Design
&
Électronique Numérique

Travaux Dirigés

de

Patrick Furon

Design & Electronique

1. AU PROGRAMME DES TD

1.1 Gérer l'espace et gérer le temps

1.1.1 Une approche duale du problème

- Des blocs fonctionnels communiquant entre eux au moyen de routes de communication numérique
- Sous la gouverne d'un séquenceur ou machine d'état qui établit les règles de circulation

1.1.2 Les fonctionnalités:

- Tester, comparer, décaler, décoder, valider, compresser, compacter, aiguiller, calculer, interfacer, mémoriser, compter, temporiser, séquencer, sans oublier les fonctions booléennes maîtrisées en logique positive et négative

1.1.3 Le matériel du "mécano-légo-logique":

- portes ET, OU, NON, NAND, NOR, XOR, multiplexeur, démultiplexeur, décodeur, buffer 3S, monostables, bascules SR, D, JK, T, flip-flop & latch transparent, compteurs programmables, registres à décalage, mémoires, sorties à collecteur ouvert ou 3 états, RAM, ROM, EEPROM, CMOS+ pile lithium, PLA et PAL, encodeurs, décaleurs, ALU, comparateurs, registres divers, microcontrôleurs…(sont soulignés les éléments de DigSim)

1.1.4 Travail sur des exemples

- Réalisations de multiples manières d'une fonction combinatoire: avec des portes, des multiplexeurs, des décodeurs, des mémoires RAM ou ROM, des PLA ou PAL
- Utilisation des compteurs programmables
- Exemples combinatoires: multiplexer, démultiplexer, décaler, décoder une information
- Exemples séquentiels: mémoriser, enclencher, déclencher, décaler
- Codages un signal sous forme série; émission et réception de signaux série
- Gestion du temps et du trafic des données au moyen de séquenceurs
    - Séquenceurs câblés synchrones ou asynchrones à bascules
    - Séquenceurs à flip-flop D et mémoire contenant une table de fonctionnement
    - Séquenceurs à compteur programmable, multiplexeur et mémoire de micro-programme

1.2 Synchrone ou asynchrone?

- Pourquoi et comment faire du synchrone
- Quand et comment faire de l'asynchrone

1.3 S'interfacer

- Microprocesseurs
- Mémoires
- Décodage d'adresses partiel ou complet; différentes façons de faire

1.4 Équilibrer matériel et logiciel, "Hard"&"Soft"

- Qualités et limites respectives du hardware et du software
- Exploiter la complémentarité de l'un et de l'autre avec un juste dosage
- Tendance vers des produits intégrant l'un et l'autre
- Exemples de systèmes multi-bus et de systèmes multi-processeurs

1.5 Règles d'hygiène

- Alimentation des circuits et règles pour un bon découplage
- Comment rendre les signaux "propres & synchrones"
- Interfaçage entre familles logiques de rapidités différentes

2. Design = dessein, intention ;dessin, modèle, plan, ébauche ; préméditation ;prévoir, concevoir ; dessiner, étudier, établir...

2.1 Réflexions sur le design

par Denis Poussart

http://www.gel.ulaval.ca/~poussart/gel21405/design.html

Le cours de Systèmes électroniques - numériques a pour objectif central l'acquisition et le développement de l'esprit de synthèse et de la démarche du design. Il utilise comme véhicule de cet apprentissage le domaine des systèmes électroniques réalisés par des méthodes analogiques et numériques.

Le design constitue l'activité par excellence de l'ingénieur et de sa faculté innovatrice. Néanmoins, il n'est pas facile d'en circonscrire précisément la nature. En effet, lorsqu'il s'exerce à un niveau élevé, le design fait appel à un faisceau de considérations, combinées les unes aux autres de manières subtiles et procédant selon une démarche en apparence souvent obscure. Il devient illusoire d'essayer d'en fournir une description rigoureuse.

Cette activité est cependant si importante qu'il convient d'en examiner les principales facettes. D'autant plus que l'électronique et les systèmes numériques, parmi les divers scénarios possibles d'apprentissage en design, sont particulièrement privilégiée par les développements technologiques contemporains. Rares sont les disciplines qui permettent en effet d'explorer aussi aisément les rapports théorie - pratique, concepts - applications, modèle - réalité, abstractions - compromis, qui sont à la base de la démarche du design technique.

L'ingénieur engagé en design reconnaît trois objets essentiels qu'il doit éventuellement intégrer:

- Il existe un problème précis, un besoin identifié
- dont on recherche concrètement une solution satisfaisante
- à l'intérieur de contraintes diverses.

Précisons quelque peu.

Un travail de design débute par l'identification d'un problème. L'existence même d'une nouvelle technologie peut parfois susciter la "découverte" de questions jusque-là ignorées. Mais en l'absence de besoins réels, concrets, une technique nouvelle ne peut demeurer qu'une "*solution à la recherche d'un problème".*

Fréquemment, de plus en plus fréquemment d'ailleurs au fur et à mesure que l'électronique envahit des champs d'applications toujours plus vastes, *la source primaire du problème est externe*. Par exemple, on souhaitera contrôler les paramètres d'une serre (agriculture), ou réduire le blocage de freins (mécanique) ou encore inspecter la qualité de produits manufacturés (robotique). Par voie de conséquence, ce problème peut compter de nombreux éléments imprécis, ou qui sont exprimés, au départ, sous une forme qui n'est pas exclusivement "électronique". Ainsi, afin de répondre à des besoins en communications et information (considérations socio-économiques), on cherchera à développer de nouveaux modes d'accès distribué à des banques de données. Ceci fera intervenir des questions d'interactions homme machine (ergométrie) encore difficilement quantifiables. Ou, afin d'améliorer la condition de patients affectés de tel problème (considérations médicales et humanitaires), on cherchera à mettre au point tel type de prothèse, alors même que de nombreuses données fondamentales sont encore incomplètes. La première tâche du responsable en design va donc consister à bien comprendre, dès le départ, le *pourquoi* du problème. En interaction très active avec l'utilisateur éventuel, il va s'efforcer de préciser, autant que possible, les paramètres pertinents.

Cette première phase du design se concrétise par la rédaction d'un cahier des charges. Elle est doublement importante. Elle va évidemment conditionner la recherche de la solution. Mais surtout c'est sur sa base, et sa base seule, que le produit final sera jugé. Les erreurs commises à ce stage seront la plupart du temps très coûteuses, ou même catastrophiques. L'ingénieur ne devra jamais perdre de vue la finalité du produit: la valeur d'une astuce technique, si brillante soit elle, ne saura jamais masquer la défaillance d'un produit qui ne répond pas aux attentes de l'usager.

Les paramètres généraux du problème étant arrêtés, du moins provisoirement, la recherche du comment de la solution peut alors débuter. On peut en distinguer deux grands aspects, en apparence contradictoires, mais en fait complémentaires.

Dans un premier temps, en s'appuyant sur l'imagination d'abord, et l'analyse ensuite, la démarche va rechercher une solution de principe qui simplifie volontairement le problème posé. Elle va retourner le problème "sous toutes ses coutures", en cherchant les lignes directrices, en identifiant les concepts qui s'y raccrochent et qui puissent être exploités, bref en évoluant dans un univers abstrait, idéal, provisoirement libéré de la multitude des détails particuliers qu'il faudra éventuellement aborder. C'est l'expérience et le sens physique qui vont être les guides de ce "débroussaillage". Ils suggéreront les ordres de grandeur, et partant la justification, d'abord intuitive, puis quantifiée par l'analyse, de l'aspect secondaire ou non secondaire de tel ou tel détail.

Cette phase aboutira à l'architecture du circuit ou du système, à une vision globale de l'ensemble, vision claire parce que non entravée par des détails à ce stade superflus. L'ensemble est maintenant segmenté en blocs correspondant à des frontières naturelles, conformes aux ressources technologiques. De plus ces blocs sont reliés les uns aux autres par des relations bien précises. Cette représentation est exprimée sous la forme de blocs diagrammes, d'organigrammes, de diagrammes d'états ou de diagrammes temporels. Une façon efficace de procéder sera d'ailleurs de faire comme s'il s'agissait de l'écriture d'un logiciel bien structuré. Cette comparaison est à ce point valable que la distinction formelle entre la synthèse d'un système électronique contemporain et la rédaction d'un logiciel tend rapidement à s'estomper. De plus en plus on simule des systèmes entiers sur ordinateur avant de les assembler, et on remplace des blocs jadis réalisés par circuits par des modules exécutés par microprocesseur. Ceci aboutit à un ensemble de circuits (algorithmes) segmentés en blocs et sous-blocs (routines, procédures)

modulaires et de dimensions suffisamment modestes pour être visualisés d'un seul coup d'œil et synthétisés (programmés) avec un très faible risque d'erreur.
Le second temps de ce "comment" de la solution va voir s'effectuer la mise en forme proprement dite du circuit ou du système. Chaque bloc sera successivement abordé et synthétisé en tenant compte d'un ensemble de détails: choix des composants, études de sensibilité, de précision, de stabilité, prévisions de moyens de diagnostic et de maintenance, mise en boîte physique ("packaging"), alimentation, protection contre les interférences et les surcharges, considérations touchant les techniques d'assemblage, fiabilité, et, bien entendu, coûts. C'est ici, par exemple, que l'ingénieur devra savoir qu'un potentiomètre de qualité peut coûter beaucoup plus cher que l'amplificateur opérationnel dont il souhaite ajuster l'erreur statique: un meilleur ampli, plus cher mais ne nécessitant pas de potentiomètre, ne coûterait-il pas, finalement, moins cher? Et il devra savoir que les connecteurs, certes commodes pour la vérification et la réparation, diminuent considérablement la fiabilité d'un ensemble complexe, et que l'économie sur le découplage des lignes d'alimentation est presque toujours mal placée, ou qu'une ligne de masse inadéquatement distribuée sur une carte imprimée peut avoir des conséquences catastrophiques pour le bon fonctionnement d'un circuit et que la changer en cours de production est presque impensable, ou que telle fonction jadis réalisée de façon analogique devrait aujourd'hui être faite de façon numérique, etc. ...
Cette phase est longue, ardue, coûteuse. Elle exige un travail méthodique dont la seule documentation représente une partie considérable.
Si la répartition définie dans la phase précédente a été faite correctement, cette tâche fastidieuse pourra cependant être effectuée de façon systématique, modulaire, et facilement répartie entre plusieurs personnes.
On a brièvement décrit le caractère distinctif de chaque aspect d'un design: identification, conception, réalisation. Si on a parlé de phases, il ne faudrait surtout pas conclure qu'il s'agit d'opérations parfaitement isolées les unes des autres. Bien au contraire, et c'est précisément les interactions entre ces phases, nombreuses et subtiles, qui rendent l'analyse de ce processus d'innovation si difficile à décrire. En effet on imagine aisément qu'un architecte qui ne dessinerait un édifice qu'à partir de critères esthétiques et humains (pourtant de première importance) et qui ignorerait les contraintes de la mécanique des structures ou les propriétés des matériaux, ferait fausse route. S'il a du génie, il saura au contraire intégrer, et finalement exploiter, ces limites en un tout harmonieux.
Il en va de même dans tout design. La conception électronique sera futile si, tout en respectant les objectifs identifiés, elle ne s'articule pas sur une connaissance approfondie de la technologie courante, de ce qu'elle offre et à quels prix, et de son évolution future prévisible. Et justement parce qu'il n'est nul autre domaine technique où l'évolution soit si rapide, une connaissance sans cesse mise à jour des composants, modules, sous-systèmes, fonctions et outils des support disponibles est essentielle à quiconque vise un design concurrentiel. Ainsi, afin de mener à bien la phase initiale de définition du projet, l'ingénieur devra bien savoir ce qu'il est très facile de réaliser aujourd'hui en électronique, ce qu'il est possible de faire à un coût plus élevé mais encore raisonnable, et ce qu'il est illusoire de tenter sauf dans des applications exceptionnelles. La courbe performance vs coût a toujours la même allure: augmentation graduelle jusqu'à un niveau seuil, avec progression très rapide au-delà (ceci étant dû, en bonne partie, à la réduction d'économie d'échelle dans les niveaux de performances rarement exigés). Encore aujourd'hui, les deux derniers bits d'un convertisseur analogique numérique de 16 bits / 10 microsecondes coûtent à eux seuls aussi cher que les 14 autres. Ce seuil limite évolue constamment et le concepteur efficace doit savoir où il se situe aujourd'hui et où il se situera bientôt.
Encore en rapport avec cette interdépendance entre les diverses phases d'un projet, comment élaborer un bloc diagramme qui va segmenter un circuit en parties analogiques et numériques si, en même temps, ne sont pas clairement présents à l'esprit les avantages, les inconvénients, les imperfections et les problèmes de chaque approche. Ainsi il faudra se demander si l'économie et la précision obtenues par une approche numérique dans tel bloc ne risquent pas d'être trop chèrement payées par le bruit ainsi produit sur les lignes d'alimentation et ses conséquences sur tel autre bloc analogique: un excellent découplage sera-t-il suffisant ou va-t-on devoir recourir à des alimentations distinctes? .....
Si on peut attribuer une pondération à de tels facteurs, il est possible d'aborder de façon rigoureuse l'optimalisation d'un circuit. Et cela se fait, effectivement. Mais parce que le nombre de contraintes est souvent démesuré, que le coût ou la valeur des inconvénients et des avantages est lui-même difficile à préciser ("combien vaut une réputation de fiabilité? ...") le jugement pragmatique, le sens physique, l'information sur l'état de la technique demeurent, dans le concret, aussi importants que la compréhension profonde des concepts de base.

2.2 C'est seulement l'expérience acquise personnellement comme la somme de toutes les erreurs comprises qui permet, en définitive, de développer cette aptitude à un degré élevé. C'est là que le travail expérimental de laboratoire, où se confrontent l'abstraction du modèle et la réalité concrète, trouve sa justification. Mais une "expérience" ne laisse de traces durables que si elle s'accompagne d'un esprit critique, qui ne se satisfait pas d'une vérification approximative et banale, mais qui plutôt observe, cherche, ... et finalement comprend. Tout est là.

3. Quelques éléments du « mécano logique » : composants et fonctions associées

Ce tableau est réalisé dans le but de suggérer des possibilités de réalisation de différentes fonctions. il n'est pas exhaustif.

	Les routes : communication & manipulation des données et des adresses									la gestion et le séquencement			
fonctions composants	inverser	test de parité	comparer	décaler	décoder valider	compresser compacter aiguiller	décompresser décompacter aiguiller	interfacer	mémoriser	état	test des transitions	actions	temporisations
ou exclusif	x	x	x								x		
décodeur-démultiplexeur					x		x						
multiplexeur				x		x					x	x	
décaleur statique				x									
buffer, transceiver driver de ligne	x			x		x	x	x				x	
circuits arithmétiques et logiques (ALU)	x	x	x	x								x	
comparateurs			x										
sorties collecteur ouvert (+ résistance de pull up) « OU câblé)						x		x					
sorties 3 états (L, H & haute impédance)						x		x					
trigger de Schmitt								x					
monostables													x
bascules RS, D, JK, flip-flop				x					x	x		x	
flip-flop D, latch transparents				x		x	x	x	x	x		x	
registres à lecture et écriture simultanées								x	x				
registres FIFO							x	x	x				
compteurs			x							x			x
registres à décalage et piles SISO, SIPO, PISO, PIPO			x	x		x	x	x	x				
RAM								x	x			x	
ROM												x	
EEPROM, RAM+pile lithium	x	x	x	x	x	x	x	x	x			x	
PLA PAL	x	x	x	x	x	x	x	x	x	x	x	x	x
FPGA	x	x	x	x	x	x	x	x	x	x	x	x	x
Microcontrôleurs	x	x	x	x	x	x	x	x	x	x	x	x	x

Ne pas sous estimer l'emploi des portes élémentaires ET, OU, NON, NAND, NOR

4. Expérimenter et partager son expérience sur Internet avec DigSim

Digital Simulator [MesSchemas/Demo.dig]

File Edit Passive Gates Combinational... Bi-stable Counters & Registers Display Miscellaneous Simulate Speed Help

*CETTE VERSION DE DIGSIM_ENSICAEN A ETE REVUE ET CONTIENT DE NOMBREUX ET NOUVEAUX COMPOSANTS
SI CETTE SIMULATION NE S'AFFICHE PAS CORRECTEMENT ET QU'UN MESSAGE D'ERREUR APPARAIT C'EST PARCEQUE
VOUS POSSEDEZ UNE ANCIENNE VERSION DE DIGSIM QUI EST DECLAREE PAR UN CLASSPATH DANS VOTRE MACHINE
IL VOUS FAUT ALORS TELECHARGER ET INSTALLER CETTE NOUVELLE VERSION OU DESACTIVER PROVISOIREMENT L'ANCIENNE

Compteurs et affichage

Appuyer sur les boutons poussoir

Bascule /S /R asynchrone

REMARQUES

* PATIENCE SI VOTRE CONNEXION EST LENTE,
* POUR LANCER LA SIMULATION, CLIQUER SUR L'ECLAIR OU APPUYEZ SUR LA TOUCHE ENTREE DE VOTRE CLAVIER
* SI VOTRE ANTIVIRUS EST ACTIF, LES EXEMPLES DE "File Open example" SERONT TRES LONGS A OUVRIR
* SI VOUS AVEZ DECLARE DIGSIM SUR VOTRE MACHINE AU MOYEN D'UN CLASSPATH
  VOUS POUVEZ AUSSI ACCEDER AUX FICHIERS DE SIMULATION DU REPERTOIRE "DigSim/examples"
  AU MOYEN DE "File Open File" ou du raccourci "Ctrl O" (voir NOTES à LIRE)
* NE PAS OUBLIER QUE VOUS POUVEZ MODIFIER LA VITESSE DE SIMULATION DANS "Simulate Options"
  C'EST PARFOIS UNE NECESSITE SUR LES MACHINES RAPIDES
* VOUS DISPOSEZ D'UN ANALYSEUR LOGIQUE dans "Simulate Show analyser";
  ATTENTION il ne fonctionne que si DigSim est déclaré par un CLASSPATH
* VOUS POUVEZ IMPORTER DES MORCEAUX DE SIMULATIONS AU MOYEN DE "Edit Paste from"
* LE PASSAGE DE LA VERSION NOIR ET BLANC A LA VERSION COULEUR DEMANDE A CE QUE VOUS FERMIEZ
  LE NAVIGATEUR AVANT DE LE RELANCER

Move the cursor to a component body or wire-end and press a mouse button.

Attention : fenêtre d'applet

5. Le microprocesseur 8085 et sa périphérie

5.1 Généralités

- C'est un système séquentiel qui dispose d'une horloge à quartz permettant un « timing » précis et qui fonctionne grâce à un logiciel ou « software » qui sera vu au second semestre...et à un matériel ou « hardware » *(cf. schéma type) auquel nous allons nous intéresser.*
    - signaux de commande, de dialogue et de communication
        - 16 bits d'adresses et 8 bits de données
            - bus non multiplexé A8-15
            - bus multiplexé AD0-7
        - signaux **/RD, /WR**, ALE et interprétation des *cycles de lecture et d'écriture*
        - autres signaux du 8085 :
            - /RST IN & RST OUT pour l'état initial (reset)
            - CLK horloge de fréquence d'égale à la moitié de celle du quartz
            - READY pour l'interfaçage avec des périphériques lents
            - HOLD & HOLDA « handshake » pour transfert rapide (mode DMA)
            - RST 5.5, RST 6.5, RST 7.5, INTR, /INTA, TRAP (interruptions)
            - S0, S1, IO /M d'état du cycle machine
            - SID & SOD (entrées et sorties série)
    - éléments périphériques RAM, ROM, PORTS, TIMER, CAN, CNA & Registres divers
    - démultiplexage du bus multiplexé AD0-7 *(cf. sujet examen 1996)*

5.1.1 Décodage des adresses

- l'adresse est sur 16 bits : $[A_{15},A_{14},A_{13}...A_1,A_0]_2 = [X\ X\ X\ X]_{16}$
- les périphériques sont de tailles multiples, ce qui demande à choisir entre décodage complet et/ou décodage partiel → compromis « hardware » - « software »
- les outils classiques du décodage sont
    - le décodeur - démultiplexeur 74HC/HCT138 (≅ 8205) *(cf. data sheet)*
    - le comparateur 8 bits 74HC/HCT688
    - les PAL & PLA

5.1.2 Multiplicité des solutions matérielles

- solutions standard et solutions adaptées ?
- les solutions pour mieux gérer le temps font appel à une maîtrise du hardware, des "astuces", décodages et bus « à façon
- les deux plateaux de la balance
    - ordre de facilité de mise en œuvre : soft → synchrone → asynchrone
    - voir le compromis « hard »-« soft » et rapidité – compacité-intégration
        - réaliser le principal en soft,
            - y intégrer des parties hard synchrones aux endroits opportuns
                - en y incluant éventuellement une ou plusieurs parties asynchrones.
- solutions passées, présentes et à venir...

5.1.3 Exercices et projets :

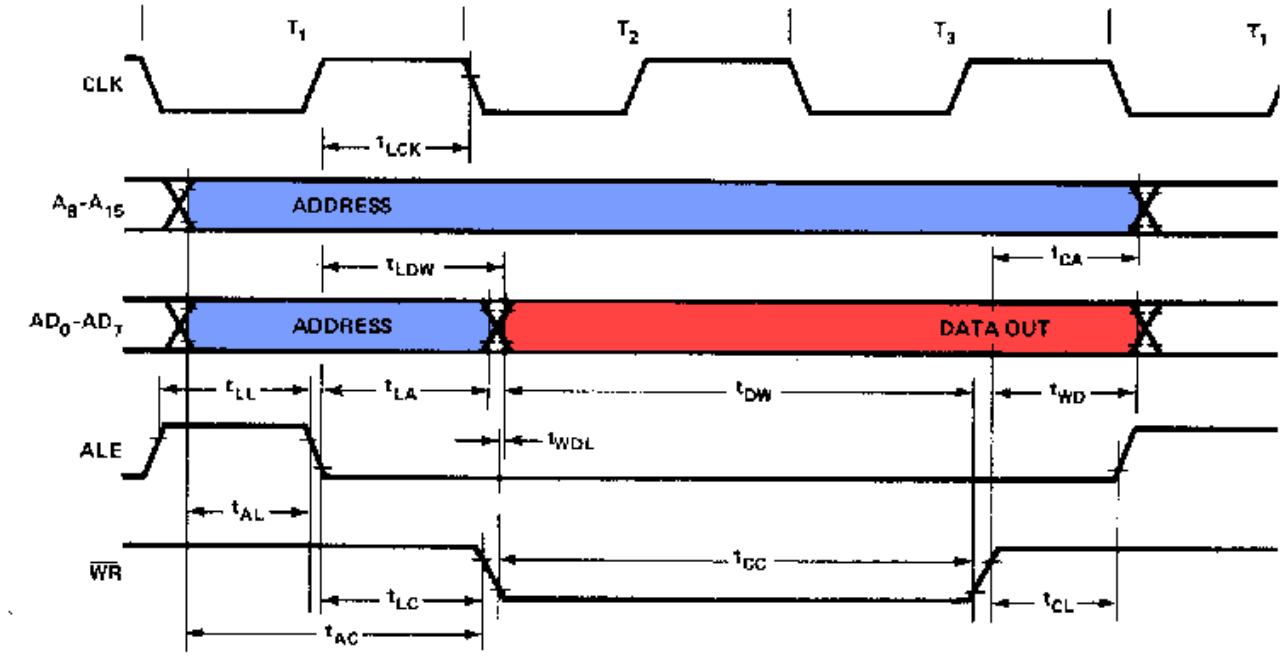
- démultiplexage du bus du 8085
- calcul d'un produit de tableaux & transfert de données
- réalisation matérielle d'un filtre numérique temps réel
- ...

5.2 Exemple de programme pour le 8085

```
Counter List File...

;Four delay schemes and different formats
;to demonstrate upper case or lower case is OK!
;
;This has been machine assembled.
;You should understand each program completely !

2000:            ! 10           ORG   2000H     ;Must be valid RAM 2000 - 27FF
2000:    3E00    ! 11           MVI   A,0       ;Zero A Register
2002:    D303    ! 12           OUT   03        ;Send to LED Port #3
2004:    3C      ! 13           INR   A         ;Add one to A
2005:    C30220  ! 14           JMP   2002H     ;This is very fast...

2100:            ! 17           ORG   2100H
2100:    3E00    ! 18           MVI   A,0       ;Zero Accumulator
2102:    D303    ! 19 OUTIT     OUT   03        ;Put it where you can see it
2104:    3C      ! 20           INR   A         ;Add one to A
2105:    06FF    ! 21           MVI   B,0FFH    ;Want to have Maximum Delay
2107:    05      ! 22 LOOP      DCR   B         ;One less in B
2108:    C20721  ! 23           JNZ   LOOP      ;Circle like a Buzzard til Zer
210B:    C30221  ! 24           JMP   OUTIT     ;Done with delay, procede

2200:            ! 27           ORG   2200H
2200:    3E00    ! 28           MVI   A,0       ;zero the a register
2202:    D303    ! 29 BOB       OUT   03        ;out to lights
2204:    3C      ! 30           INR   A         ;add one to accumulator
2205:    06FF    ! 31 DELAY     MVI   B,0FFH    ;start of delay
2207:    0EFF    ! 32 FILLC     MVI   C,0FFH    ;fill c too with max delay
2209:    0D      ! 33 RUSTI     DCR   C         ;decrement c
220A:    C20922  ! 34           JNZ   RUSTI     ;rusti is my dog!
220D:    05      ! 35           DCR   B         ;c is empty...
220E:    C20722  ! 36           JNZ   FILLC     ;do c again
2211:    00      ! 37           NOP            ;delay is over 255 x 255 loops
2212:    C30222  ! 38           JMP   BOB       ;so back to the counter

2300:            ! 42           ORG   2300H
2300:    313B23  ! 43           LXI   SP,STACK  ;MUST SET SP IF USING SUBROUTI
2303:    AF      ! 44           XRA   A         ;ZERO A
2304:    D303    ! 45 OUTITT    OUT   03        ;OUT TO THE LIGHTS
2306:    CD0522  ! 46           CALL  DELAY     ;SIMPLE CALL TO SUBROUTINE
2309:    3C      ! 47           INR   A         ;BACK FROM SUB, NOW ADD ONE TO
230A:    C30423  ! 48           JMP   OUTITT    ;CONTINUE MAIN PROGRAM
230D:    57      ! 49 DELAYY    MOV   D,A       ;SAVE A
230E:    010080  ! 50           LXI   B,8000H   ;HALF MAXIMUM DELAY
2311:    0B      ! 51 LOOOP     DCX   B
2312:    00      ! 52           NOP
2313:    00      ! 53           NOP
2314:    78      ! 54           MOV   A,B
2315:    B1      ! 55           ORA   C
2316:    C21123  ! 56           JNZ   LOOOP
2319:    7A      ! 57           MOV   A,D       ;RESTORE A
231A:    C9      ! 58           RET            ;RETURN TO MAIN PROGRAM
231B:            ! 59           DS    20H       ;SAVE 32 BYTES FOR THE STACK
233B:            ! 60 STACK     DS    1         ;THE STACK STARTS HERE
233C:            ! 61           END            ;END TELLS THE ASSEMBLER TO ST


End of Assembly
No Errors
```

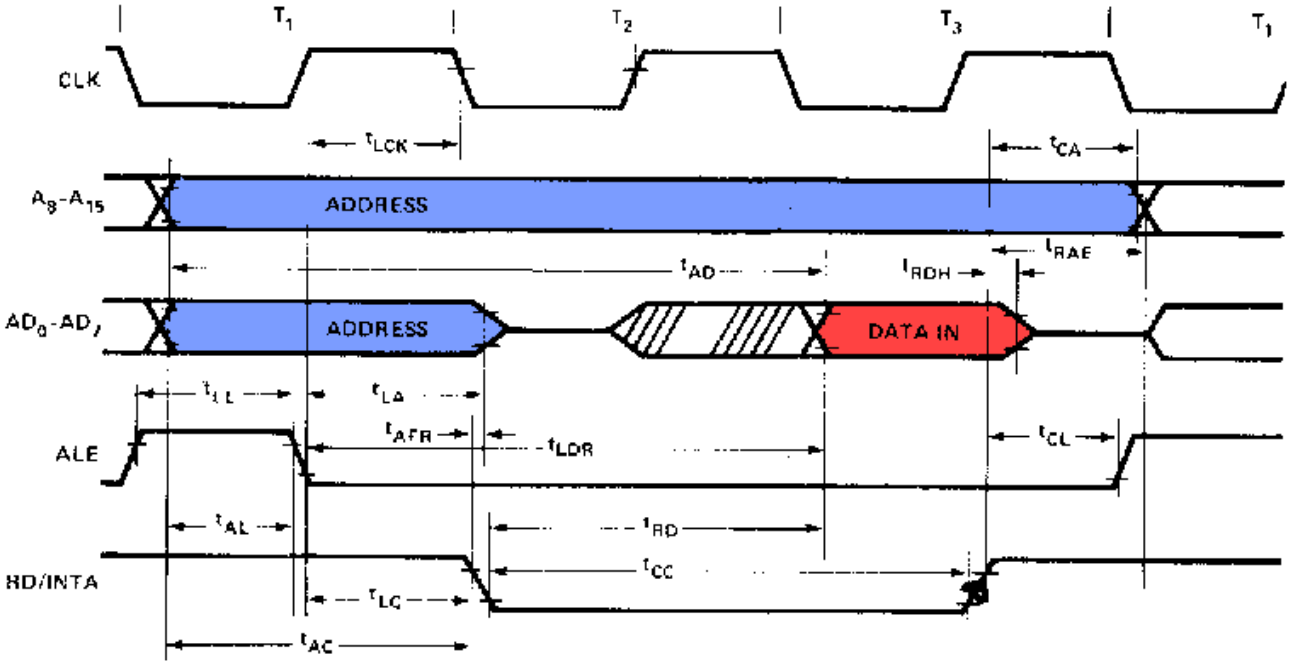
5.3 Cycles d'écriture et de lecture du 8085

5.3.1 Cycle d'écriture: le front montant de /WR enregistre D_{0-7} dans le composant interfacé



t_{AC}	t_{AD}	t_{AL}	t_{AFR}	t_{CA}	t_{CC}	t_{CL}	t_{LA}	t_{LC}	t_{LCK}	t_{LDR}	t_{LL}	t_{RAE}	t_{RD}	t_{RDH}
>270	<575	>115	<0	>120	>400	>50	>100	>130	>100	>460	>140	>150	<300	>0

5.3.2 Cycle de lecture: le front montant de /RD enregistre D_{0-7} dans le microprocesseur

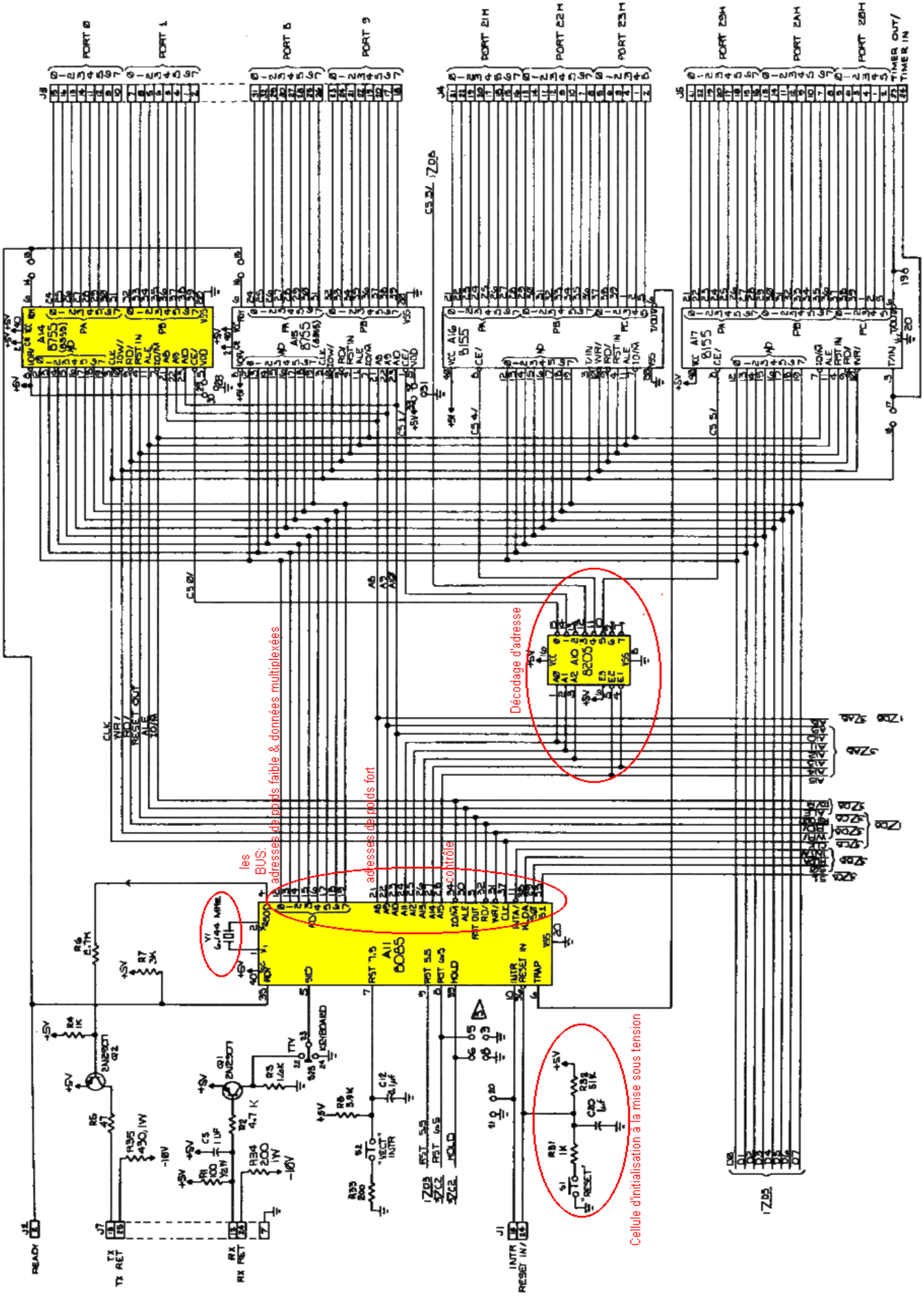


t_{AC}	t_{AL}	t_{CA}	t_{CC}	t_{CL}	t_{DW}	t_{LA}	t_{LC}	t_{LCK}	t_{LDW}	tLL	t_{WD}	t_{WDL}
>270	>115	>120	>400	>50	>420	>100	>130	>100	<200	>140	>100	<40

Caractéristiques de 8085AH ; les valeurs de ces tableaux sont exprimés en ns
En salle de TP, la fréquence du quartz est de 6,144Mhz, ce qui donne un temps de cycle Tcy=320ns

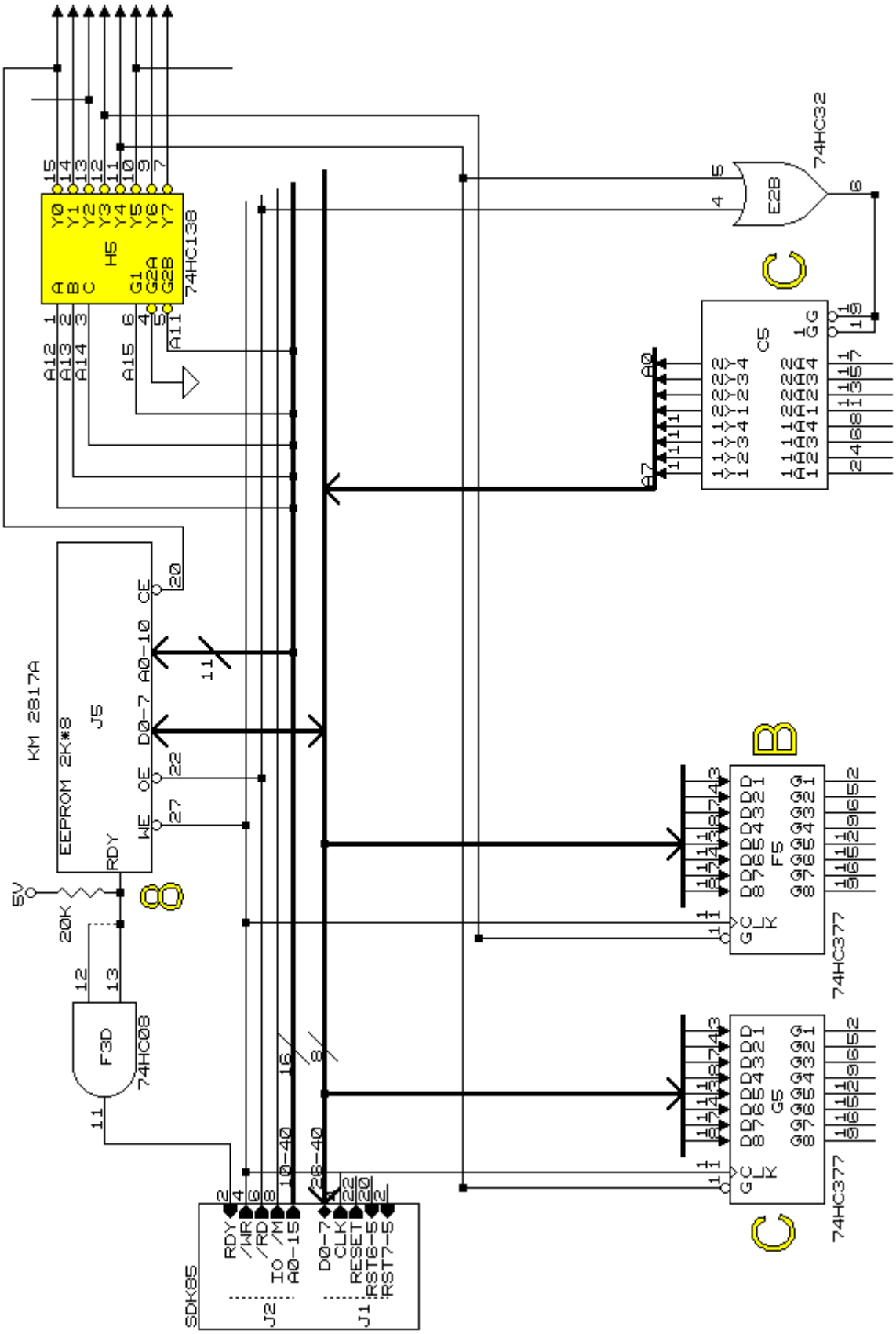
5.4 Plan de la maquette SDK85

*(8155 = RAM 256*8 +timer + ports A, B, C ; 8755 = EPROM 2K*8 + ports A, B)*

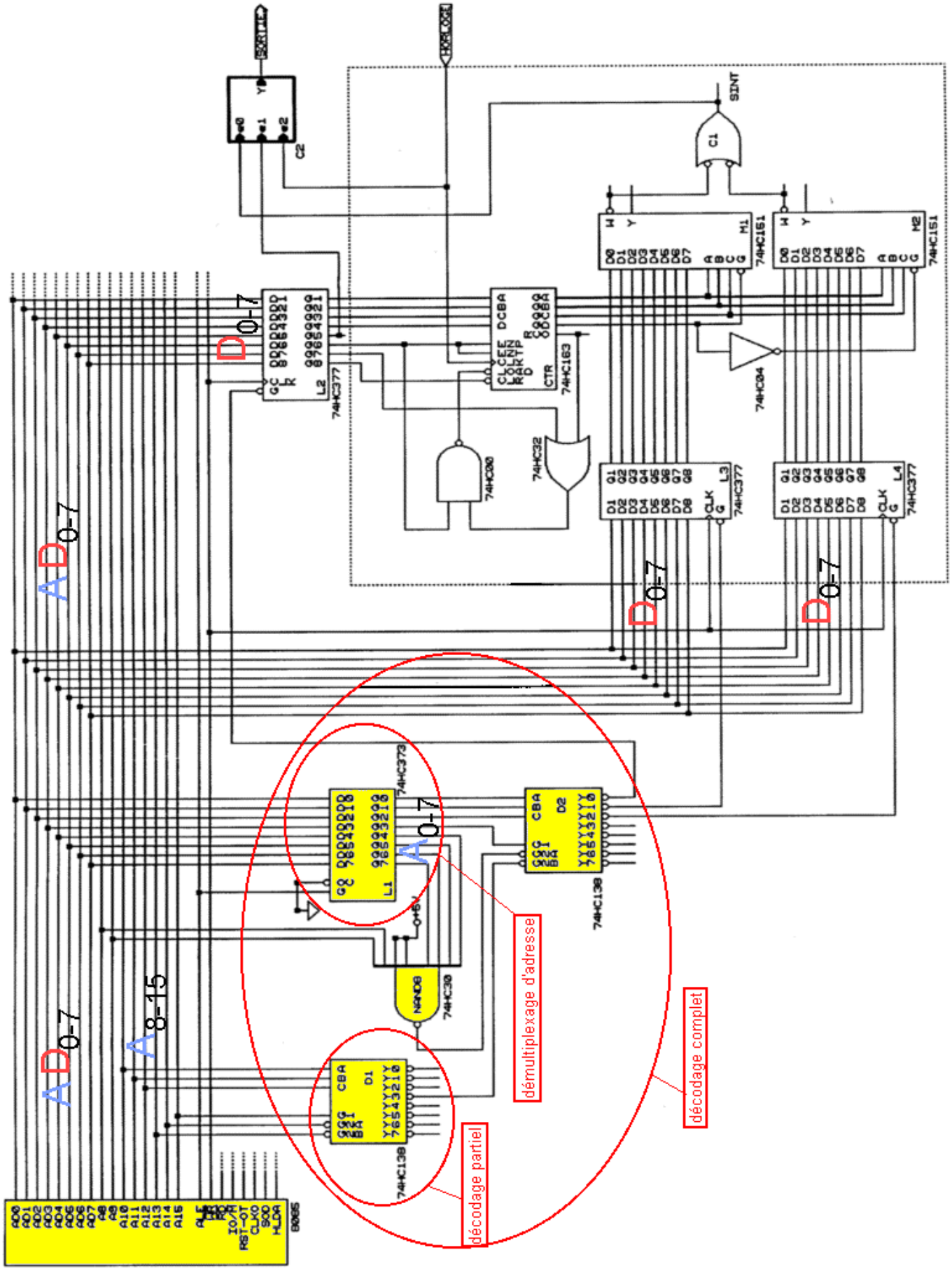


6. Exercices sur les bus et microprocesseurs

6.1 Décodage partiel de l'adresse

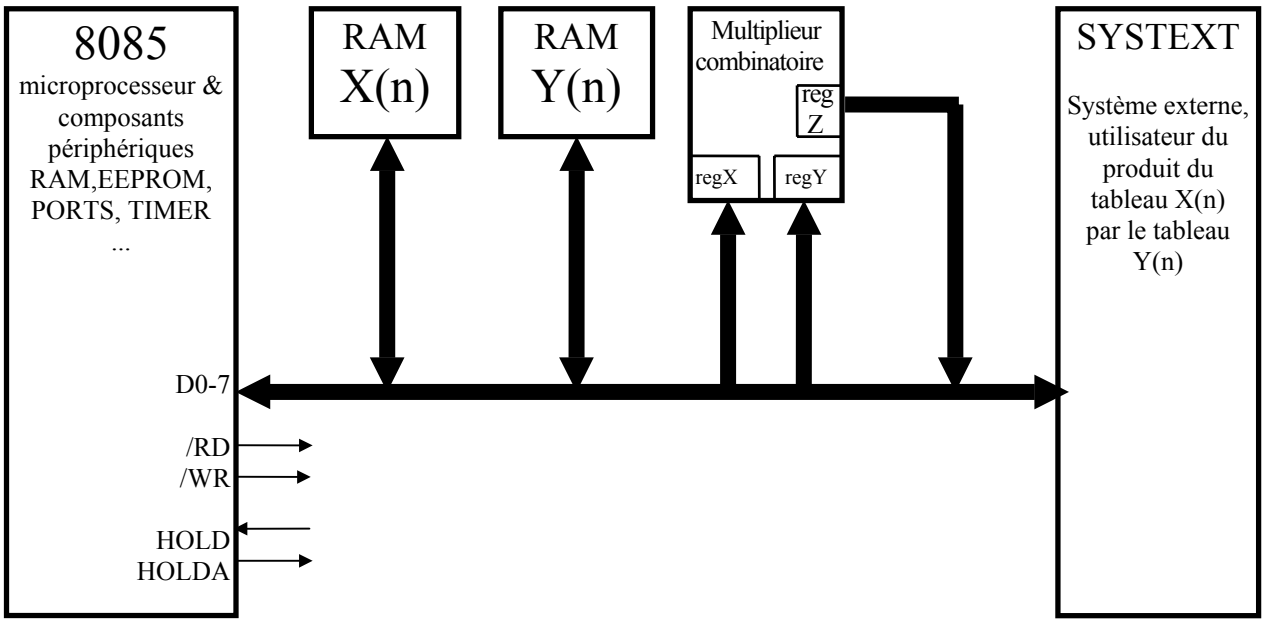


6.2 Démultiplexage du bus multiplexé AD_{0-7} et décodage complet de l'adresse



7. Compromis matériel-logiciel, "hard" & "soft", "espace", "temps"

7.1 Produit & transfert de deux tableaux : un compromis entre « hard » et « soft » :



Soit à faire le produit 2 à 2 de données contenues dans deux tableaux de 1024 éléments chacun. Soit Z(n)=X(n)*Y(n), pour n=0 à 1023, les produits à réaliser et à déposer dans SYSTEXT. Pour ne pas faire de multiplication logicielle (solution soft), on utilise un multiplieur combinatoire (solution hard) qui effectue le produit X*Y en 50ns. Ce multiplieur dispose de deux registres 8 bits d'entrée REGX, REGY et de deux registres de sortie 8 bits REGZSup et REGZInf.
Nous supposerons que les tableaux X(n) et Y(n) ont déjà été remplis par le système architecturé autour du microprocesseur.
Les solutions suivantes sont envisageables. Mélangeages entre elles, elles possèdent chacune des variantes:

7.1.1 Solution n°1, standard

Le microprocesseur est seul et muni d'une structure standard de lecture et d'écriture et de décodage d'adresse.
Supposons que l'adresse de X soit pointée par HL, celle de Y par DE, et que
@(SYSTEXT)=8000_H , @(REGX)=8001_H ,@(REGY)=8002_H , @(REGZS)=8003_H et @(REGZI)=8004_H
Le microprocesseur gère donc les adresses et doit réaliser 1024 fois les opérations suivantes correspondant chacune à plusieurs instructions d'une durée d'environ 1 µs chacune:

cycles			instructions	
1.	lecture soft de X(n) qui est mis dans l'accumulateur	(/RD)	*MOV A,M*	*7*
2.	écriture soft de l'accumulateur dans REGX du multiplieur	(/WR)	*STA 800113*	
3.	lecture soft de Y(n) qui est mis dans l'accumulateur	(/RD)	*LDAX D*	7
4.	écriture soft de l'accumulateur dans REGY du multiplieur	(/WR)	*STA 800213*	
5.	lecture soft de Z(n) contenu dans REGZS qui est mis dans l'accumulateur	(/RD)	*LDA 8003*	*13*
6.	écriture soft de l'accumulateur dans SYSTEXT	(/WR)	*STA 800013*	
7.	lecture soft de Z(n) contenu dans REGZI qui est mis dans l'accumulateur	(/RD)	*LDA 8004*	*13*
8.	écriture soft de l'accumulateur dans SYSTEXT	(/WR)	*STA 800013*	
9.	incrémenter DE et HL			

Cette solution nécessite un temps de transfert d'environ 35ms.

- *Quelles astuces matérielles (hard) et logicielles (soft) permettraient de mieux employer les registres et d'économiser des instructions et du temps de calcul.*

7.1.2 Solution n°2, basée sur des opérations simultanées de lecture et d'écriture

Le microprocesseur est associé à une structure adaptée de décodage d'adresse et de transfert dans les mémoires, registres du multiplieur et SYSTEXT. Il gère les adresses et peut dans ce cas réaliser ces opérations par paires 1&2, 3&4, 5&6, 7&8. Dans ce cas, le contenu de l'accumulateur ne sert pas et le microprocesseur doit réaliser 1024 fois les opérations suivantes correspondant chacune à un jeu d'instructions. Dans cette version le signal /RD du microprocesseur est utilisé pour réaliser des opérations simultanées de lecture et d'écriture et donc /WR n'est jamais utilisé. Par rapport à la version précédente, il y a un gain de vitesse de l'ordre d'un facteur 2.

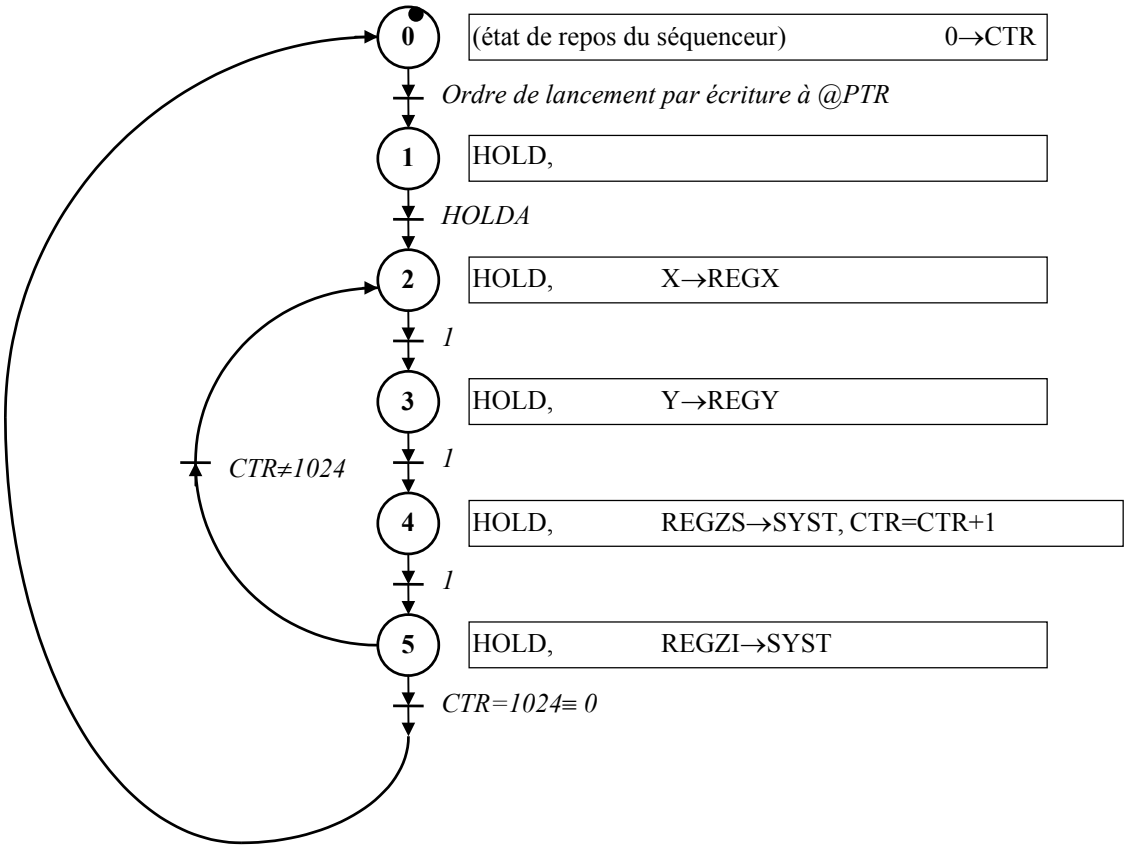
- lecture soft de X(n) qui est ainsi mis sur le bus et écriture hard par /RD dans REGX du multiplieur *MOV A,M*
- lecture soft de Y(n) qui est ainsi mis sur le bus et écriture hard par /RD dans REGY du multiplieur LDAX D
- lecture soft de Z(n) dans REGZS qui est ainsi mis sur le bus et écriture hard par /RD dans SYSTEXT LDA 8003
- lecture soft de Z(n) dans REGZI qui est ainsi mis sur le bus et écriture hard par /RD dans SYSTEXT LDA8004
- incrémenter n

- ***Problème à résoudre:*** *comment effectuer un décodage d'adresses et une gestion de /RD et/WR qui permette ces lectures et écritures simultanées?*

7.1.3 Solution n°3, basée sur un transfert avec accès direct aux mémoires (DMA)

Le microprocesseur "lâche la main" car le matériel réalisé autorise un transfert du type DMA supervisé par un processeur de transfert PTR. Sur demande de HOLD en provenance du PTR, le microprocesseur répond par son accord (HOLDA). Il laisse ainsi les bus d'adresses et de données et les signaux /RD et /WR à la libre disposition du PTR et d'un système de gestion des adresses. Cette solution qui autorise le fonctionnement lecture/écriture simultanés précédent se fait à la cadence du processeur de transfert qui n'est limitée que par le temps de calcul du produit X*Y ou par la rapidité des bus utilisés. Elle peut ainsi permettre de réaliser un transfert en 1024*3*50ns, soit 160µs, soit un gain de rapidité de l'ordre de 200 par rapport à la première solution.

- *Comment réaliser ce processeur de transfert et le système de gestion des adresses?*
  On peut utiliser un compteur modulo 1024 (CTR) affecté à l'adressage des mémoires X et Y et un processeur de transfert PTR qui est une machine d'états construite selon l'architecture "Flip-flop+ Mémoire" ou "Compteur + Multiplexeur + Mémoire" ou "Bascules JK", le tout étant réalisé avec des ***éléments discrets et/ou des PAL/PLA ou FPGA***
  Voici ci-dessous un exemple de RdP du processeur de transfert.
  Noter qu'il est aussi possible de trouver des variantes fusionnant PTR et CTR.!



7.1.4 Solution n°4, basée sur la création de bus internes spécifiques

On peut aller encore plus loin dans cette démarche en créant des bus internes spécifiques plus ou moins (in)dépendants des bus du microprocesseur...(page suivante)

7.1.5 Quelle est la meilleure solution. Comment choisir?

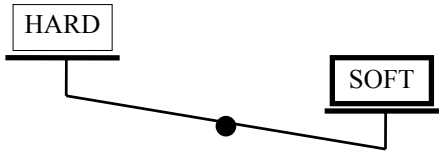
On voit sur cet exemple simple la **multiplicité** des solutions et le **compromis** à trouver entre matériel et logiciel.

Chacune des solutions précédentes peut s'associer aux autres pour résoudre le problème posé et effectuer les opérations demandées en un temps record, mais à quel prix et pour quel encombrement.
Certaines astuces ne coûtent rien si ce n'est de savoir les imaginer, d'autres ont un prix.

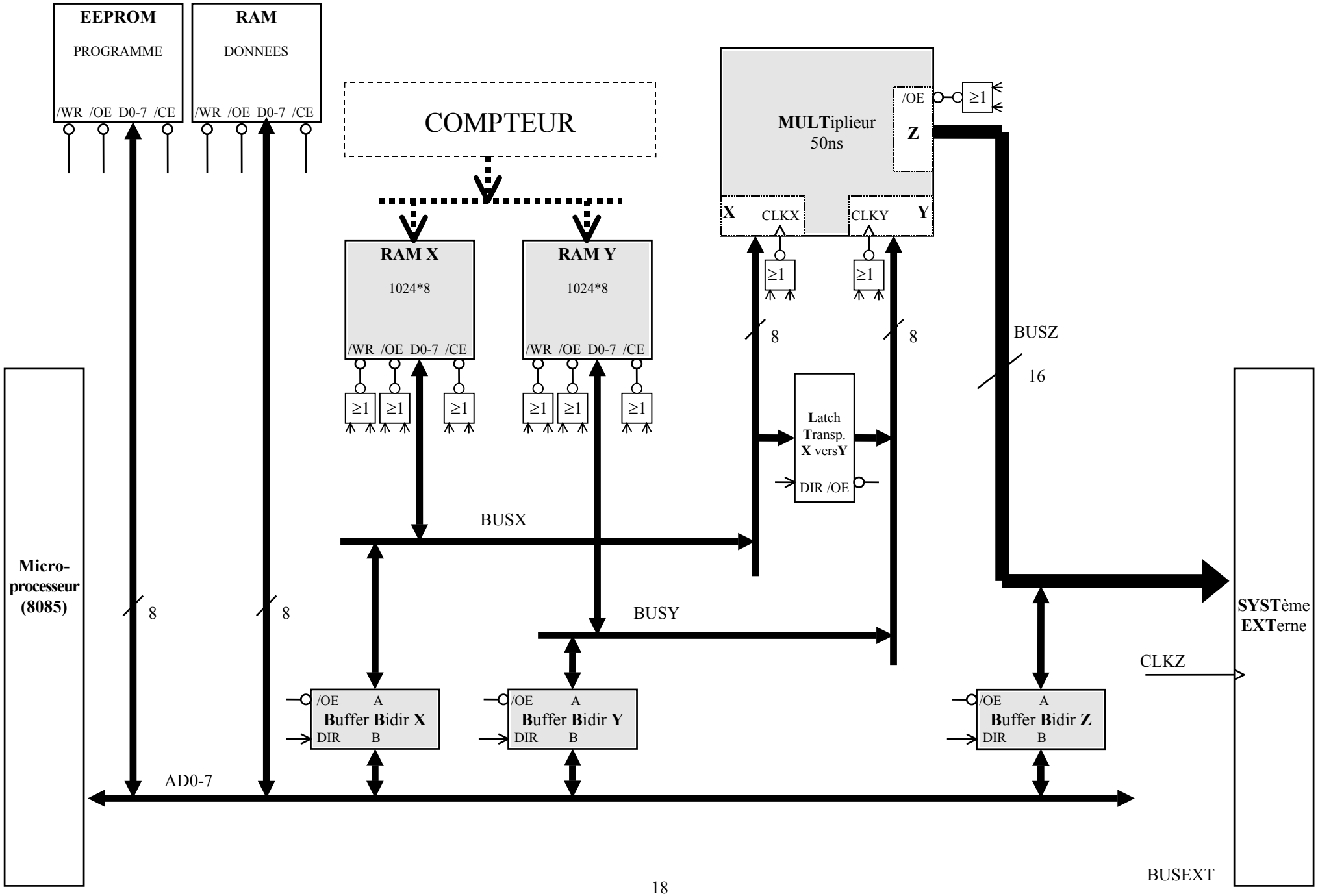
Il est évident que l'on ne peut choisir que parmi les solutions qui nous sont accessibles
On est donc ramené à une question d'ouverture, de choix et d'équilibre.

Vous pouvez aussi compléter ce tableau

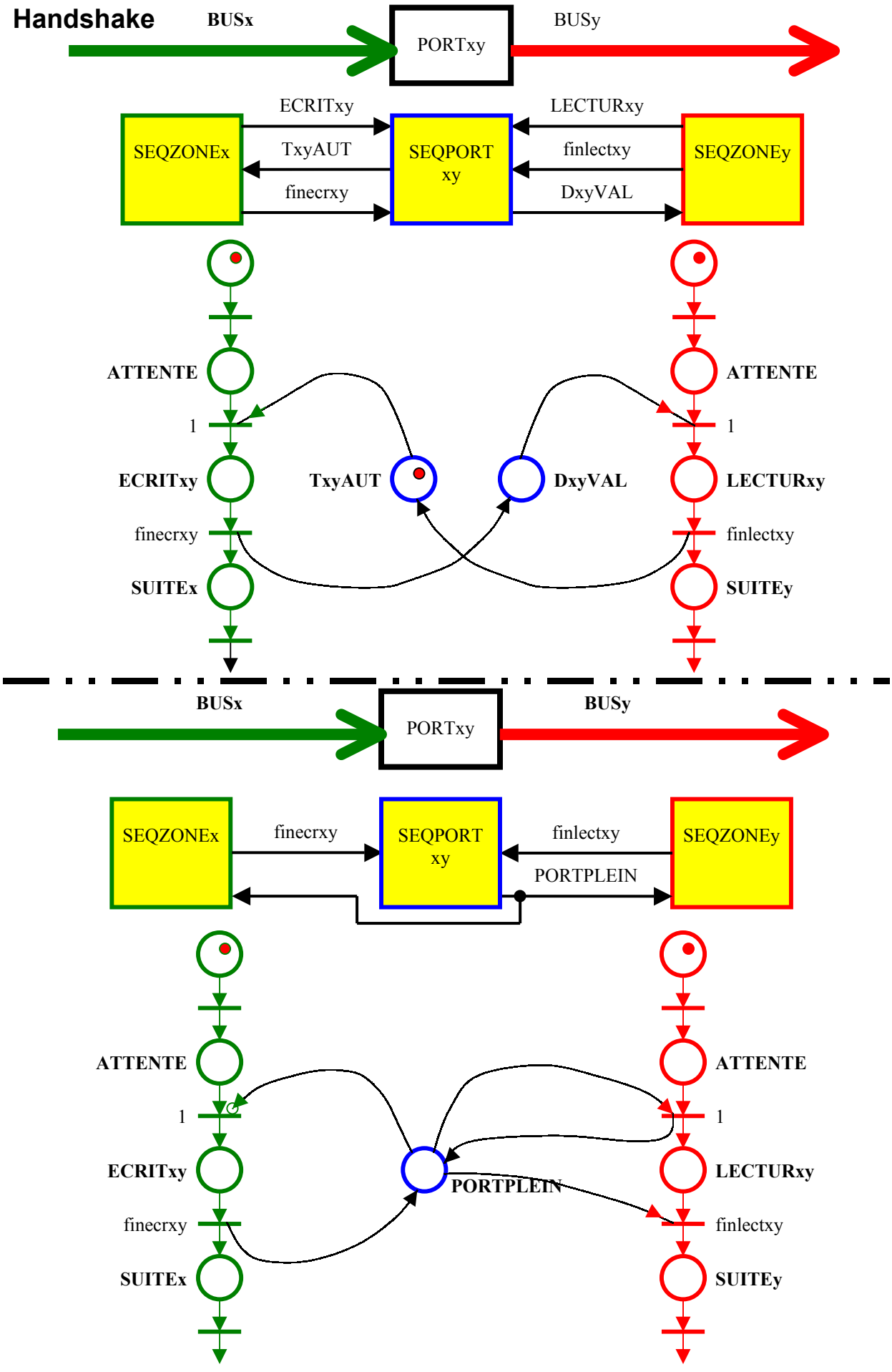
	Soft seul	**Soft+Hard**	**Hard seul**
Temps de développement			
Vitesse d'exécution			
Encombrement			
Convivialité			
Prix			



Libre à vous d'imaginer et de choisir votre solution qui équilibrera la balance!


EEPROM
PROGRAMME
/WR /OE D0-7 /CE
RAM
DONNEES
/WR /OE D0-7 /CE
COMPTEUR
MULTiplieur
50ns
/OE
Z
≥1
X
CLKX
CLKY
Y
RAM X
1024*8
/WR /OE D0-7 /CE
RAM Y
1024*8
/WR /OE D0-7 /CE
≥1
8
8
BUSZ
16
Latch
Transp.
X versY
DIR /OE
BUSX
BUSY
Micro-
processeur
(8085)
8
8
AD0-7
/OE A
Buffer Bidir X
DIR B
/OE A
Buffer Bidir Y
DIR B
/OE A
Buffer Bidir Z
DIR B
SYSTème
EXTerne
CLKZ
BUSEXT

8. Handshake


BUSx
PORTxy
BUSy
ECRITxy
LECTURxy
SEQZONEx
TxyAUT
SEQPORT xy
finlectxy
SEQZONEy
finecrxy
DxyVAL
ATTENTE
ATTENTE
1
1
ECRITxy
TxyAUT
DxyVAL
LECTURxy
finecrxy
finlectxy
SUITEx
SUITEy
BUSx
PORTxy
BUSy
SEQZONEx
finecrxy
SEQPORT xy
finlectxy
SEQZONEy
PORTPLEIN
ATTENTE
ATTENTE
1
1
ECRITxy
PORTPLEIN
LECTURxy
finecrxy
finlectxy
SUITEx
SUITEy

9. Autres exemples

9.1 Filtre numérique

Il s'agit de concevoir un filtre numérique d'ordre k qui, recevant un signal d'entrée analogique, fournit en sortie un signal analogique filtré.

Fonction de transfert du filtre (pour k=3):

$$F(z) = \frac{b_3 z^{-1} + b_2 z^{-2} + b_1 z^{-3}}{1 + a_3 z^{-1} + a_2 z^{-2} + a_1 z^{-3}} = \frac{S(z)}{E(z)}$$

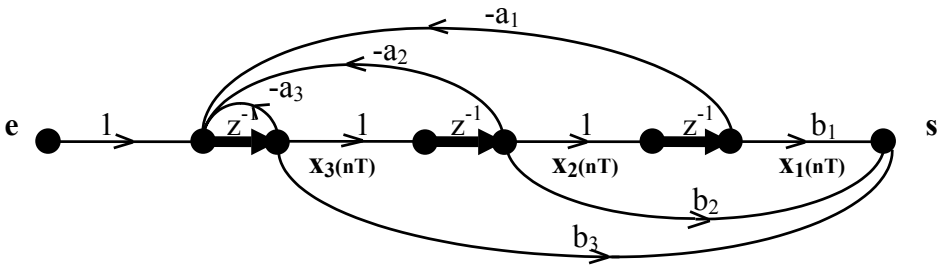
Équations d'état associées à la forme canonique gouvernable de F(z):

- $x_1(n+1) = x_2(n)$
- $x_2(n+1) = x_3(n)$
- $x_3(n+1) = -[a_1x_1(n) + a_2x_2(n) + a_3x_3(n)] + e(n)$

avec

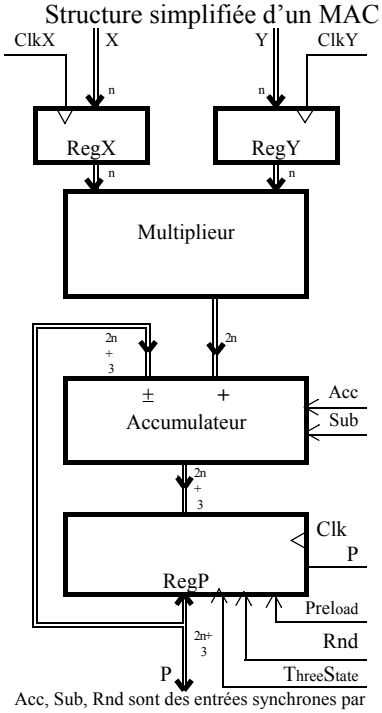
- $s(n) = b_1x_1(n) + b_2x_2(n) + b_3x_3(n)$

Graphe de fluence de la forme canonique gouvernable:



1. Étudier et dessiner le schéma d'une réalisation matérielle de F(z) utilisant autant de multiplieurs et d'additionneurs soustracteurs qu'il est nécessaire (k=3). Quels sont les avantages et les inconvénients de cette réalisation?
2. Étudier et dessiner le schéma d'une réalisation matérielle d'un filtre d'ordre k utilisant un seul multiplieur-accumulateur (MAC). Avantages et inconvénients de cette réalisation?
   - 2.1. Quel chemin (matériel, bus...) pour les données et les coefficients?
   - 2.2. Quel chemin (matériel, bus...) pour les adresses de ces données et coefficients?
   - 2.3. Quel séquenceur pour assurer la gestion de l'ensemble?
   - 2.4. Comment interfacer ce filtre et un microprocesseur pour entrer les coefficients?

Structure simplifiée d'un MAC



Acc, Sub, Rnd sont des entrées synchrones par rapport à ClkX + ClkY ; Prel est une entrées synchrone vis à vis de ClkP ; n= 8 ou 16

2's complement	décomposition avec la pondération -1, 1/2, 1/4, 1/8				base 10	autre exemple avec pondération 8, 4, 2, 1
1000	= -1	+0	+0	+0	**-1,000**	**-8**
1001	= -1	+0	+0	+0,125	**-0,875**	**-7**
1010	= -1	+0	+0,25	+0	**-0,750**	**-6**
1011	= -1	+0	+0,25	+0,125	**-0,625**	**-5**
1100	= -1	+0,5	+0	+0	**-0,500**	**-4**
1101	= -1	+0,5	+0	+0,125	**-0,375**	**-3**
1110	= -1	+0,5	+0,25	+0	**-0,250**	**-2**
1111	= -1	+0,5	+0,25	+0,125	**-0,125**	**-1**
0000	= 0	+0	+0	+0	**0,000**	**0**
0001	= 0	+0	+0	+0,125	**+0,125**	**1**
0010	= 0	+0	+0,25	+0	**+0,250**	**2**
0011	= 0	+0	+0,25	+0,125	**+0,375**	**3**
0100	= 0	+0,5	+0	+0	**+0,500**	**4**
0101	= 0	+0,5	+0	+0,125	**+0,625**	**5**
0110	= 0	+0,5	+0,25	+0	**+0,750**	**6**
0111	= 0	+0,5	+0,25	+0,125	**+0,875**	**7**

Exemple sur 4 bits du format binaire complément à deux (2's complement):

9.2 Fonctionnement d'un multiplieur accumulateur (cf. filtre numérique et FFT)

X sur 16 bits & Y sur 16 bits ⇒ le résultat X*Y sera sur 32 bits.
⇒ son cumul par additions ou soustractions successives peut dépasser 32 bits. Donc, avec ses 35 bits de sortie, ce MAC autorise un dépassement de 3 bits.

Registres d'entrée du MAC

Sur le front ↑ de ClkX	Sur le front ↑ de ClkY
RegX = X	RegY = Y

Fonctionnement de l'accumulateur

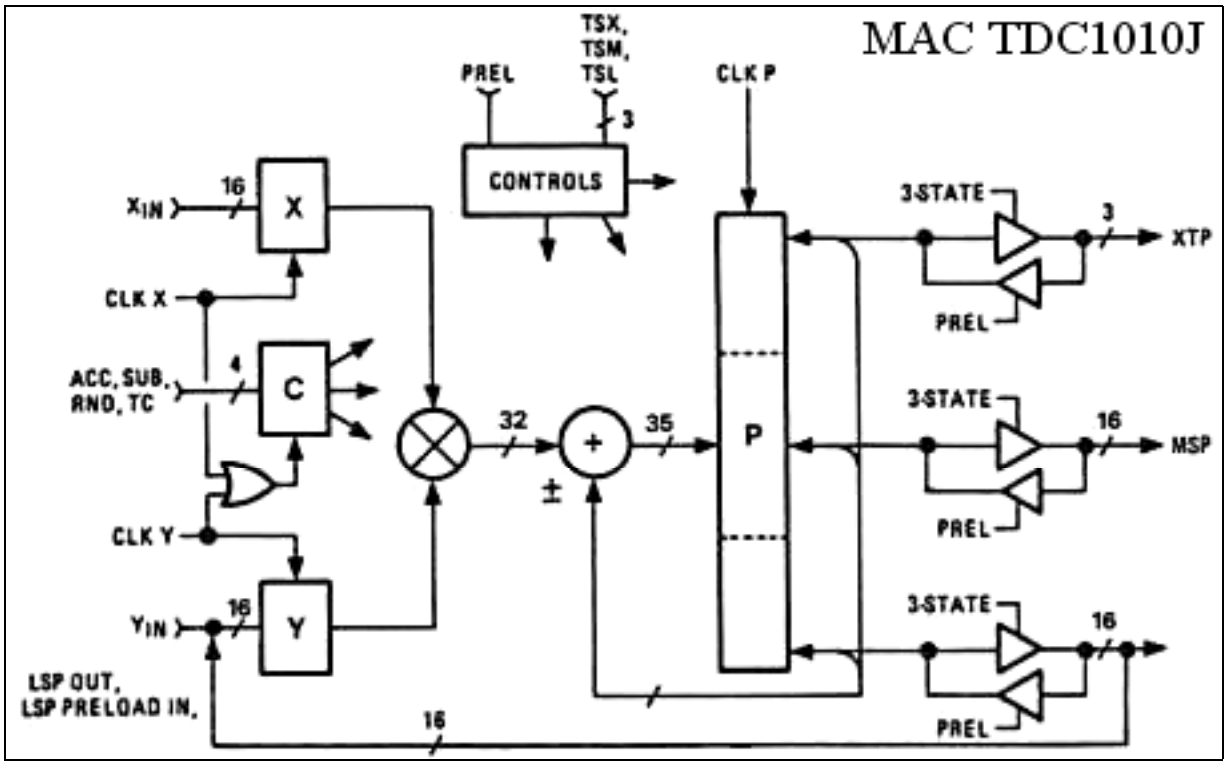
Prel	Acc*	Sub*	Sur le front ↑ de ClkP	Commentaires
0	0	-	$RegP^+ = RegX * RegY$	Multiplication seule
0	1	0	$RegP^+ = RegX * RegY + RegP$	Multiplication & addition
0	1	1	$RegP^+ = RegX * RegY - RegP$	Multiplication & soustraction
1	-	-	$RegP^+ = P$	Pré chargement du registre

Acc et Sub sont latchés sur ↑ClkX ou ↑ClkY ; Acc et Sub* sont leurs valeurs enregistrées
Abréviations : **Prel**oad, **Acc**umulate, **Sub**tract ; - = indifférent

Sorties du MAC

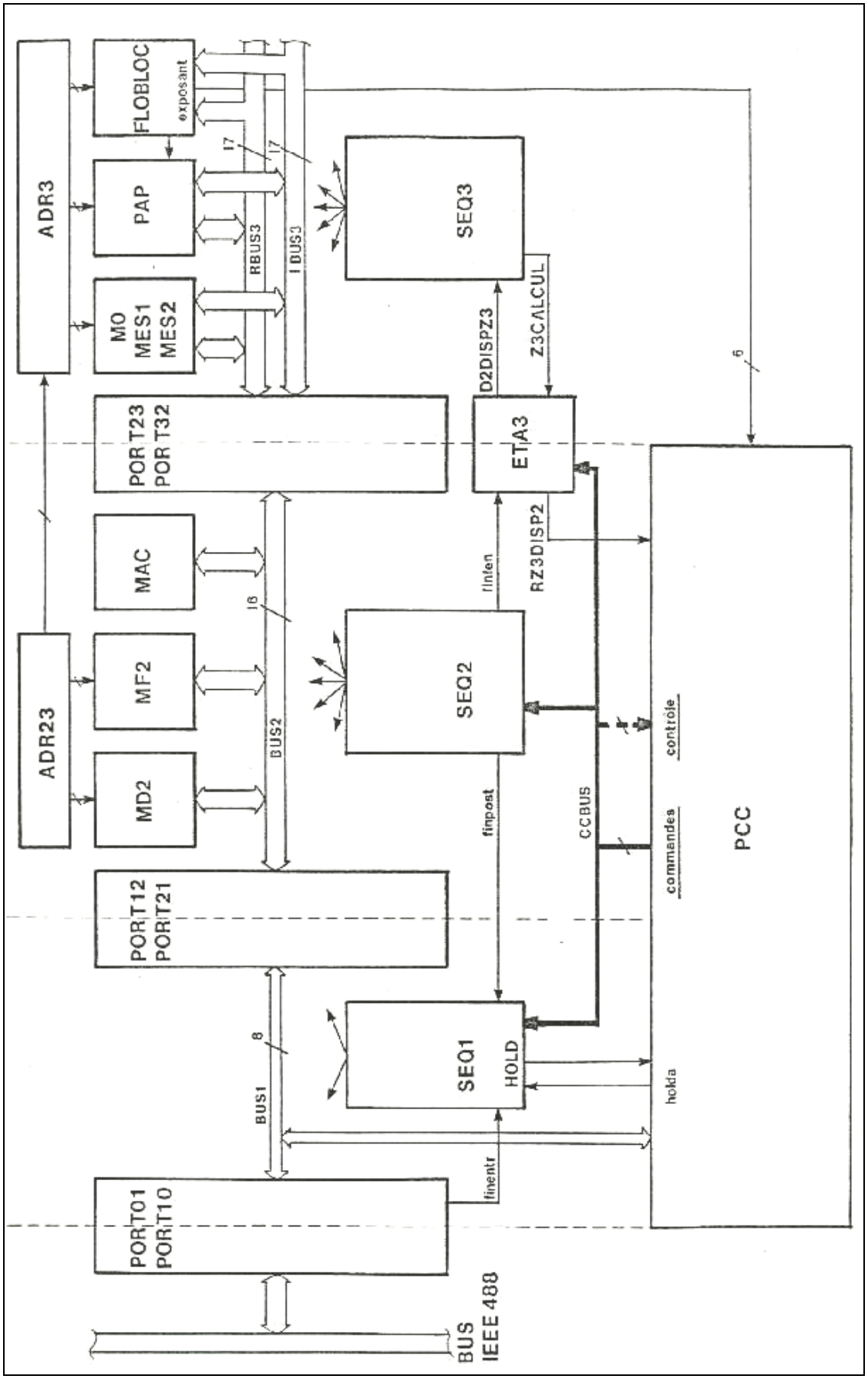
Rnd*	TS	P	Commentaires
0	0	P = RegP	La sortie P est sur 3+32 bits
1	0	P = RegP	La sortie P est arrondie à 3+16 bits
-	1	P = Haute impédance	

Rnd est latché sur ↑ClkX ou ↑ClkY ; Rnd est la valeur enregistrée
Abréviations : **R**ou**nd**, **T**hree **S**tate

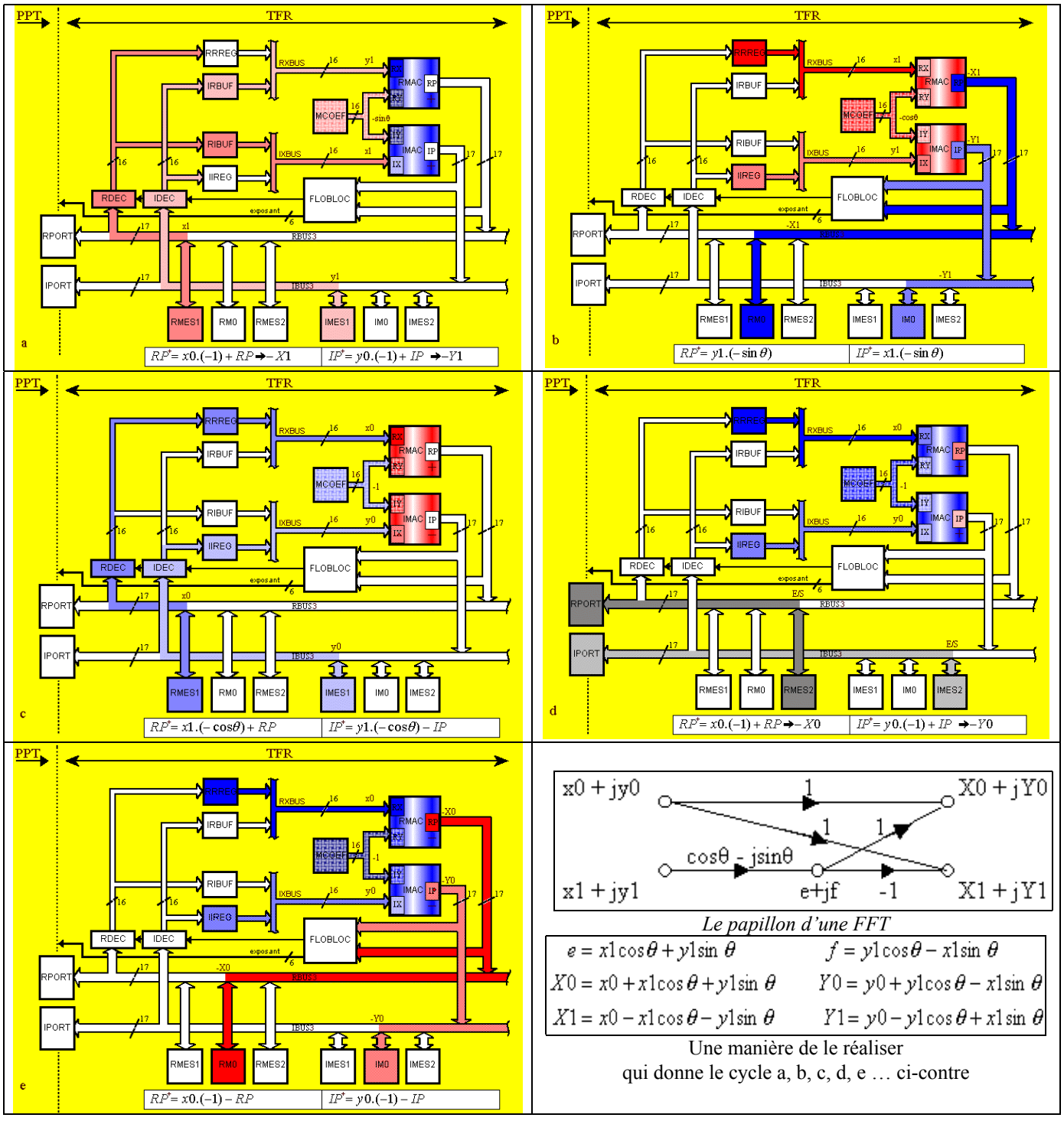


9.3 Exemple d'une FFT

9.3.1 Vue d'ensemble



9.3.2 Hardware et timing du papillon de la FFT



Le papillon d'une FFT

$$e = x1\cos\theta + y1\sin\theta \qquad f = y1\cos\theta - x1\sin\theta$$
$$X0 = x0 + x1\cos\theta + y1\sin\theta \qquad Y0 = y0 + y1\cos\theta - x1\sin\theta$$
$$X1 = x0 - x1\cos\theta - y1\sin\theta \qquad Y1 = y0 - y1\cos\theta + x1\sin\theta$$

Une manière de le réaliser
qui donne le cycle a, b, c, d, e … ci-contre

Phase	a	b	c	d	e	a	b	c
RBUS3,IBUS3	x1,y1	-X1,-Y1	x0,y0	*Opération d'E/S avec RMES2, IMES2*	-X0,-Y1	x1,y1	-X1,-Y1	x0,y0
Lecture de	RMES1 IMES1		RMES1 IMES1			RMES1 IMES1		RMES1 IMES1
IRBUF	y1	-	-	-	-	y1	-	-
RIBUF	x1					x1		
RRREG	-	x1	x0	x0	x0	-	x1	x0
IIREG		y1	y0	y0	y0		y1	y0
RXBUX	y1	x1	x0	x0	x0	y1	x1	x0
IXBUS	x1	y1	y0	y0	y0	x1	y1	y0
MCOEF	-sinθ	-cosθ	-1	-1	-1	-sinθ	-cosθ	-1
Calcul RMAC Calcul IMAC	x0.(-1)+RP y0.(-1)+IP	y1.(-sinθ) x1.(-sinθ)	x1.(-cosθ)+RP y1.(-cosθ)-IP	x0.(-1)+RP y0.(-1)+IP	x0.(-1)-RP y0.(-1)-IP	x0.(-1)+RP y0.(-1)+IP	y1.(-sinθ) x1.(-sinθ)	x1.(-cosθ)+RP y1.(-cosθ)-IP
RP IP	-[-x1cosθ-y1sinθ] -[-y1cosθ+x1sinθ]	-[x0-x1cosθ-y1sinθ] -[y0-y1cosθ+x1sinθ]	-y1sinθ -x1sinθ	-[x1cosθ+y1sinθ] -[y1cosθ-x1sinθ]	-[x0+x1cosθ+y1sinθ] -[y0+y1cosθ-x1sinθ]	-[-x1cosθ-y1sinθ] -[-y1cosθ+x1sinθ]	-[x0-x1cosθ-y1sinθ] -[y0-y1cosθ+x1sinθ]	-y1sinθ -x1sinθ
Ecriture dans		RM0 IM0			RM0 IM0		RM0 IM0	
RBUS3,IBUS3	x1,y1	-X1,-Y1	x0,y0	*Opération d'E/S avec RMES2, IMES2*	-X0,-Y1	x1,y1	-X1,-Y1	x0,y0
En jaune les opérations concernant le papillon en cours; en gris clair celles du papillon précédent; en gris foncé celles du papillon suivant.								

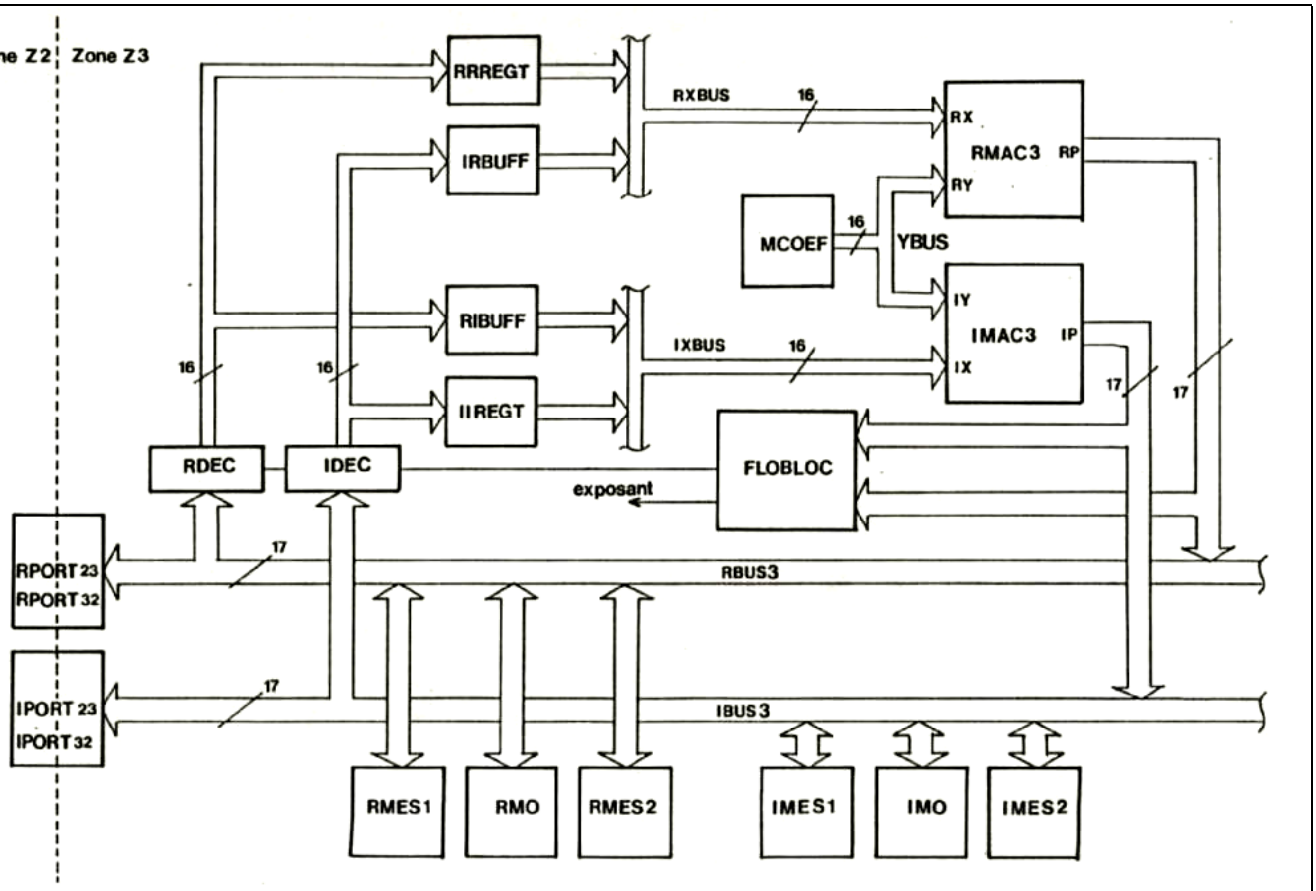


Figure II-8 : Architecture du chemin des données du processeur P3 ou processeur FFT.

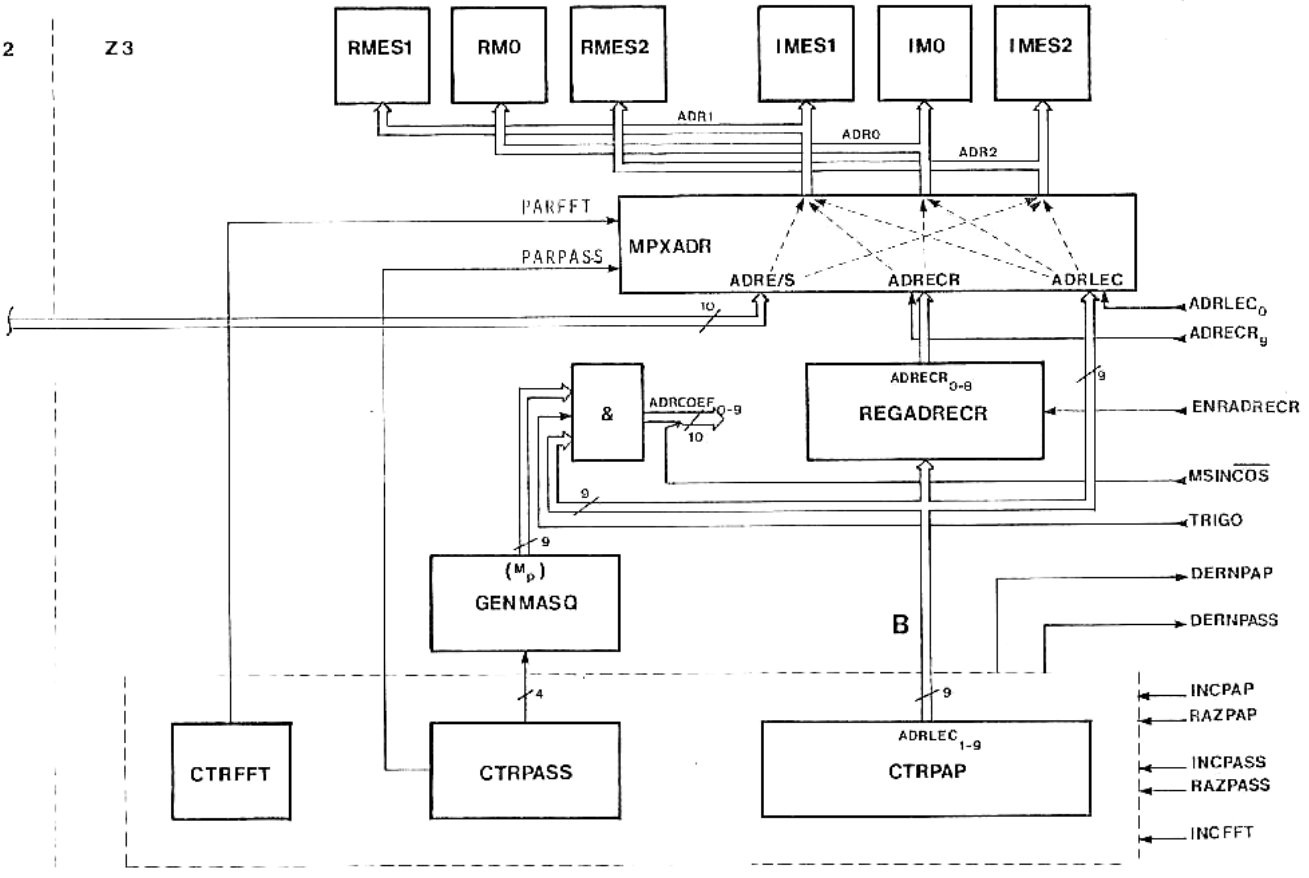
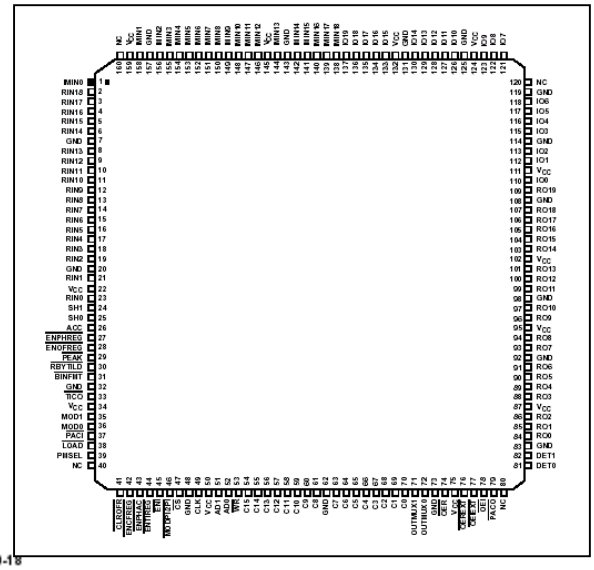


Figure II.10 : Architecture de calcul des adresses des mémoires et de la table de coefficients.

9.4 Architecture interne d'un DSP, le HSP 45116



160 broches…

Le cœur de ce DSP où l'on peut reconnaître 2 MAC

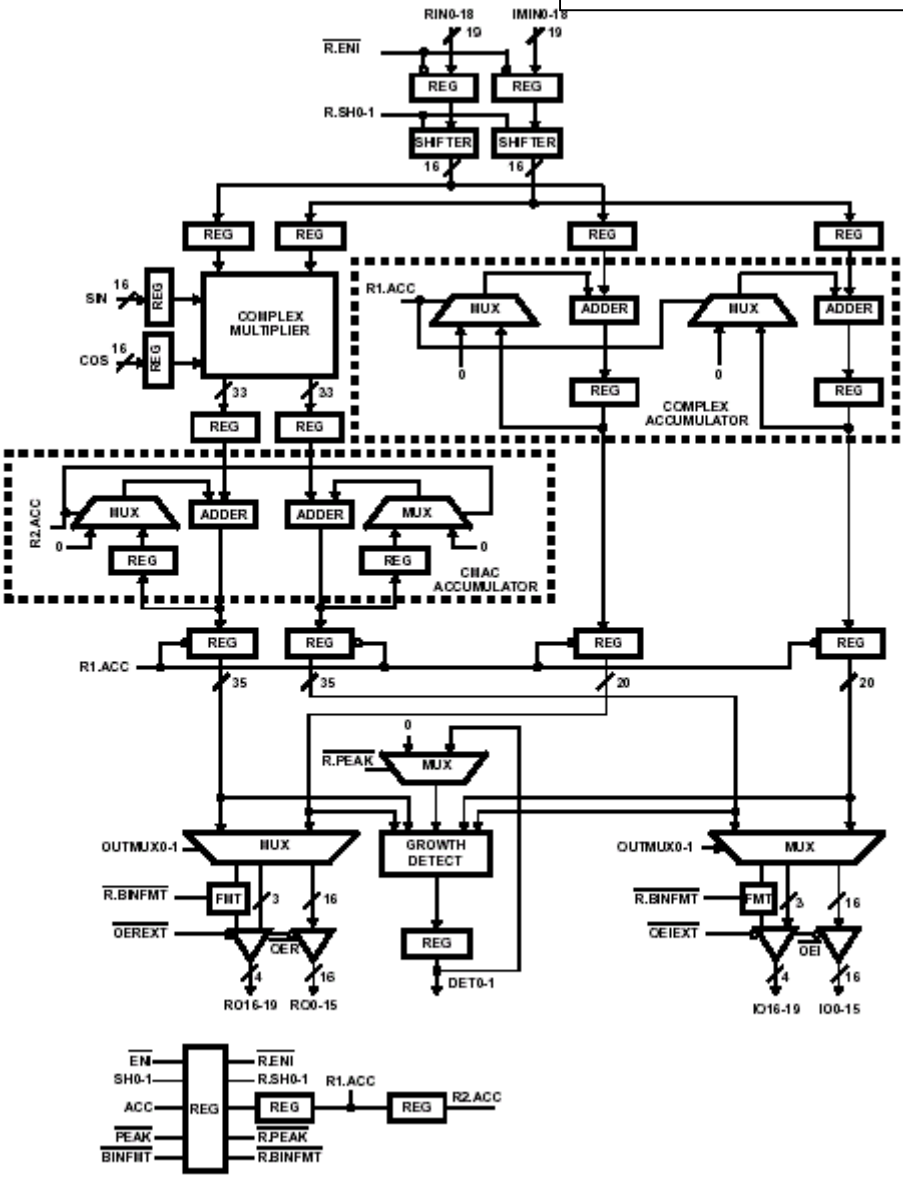
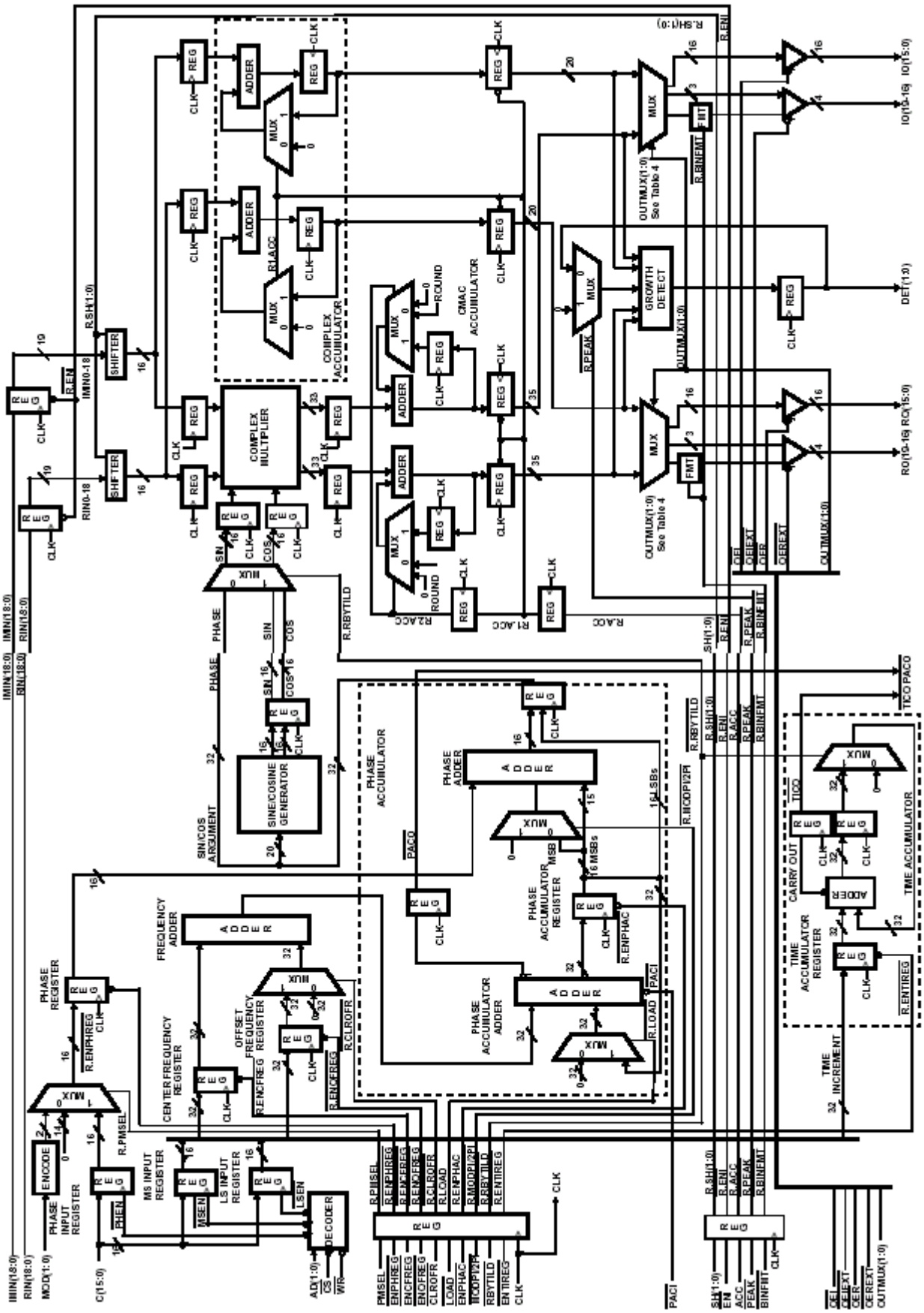


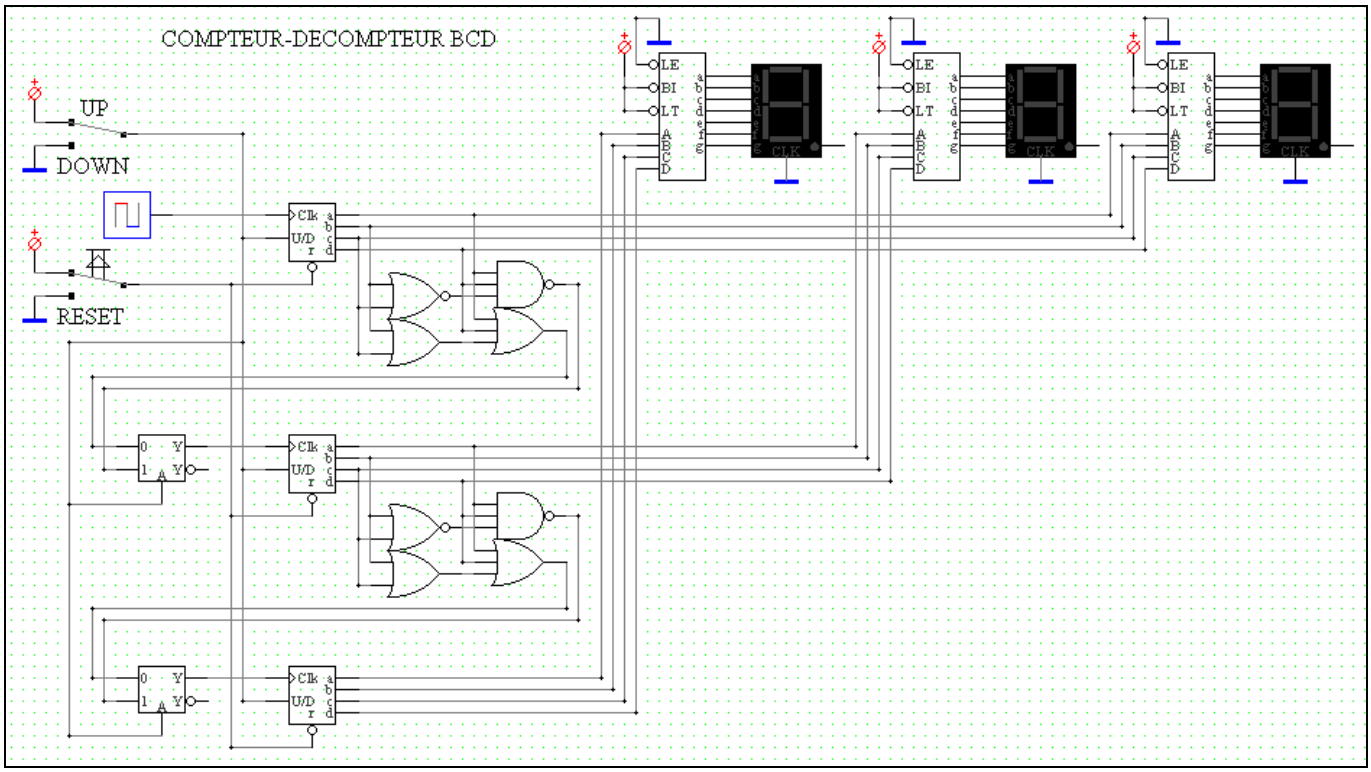
FIGURE 2. COMPLEX MULTIPLIER/ACCUMULATOR; ALL REGISTERS CLOCKED BY CLK

Une vue du processeur dans son intégralité montrant la multitude des fonctions et chemins intégrés dans ce boîtier

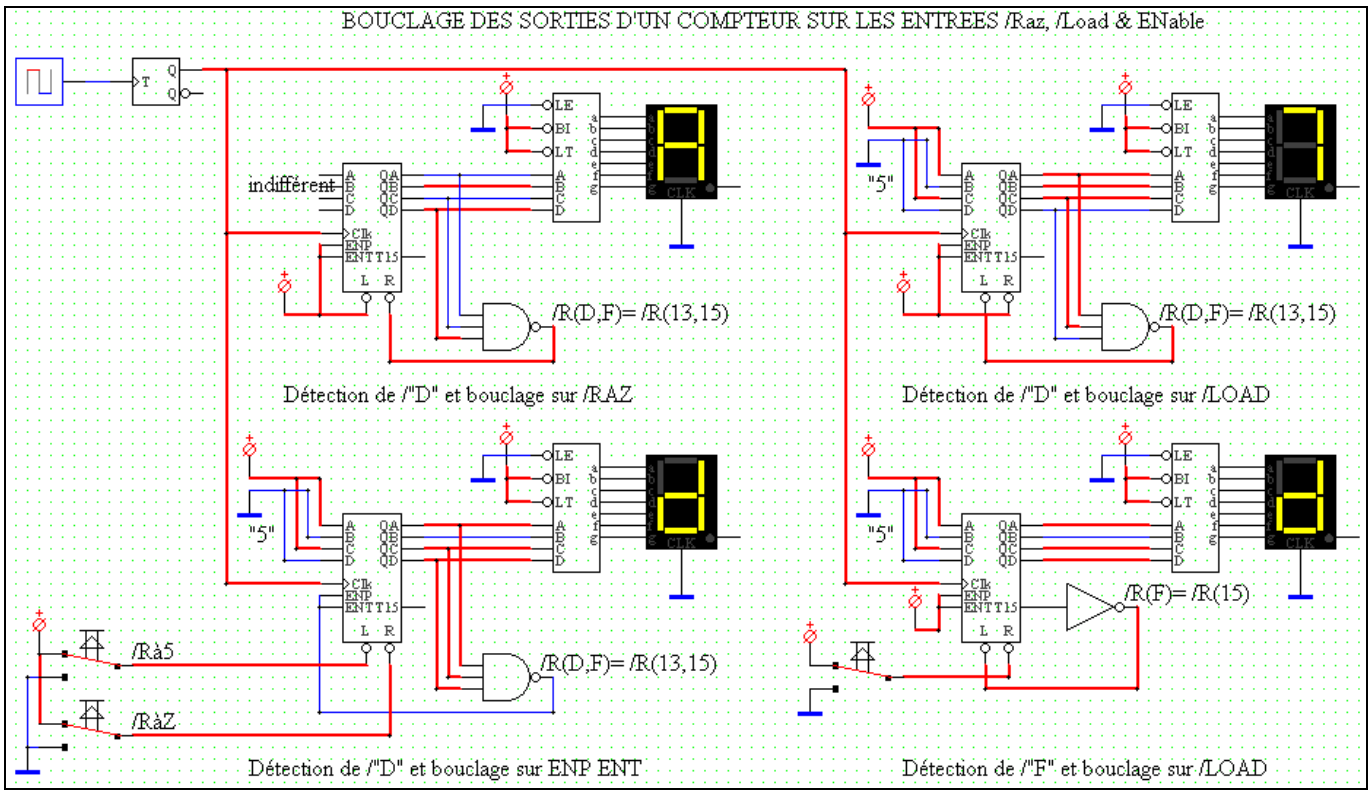


10. Utiliser les compteurs

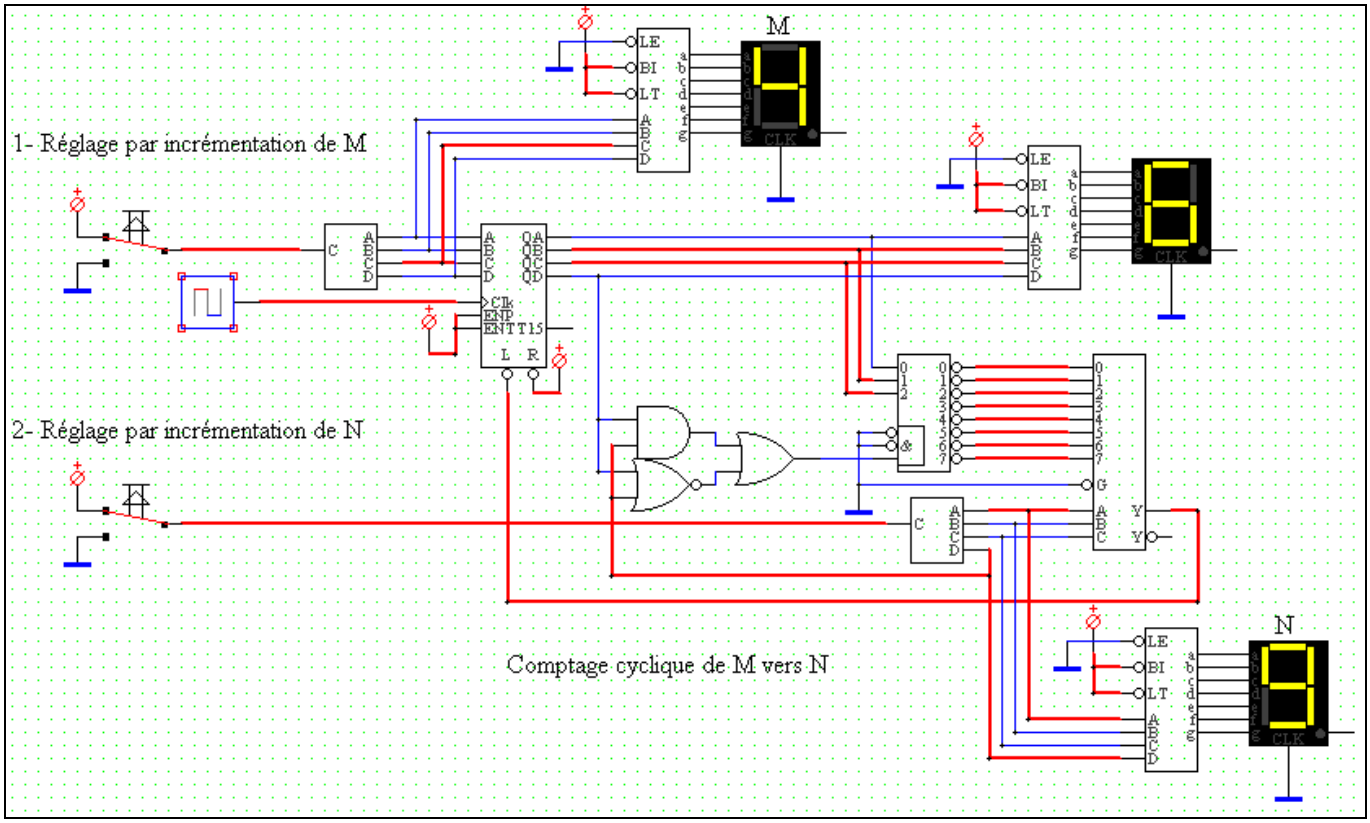
10.1 Compteur-décompteur BCD asynchrone (simulation DigSim)



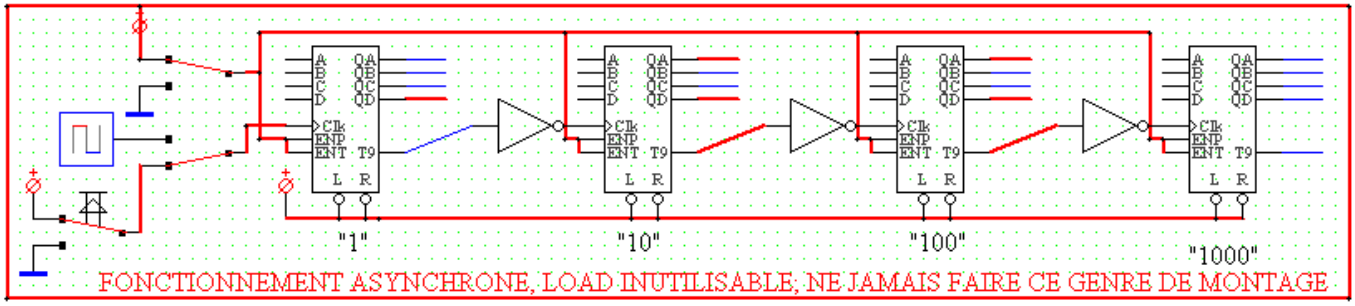
10.2 Bouclage des sorties d'un Compteur sur ses entrées /Raz, /Load & Enable (simulation DigSim)

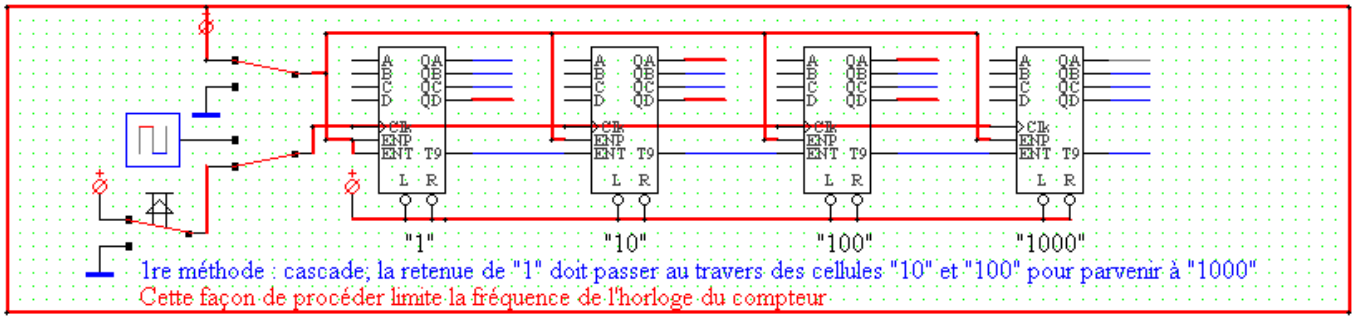


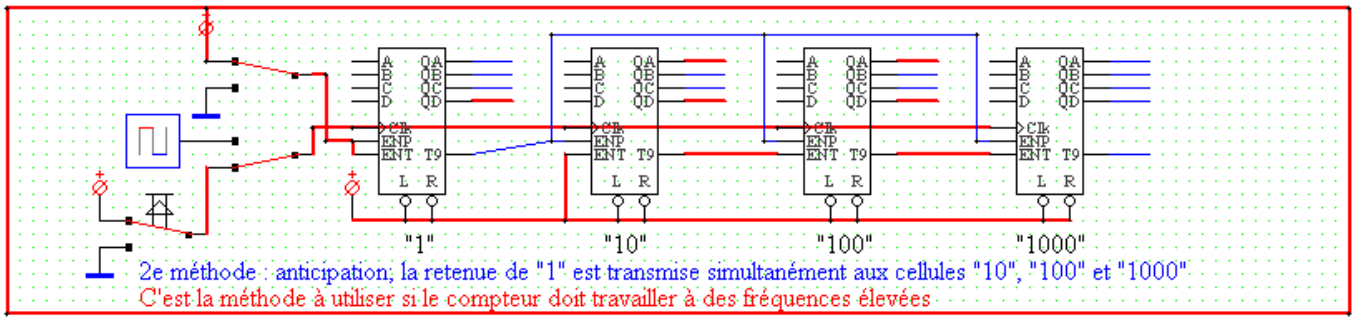
10.3 Comptage de M vers N (simulation DigSim)



10.4 Mise en cascade de compteurs (simulation DigSim)



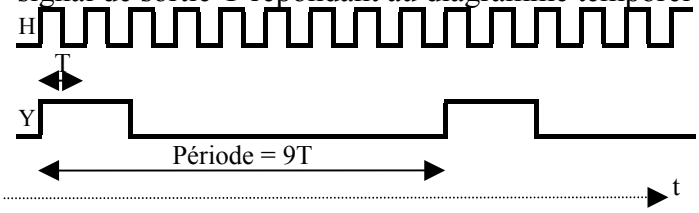




11. Créer des timing

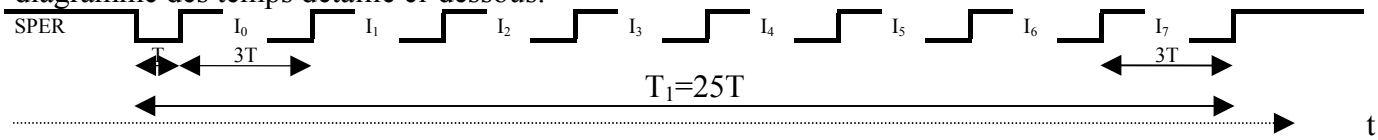
11.1 Comptage

Vous disposez d'une horloge H de période T. En utilisant un compteur et quelques composants annexes si nécessaire, faites la synthèse d'un système logique qui fournisse sur l'une des sorties du compteur le signal de sortie Y répondant au diagramme temporel ci-contre.

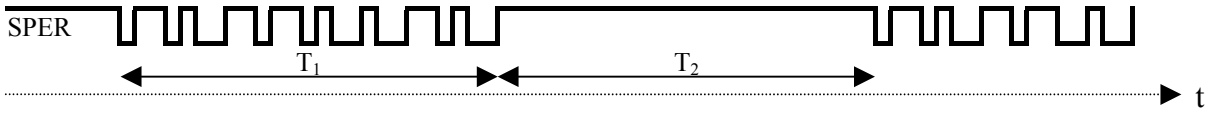


11.2 Transmission de données

Soient 8 entrées nommées I_7, I_6,...,I_0 (I_7: poids fort et I_0 poids faible) et SPER la sortie périodique d'un système logique à réaliser que l'on appellera PISO. Cette sortie doit véhiculer les données I_7 à I_0 selon le diagramme des temps détaillé ci-dessous.



La période de SPER est égale à T_1+T_2 avec $T_1=T_2=25T$. Le diagramme temporel suivant en est une illustration lorsque les données $I_{7\text{-}0}$ sont, par exemple, fixées à la valeur $01001101_2 = 4D_H$.



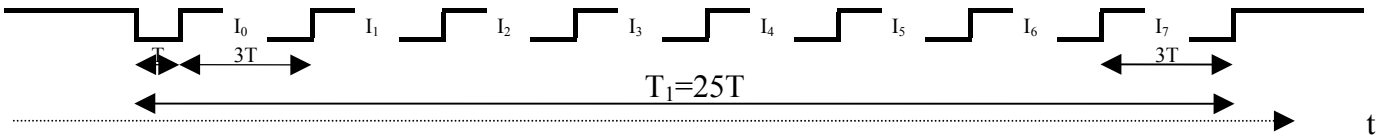
Dessinez un schéma logique de PISO. Indiquez le nom des composants utilisés et commentez brièvement.

11.3 Clé numérique

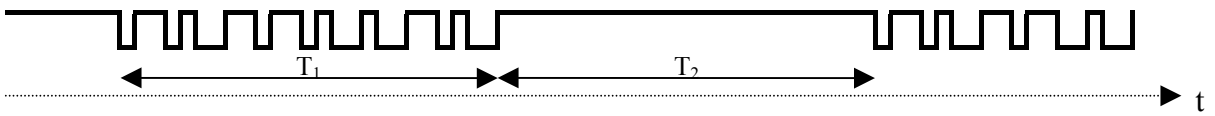
Recréer la fonctionnalité du MM53200 (sur 8 bits pour simplifier) dans son contexte d'utilisation en clé numérique. Les 8 entrées seront nommées I_7, I_6,...,I_0 (I_7: poids fort et I_0 poids faible).
La fonction à réaliser sera configurée en émission ou en réception par une entrée

- en émission, le signal émis contient l'horloge et les 8 bits d'information
- en réception, la sortie devra passer à 1 quand le signal en provenance de l'émetteur sera reçu correctement 3 fois de suite

Détails :
Le signal émis est périodique ; il véhicule les données I_7 à I_0 selon le diagramme des temps ci-dessous.

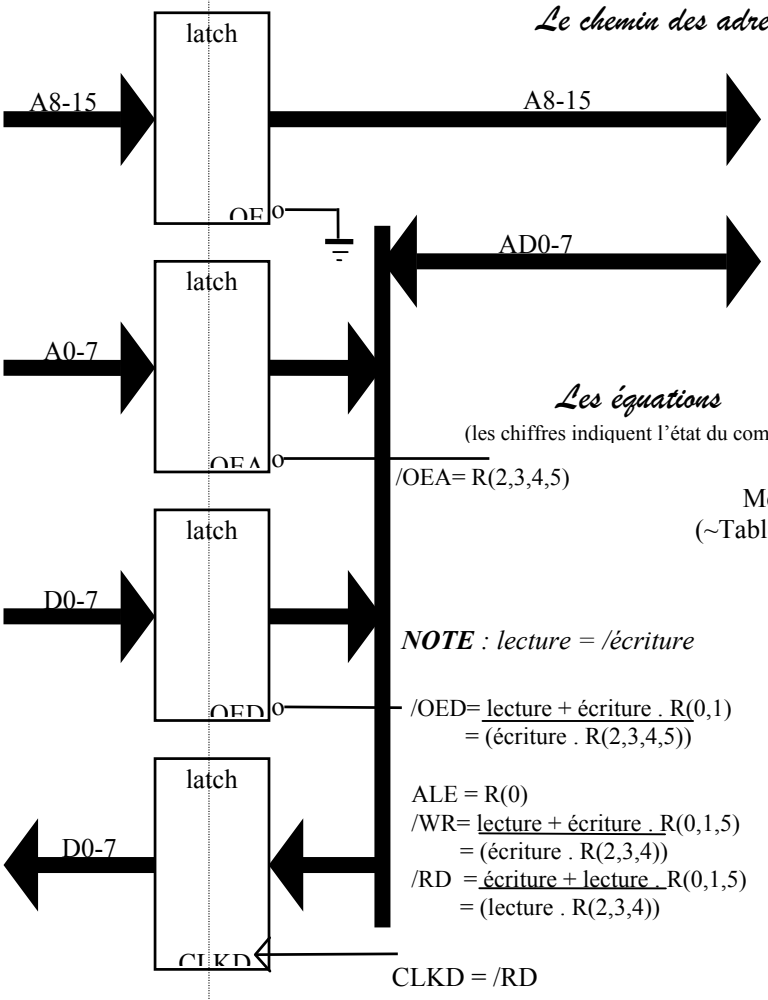


Sa période est égale à T_1+T_2 avec $T_1=T_2=25T$.
Le diagramme temporel illustre ce que l'on peut observer lorsque les données $I_{7\text{-}0} = 01001101_2 = 4D_H$.



11.4 Recréer le timing du 8085

Le chemin des adresses et données



Les équations

(les chiffres indiquent l'état du compteur

/OEA= R(2,3,4,5)

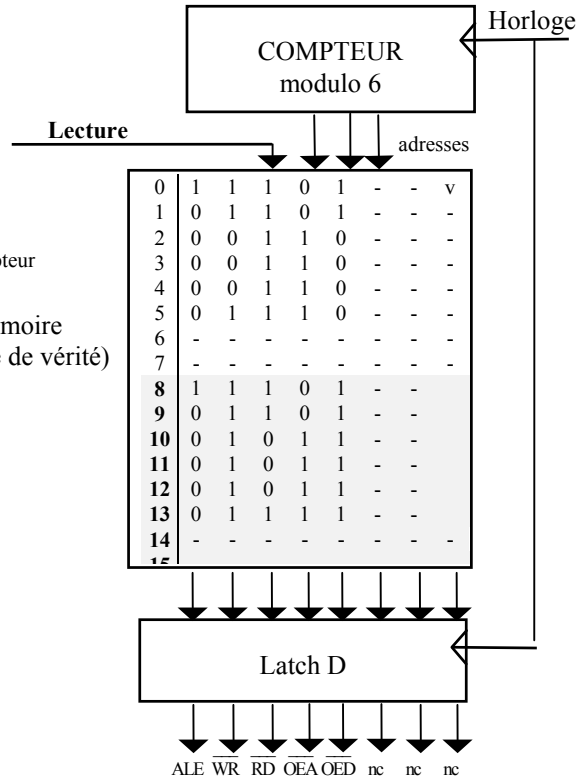
/OED= lecture + écriture . R(0,1)
= (écriture . R(2,3,4,5))

***NOTE** : lecture = /écriture*

ALE = R(0)
/WR= lecture + écriture . R(0,1,5)
= (écriture . R(2,3,4))
/RD = écriture + lecture . R(0,1,5)
= (lecture . R(2,3,4))

CLKD = /RD

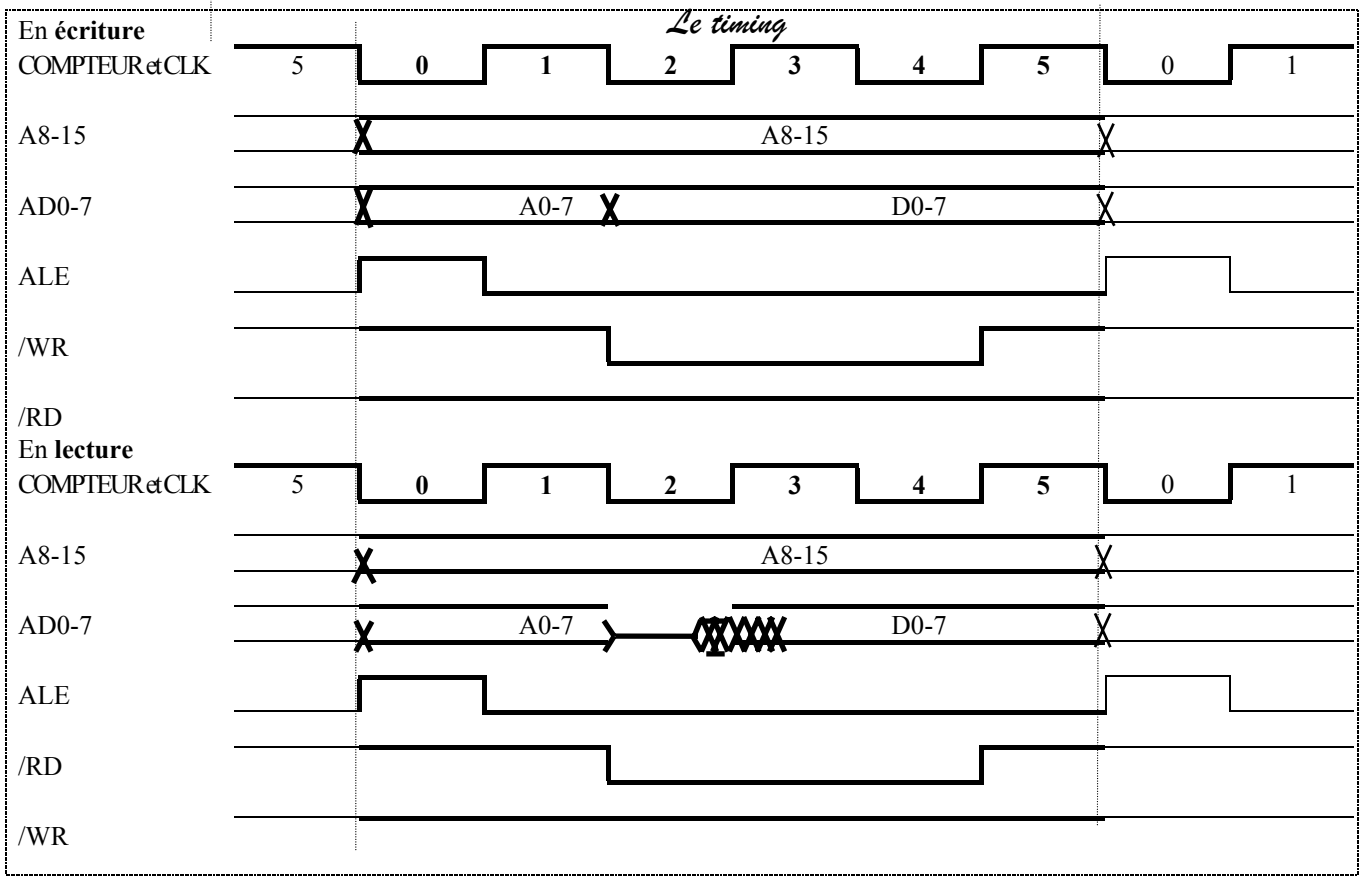
Une réalisation possible



Mémoire
(~Table de vérité)

0	1	1	1	0	1	-	-	v
1	0	1	1	0	1	-	-	-
2	0	0	1	1	0	-	-	-
3	0	0	1	1	0	-	-	-
4	0	0	1	1	0	-	-	-
5	0	1	1	1	0	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	1	1	1	0	1	-	-	
9	0	1	1	0	1	-	-	
10	0	1	0	1	1	-	-	
11	0	1	0	1	1	-	-	
12	0	1	0	1	1	-	-	
13	0	1	1	1	1	-	-	
14	-	-	-	-	-	-	-	-

Le timing



12. Transmission

Tiré de http://f6css.free.fr/nrz_nrzi.htm et de http://www.commentcamarche.net/transmission/transnum.php3

12.1 Codage N R Z

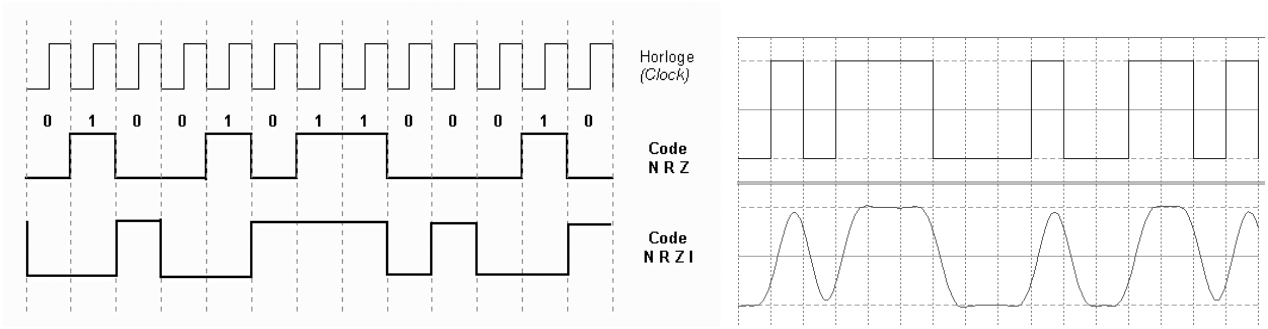
Le code NRZ est le moyen le plus simple de coder l'information: il y a une relation directe entre la valeur logique et une grandeur physique par exemple une tension électrique. On peut imaginer de coder le "**1**" logique par **+5V** est le "**0**" par **0V** (dans ce cas le code est dit unipolaire). On pourrait tout aussi bien coder le "**1**" par **-12v** et le "**0**" par **+12v**. C'est donc un niveau électrique qui "matérialise" l'information et toute inversion de polarité se traduira par une inversion de l'information. Que signifie le sigle NRZ ? il s'agit de *Non Return to Zero,* c'est à dire que le signal ne comporte pas de transition "retour vers zéro" (électrique) pendant une période bit (par opposition aux codes RZ).

12.2 Codage N R Z I

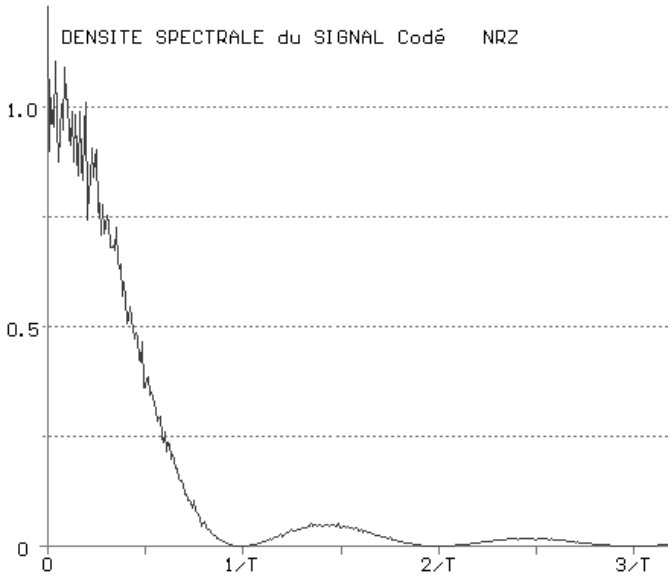
Ce codage n'utilise pas directement le niveau mais le **changement** de niveau pour coder les valeurs logiques: il est du type différentiel. Par exemple pour un "0" logique on provoquera une transition de niveau et il n'y aura aucun changement pour le "1" logique.

Dans ce cas le code est du type ***N**on **R**eturn to **Z**ero **I**nverted on space ,* c'est à dire code NRZ avec inversion de signal sur un *"space" (0 logique).* C'est le codage **NRZI**-S qui est utilisé en transmission packet radio. L'intérêt principal de ce codage est qu'il est insensible aux inversions de polarité du signal pouvant intervenir dans la chaîne de transmission.

La figure suivante illustre les codages NRZ et NRZI. Le signal horloge nous rappelle qu'il s'agit de transmission synchrone. Pour le codage NRZI on suppose ici que le niveau repos était un niveau "haut".



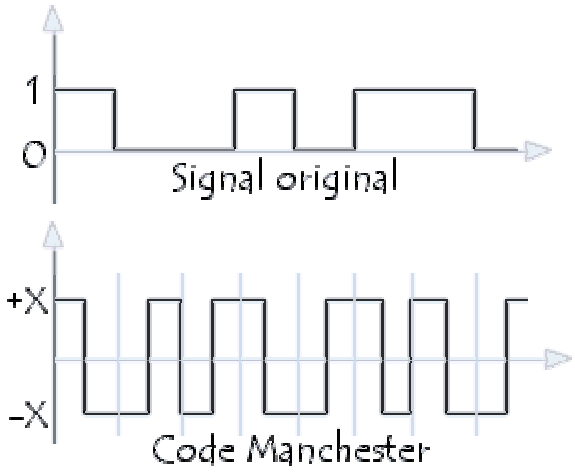
La densité spectrale de puissance d'un signal NRZ (ou NRZI) à l'allure suivante (T = période bit) :



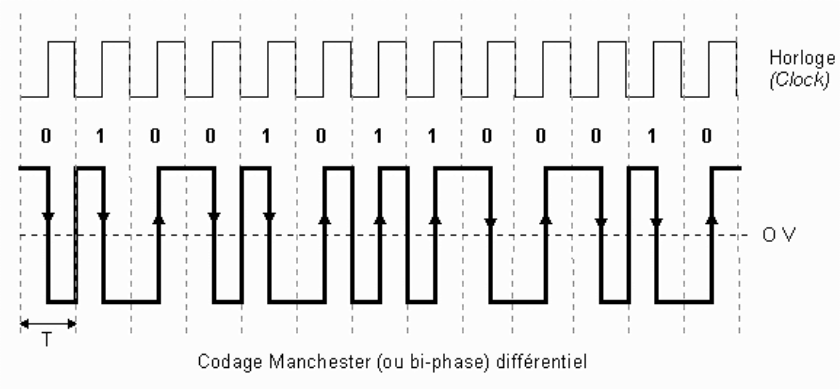
On constate que le signal NRZ (ou variante NRZI) comporte beaucoup d'énergie dans les basses fréquences voire même une composante continue (en fonction du nombre de 1 et de 0 !). Si l'on veut transmettre ce signal sans déformation il faudrait, en théorie, une bande passante allant de "0" à l'infini.

12.3 Codage Manchester ou bi-phase et codage Manchester différentiel

Le codage Manchester est obtenu par le mélange (opération logique OU-exclusif) d'un signal horloge et d'un signal NRZ.



Le Manchester différentiel s'obtient de la même manière en utilisant le codage N R Z I.



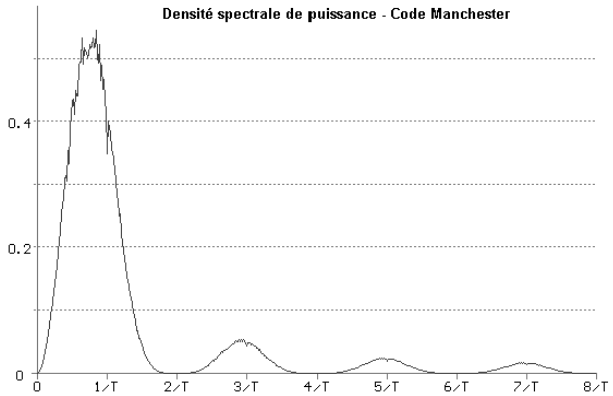
On constate une inversion du sens des transitions du signal en milieu de période bit (durée = T) sur les valeurs "0" logiques.

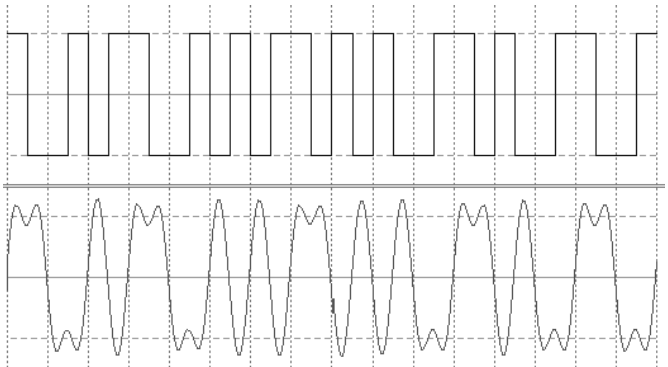
Avantages du code Manchester :

- Valeur moyenne du signal nulle (pas de composante continue et peu d'énergie dans les fréquences basses).
- La densité spectrale du signal présente un maximum vers la fréquence "0,75 / T" (**T = période bit**).
- Synchronisation facile (Il y a systématiquement une transition de signal durant une période bit).

Inconvénients :

- Nécessité d'une bande passante minimum de l'ordre de 1,5 x (1 / T). Par exemple pour un débit de 2400 bits/s il faudrait 1,5 x 2400 Hz = 3600 Hz.

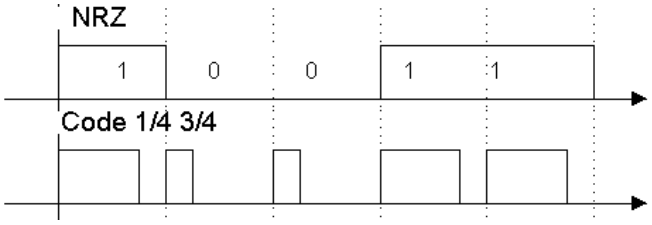




La courbe ci-dessus montre l'allure du signal Manchester après passage dans un filtre passe bas idéal dont la fréquence de coupure est de 1.6 x 1/T.

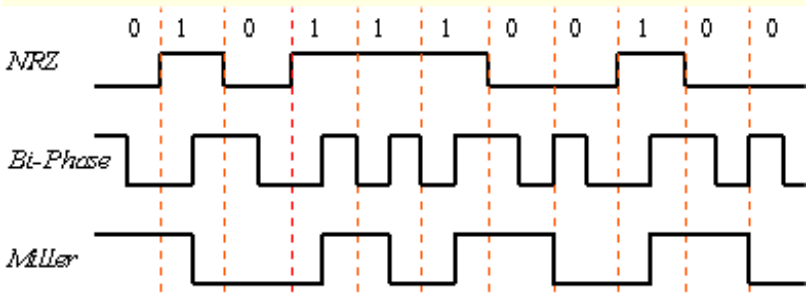
12.4 Autres codages …

12.4.1 Code ¼ ¾

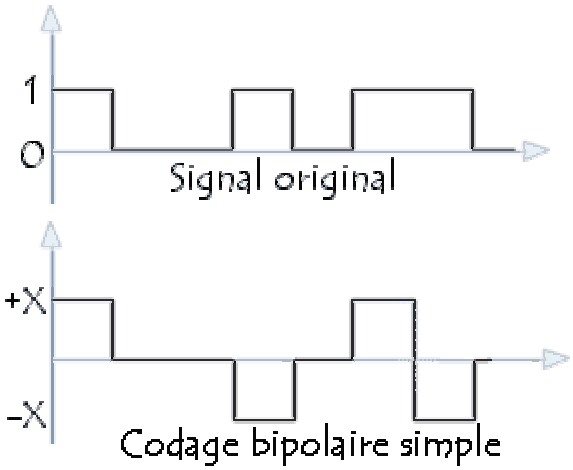


12.4.2 Codage Delay mode ou code de Miller

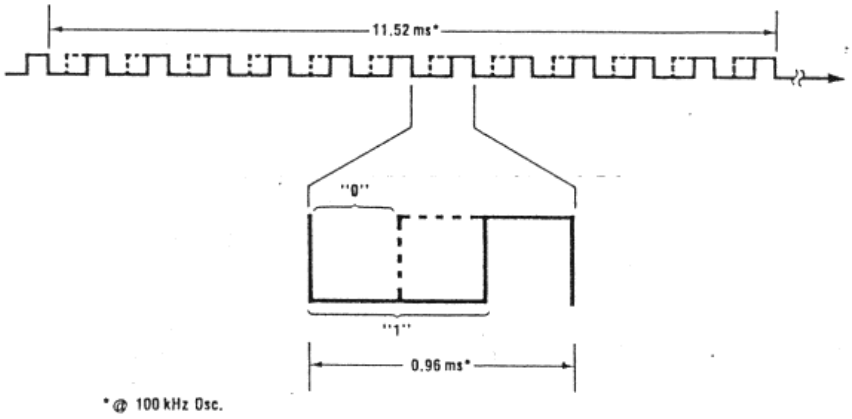
Le code de Miller est obtenu en divisant le signal bi-phase ou Manchester par 2.



12.4.3 Codage bipolaire simple



12.4.4 Le codage employé pour le MM53200



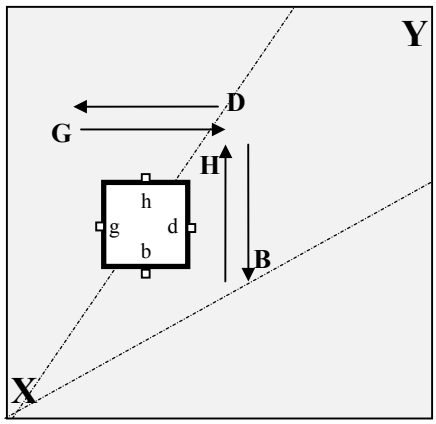
- De quel type est-il?
- Aurait-on pu faire mieux?
- Quels en auraient été les avantages?

13. Séquenceurs

13.1 Enclenchement et déclenchement des bascules RS et JK

S	R	Q^+	$/Q^+$	*Mode ou action*
0	**0**	**Q**	/Q	*mémoire, maintien*
0	**1**	**0**	1	*reset, déclenchement, mise à 0*
1	**0**	**1**	0	*set, enclenchement, mise à 1*
1	**1**	Q=/Q=0ou1		*à utiliser avec précaution (voir*)*

Bascule RS (asynchrone) : $Q^+ = S + \overline{R}Q$ (*enclenchement prioritaire)
ou : $Q^+ = \overline{R}(S+Q)$ (*déclenchement prioritaire)



m

J	K	Q^+	$/Q^+$	*Mode ou action*
0	**0**	**Q**	/Q	*mémoire, maintien*
0	**1**	**0**	1	*reset, déclenchement*
1	**0**	**1**	0	*set, enclenchement*
1	**1**	**/Q**	Q	*toggle, bascule*

Bascule JK (synchrone) : $Q^+ = J\,\overline{Q} + \overline{K}\,Q$

D	Q^+	*Mode ou action*
0	**0**	*reset, déclenchement, mise à 0*
1	**1**	*set, enclenchement, mise à 1*

Bascule D (synchrone) : $Q^+ = D$

T	Q^+	*Mode ou action*
0	**Q**	*mémoire, maintien*
1	**/Q**	*toggle, bascule*

Bascule T (synchrone) : $Q^+ = T \oplus Q$

13.2 Commandes de déplacement d'un chariot

Réaliser le système logique qui permet de déplacer un chariot de la manière suivante:
Suite à une action sur le bouton poussoir de mise en marche **m** , ce chariot se déplacera de **X** vers **Y** puis retournera en **X** pour s'y arrêter.

- **H** et **B** sont les commandes tout ou rien du moteur qui est associé au déplacement vertical.
- **G** et **D** sont les commandes tout ou rien du moteur qui est associé au déplacement latéral.
- **h** et **b** sont les capteurs de position haute et basse situés sur le chariot.
- **g** et **d** sont les capteurs de position gauche et droite situés sur le chariot.

1 On associe une bascule RS (réalisation asynchrone) ou JK (réalisation synchrone) à chaque commande de moteur, soit 4 bascules au total.
Déterminer les 8 équations associées à ce type de réalisation, soit

S_H ou J_H=	S_D ou J_D=	S_B ou J_B=	S_G ou J_G=
R_H ou K_H=	R_D ou K_D=	R_B ou K_B=	S_G ou K_G=

2 On définit deux états appelés **ALLER** et **RETOUR.** On associe une bascule à chacun de ces états.

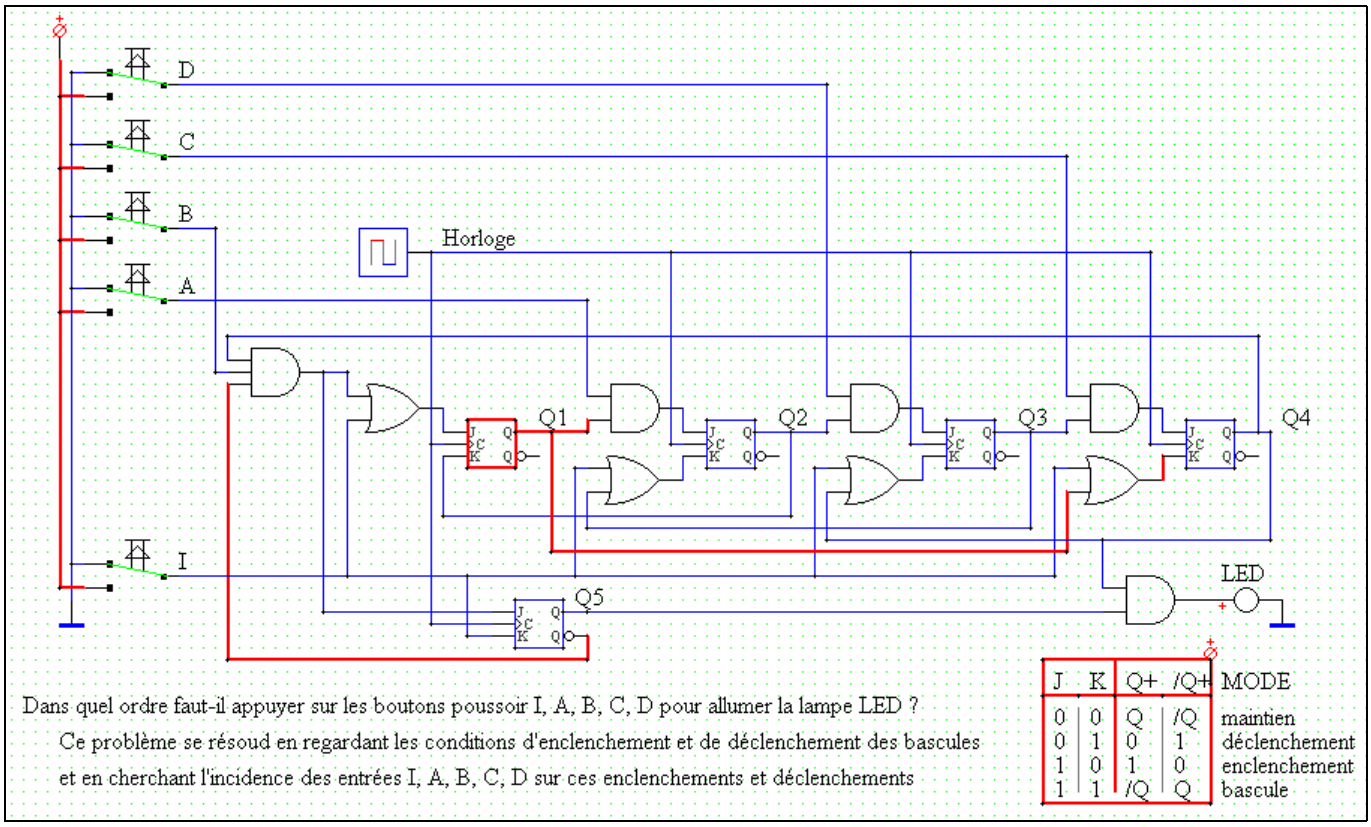
2.1 Déterminer les 4 équations associées à cette réalisation à deux bascules, soit

S_{ALLER}=	S_{RETOUR}=
R_{ALLER}=	R_{RETOUR}=

2.2 Déterminer les équations de sortie (commande des moteurs), soit

H=	**B**=
D=	**G**=

13.3 Séquenceur à JK (simulation DigSim)

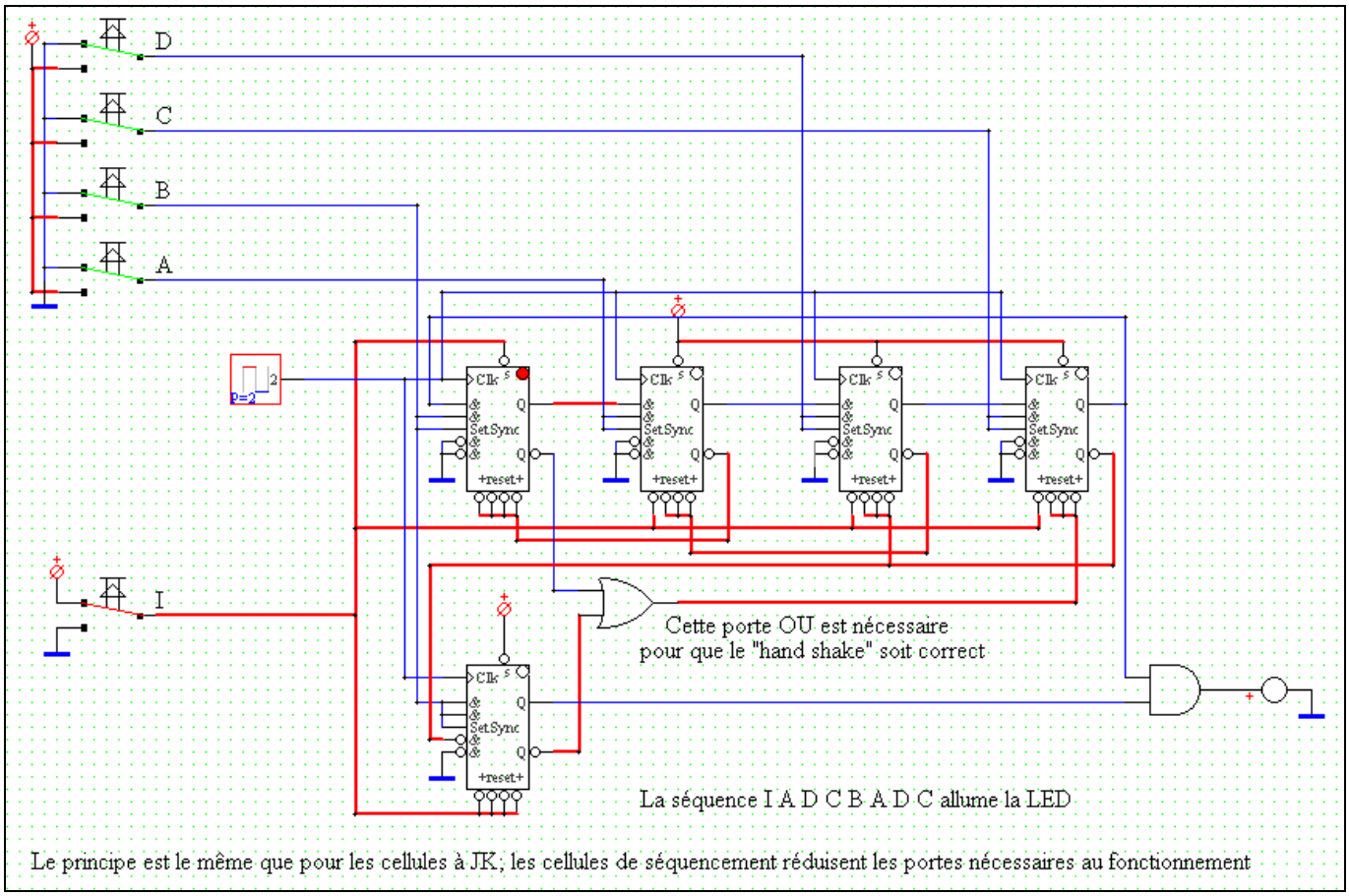


J	K	Q+	/Q+	MODE
0	0	Q	/Q	maintien
0	1	0	1	déclenchement
1	0	1	0	enclenchement
1	1	/Q	Q	bascule

Dans quel ordre faut-il appuyer sur les boutons poussoir I, A, B, C, D pour allumer la lampe LED ?

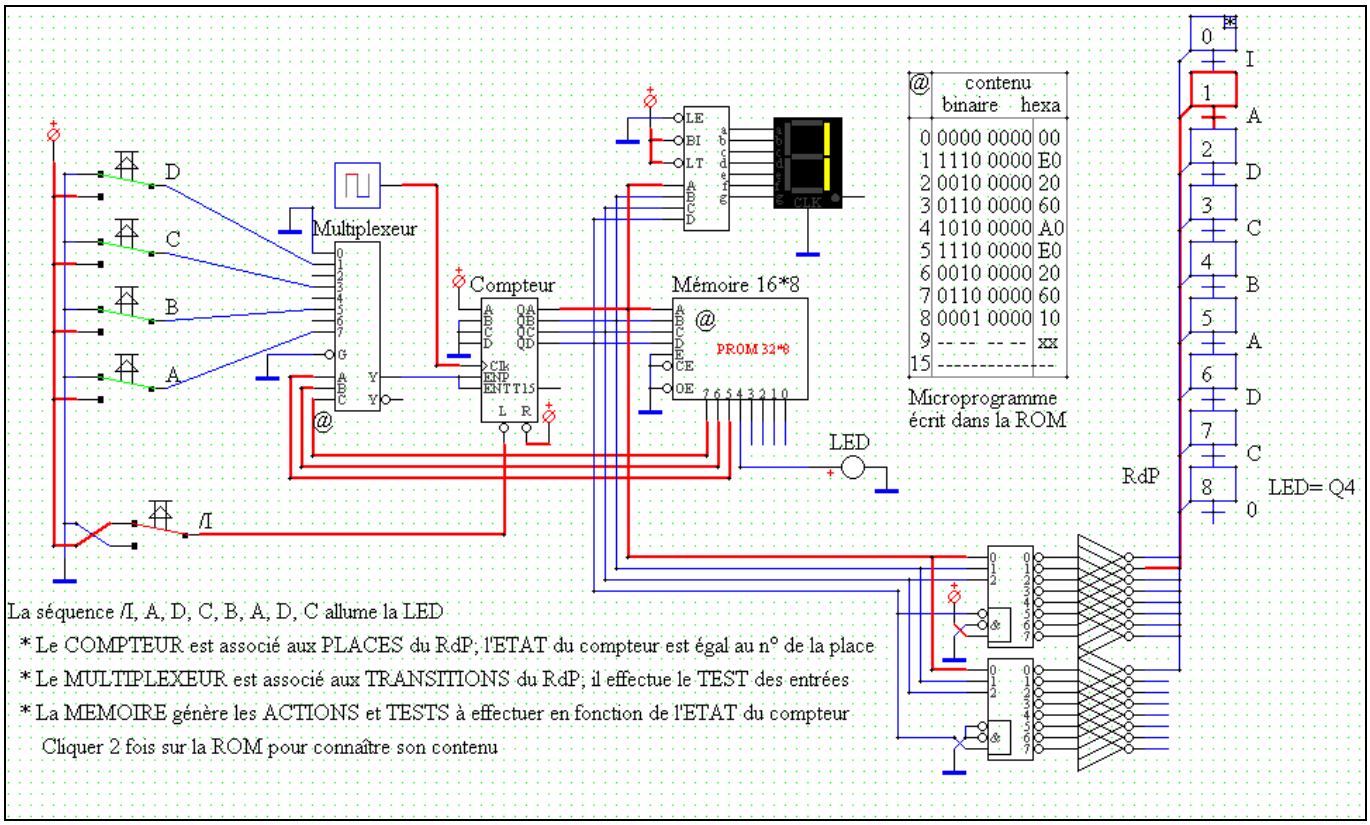
Ce problème se résoud en regardant les conditions d'enclenchement et de déclenchement des bascules et en cherchant l'incidence des entrées I, A, B, C, D sur ces enclenchements et déclenchements

13.4 Un séquenceur utilisant les cellules de séquencement de DigSim (simulation DigSim)

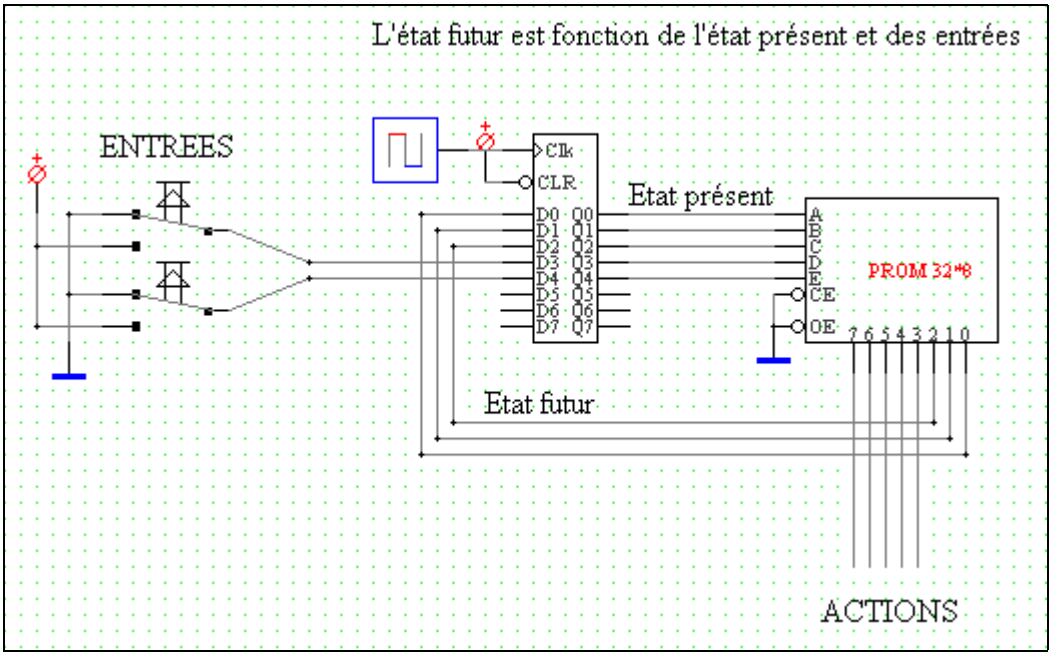


Le principe est le même que pour les cellules à JK ; les cellules de séquencement réduisent les portes nécessaires au fonctionnement

13.5 Séquenceur utilisant un compteur (simulation DigSim)



13.6 Séquenceur utilisant des bascules D (simulation DigSim)

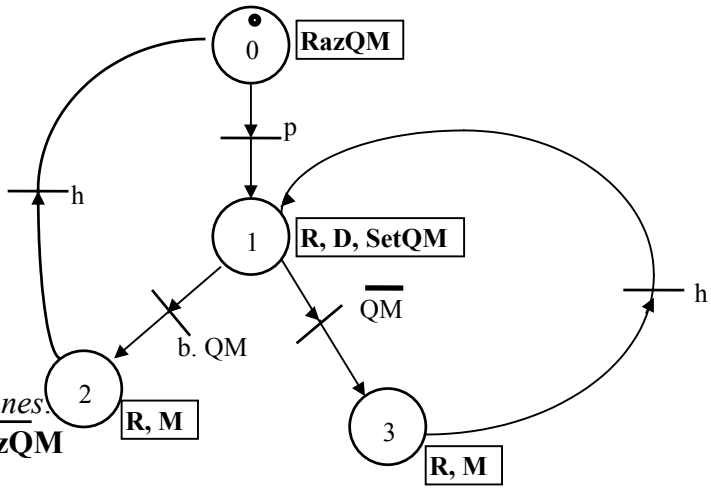


13.7 Exercices

13.7.1 Séquence de perçage

Il s'agit de réaliser une machine séquentielle synchrone fonctionnant selon le réseau de Petri de la figure 1.

Figure 1



Pour cette machine à réaliser :

- **p**ièce, **h**aut, **b**as et **QM** sont des entrées *synchrones*.
- **R**otation, **D**escente, **M**ontée, **SetQM** et **RazQM** sont les sorties.

H est une horloge de fréquence élevée.

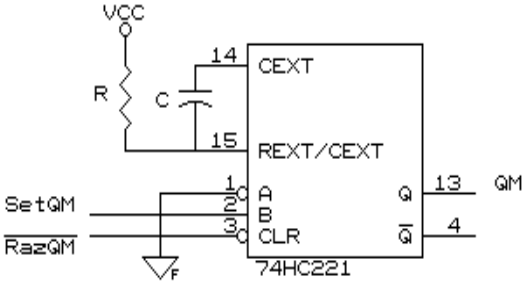


Figure 2

SetQM, RazQM et QM sont les entrées et sorties d'un Monostable 74HC221 qui réalise une temporisation de durée θ (cf. figures 2 & 3)

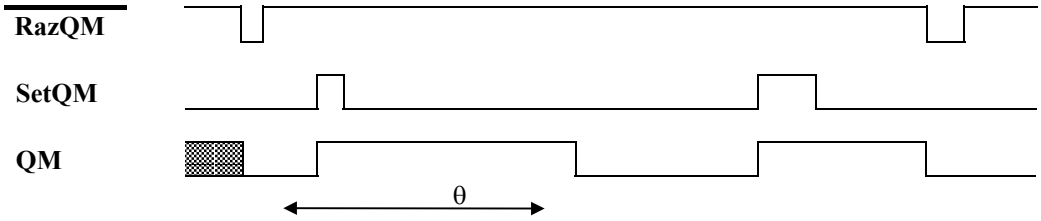


Figure 3

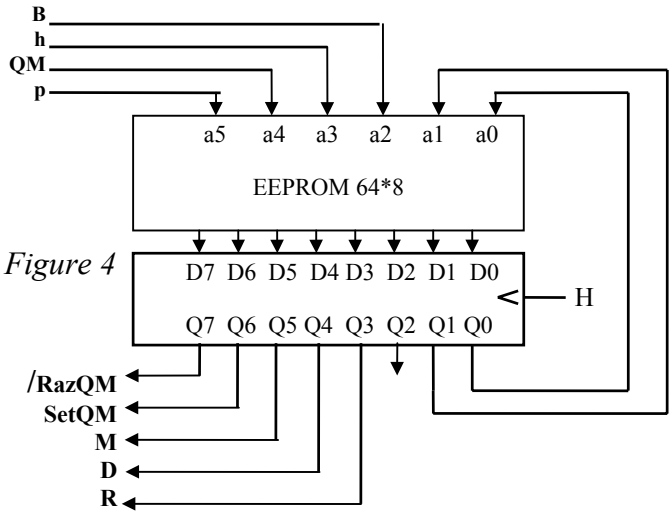


Figure 4

1- La structure de la figure 4 utilise un registre D tel le 74HC273 et une mémoire EEPROM.

Elle permet de réaliser le réseau de Pétri de la figure1.

- **Expliquez** clairement comment trouver contenu de cette mémoire, puis...
- Écrire le contenu de la mémoire sur la feuille suivante

- Le contenu de l'EEPROM

	p	QM	h	B	Q1	Q0	RazQM	SetQM	M	D	R		$Q1^+$	$Q0^+$
	adresses EEPROM						Sorties EEPROM							
	A5	A4	A3	A2	A1	A0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
adresse 0.	0	0	0	0	0	0								
adresse 1.	0	0	0	0	0	**1**								
adresse 2.	0	0	0	0	**1**	0								
adresse 3.	0	0	0	0	**1**	**1**								
adresse 4.	0	0	0	**1**	0	0								
adresse 5.	0	0	0	**1**	0	**1**								
adresse 6.	0	0	0	**1**	**1**	0								
adresse 7.	0	0	0	**1**	**1**	**1**								
adresse 8.	0	0	**1**	0	0	0								
adresse 9.	0	0	**1**	0	0	**1**								
adresse 10.	0	0	**1**	0	**1**	0								
adresse 11.	0	0	**1**	0	**1**	**1**								
adresse 12.	0	0	**1**	**1**	0	0								
adresse 13.	0	0	**1**	**1**	0	**1**								
adresse 14.	0	0	**1**	**1**	**1**	0								
adresse 15.	0	0	**1**	**1**	**1**	**1**								
adresse 16.	**0**	**1**	0	0	0	0								
adresse 17.	**0**	**1**	0	0	0	**1**								
adresse 18.	**0**	**1**	0	0	**1**	0								
adresse 19.	**0**	**1**	0	0	**1**	**1**								
adresse 20.	**0**	**1**	0	**1**	0	0								
adresse 21.	**0**	**1**	0	**1**	0	**1**								
adresse 22.	**0**	**1**	0	**1**	**1**	0								
adresse 23.	**0**	**1**	0	**1**	**1**	**1**								
adresse 24.	**0**	**1**	**1**	0	0	0								
adresse 25.	**0**	**1**	**1**	0	0	**1**								
adresse 26.	**0**	**1**	**1**	0	**1**	0								
adresse 27.	**0**	**1**	**1**	0	**1**	**1**								
adresse 28.	**0**	**1**	**1**	**1**	0	0								
adresse 29.	**0**	**1**	**1**	**1**	0	**1**								
adresse 30.	**0**	**1**	**1**	**1**	**1**	0								
adresse 31.	**0**	**1**	**1**	**1**	**1**	**1**								
adresse 32.	**1**	0	0	0	0	0								
adresse 33.	**1**	0	0	0	0	**1**								
adresse 34.	**1**	0	0	0	**1**	0								
adresse 35.	**1**	0	0	0	**1**	**1**								
adresse 36.	**1**	0	0	**1**	0	0								
adresse 37.	**1**	0	0	**1**	0	**1**								
adresse 38.	**1**	0	0	**1**	**1**	0								
adresse 39.	**1**	0	0	**1**	**1**	**1**								
adresse 40.	**1**	0	**1**	0	0	0								
adresse 41.	**1**	0	**1**	0	0	**1**								
adresse 42.	**1**	0	**1**	0	**1**	0								
adresse 43.	**1**	0	**1**	0	**1**	**1**								
adresse 44.	**1**	0	**1**	**1**	0	0								
adresse 45.	**1**	0	**1**	**1**	0	**1**								
adresse 46.	**1**	0	**1**	**1**	**1**	0								
adresse 47.	**1**	0	**1**	**1**	**1**	**1**								
adresse 48.	**1**	**1**	0	0	0	0								
adresse 49.	**1**	**1**	0	0	0	**1**								
adresse 50.	**1**	**1**	0	0	**1**	0								
adresse 51.	**1**	**1**	0	0	**1**	**1**								
adresse 52.	**1**	**1**	0	**1**	0	0								
adresse 53.	**1**	**1**	0	**1**	0	**1**								
adresse 54.	**1**	**1**	0	**1**	**1**	0								
adresse 55.	**1**	**1**	0	**1**	**1**	**1**								
adresse 56.	**1**	**1**	**1**	0	0	0								
adresse 57.	**1**	**1**	**1**	0	0	**1**								
adresse 58.	**1**	**1**	**1**	0	**1**	0								
adresse 59.	**1**	**1**	**1**	0	**1**	**1**								
adresse 60.	**1**	**1**	**1**	**1**	0	0								
adresse 61.	**1**	**1**	**1**	**1**	0	**1**								
adresse 62.	**1**	**1**	**1**	**1**	**1**	0								
adresse 63.	**1**	**1**	**1**	**1**	**1**	**1**								

- Simuler avec DigSim

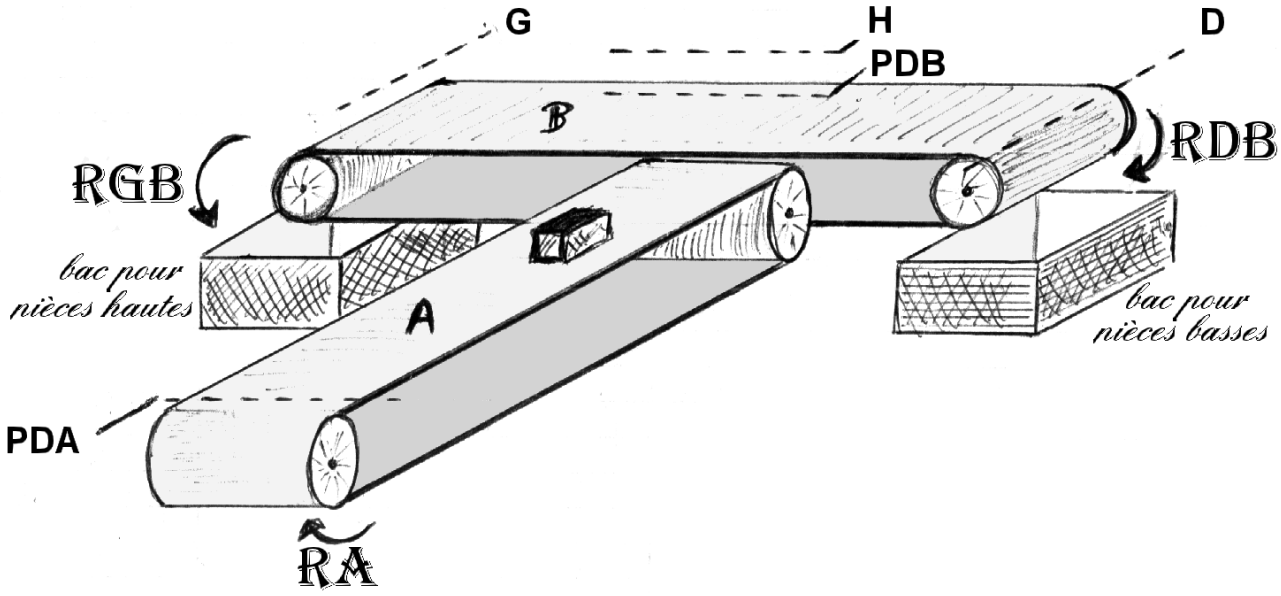
2- **Dessinez le schéma** d'un séquenceur adapté à ce problème réalisé autour d'un compteur 74HC161, de mémoires EEPROM, d'un multiplexeur 74HC151 et de quelques autres composants de votre choix.....puis **écrivez le microprogramme** de ce séquenceur et simuler avec DigSim.

3- **Comparez** les avantages et les inconvénients des réalisations micro-programmées et tabulaires

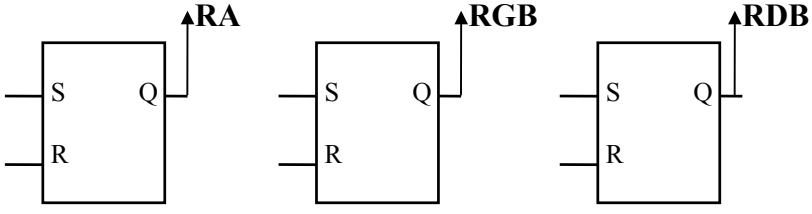
13.7.2 Convoyage de pièces sur un tapis

Une partie d'un système de convoyage de pièces est composée de deux tapis:

- *La condition suivante est assurée lors du dépôt d'une pièce sur le tapis A:*
  Une pièce ne peut être déposée au début du tapis A qu'en l'absence de pièces sur les tapis A et B.
- *Fonctionnement recherché:*
  La présence d'une pièce sur le tapis A déclenche sa mise en marche. Cette pièce arrive sur le tapis B.
  Si cette pièce est haute, le tapis B la dirige vers un bac situé à gauche
  Si cette pièce est basse, le tapis B la dirige vers un bac situé à droite.
  Les tapis A et B ne peuvent fonctionner simultanément.



- A cet effet, 5 capteurs optiques sont utilisés:
  PDA : **P**ièce détectée au **D**ébut du tapis **A** (PDA=1 quand une pièce est détectée par ce capteur)
  PDB : **P**ièce détectée au **D**ébut du tapis **B** (Noter que cette pièce est amenée par le tapis A)
  H : détection d'une pièce **H**aute
  G : arrivée d'une pièce à **G**auche, dans le bac pour pièces hautes
  D : arrivée d'une pièce à **D**roite, dans le bac pour pièces basses
- Les actions sont données par la rotation des cylindres d'entraînement des tapis
  RA : **R**otation du cylindre du tapis **A**
  RGB : **R**otation à **G**auche du cylindre du tapis **B**
  RDB : **R**otation à **D**roite du cylindre du tapis **B**



1. On se propose d'utiliser 3 bascules RS affectées chacune à une action.
   Écrire les équations d'enclenchement et de déclenchement de ces bascules .
2. Faire de même pour des bascules JK
3. Comment modifier le cahier des charges pour apporter des améliorations au convoyage ?
4. Étudier la mise en œuvre de ces améliorations.

13.7.3 Séquenceur microprogrammé et séquenceur tabulaire

Dans le TP, le séquenceur câblé SEQEMI gère l'émission et la bloque en cas d'erreur. Son fonctionnement est donné par le réseau de Petri (RdP) qui vous a été donné en TP.
Il s'agit maintenant d'étudier et de réaliser deux autres types de séquenceur non câblés ayant le même RdP que le séquenceur du TP.

- un séquenceur microprogrammé (compteur + multiplexeur + mémoire + composants annexes)
- un séquenceur tabulaire (Flip-flop + mémoire + composants annexes)

Pour conclure, vous comparerez les avantages et inconvénients de ces trois différentes versions de séquenceur :

13.7.4 Gestion de feux de circulation

Il existe des analogies mais aussi des différences entre la gestion de la circulation de véhicules sur des portions de voies communes et la celle de la circulation de données binaires (données et/ou adresses) sur des bus partagés. Ainsi, la transposition du domaine de la vie courante au domaine de la vie binaire et réciproquement, peut faciliter la compréhension, mais aussi être source d'idées et à l'origine de solutions au problème posé.
Il s'agit ici d'aborder la gestion des feux de circulation situés au carrefour de deux voies.
Il est évident que ce problème de gestion peut être abordé et résolu comme un problème de séquencement.
Vous l'approcherez de manière progressive en envisageant les cas suivants:

- Cycle fixe
- Cycle variable avec détection(s) de véhicule(s)
- Autres propositions : dans cette partie, le choix du type problème à résoudre, de l'importance des voies et de la gestion des possibles heures de pointe est laissé au libre cours de votre imagination.

13.8 Séquenceur du filtre numérique envisagé précédemment

Voir la partie relative au filtre numérique

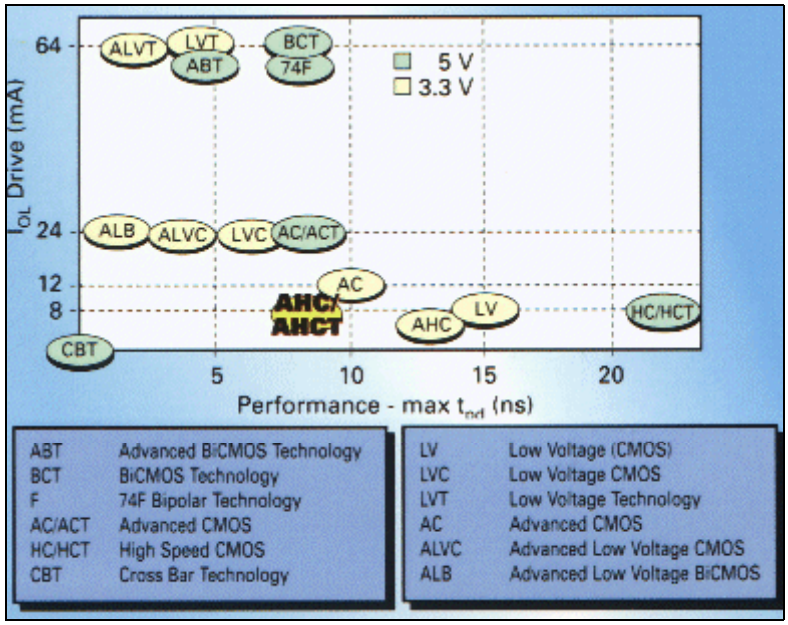
13.9 Liste non exhaustive de composants de la série 74HC

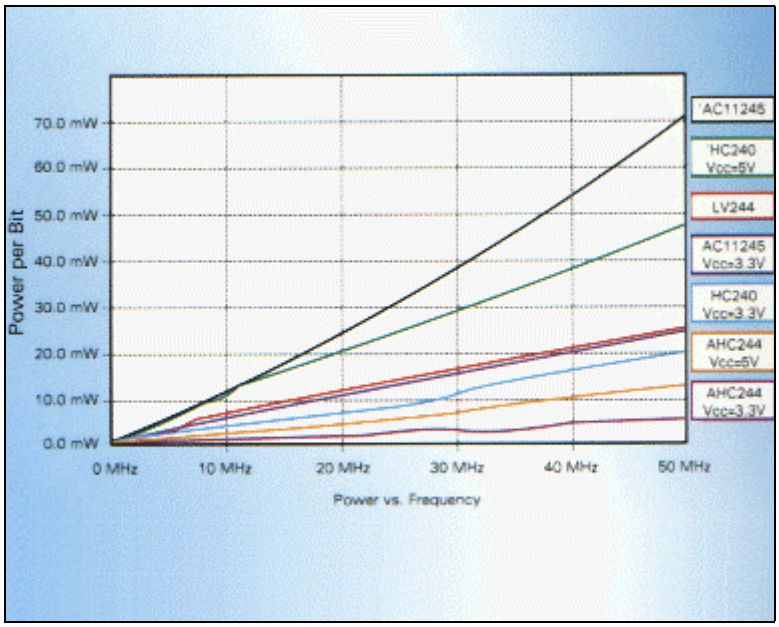
(http://www-eu2.semiconductors.com/handbook/chapter_1906.html)

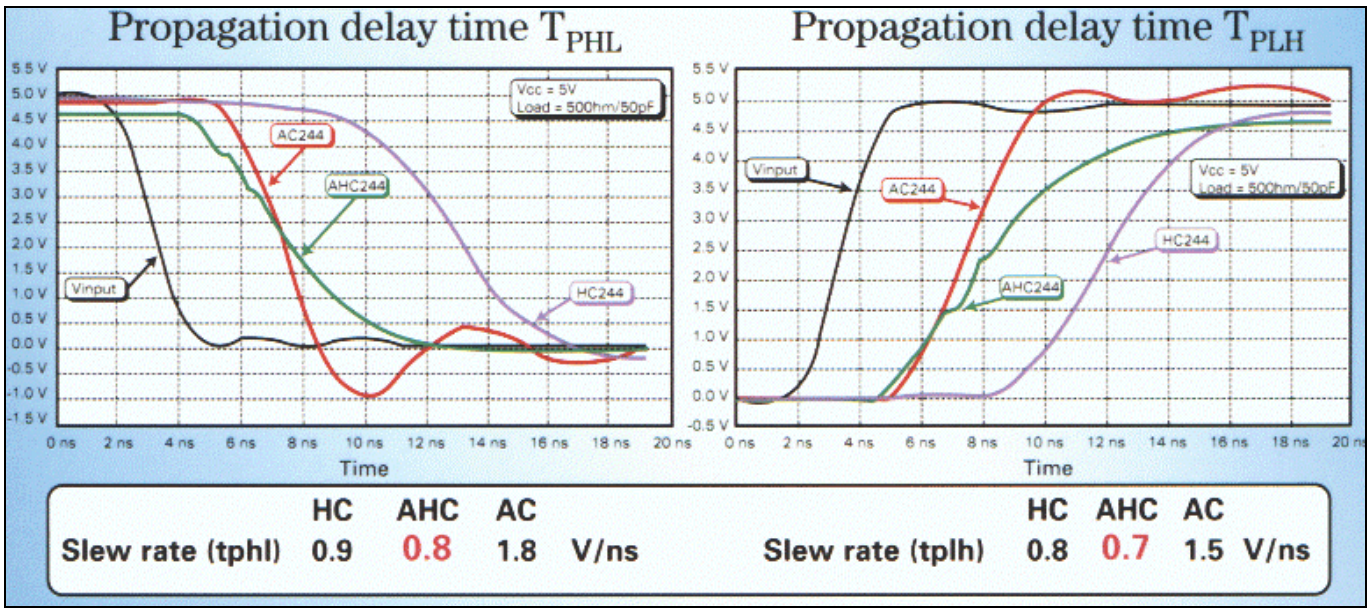
74HC/HCT00: Quad 2-input NAND gate
74HC/HCT02: Quad 2-input NOR gate
74HC/HCT03: Quad 2-input NAND gate
74HC/HCT04: Hex inverter
74HC/HCT10: Triple 3-input NAND gate
74HC/HCT107: Dual JK flip-flop with reset; negative-edge trigger
74HC/HCT109: Dual JK flip-flop with set and reset; positive-edge trigger
74HC/HCT11: Triple 3-input AND gate
74HC/HCT123: Dual retriggerable monostable multivibrator with reset
74HC/HCT125: Quad buffer/line driver; 3-state
74HC/HCT126: Quad buffer/line driver; 3-state
74HC/HCT132: Quad 2-input NAND Schmitt trigger
74HC/HCT137: 3-to-8 line decoder/demultiplexer with address latches; inverting
74HC/HCT138: 3-to-8 line decoder/demultiplexer; inverting
74HC/HCT139: Dual 2-to-4 line decoder/demultiplexer
74HC/HCT14: Hex inverting Schmitt trigger
74HC/HCT147: 10-to-4 line priority encoder
74HC/HCT151: 8-input multiplexer
74HC/HCT153: Dual 4-input multiplexer
74HC/HCT154: 4-to-16 line decoder/demultiplexer
74HC/HCT157: Quad 2-input multiplexer
74HC/HCT158: Quad 2-input multiplexer; inverting
74HC/HCT160: Presettable synchronous BCD decade counter; asynchronous reset
74HC/HCT161: Presettable synchronous 4-bit binary counter; asynchronous reset
74HC/HCT163: Presettable synchronous 4-bit binary counter; synchronous reset
74HC/HCT164: 8-bit serial-in/parallel-out shift register
74HC/HCT165: 8-bit parallel-in/serial-out shift register
74HC/HCT166: 8-bit parallel-in/serial-out shift register
74HC/HCT173: Quad D-type flip-flop; positive-edge trigger; 3-state
74HC/HCT174: Hex D-type flip-flop with reset; positive-edge trigger
74HC/HCT191: Presettable synchronous 4-bit binary up/down counter
74HC/HCT193: Presettable synchronous 4-bit binary up/down counter
74HC/HCT194: 4-bit bidirectional universal shift register
74HC/HCT20: Dual 4-input NAND gate
74HC/HCT21: Dual 4-input AND gate
74HC/HCT221: Dual non-retriggerable monostable multivibrator with reset
74HC/HCT237: 3-to-8 line decoder/demultiplexer with address latches
74HC/HCT238: 3-to-8 line decoder/demultiplexer
74HC/HCT240: Octal buffer/line driver; 3-state; inverting
74HC/HCT241: Octal buffer/line driver; 3-state
74HC/HCT243: Quad bus transceiver; 3-state
74HC/HCT244: Octal buffer/line driver; 3-state
74HC/HCT245: Octal bus transceiver; 3-state
74HC/HCT251: 8-input multiplexer; 3-state
74HC/HCT259: 8-bit addressable latch
74HC/HCT27: Triple 3-input NOR gate
74HC/HCT273: Octal D-type flip-flop with reset; positive-edge trigger
74HC/HCT273: Octal D-type flip-flop with reset; positive-edge trigger
74HC/HCT280: 9-bit odd/even parity generator/checker
74HC/HCT283: 4-bit binary full adder with fast carry
74HC/HCT299: 8-bit universal shift register; 3-state
74HC/HCT30: 8-input NAND gate
74HC/HCT32: Quad 2-input OR gate
74HC/HCT365: Hex buffer/line driver; 3-state
74HC/HCT366: Hex buffer/line driver; 3-state; inverting
74HC/HCT367: Hex buffer/line driver; 3-state
74HC/HCT368: Hex buffer/line driver; 3-state; inverting
74HC/HCT373: Octal D-type transparent latch; 3-state
74HC/HCT374: Octal D-type flip-flop; positive edge-trigger; 3-state
74HC/HCT377: Octal D-type flip-flop with data enable; positive-edge trigger
74HC/HCT390: Dual decade ripple counter
74HC/HCT393: Dual 4-bit binary ripple counter
74HC/HCT4002: Dual 4-input NOR gate
74HC/HCT40105: 4-bit x 16-word FIFO register
74HC/HCT4015: Dual 4-bit serial-in/parallel-out shift register
74HC/HCT4016: Quad bilateral switches
74HC/HCT4017: Johnson decade counter with 10 decoded outputs
74HC/HCT4020: 14-stage binary ripple counter
74HC/HCT4024: 7-stage binary ripple counter
74HC/HCT4040: 12-stage binary ripple counter
74HC/HCT4046A: Phase-locked-loop with VCO
74HC/HCT4051: 8-channel analog multiplexer/demultiplexer
74HC/HCT4052: Dual 4-channel analog multiplexer/demultiplexer
74HC/HCT4053: Triple 2-channel analog multiplexer/demultiplexer
74HC/HCT4060: 14-stage binary ripple counter with oscillator
74HC/HCT4066: Quad bilateral switches

74HC/HCT4067: 16-channel analog multiplexer/demultiplexer
74HC/HCT4075: Triple 3-input OR gate
74HC/HCT4094: 8-stage shift-and-store bus register
74HC/HCT42: BCD to decimal decoder (1-of-10)
74HC/HCT423: Dual retriggerable monostable multivibrator with reset
74HC/HCT4316: Quad bilateral switches
74HC/HCT4351: 8-channel analog multiplexer/demultiplexer with latch
74HC/HCT4353: Triple 2-channel analog multiplexer/demultiplexer with latch
74HC/HCT4511: BCD to 7-segment latch/decoder/driver
74HC/HCT4514: 4-to-16 line decoder/demultiplexer with input latches
74HC/HCT4515: 4-to-16 line decoder/demultiplexer with input latches; inverting
74HC/HCT4518: Dual synchronous BCD counter
74HC/HCT4520: Dual 4-bit synchronous binary counter
74HC/HCT4538: Dual retriggerable precision monostable multivibrator
74HC/HCT534: Octal D-type flip-flop; positive edge-trigger; 3-state; inverting
74HC/HCT540: Octal buffer/line driver; 3-state; inverting
74HC/HCT541: Octal buffer/line driver; 3-state
74HC/HCT5555: Programmable delay timer with oscillator
74HC/HCT563: Octal D-type transparent latch; 3-state; inverting
74HC/HCT564: Octal D-type flip-flop; positive-edge trigger; 3-state; inverting
74HC/HCT573: Octal D-type transparent latch; 3-state
74HC/HCT574: Octal D-type flip-flop; positive edge-trigger; 3-state
74HC/HCT594: 8-bit shift register with output register
74HC/HCT595: 8-bit serial-in/serial or parallel-out shift register with output latches; 3-state
74HC/HCT597: 8-bit shift register with input flip-flops
74HC/HCT6323A: Programmable ripple counter with oscillator; 3-state
74HC/HCT640: Octal bus transceiver; 3-state; inverting
74HC/HCT643: Octal bus transceiver; 3-state; true/inverting
74HC/HCT646: Octal bus transceiver/register; 3-state
74HC/HCT648: Octal bus transceiver/register; 3-state; inverting
74HC/HCT652: Octal bus transceiver/register; 3-state
74HC/HCT670: 4 x 4 register file; 3-state
74HC/HCT688: 8-bit magnitude comparator
74HC/HCT7030: 9-bit x 64-word FIFO register; 3-state
74HC/HCT7046A: Phase-locked-loop with lock detector
74HC/HCT73: Dual JK flip-flop with reset; negative-edge trigger
74HC/HCT74: Dual D-type flip-flop with set and reset; positive-edge trigger
74HC/HCT7403: 4-Bit x 64-word FIFO register; 3-state
74HC/HCT75: Quad bistable transparent latch
74HC/HCT7540: Octal Schmitt trigger buffer/line driver; 3-state; inverting
74HC/HCT7541: Octal Schmitt trigger buffer/line driver; 3-state
74HC/HCT7731: Quad 64-bit static shift register
74HC/HCT85: 4-bit magnitude comparator
74HC/HCT86: Quad 2-input EXCLUSIVE-OR gate
74HC/HCT9015: Nine wide Schmitt trigger buffer/line driver
74HC/HCT9114: Nine wide Schmitt trigger buffer; open drain outputs; inverting
74HC/HCT93: 4-bit binary ripple counter
74HC/HCT9323A: Programmable ripple counter with oscillator (3-State)
74HC08;74HCT08: Quad 2-input AND gate
74HC112;74HCT112: dual JK flip-flop with set and reset; negative-edge trigger
74HC175;74HCT175: quad D-type flip-flop with reset; positive-edge trigger
74HC181;74HCT181: 4-bit arithmetic logic unit
74HC1G00; 74HCT1G00: 2-input NAND gate
74HC1G02; 74HCT1G02: 2-input NOR gate
74HC1G04; 74HCT1G04: Inverter
74HC1G08; 74HCT1G08: 2-input AND gate
74HC1G125; 74HCT1G125: Bus buffer/line drivers; 3-state
74HC1G126; 74HCT1G126: Bus buffer/line driver; 3-state
74HC1G14; 74HCT1G14: Inverting Schmitt-triggers
74HC1G32; 74HCT1G32: 2-Input OR gate
74HC1G66; 74HCT1G66: Bilateral switch
74HC1G86; 74HCT1G86: 2-input EXCLUSIVE_OR gate
74HC1GU04: Inverter
74HC253;74HCT253: Dual 4-input multiplexer; 3-state
74HC40103;74HCT40103: 8-bit synchronous binary down counter
74HC4049: Hex inverting high-to-low level shifter
74HC4050: Hex high-to-low level shifter
74HC4059;74HCT4059: programmable divide-by-n counter
74HC58: Dual AND-OR gate
74HC7014: Hex non-inverting precision Schmitt-trigger
74HC7266: Quad 2-input EXCLUSIVE-NOR gate
74HCT1284: Parallel printer interface transceiver/buffer
74HCT9046A: PLL with bandgap controlled VCO
74HCU04: Hex inverter
74HC_HCT257: Quad 2-input multiplexer; 3-state
74HC_HCT258: Quad 2-input multiplexer; 3-state; inverting

13.10 Quelques caractéristiques







14. Exemple de data-sheet

Philips Semiconductors — Product specification

Dual $\overline{JK}$ flip-flop with set and reset; positive-edge trigger — 74HC/HCT109

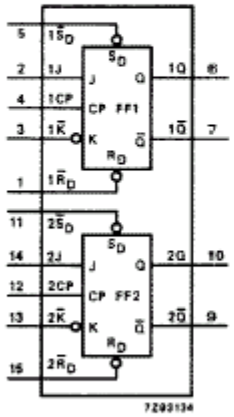


Fig.4 Functional diagram.

FUNCTION TABLE

OPERATING MODE	INPUTS					OUTPUTS	
	$\overline{S}_D$	$\overline{R}_D$	CP	J	$\overline{K}$	Q	$\overline{Q}$
asynchronous set	L	H	X	X	X	H	L
asynchronous reset	H	L	X	X	X	L	H
undetermined	L	L	X	X	X	H	H
toggle	H	H	↑	h	l	$\overline{q}$	q
load "0" (reset)	H	H	↑	l	l	L	H
load "1" (set)	H	H	↑	h	h	H	L
hold "no change"	H	H	↑	l	h	q	$\overline{q}$

Notes

1. H = HIGH voltage level
   h = HIGH voltage level one set-up time prior to the LOW-to-HIGH CP transition
   L = LOW voltage level
   l = LOW voltage level one set-up time prior to the LOW-to-HIGH CP transition
   q = lower case letters indicate the state of the referenced output one set-up time prior to the LOW-to-HIGH CP transition
   X = don't care
   ↑ = LOW-to-HIGH CP transition

FEATURES

- J, $\overline{K}$ inputs for easy D-type flip-flop
- Toggle flip-flop or "do nothing" mode
- Output capability: standard
- I_{CC} category: flip-flops

GENERAL DESCRIPTION

The 74HC/HCT109 are high-speed Si-gate CMOS devices and are pin compatible with low power Schottky TTL (LSTTL). They are specified in compliance with JEDEC standard no. 7A.

The 74HC/HCT109 are dual positive-edge triggered, $\overline{JK}$ flip-flops with individual J, $\overline{K}$ inputs, clock (CP) inputs, set ($\overline{S}_D$) and reset ($\overline{R}_D$) inputs; also complementary Q and $\overline{Q}$ outputs.

The set and reset are asynchronous active LOW inputs and operate independently of the clock input.

The J and $\overline{K}$ inputs control the state changes of the flip-flops as described in the mode select function table.

The J and $\overline{K}$ inputs must be stable one set-up time prior to the LOW-to-HIGH clock transition for predictable operation.

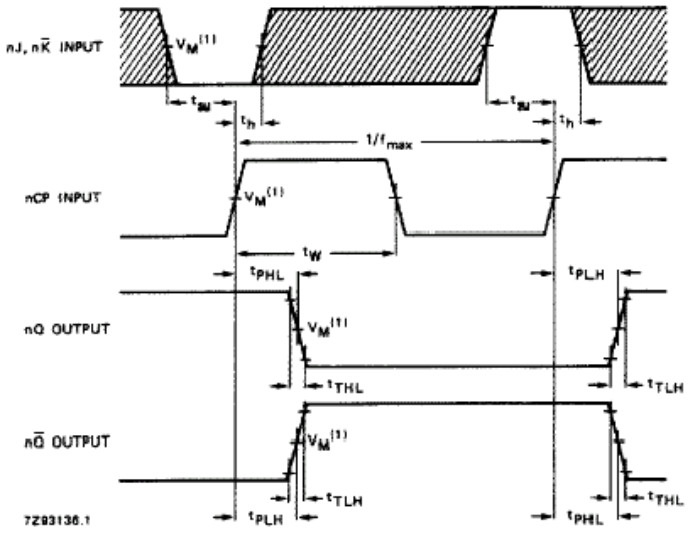
The $J\overline{K}$ design allows operation as a D-type flip-flop by tying the J and $\overline{K}$ inputs together.

Schmitt-trigger action in the clock input makes the circuit highly tolerant to slower clock rise and fall times.

QUICK REFERENCE DATA

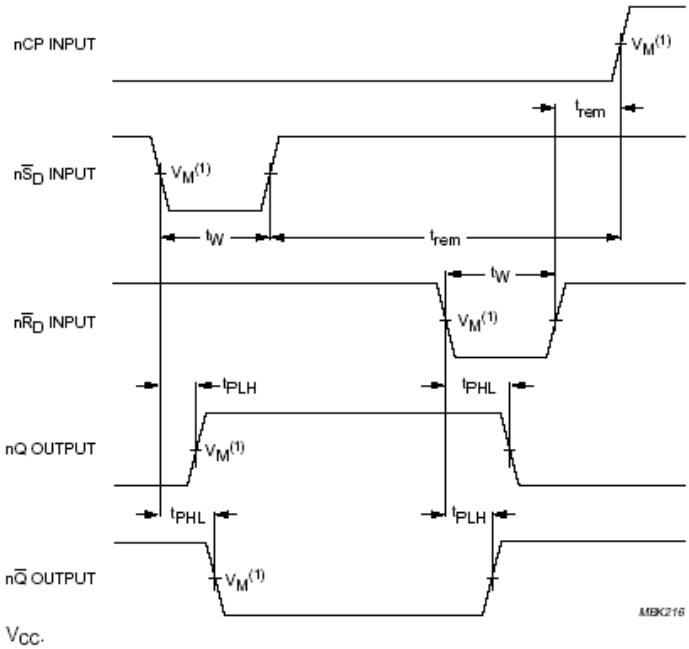
GND = 0 V; T_{amb} = 25 °C; $t_r = t_f$ = 6 ns

SYMBOL	PARAMETER	CONDITIONS	TYPICAL		UNIT
			HC	HCT	
t_{PHL}/t_{PLH}	propagation delay nCP to nQ, n$\overline{Q}$ n$\overline{S}_D$ to nQ, n$\overline{Q}$ n$\overline{R}_D$ to nQ, n$\overline{Q}$	C_L = 15 pF; V_{CC} = 5 V	 15 12 12	 17 14 15	 ns ns ns
f_{max}	maximum clock frequency		75	61	MHz
C_I	input capacitance		3.5	3.5	pF
C_{PD}	power dissipation capacitance per flip-flop	notes 1 and 2	20	22	pF



The shaded areas indicate when the input is permitted to change for predictable output performance.

Fig.6 Waveforms showing the clock (nCP) to output (nQ, nQ̄) propagation delays, the clock pulse width, the nJ, nK̄ to nCP set-up, the nCP to nJ, nK̄ hold times, the output transition times and the maximum clock pulse frequency.



(1) HC: V_M = 50%; V_I = GND to V_{CC}.
HCT: V_M = 1.3 V; V_I = GND to 3 V.

Fig.7 Waveforms showing the set (nS̄D) and reset (nR̄D) input to output (nQ, nQ̄) propagation delays, the set and reset pulse widths and the nR̄D, nS̄D to nCP removal time.

SYMBOL	PARAMETER	T_{amb} (°C)							UNIT	TEST CONDITIONS	
		74HC								V_{CC} (V)	WAVEFORMS
		+25			−40 to +85		−40 to +125				
		min.	typ.	max.	min.	max.	min.	max.			
t_{PHL}/ t_{PLH}	propagation delay nCP to nQ, $n\overline{Q}$		50 18 14	175 35 30		220 44 37		265 53 45	ns	2.0 4.5 6.0	Fig.6
t_{PLH}	propagation delay $n\overline{S}_D$ to nQ		30 11 9	120 24 20		150 30 26		180 36 31	ns	2.0 4.5 6.0	Fig.7
t_{PHL}	propagation delay $n\overline{S}_D$ to $n\overline{Q}$		41 15 12	155 31 26		195 39 33		235 47 40	ns	2.0 4.5 6.0	Fig.7
t_{PHL}	propagation delay $n\overline{R}_D$ to nQ		41 15 12	185 37 31		230 46 39		280 56 48	ns	2.0 4.5 6.0	Fig.7
t_{PLH}	propagation delay $n\overline{R}_D$ to $n\overline{Q}$		39 14 11	170 34 29		215 43 37		255 51 43	ns	2.0 4.5 6.0	Fig.7
t_{THL}/ t_{TLH}	output transition time		19 7 6	75 15 13		95 19 16		110 22 19	ns	2.0 4.5 6.0	Fig.6
t_W	clock pulse width HIGH or LOW	80 16 14	19 7 6		100 20 17		120 24 20		ns	2.0 4.5 6.0	Fig.6
t_W	set or reset pulse width HIGH or LOW	80 16 14	14 5 4		100 20 17		120 24 20		ns	2.0 4.5 6.0	Fig.7
t_{rem}	removal time $n\overline{S}_D$, $n\overline{R}_D$ to nCP	70 14 12	19 7 6		90 18 15		105 21 18		ns	2.0 4.5 6.0	Fig.7
t_{su}	set-up time nJ, $n\overline{K}$ to nCP	70 14 12	17 6 5		90 18 15		105 21 18		ns	2.0 4.5 6.0	Fig.6
t_h	hold time nJ, $n\overline{K}$ to nCP	5 5 5	0 0 0		5 5 5		5 5 5		ns	2.0 4.5 6.0	Fig.6
f_{max}	maximum clock pulse frequency	6.0 30 35	22 68 81		5.0 24 28		4.0 20 24		MHz	2.0 4.5 6.0	Fig.6

15. En résumé

15.1 Décomposer un système en blocs fonctionnels structurés

- Définir les modules fonctionnels et leurs noms
- Déterminer les signaux de dialogue entre ces modules ainsi que leurs noms
- **Doser** le "cocktail" software + hardware synchrone intégrant un soupçon d'asynchrone
  - Équilibrer logiciel et matériel, hard et soft, ce qui prendra de l'espace et ce qui prendra du temps
  - Privilégier le synchrone et savoir utiliser l'asynchrone ponctuellement et à bon escient
- Pour l'analyse d'un schéma, partir de la fin et savoir effectuer une lecture en logique positive ou négative

15.2 Concevoir les routes de communication numérique au moyen d'éléments du "mécano logique"

- NAND, NOR, XOR
- Décodeurs-démultiplexeurs et multiplexeurs
- Mémoires (RAM, ROM, EEPROM…)
- Bascules RS, D, JK,T
- Flip-flops, latches, registres (SISO, SIPO, PISO, PIPO, FIFO, FILO)
- Compteurs programmables : savoir jouer avec
- PLA, PAL, FPGA...
- Sorties totem-pole, collecteur ou drain ouvert, 3 états, buffer
- Buffer directionnel et bidirectionnel
- Les bus
  - Basse et haute impédance, 3 états, collecteur ouvert
  - Bus multiplexé et non multiplexé
  - Systèmes multi-bus; penser à
    - Driver de bus bidirectionnel ou non
    - Registres flip-flop
    - Registres latch (transparents)
- Décoder des adresses
  - Décodage partiel et décodage complet
  - Décodeur-démultiplexeur, comparateur, PLA & PAL

15.3 Gérer le temps

15.3.1 Temporisations

- Elles peuvent être réalisées numériquement (compteurs) ou analogiquement (monostables) et être ré-enclenchables ou non ré-enclenchables.

15.3.2 Séquenceurs ou machines d'état

Au choix:

- Utiliser des RS (asynchrone) ou des JK (synchrone) ou avec toutes les entrées J, K, H, R, S
  - Déterminer les états nécessaires
  - Questions à poser :
    - **Qu'est-ce qui enclenche** cet état → S= ou J=
    - **Qu'est-ce qui déclenche** cet état → R= ou K=
    - Attention: vérifier que l'on n'utilise pas les combinaisons R=S=1 ou J=K=1
- Utiliser une structure Flip-flop D + Mémoire (structure adaptée à une modélisation par RdP)
- Utiliser une structure Compteur + Multiplexeur + Mémoire (modélisation par RdP)
  - CTR réalise les états,

- MUX sélectionne les transitions
- MEM réalise les champs d'action, de sensibilité aux transitions, de chargement...
- ajouter d'autres éléments suivant convenance pour des sauts conditionnels...

- **Noter qu'il y aura toujours lieu d'imposer l'état initial du système**

15.4 Règles d'hygiène : découplage, trigger et (re)-synchronisation

15.4.1 Soigner les alimentation et découpler correctement

- Les liaisons sont résistives, selfiques, capacitives; minimiser leur longueur
- **Les entrées CMOS sont capacitives et nécessitent pour chaque transition de leurs entrées un appoint de courant qui doit être fourni par une réserve d'énergie locale immédiatement accessible,...**
  - Ce qui nécessite une bonne implantation des masses et des découplages efficaces, qui ne peuvent être réalisés que par …
  - Une utilisation adéquate des condensateurs de découplage en tenant compte qu'ils sont R & L & C

type	capacité	taille relative	courant de fuite	self	utilisation	
					alimentations et découplage	intégrateurs, oscillateurs, monostables
chimique	+++	- - -	+++	+++	efficace en B.F.,nul en HF	problème de fuites
plastique, mylar	++	- -	- - -	+	intermédiaire	usage conseillé
céramique	+	+	+	- - -	efficace en H.F., nul en BF	

Note: HF= Haute Fréquence, BF= Basse Fréquence

- Attention, une entrée de porte logique CMOS laissée en l'air augmente considérablement la consommation du circuit, car elle fait "antenne" et réagit aux signaux reçus; donc ne jamais faire

15.4.2 Raisonner en termes d'interfaçage pour passer d'un type de logique à un autre

Toute liaison entre composants logiques de types différents masque des problèmes d'interfaçage dont il faut tenir compte et surtout ne pas ignorer, car chaque entrée a sa rapidité propre, son seuil logique propre et toute comparaison d'une entrée avec son seuil logique se fait avec du bruit

- **Passage logique "lente" vers logique "rapide"**
  - 1er problème: influence du bruit pendant le temps de passage au voisinage du seuil;
    SOLUTION: intercaler un TRIGGER
  - 2nd problème: chaque porte a son niveau logique
    SOLUTION: intercaler une porte "RAPIDE"

SOLUTION COMMUNE à ces deux problèmes: le **TRIGGER "RAPIDE"**

- Se souvenir qu'il n'y a jamais de signaux synchrones et que les sorties d'un système synchrone sont asynchrones(cf. mesures effectuées en TP sur les flip-flop)
  SOLUTION: **SYNCHRONISER** sur un front d'horloge

- **Passage logique "rapide" vers logique "lente"**
  - Largeur d'impulsion trop courte pour la logique lente, ce qui fait qu'un signal trop court ne serait pas vu!
    SOLUTIONS: élargir l'impulsion, employer un monostable, faire un transfert utilisant un handshake

15.4.3 Penser à synchroniser et re-synchroniser si nécessaire

Se souvenir……qu'il est « presque toujours » nécessaire
de (re)synchroniser entrées et actions

15.5 Tables de fonctionnement de quelques composants de la série 74xx

G1	/G2A	/G2B	C B A	/Y0	/Y1	/Y2	/Y3	/Y4	/Y5	/Y6	/Y7
0	-	-	- - -	1	1	1	1	1	1	1	1
-	1	-	- - -	1	1	1	1	1	1	1	1
-	-	1	- - -	1	1	1	1	1	1	1	1
1	0	0	0 0 0	0	1	1	1	1	1	1	1
1	0	0	0 0 1	1	0	1	1	1	1	1	1
1	0	0	0 1 0	1	1	0	1	1	1	1	1
1	0	0	0 1 1	1	1	1	0	1	1	1	1
1	0	0	1 0 0	1	1	1	1	0	1	1	1
1	0	0	1 0 1	1	1	1	1	1	0	1	1
1	0	0	1 1 0	1	1	1	1	1	1	0	1
1	0	0	1 1 1	1	1	1	1	1	1	1	0

- 74HC138 : **décodeur-démultiplexeur** 1 vers 8 muni de trois entrées de validation.

/G	C	B	A	Y	/W=/Y
1	-	-	-	0	1
0	0	0	0	D0	/D0
0	0	0	1	D1	/D1
0	0	1	0	D2	/D2
0	0	1	1	D3	/D3
0	1	0	0	D4	/D4
0	1	0	1	D5	/D5
0	1	1	0	D6	/D6
0	1	1	1	D7	/D7

- 74HC151 : **multiplexeur** 8 vers 1 muni d'une entrée de validation et de deux sorties complémentaires.

Modes	/CLR	CLK	ENP	ENT	/LOAD	Q^+_{DCBA}
RAZ	0	-:HC160 & 161 ↑:HC162 & 163	-	-	-	0000
Chargement	1	↑	-	-	0	DCBA
Comptage	1	↑	1	1	1	Q+1
Maintien	1	↑	-	0	1	Q
Maintien	1	↑	0	-	1	Q

- 74HC160 : **compteur** synchrone de type décade, à chargement synchrone et à RAZ asynchrone. Sa sortie RCO est à 1 quand ENT=1 et que le compteur est dans l'état 9.
- 74HC161 : **compteur** binaire synchrone à chargement synchrone et à RAZ asynchrone. Sa sortie RCO est à 1 quand ENT=1 et que le compteur est dans l'état 15.
- 74HC162 : **compteur** synchrone de type décade, à chargement synchrone et à RAZ synchrone. Sa sortie RCO est à 1 quand ENT=1 et que le compteur est dans l'état 9.
- 74HC163 : **compteur** binaire synchrone à chargement synchrone et à RAZ synchrone. Sa sortie RCO est à 1 quand ENT=1 et que le compteur est dans l'état 15.

/CLR	/A	B	Q	/Q
0	-	-	0	1
-	1	-	0	1
-	-	0	0	1
1	0	↑		
1	↓	1		
↑	0	1		

- 74HC123 : contient deux **monostables réenclenchables**.
- 74HC221 : contient deux **monostables non réenclenchables**.

/1G	$1A_n$	$1Y_n$
0	0	0
0	1	1
1	-	Z

/2G	$2A_n$	$2Y_n$
0	0	0
0	1	1
1	-	Z

- 74HC244 : double **buffer** de 4 bits, à sorties 3-state.

Modes	/G	DIR	A_i	B_i
Entrée B & sortie A	0	0	$=B_i$	
Entrée A & sortie B	0	1		$=A_i$
3-State	1	-	Z	Z

- 74HC245 : **transceiver** (buffer bidirectionnel) 8 bits 3-state.

Modes	/CLR	CLK	A	B	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H
RAZ	0	-	-	-	0	0	0	0	0	0	0	0
Z												
Shift	1	↑	a	b	a.b	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G

74HC164 : **registre à décalage** 8 bits « SIPO »: entrées en « série » et sorties en « parallèle » + entrée de RAZ

Modes	/CLR	CLK	Q_i^+
RAZ	0	-	0
Chargement	1	↑	D_i

- 74HC273 : **flip-flop D** edge triggered 8 bits + entrée de RAZ.

Modes	/OE	Enable G	Bascules internes q_i^+	sorties Q_i
Transparent	0	1	D_i	D_i
Mémorisation et lecture	0	0	q_i	q_i
Mémo & haute impédance	1	0	q_i	Z
Transparent & hte impédance	1	1	D_i	Z

- 74HC373 : **latch D transparent** de 8 bits avec sorties 3-state.

/OE	CLK	Latches internes q_i^+	Sorties Q_i
0	↑	D_i	q_i
0	0 ou 1 ou ↓	q_i	q_i
1	↑	D_i	Z
1	0 ou 1 ou ↓	q_i	Z

- 74HC374 : **flip-flop D** edge triggered 8 bits ;sorties 3-state.

	/G	CLK	Q_i^+
Maintien	1	-	Q_i
Chargement	0	↑	D_i

- 74HC377 : **flip-flop D** edge triggered 8 bits + entrée de validation.

/G2	/G1	A_n	Y_n
0	0	0	0
0	0	1	1
-	1	-	Z
1	-	-	Z

- 74HC541 : **buffer** (étage tampon) de 8 bits, à sorties 3-state.

NB: « ↑ & ↓ » = fronts montants & descendants
« - » = combinaison indifférente
« Z » = état haute impédance