PC.

Précaution à prendre:

La commande "inc" reste actif du début de l'injection de l'adresse dans l'incrémenteur jusqu'après l'inhibition de la commande "out-ampli" (pour qu'elle passe sur le bus).

Dans le cas où "inc" change de valeur, alors que les transistors de passage à la sortie de l'incrémenteur ne sont pas totalement coupés, l'incrémenteur change de mode. On trouve à sa sortie des redondances due à un nouveau calcul de l'adresse (car on décrémente avec "inc" = 0).
3.3-c) Résultat

Le vecteur d'entée est 1001100111111111.

Après incrémentation, le vecteur est devenu 1001101000000000.

L'incrémenteur est capable d'effectuer le calcul dans une seule période d'horloge (fréquence de 25 MHz).

D'après les courbes, le résultat est reçu après un temps qui ne dépasse pas les 10-ns (courbes "b9" et "a9"). C'est le temps que l'incrémenteur a prit.

Il n'y a pas de redondance à la sortie de la nouvelle adresse de l'incrémenteur, car il est conçu de manière à ce qu'il injecte les signaux de sorties à partir des inverseurs qui n'ont que les deux états logiques 0 ou 1.

Les plots présentent des capacités de 0,34375 pF. Cette valeur est donnée par l'extraction du dessin d'un plot (Pad) qui à la forme d'un carré en métal de coté 125 µm. Ces plots sont chargés à partir des amplificateurs dans un temps de: t\text{montée} = 10 \text{ ns}, t\text{descente} = 5 \text{ ns}

Les amplificateurs sont des cellules standards, dont la forme est donnée par la figure suivante:

Figure 3.8: schéma logique de l'amplificateur unidimensionnel. On prend Ln=Lp= 3µm.

3.4) "MACRO-CELL" de toute l'unité

Le Macro-cell est constitué par les blocs 1, 2 et 3 et des transistors de passage. Les signaux de sortie du Macro-cell doivent être amplifier par les amplificateurs, comme dans figure 3.8. L'unité de traitement du µP Z-80 est donc constituée par le Macro-cell, des amplificateurs, et des plots.
3.4-a) Description

Les blocs 1, 2 et 3 ont été conçus et simulés avec la fréquence 25 MHz.

Comme c'est prévu au cours de l'étude de l'architecture de l'unité, le reste est à regrouper ces trois blocs en les reliant par des transistors de passage.

On a remarqué que la longueur du dessin de layout est cinq fois sa largeur. Une idée est de découper l'unité en deux parties, et de les mettre l'une sur l'autre. De cette façon on peut avoir un layout plus compacte, comme dans la figure 3.9.

Figure 3.9: répartition de l'unité de traitement. On a coupé bloc2 en deux pour pouvoir optimiser la géométrie de l'unité en carré.

Les lignes de l'alimentation ne doivent être, en aucun cas, connecter par du polysilicium qui présente une résistance 10000 fois plus grande que celle de l'aluminium. Raison à la quelle, on distribue l'alimentation depuis les plots jusqu'aux transistors (sources et drains) à travers des pistes d'Aluminium (Al).

Résistance du poly-silicium $\cong 30 \, \Omega / \text{carré}$. Résistance de l'Al $\cong 0,003 \, \Omega / \text{carré}$

Si on raccorde l'alimentation à une piste de poly-silicium occupant une surface $S$, ce n'est plus le 5 V qui arrive aux transistors mais $v = (5 - r \times i) < 5$ volts, et de la masse non plus zéro volt mais $v = r \times i > 0$ volt.

avec $r$: résistance de la piste tel que $r = 30 \, \Omega / \text{carré} \times S$

et $i$: le courant qui circule dans la piste de poly-silicium (d'ordre de quelques
centaines de µA).

3.4-b) Layout

Le layout du Macro-cell présente une longueur 1,57 fois la largeur, soit: \( \Delta x = 2706 \, \mu m \), \( \Delta y = 1722 \, \mu m \).

Il englobe 1272 transistors PMOS, 2043 transistors NMOS.

2017 S-D capacités (\( \approx 1,5 \) à 300 fF) et 2429 P capacités (\( \approx 1 \) à 45 fF).

Les amplificateurs peuvent être glissés entre les plots externes de manière à ce qu'on gagne sur la surface libre pour le "Macro-cell". Toute l'unité de traitement est constituée par 3495 transistors.

3.4-c) Simulation

Étant donnée que les blocs 1, 2 et 3 fonctionnent avec la fréquence de 25 MHz, une simulation de toute l'unité consiste à voir le temps mis pour qu'une information passe d'une extrémité de l'unité à l'autre extrémité. En fait, Ces blocks supportent même une fréquence plus élevée dans l'ordre de 30 MHz, car les différents temps de charges n'ont pas dépassé les 15 ns.

L'idée est de passer une information du registre ACT (bloc1) au registre PC (bloc3) à travers les deux bus de données et d'adresses, et de voir le temps mis tout le long de l'unité de l'extrème à l'extrême.

La fréquence est fixée à 25 MHz. A partir de \( t = 80 \, ns \) on active les signaux de lecture du registre ACT et d'écriture du registre PC, en effectue alors une écriture du ACT au PC.

3.4-d) Résultat

Pour charger le registre PC à partir du registre ACT on prend un temps de: \( t_{montée} = 6 \, ns \), \( t_{descente} = 3 \, ns \).

Étant donnée que toutes les cellules fonctionnent avec la fréquence de 25 MHz, tous les sous-blocs combinatoires FULL-ADDER, Incrémenteur, et ROTATE, et les blocs 1, 2 et 3 fonctionnent aussi correctement. On peut conclure que l'unité de traitement, ainsi conçu, peut supporter cette fréquence de 25 MHz.
CONCLUSION

L'unité de traitement du processeur Z-80 est conçue en technologie SOI des transistors CMOS, en partant des cellules aux sous-blocs, puis aux blocs jusqu'à toute l'unité. Elle a demandé 3315 transistors et elle occupe une surface fictive (sans compter les plots) de 4,66 cm².

La méthode de conception "full-custom design" des circuits intégrés exige, pas seulement de résoudre le problème au niveau porte logique, mais plus au niveau transistor. Autrement dit, "Full custom design" demande une connaissance de la technologie des semiconducteurs et des transistors, l'architecture des blocs combinatoires synchrones et asynchrones, "circuitry" des portes logiques, conception des circuits logiques et analogues, dessin de Layout des circuits, et la simulation.

Cette méthode demande beaucoup de temps de conception: dessiner, simuler, dimensionner les transistors, et puis re-dessiner de sorte qu'on arrive enfin à la conception du circuit qui soit le plus compacte et le plus rapide que possible. Donc c'est une perte sur le temps de conception, mais un gain très important sur la surface de silicium. La densité d'occupation par les transistors doit être importante pour baisser le coût de conception des chips électroniques.

Pour la fréquence de fonctionnement de l'unité de traitement, la simulation est faite pour toutes les parties de l'unité (cellules, blocs..) avec une fréquence de 25 MHz. Cette fréquence est choisie à partir de nombreuses simulations à fréquences différentes, que ce soit au cours de la conception d'une cellule ou de tout un bloc. La fréquence maximale, que peut supporter cette unité, ne se fixera définitivement qu'après la fabrication. Donc notre fréquence n'est autre qu'une approche de la réalité.
**OUVRAGES DE REFERENCE**

*Digital MOS Integrated Circuits II*
with Applications to Processors and Memory design.
Mohamed I Elmasry
IEEE Solid-State Circuits council, sponsor 1992

*Digital CMOS Circuit Design*
Marco Annaratone
Kluwer Academic Publisher 1986

*Silicon on Insulator Technology*
Materials to VLSI
Jean-Pierre Colinge
Kluwer Academic Publisher 1991

*The VLSI Designer’s Library*
John Newkirk, Robert Mathews
Addition-Wesley Publishing Company, Inc. 1983

*MOS Circuits Clocking*
Charles Trullemans
Intensive summer course on:
CMOS & BiCMOS VLSI Design’s 93 (EPFL)

*Z80-CPU  Z80A-CPU*
Zilog – Technical Manual
Heliagraph pvba. Computer Applications International
ANNEXE 1

Tables d’instructions:
Mnémoniques du processeur Z-80
Après Zilog
<table>
<thead>
<tr>
<th>Instruction</th>
<th>C</th>
<th>Z</th>
<th>V</th>
<th>S</th>
<th>N</th>
<th>H</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD A, 1; ADC A, 1</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>8-bit add or add with carry</td>
</tr>
<tr>
<td>SUB 1, SBC A, 1, CP, 1, NEG</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>8-bit subtract, subtract with carry, compare and negate accumulator</td>
</tr>
<tr>
<td>AND s</td>
<td>O</td>
<td>P</td>
<td>O</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>Logical operations</td>
</tr>
<tr>
<td>OR s, XOR s</td>
<td>O</td>
<td>P</td>
<td>O</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>Logical operations</td>
</tr>
<tr>
<td>INC s</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>8-bit increment</td>
</tr>
<tr>
<td>DEC m</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>8-bit decrement</td>
</tr>
<tr>
<td>ADD DD, m</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>16-bit add</td>
</tr>
<tr>
<td>ADC HL, m</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>16-bit add with carry</td>
</tr>
<tr>
<td>SBC HL, m</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>16-bit subtract with carry</td>
</tr>
<tr>
<td>RLA, RLCA, RRA, RRCA</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Rotate accumulator</td>
</tr>
<tr>
<td>RL, RLC, RR, RRC</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Rotate and shift location m</td>
</tr>
<tr>
<td>SLA, SRA, SRL, m</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Rotate and shift location m</td>
</tr>
<tr>
<td>RLD, RRD</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Rotate right and left</td>
</tr>
<tr>
<td>DAA</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Decimal adjust accumulator</td>
</tr>
<tr>
<td>CPL</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Complement accumulator</td>
</tr>
<tr>
<td>SCF</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Set carry</td>
</tr>
<tr>
<td>CCF</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Complement carry</td>
</tr>
<tr>
<td>IN, INX, OUT, OUTX</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Input register indirect</td>
</tr>
<tr>
<td>INR, INDR, OUTR, OUTDR</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Block input and output</td>
</tr>
<tr>
<td>LD, LDD</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Block transfer instructions</td>
</tr>
<tr>
<td>LD, LD, LD, LDH, LDH, LDH</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Block transfer instructions</td>
</tr>
<tr>
<td>LD, LDH, LDH, LDH, LDH, LDH</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Block transfer instructions</td>
</tr>
<tr>
<td>LD A, 1, LD A, R</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag</td>
</tr>
<tr>
<td>BIT b, s</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>The state of bit b of location s is copied into the Z flag</td>
</tr>
<tr>
<td>NOTC</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>$</td>
<td>Negate accumulator</td>
</tr>
</tbody>
</table>

The following notation is used in this table:

**Symbol**

- **C**: Carry flag. C=1 if the operation produced a carry from the MSB of the operand or result.
- **Z**: Zero flag. Z=1 if the result of the operation is zero.
- **S**: Sign flag. S=1 if the MSB of the result is set.
- **P/V**: Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result is even, P/V=0 if the result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
- **H**: Half carry flag. H=1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
- **N**: Add/Subtract flag. N=1 if the previous operation was a subtract.
- **I**: The flags are unaffected by the operation.
- **O**: The flag is set by the operation.
- **A**: The flag is not set by the operation.
- **Y**: The flag is a "don't care."
- **P**: P/V flag affected according to the overflow result of the operation.
- **V**: P/V flag affected according to the parity result of the operation.
- **R**: Any one of the CPU registers A, B, C, D, E, H, L.
- **M**: Any 16-bit location for all the addressing modes allowed for the particular instruction.
- **I**: Any 16-bit location for all the addressing modes allowed for that instruction.
- **R**: Any one of the two index registers IX or IY.
- **F**: Flag or carry.
- **s**: 8-bit value in range <0, 255>.
- **m**: 16-bit value in range <0, 65535>.

**Summary of Flag Operation**

<p>| TABLE 8.0-1 |</p>
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>C</th>
<th>Z/P/N</th>
<th>I</th>
<th>H</th>
<th>74</th>
<th>54</th>
<th>3</th>
<th>16</th>
<th>No. of BYTES</th>
<th>No. of Cycles</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD r, r'</td>
<td>r' ← r</td>
<td>000</td>
<td>A</td>
<td>0110</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0</td>
<td>0 7</td>
<td>0 0 0</td>
<td>B</td>
</tr>
<tr>
<td>LD r, n</td>
<td>r ← n</td>
<td>000</td>
<td>A</td>
<td>0110</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0</td>
<td>0 7</td>
<td>0 0 0</td>
<td>C</td>
</tr>
<tr>
<td>LD r, (HL)</td>
<td>r ← (HL)</td>
<td>0110 0 0 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>LD r, (IX+d)</td>
<td>r ← (IX+d)</td>
<td>0110 0 0 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>LD r, (IY+d)</td>
<td>r ← (IY+d)</td>
<td>0110 0 0 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>LD (HL), r</td>
<td>(HL) ← r</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>LD (IX+d), r</td>
<td>(IX+d) ← r</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>LD (IY+d), r</td>
<td>(IY+d) ← r</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>LD (HL), n</td>
<td>(HL) ← n</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>LD (IX+d), n</td>
<td>(IX+d) ← n</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>LD (IY+d), n</td>
<td>(IY+d) ← n</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>LD A, (B), A</td>
<td>A ← (B)</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>LD A, (DE), A</td>
<td>A ← (DE)</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>LD A, (nn), A</td>
<td>A ← (nn)</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>LD (BC), A</td>
<td>(BC) ← A</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>LD (DE), A</td>
<td>(DE) ← A</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>LD (nn), A</td>
<td>(nn) ← A</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>LD A, I</td>
<td>A ← I</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>LD A, R</td>
<td>A ← R</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>LD A, A</td>
<td>A ← A</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>LD R, A</td>
<td>R ← A</td>
<td>0011 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- 'r, r' means any of the registers A, B, C, D, E, H, I.
- IFF is the content of the interrupt enable flip-flop (IFF) is copied into the Z/P/N flag.
- The flag notation: 0 = flag not affected, 1 = flag reset, 0 = flag set, X = flag is unknown.
- 't' = flag is affected according to the result of the operation.

**8-BIT LOAD GROUP**

**TABLE 7.0-1**
<table>
<thead>
<tr>
<th>Movement</th>
<th>Dynamic Operation</th>
<th>Flag</th>
<th>Op Code</th>
<th>No. of Bytes</th>
<th>No. of M Cycles</th>
<th>No. of Y Bytes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD, 6-8</td>
<td>d0 - n0</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 6, 7</td>
<td>EX - n0</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 7, 8</td>
<td>FY - n0</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 8, 6</td>
<td>H - (n0 +1) L - n0</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 6, 6</td>
<td>d0d2 - (n0 +1) d0d1 - (n0)</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 7, 7</td>
<td>EX - (n0 +1) X - (n0)</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 8, 8</td>
<td>FY - (n0 +1) Y - (n0)</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 6, 6</td>
<td>d0d2 - d0d1</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 7, 7</td>
<td>EX - EX</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
<tr>
<td>LD, 8, 8</td>
<td>FY - FY</td>
<td>F</td>
<td>6 563 219</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>d0 F0</td>
</tr>
</tbody>
</table>

Notes: 6d is any of the register pairs BC, D1, E1, D7.
8d is any of the register pairs A7, BC, DE, HL.
6d, 6, 7, 8, 6, 8 are in any order and in any order eight bits of the register pair respectively.
6d, 7, 7, 8, 8, 6, 8 are in any order and in any order eight bits of the register pair respectively.

Flag Notation: e = flag not affected, g = flag not set, l = flag set, x = flag is unknown.

16-BIT LOAD GROUP

TABLE 7.0-2
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of</th>
<th>No. of</th>
<th>No. of</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operation</td>
<td></td>
<td></td>
<td>Bytes</td>
<td>Cycles</td>
<td>T States</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EX DE, HL</td>
<td>DF = HL</td>
<td>*</td>
<td>DA = AF</td>
<td>00 001 000</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>EX (SP), IX</td>
<td>IX ← (SP + 1)</td>
<td>*</td>
<td>*</td>
<td>11 011 011</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>EX (SP), IX</td>
<td>IX ← (SP + 1)</td>
<td>*</td>
<td>*</td>
<td>11 111 101</td>
<td>2</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>EX (SP), IX</td>
<td>IX ← (SP)</td>
<td>*</td>
<td>*</td>
<td>11 100 011</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>LDI</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LDI</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LDR</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>LDRR</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LDD</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>LDD</td>
<td>DF ← (HL)</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>CPI</td>
<td>A ← (HL)</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11 101 101</td>
<td>2</td>
</tr>
<tr>
<td>CPI</td>
<td>A ← (HL)</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11 101 101</td>
<td>2</td>
</tr>
<tr>
<td>CPDR</td>
<td>A ← (HL)</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11 101 101</td>
<td>2</td>
</tr>
<tr>
<td>CPDR</td>
<td>A ← (HL)</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11 101 101</td>
<td>2</td>
</tr>
</tbody>
</table>

Notes: 1. P = flag set if the result of BC = 1, otherwise P = 0 |
        2. Z = flag set if A = (HL), otherwise Z = 0 |

Flag Notation: * = flag not affected, 0 = flag reset, 1 = flag set, X = flag unknown, I = flag affected according to the result of the operation |

**EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP**

**TABLE 7.0-3**
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of Bytes</th>
<th>No. of M Cycles</th>
<th>No. of T States</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD A, r</td>
<td>A - A + r</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 1 0000 010</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>r     Reg.</td>
</tr>
<tr>
<td>ADD A, n</td>
<td>A - A = n</td>
<td>C, S, V</td>
<td>1 V 1 0 1 1 1 0 000 110</td>
<td>2</td>
<td>7</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>ADD A, (HL)</td>
<td>A - A + (HL)</td>
<td>C, S, V</td>
<td>1 V 1 0 1 1 1 0 000 110</td>
<td>1</td>
<td>2</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>ADD A, (IX+d)</td>
<td>A - A + (IX+d)</td>
<td>C, S, V</td>
<td>1 V 1 0 1 1 0 1 1 01 1</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>ADD A, (Y+d)</td>
<td>A - A + (Y+d)</td>
<td>C, S, V</td>
<td>1 V 1 0 1 1 1 1 1 1 1 1 1 01 0</td>
<td>3</td>
<td>5</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 V 1 1 1 1 0 0 0 1 01 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC A, r</td>
<td>A - A + s + CV</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 1 0 00 010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB r</td>
<td>A - A - s</td>
<td>C, S, V</td>
<td>1 V 1 1 0 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>SBC A, r</td>
<td>A - A - s - CV</td>
<td>C, S, V</td>
<td>1 V 1 1 0 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>AND r</td>
<td>A - A &amp; s</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 1 1 1 1 1 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>XOR r</td>
<td>A - A ^ s</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 1 1 1 1 1 1 1 1 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>CP r</td>
<td>A - s</td>
<td>C, S, V</td>
<td>1 V 1 1 1 1 1 1 1 1 1 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>INC r</td>
<td>(HL) + 1</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>INC (HL)</td>
<td>(HL) + 1</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>INC (IX+d)</td>
<td>(IX+d) + 1</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>INC (Y+d)</td>
<td>(Y+d) + 1</td>
<td>C, S, V</td>
<td>1 V 1 0 1 0 0 0 0 1 00 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>DEC m</td>
<td>m = m - 1</td>
<td>C, S, V</td>
<td>1 V 1 1 1 1 1 1 1 1 1 1 01 1</td>
<td>1</td>
<td>4</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

Notes: The V symbol in the P.V. flag column indicates that the P.V. flag contains the overflow of the result of the operation. Similarly, the P.V. symbol indicates parity. V = 1 means overflow. V = 0 means no overflow. P = 1 means parity of the result is even. P = 0 means parity of the result is odd.

Flag Notation: * = flag not affected, C = carry set, I = interrupt in, X = flag is unknown. * = flag is affected according to the result of the operation

B-BIT ARITHMETIC AND LOGICAL GROUP

TABLE 7.04

94
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Plan</th>
<th>D-Code</th>
<th>No. of Bytes</th>
<th>No. of M Cycles</th>
<th>No. of Y States</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAA</td>
<td>Converts acc. content into packed BCD following and or subtraction with packed BCD operands</td>
<td>1 1 1 1 * 1 1</td>
<td>00 100 111</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>Decimal adjust accumulator</td>
</tr>
<tr>
<td>CPL</td>
<td>A = X</td>
<td>* * * * 1 1</td>
<td>00 101 111</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>Complement accumulator (one's complement)</td>
</tr>
<tr>
<td>NEG</td>
<td>A = 0 - A</td>
<td>1 1 V 1 1 1</td>
<td>11 101 101</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>Negate acc. (two's complement)</td>
</tr>
<tr>
<td>CCF</td>
<td>CY = CY</td>
<td>* * * * 0 X</td>
<td>00 111 111</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>Complement carry flag</td>
</tr>
<tr>
<td>SCF</td>
<td>CY = 1</td>
<td>* * * * * 0</td>
<td>00 110 111</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>Set carry flag</td>
</tr>
<tr>
<td>NOP</td>
<td>No operation</td>
<td>* * * * * *</td>
<td>00 000 000</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td>CPU halted</td>
<td>* * * * * *</td>
<td>01 110 110</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>IF = 0</td>
<td>* * * * * *</td>
<td>11 110 011</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>EI</td>
<td>IF = 1</td>
<td>* * * * * *</td>
<td>11 111 011</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>IM 0</td>
<td>Set interrupt mode 0</td>
<td>* * * * * *</td>
<td>11 101 101</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>IM 1</td>
<td>Set interrupt mode 1</td>
<td>* * * * * *</td>
<td>01 000 110</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>IM 2</td>
<td>Set interrupt mode 2</td>
<td>* * * * * *</td>
<td>11 101 101</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Notes: IF indicates the interrupt enable flip-flop
       CY indicates the carry flip-flop.

Flag Notation: * = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
       1 = flag is affected according to the result of the operation.

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 7.0-5

95
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of Bytes</th>
<th>No. of CYCLES</th>
<th>No. of Status</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD HL, w</td>
<td>HL = HL + w</td>
<td>1 w w + 0 X</td>
<td>00 00 001</td>
<td>1 3 11</td>
<td>m Reg.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC HL, w</td>
<td>HL=HL+w + CY</td>
<td>1 V 1 0 X</td>
<td>11 101 001</td>
<td>2 4 15</td>
<td>0 BC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC HL, m</td>
<td>HL=HL-m - CY</td>
<td>1 V 1 0 X</td>
<td>11 101 011</td>
<td>2 4 15</td>
<td>10 HL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD IX, pp</td>
<td>IX = IX + pp</td>
<td>1 m m + 0 X</td>
<td>11 011 001</td>
<td>2 4 15</td>
<td>15 SP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD IY, r</td>
<td>IY = IY + r</td>
<td>1 m m + 0 X</td>
<td>11 011 001</td>
<td>2 4 15</td>
<td>16 SP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INC m</td>
<td>m = m + 1</td>
<td>1 * * * * *</td>
<td>00 00 011</td>
<td>1 1 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INC IX</td>
<td>IX = IX + 1</td>
<td>1 * * * * *</td>
<td>11 011 011</td>
<td>2 2 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INC IY</td>
<td>IY = IY + 1</td>
<td>1 * * * * *</td>
<td>11 111 011</td>
<td>2 2 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC m</td>
<td>m = m - 1</td>
<td>1 * * * * *</td>
<td>00 00 001</td>
<td>1 1 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC IX</td>
<td>IX = IX - 1</td>
<td>1 * * * * *</td>
<td>11 011 011</td>
<td>2 2 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC IY</td>
<td>IY = IY - 1</td>
<td>1 * * * * *</td>
<td>11 011 011</td>
<td>2 2 10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: m is any of the register pairs BC, DE, HL, SP.
pp is any of the register pairs BC, DE, IX, SP.
r is any of the register pairs BC, DE, IY, SP.

Flag Notation: = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, 
1 = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP
TABLE 7.0-6

96
<table>
<thead>
<tr>
<th>Masnomic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of Bytes</th>
<th>No. of M Cycles</th>
<th>No. of T States</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLCA</td>
<td></td>
<td></td>
<td></td>
<td>00 000 133</td>
<td>1</td>
<td>1</td>
<td>4 Rotate left circular accumulator</td>
</tr>
<tr>
<td>RLA</td>
<td></td>
<td></td>
<td></td>
<td>00 010 133</td>
<td>1</td>
<td>1</td>
<td>4 Rotate left accumulator</td>
</tr>
<tr>
<td>RRCA</td>
<td></td>
<td></td>
<td></td>
<td>00 001 133</td>
<td>1</td>
<td>1</td>
<td>4 Rotate right circular accumulator</td>
</tr>
<tr>
<td>RRA</td>
<td></td>
<td></td>
<td></td>
<td>00 011 133</td>
<td>1</td>
<td>1</td>
<td>4 Rotate right accumulator</td>
</tr>
<tr>
<td>RLC r</td>
<td></td>
<td></td>
<td></td>
<td>11 001 011</td>
<td>2</td>
<td>2</td>
<td>8 Rotate left circular register r</td>
</tr>
<tr>
<td>RLC (HL)</td>
<td></td>
<td></td>
<td></td>
<td>00 010 133</td>
<td>2</td>
<td>4</td>
<td>15 r Reg</td>
</tr>
<tr>
<td>RLC (IX+n)</td>
<td></td>
<td></td>
<td></td>
<td>11 011 101</td>
<td>4</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>RLC (Y+n)</td>
<td></td>
<td></td>
<td></td>
<td>11 011 101</td>
<td>4</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>RLC m</td>
<td></td>
<td></td>
<td></td>
<td>00 110</td>
<td></td>
<td></td>
<td>1 Instruction format &amp; stages are as shown for RLC m. To form new OP-code replace 000 of RLC m with shown code</td>
</tr>
<tr>
<td>RRC m</td>
<td></td>
<td></td>
<td></td>
<td>00 111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RR m</td>
<td></td>
<td></td>
<td></td>
<td>01 100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLA m</td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRA m</td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRL m</td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLD</td>
<td></td>
<td></td>
<td></td>
<td>11 101 101</td>
<td>2</td>
<td>5</td>
<td>18 Rotate digit left and right between the accumulator and location (HL) The content of the upper half of the accumulator is unaffected</td>
</tr>
<tr>
<td>RRD</td>
<td></td>
<td></td>
<td></td>
<td>01 100 111</td>
<td>2</td>
<td>5</td>
<td>18</td>
</tr>
</tbody>
</table>

Flag Notation:  * = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, 1 = flag is affected according to the result of the operation.

**ROTATE AND SHIFT GROUP**
**TABLE 7.0-7**
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>OpCode</th>
<th>No. of Bytes</th>
<th>No. of Cycles</th>
<th>No. of T States</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT b, r</td>
<td>(Z \rightarrow r)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>2</td>
</tr>
<tr>
<td>BIT b, (HL)</td>
<td>(Z \rightarrow (HL))</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>2</td>
</tr>
<tr>
<td>BIT b, (IX+d)</td>
<td>(Z \rightarrow (IX+d))</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>4</td>
</tr>
<tr>
<td>BIT b, (IY+d)</td>
<td>(Z \rightarrow (IY+d))</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>4</td>
</tr>
<tr>
<td>SET b, r</td>
<td>(s_b \rightarrow 1)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>2</td>
</tr>
<tr>
<td>SET b, (HL)</td>
<td>((HL) \rightarrow 1)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>2</td>
</tr>
<tr>
<td>SET b, (IX+d)</td>
<td>((IX+d) \rightarrow 1)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>4</td>
</tr>
<tr>
<td>SET b, (IY+d)</td>
<td>((IY+d) \rightarrow 1)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>4</td>
</tr>
<tr>
<td>RES b, m</td>
<td>(s_b \rightarrow 0)</td>
<td>(X)</td>
<td>(V)</td>
<td>(N)</td>
<td>(H)</td>
<td>76 543 210</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes: The notation \(s_b\) indicates bit \((0\ to \ 7)\) or location \(m\).

Flag Notation: 0 = flag not affected, O = flag reset, I = flag set, X = flag unknown, r = flag is affected according to the result of the operation.

To form new OpCode replace \(11\) of SET b, m with \(00\). Flags and time states for SET instruction.

BIT SET, RESET AND TEST GROUP

TABLE 7.0-B
<table>
<thead>
<tr>
<th>Macro</th>
<th>Symmetric Operation</th>
<th>Flags</th>
<th>Op. Code</th>
<th>No. of Bytes</th>
<th>No. of Instruction Cycles</th>
<th>No. of Status Bytes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP ea</td>
<td>PC -- ea</td>
<td>a a</td>
<td></td>
<td>11 600 011</td>
<td>3</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>JP ea, an</td>
<td>IF condition cc true PC = ea, otherwise continue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JR e</td>
<td>PC -- PC + e</td>
<td>a a a a</td>
<td>00 011 000</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>JR C, e</td>
<td>IF C = 0, continue</td>
<td>a a a a</td>
<td>00 111 000</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>If condition not met</td>
</tr>
<tr>
<td>JR NC, e</td>
<td>IF C = 1, PC -- PC + e</td>
<td>a a a a</td>
<td>00 110 000</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>If condition not met</td>
</tr>
<tr>
<td>JR Z, e</td>
<td>IF Z = 0, continue</td>
<td>a a a a</td>
<td>00 101 000</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>If condition not met</td>
</tr>
<tr>
<td>JR NZ, e</td>
<td>IF Z = 1, PC -- PC + e</td>
<td>a a a a</td>
<td>00 100 000</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>If condition not met</td>
</tr>
<tr>
<td>IP (HL)</td>
<td>PC = HL</td>
<td>a a a a</td>
<td>11 101 001</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>IP (IX)</td>
<td>PC -- IX</td>
<td>a a a a</td>
<td>11 011 101</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>IP (IY)</td>
<td>PC -- IY</td>
<td>a a a a</td>
<td>11 111 101</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DJNZ e</td>
<td>B -- B - 1</td>
<td>a a a a</td>
<td>00 010 000</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>IF B = 0</td>
</tr>
<tr>
<td></td>
<td>IF B = 0, continue</td>
<td>a a a a</td>
<td>00 010 000</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Notes:  
e represents the extension in the relative addressing mode  
s is a signed two's complement number in the range <-126, 127>  
e-2 in the op-code provides an effective address of PC + s as PC is incremented by 2 prior to the addition of s.  
Flag Notation:  
* = flag not affected, O = flag reset, 1 = flag set, X = flag is unknown,   
S = flag is affected according to the result of the operation.
<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of Byins</th>
<th>No. of M Cycles</th>
<th>No. of T States</th>
<th>Comment(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL mn</td>
<td>(SP-1)→PCH (SP-2)→PCL</td>
<td>* * * * *</td>
<td>01 001 010</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td>PC→n</td>
<td></td>
<td>− n −</td>
<td></td>
<td></td>
<td></td>
<td>If cc is false</td>
</tr>
<tr>
<td>CALL cc, an</td>
<td>If condition cc is false continue, otherwise same as CALL mn</td>
<td>* * * *</td>
<td>11 cc 100</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>− a −</td>
<td></td>
<td></td>
<td></td>
<td>If cc is true</td>
</tr>
<tr>
<td>RET</td>
<td>PC=(SP)</td>
<td>* * * *</td>
<td>01 001 001</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td>PC=(SP+1)</td>
<td></td>
<td>− a −</td>
<td></td>
<td></td>
<td></td>
<td>If cc is false</td>
</tr>
<tr>
<td>RET cc</td>
<td>If condition cc is false continue, otherwise same as RET</td>
<td>* * * *</td>
<td>11 cc 000</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>− a −</td>
<td></td>
<td></td>
<td></td>
<td>If cc is true</td>
</tr>
<tr>
<td></td>
<td>cc</td>
<td></td>
<td>Condition</td>
<td></td>
<td></td>
<td></td>
<td>[Diagram]</td>
</tr>
<tr>
<td>RETI</td>
<td>Return from interrupts</td>
<td>* * * *</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
<td>14</td>
<td>[Diagram]</td>
</tr>
<tr>
<td>RETN</td>
<td>Return from non maskable interrupt</td>
<td>* * * *</td>
<td>11 101 101</td>
<td>2</td>
<td>4</td>
<td>14</td>
<td>[Diagram]</td>
</tr>
<tr>
<td>RET P</td>
<td>(SP-1)→PCH (SP-2)→PCL</td>
<td>* * * *</td>
<td>11 t 111</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td>PCn=0</td>
<td></td>
<td>− a −</td>
<td></td>
<td></td>
<td></td>
<td>[Diagram]</td>
</tr>
<tr>
<td></td>
<td>PC&lt;sub&gt;P&lt;/sub&gt;−P</td>
<td></td>
<td>− a −</td>
<td></td>
<td></td>
<td></td>
<td>[Diagram]</td>
</tr>
</tbody>
</table>

Flag Notation: 0 = flag not affected, 1 = flag reset, X = flag is unknown  
t = flag is affected according to the result of the operation.
<table>
<thead>
<tr>
<th>Maremonic</th>
<th>Symbolic Operation</th>
<th>Flags</th>
<th>Op-Code</th>
<th>No. of Bytes</th>
<th>No. of M Cycles</th>
<th>No. of T States</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN A. (C)</td>
<td>A ← (x) n ←</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>11</td>
<td>(A_0 \rightarrow A_13) (A_0 \rightarrow A_7)</td>
</tr>
<tr>
<td></td>
<td>r ← (C) if r ≠ 110 only the flags will be affected</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>12</td>
<td>(A_0 \rightarrow A_7) (A_0 \rightarrow A_7)</td>
</tr>
<tr>
<td>INI</td>
<td>(HL) ← (C)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>100</td>
<td>010</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL + 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Repeat until B = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IND</td>
<td>(HL) ← (C)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>101</td>
<td>010</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL - 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Repeat until B = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDR</td>
<td>(HL) ← (C)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>111</td>
<td>010</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL - 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Repeat until B = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUT (O) A</td>
<td>A ← A</td>
<td>X X X X</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>OUT (C), r</td>
<td>C ← r</td>
<td>X X X X</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>OUTI</td>
<td>(C) ← (HL)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>100</td>
<td>011</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL + 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>OTIR</td>
<td>(C) ← (HL)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>100</td>
<td>011</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL + 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Repeat until B = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTD</td>
<td>(C) ← (HL)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>100</td>
<td>011</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL - 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>OTDR</td>
<td>(C) ← (HL)</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B ← B - 1</td>
<td>10</td>
<td>100</td>
<td>011</td>
<td>2</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>HL ← HL - 1</td>
<td>X X X X</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Repeat until B = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: * = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ① = flag is affected according to the result of the operation.

INPUT AND OUTPUT GROUP
TABLE 7.0-11
ANNEXE 2

Règles de conception

3µm-5V CMOS SOI Technologie
CMOS 3µm - DESIGN RULES

REGLES DE CONCEPTION DES CIRCUITS INTEGRES
EN TECHNOLOGIE CMOS GRILLE POLYSILICICUM 3µm

Réf.: DM/BGi
Date:
Version:
Règles de layout LABCMOS3

1.0 Informations générales

1.1 Toutes les dimensions dessinées sont supposées être les dimensions réelles sur silicium.

1.2 Les règles de layout énoncées dans ce document sont exprimées en multiples de lambda et lambda = 1 micron. (Ceci amène plusieurs changements par rapport à la version précédente ; ces changements sont signalés par un • dans la marge de la règle modifiée. Pour rappel, l’ensemble des motifs doit être dessiné sur une grille de 0.5 lambda.)

1.3 Les règles de layout présentées dans ce document sont donc adimensionnelles et indépendantes du processus technologique.

2.0 Définition des différents niveaux

<table>
<thead>
<tr>
<th>CNW</th>
<th>N-well</th>
<th>Définit les puits pour les transistors P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPD</td>
<td>Diffusion p+</td>
<td>Définit les zones diffusées p+ et les canaux des transistors P.</td>
</tr>
<tr>
<td>CND</td>
<td>Diffusion n+ (zone active)</td>
<td>Définit les zones diffusées n+ et les canaux des transistors N.</td>
</tr>
<tr>
<td>CP</td>
<td>Polysilicium</td>
<td>Définit les grilles poly et les interconnexions poly.</td>
</tr>
<tr>
<td>CC</td>
<td>Contact</td>
<td>Définit l’ouverture des contacts sur la diffusion et le poly.</td>
</tr>
<tr>
<td>CM</td>
<td>Metal</td>
<td>Définit les interconnexions métalliques.</td>
</tr>
<tr>
<td>CG</td>
<td>Passivation</td>
<td>Définit les ouvertures dans la couche de passivation.</td>
</tr>
</tbody>
</table>
3.0 Règles de dessin

3.1 Puits

1. A 3.1.1 largeur minimale: \( d_1 = 6 \)

1. B 3.1.2 espace minimum:
   a) pour les puits équipotentiels: \( d_2 = 1.4 \)
   b) pour les puits non-equipotentiels: \( d_2 = 1.4 \)

![Diagram of N-well and p-well](image)

3.2 Diffusions (zone active)

2. A 3.2.1 largeur minimale: \( d_3 = 3 \)

2. B 3.2.2 largeur minimale d'un transistor: \( d_4 = 3 \)

4. F 3.2.3 débordement minimum de la diffusion de drain et de source de part et d'autre de la grille: \( d_5 = 4 \)

3.2.4 distance minimale entre la grille d'un transistor et une diffusion de type opposé (contact de puits ou de substrat): \( d_6 = 4 \)

2. D 3.2.5 distance minimale entre diffusions portées à des potentiels différents ou égaux: \( d_7 = 4 \)

2. E 3.2.6 distance minimale entre diffusions équipotentielles: \( d_8 = 4 \)

2. F 3.2.7 débordement du puits autour d'une diffusion p+ (source ou drain d'un transistor P-MOS): \( d_9 = 2 \)

2. H 3.2.8 débordement du puits autour d'une diffusion n+ (contact de puits): \( d_{10} = 0 \)

2. C 3.2.9 distance minimale entre puits et diffusion n+ (source ou drain d'un transistor N-MOS): \( d_{11} = 10 \)

2. F 3.2.10 distance minimale entre puits et diffusion p+ (contact de substrat): \( d_{12} = 6 \)

Diffusion P+

5. A. Débordement minimum de la diffusion P+ sur la zone active \( d = 8 \)

5. B. Distance minimale entre la diffusion p+ et la zone active \( d = 8 \)

5. C. Distance minimale entre diffusion p+ \( d = 8 \)

5. D. Distance minimale entre diffusion p+ et transistors canaux N \( d = 4 \)

5. E. Débordement minimum de la diffusion p+ autour d'un transistor canaux p (source et drain) \( d = 4 \)

5. F. Largeur minimale \( d = 3 \)
3.3 Polysilicium

densité de courant max : 160 mA/μm²

4.A.3.3.1 largeur minimale: d₁₅=3
4.B.3.3.2 longueur minimale de la grille d'un transistor: d₁₆=3
4.C.3.3.3 distance minimale entre deux bandes de poly: d₁₇=3
4.E.3.3.4 extension minimale de la grille d'un transistor sur l'oxyde de champ: d₁₈=4.5
4.O.3.3.5 distance minimale entre poly et diffusion: d₁₉=4.5
3.4 Contacts

3.4.1 dimensions du contact:
- d20 = 3
- d21 = 2
- d22a = 1.5
- d22b = 1.5
- d23 = 1.5

3.4.2 distance minimale entre deux contacts:
- 3 x 3

3.4.3 débordement minimal du poly autour du trou de contact:

3.4.4 débordement minimal autour du trou de contact
- a) de la zone active
- b) d'une diffusion

3.4.5 débordement minimal du métal autour du trou de contact:
- d24a = 3
- d24b = 3

3.4.6 distance minimale entre la grille d'un transistor et un contact

3.5 Metal f.

3.5.1 largeur minimale:
- d25 = 4

3.5.2 distance minimale entre 2 pistes dont la plus large est
- a) de largeur inférieure à 7.5:
- b) de largeur comprise entre 7.5 et 18:
- d26a = 3
- d26b = 6
4.0 Règles générales

4.1 Le N-well est connecté à la tension d'alimentation la plus positive (VDD), le substrat est connecté à la tension d'alimentation la plus négative (VSS).

4.2 Tous les dispositifs connectés aux pôts d'entrée et sortie doivent être reliés aux alimentations (VDD et VSS) par l'intermédiaire d'un métal; puits et substrat ne peuvent jamais être utilisés pour alimenter une diffusion.
4.3 Un transistor n (resp. p) ne peut jamais être distant de plus de 0.05\(\) du contact de substrat (resp. contact de puits) le plus proche.

4.4 Tous les éléments connectés aux entrées ou sorties doivent avoir un anneau de garde.

4.5 Toutes les entrées de circuit doivent être munies d'un circuit de protection contre les décharges électrostatiques.
### PARAMÈTRES ÉLECTRIQUES – TECHNO CMOS 3 MICRONS

#### A) Paramètres mesurés directement sur les tranches de test.

<table>
<thead>
<tr>
<th>* Concentrations en impuretés</th>
<th>substrat P</th>
<th>$1 \times 10^{15}$ cm$^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-Well</td>
<td>$6 \times 10^{15}$ cm$^{-3}$</td>
<td></td>
</tr>
<tr>
<td>S&amp;D Pmos</td>
<td>$5 \times 10^{19}$ cm$^{-3}$</td>
<td></td>
</tr>
<tr>
<td>S&amp;D Nmos</td>
<td>$1 \times 10^{20}$ cm$^{-3}$</td>
<td></td>
</tr>
<tr>
<td>canal</td>
<td>$1,15 \times 10^{16}$ cm$^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>* Profondeurs des jonctions</th>
<th>NMOS</th>
<th>0,25 μm</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMOS</td>
<td>0,47 μm</td>
<td></td>
</tr>
<tr>
<td>N-Well</td>
<td>3,0 μm</td>
<td></td>
</tr>
<tr>
<td>canal</td>
<td>0,28 μm</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>* Capacités</th>
<th>zone N$^+$</th>
<th>0,15 fF/μm$^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>zone P$^+$</td>
<td>0,25 fF/μm$^2$</td>
<td></td>
</tr>
<tr>
<td>oxyde MOS</td>
<td>0,63 fF/μm$^2$</td>
<td></td>
</tr>
</tbody>
</table>

| * Epaisseur de l'Oxyde MOS   | 550 Å       |

<table>
<thead>
<tr>
<th>* Tensions de seuil sur oxyde épais</th>
<th>canal N</th>
<th>14 V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>canal P</td>
<td>-12 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>* Résistivité des couches</th>
<th>N$^+$</th>
<th>$33 , \Omega/\square$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P$^+$</td>
<td>$49 , \Omega/\square$</td>
</tr>
<tr>
<td></td>
<td>Poly</td>
<td>$27 , \Omega/\square$</td>
</tr>
</tbody>
</table>

---

UCL - Laboratoire de microélectronique - Ch. Arsenault - Novembre 12 1987