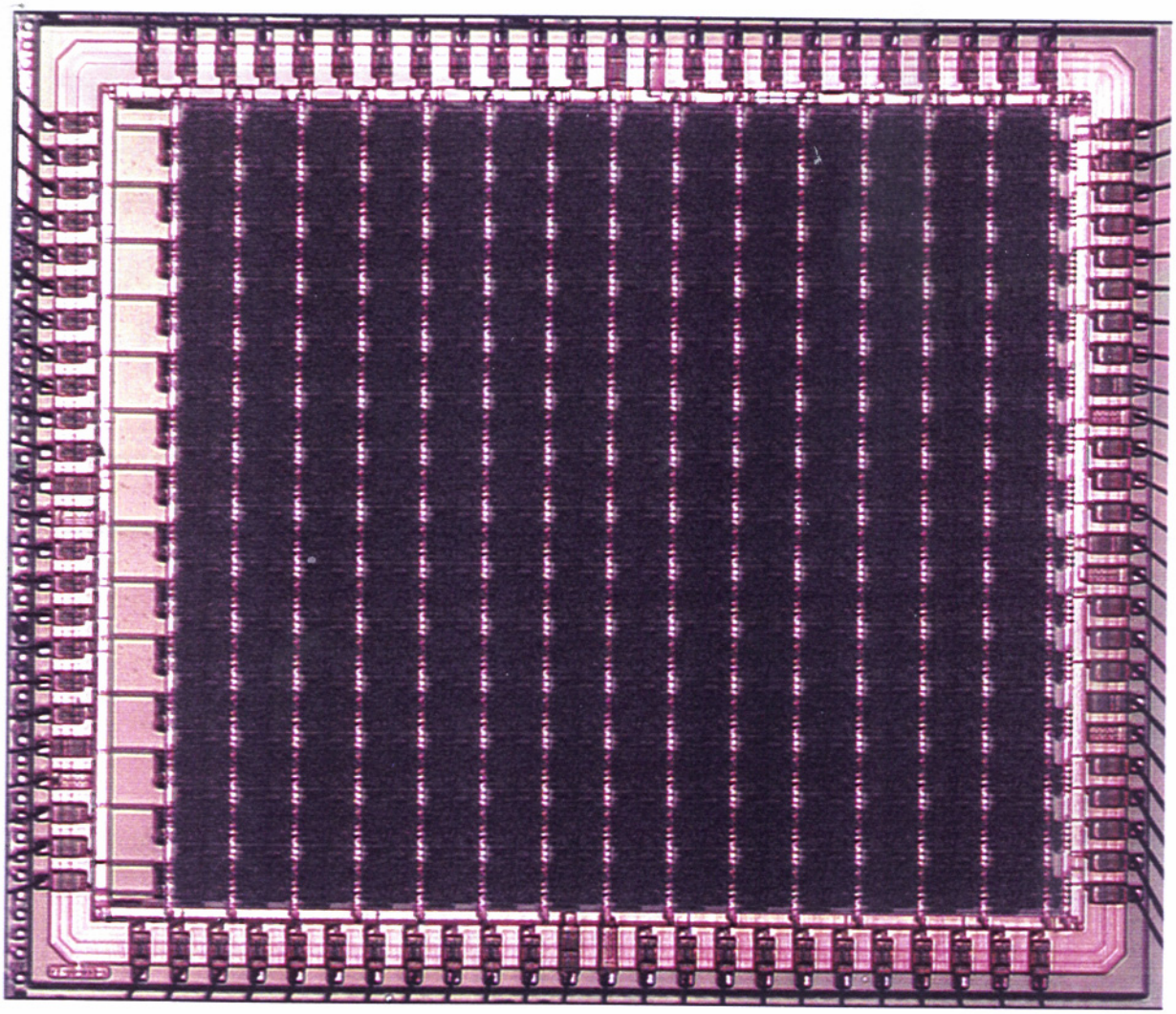
ANNEXES

- photomicrographie du circuit AMPHIN (§4.5)
- programmes en assembleur (§5.4)
    - "produit.asm"
    - "recons_iter.asm"
    - "recons_bloc.asm"
    - "recons_new.asm"
    - "rotate.asm"
- visualisation du plan mémoire destination au cours de l'exécution de l'algorithme de rotation (§5.4.4)
- schéma niveau portes de l'interface de communication non-bloquante (§5.5.2.b)



FT-VD-033-1

Nov 7 1996 11:14:16 **produit.asm** Page 1

```
1     ;
2     ; produit matriciel 16*16
3     ; elements dans [-128,127]
4     ; resultat sur 19 bits signes
5     ;
6     ; A UTILISER AVEC reseau4.c
7     ;
8
9     ;r0=aij
10    ;r1=bij
11    ;r14=x_PE=j
12    ;r15=y_PE=i
13    ;mem #0=aik
14    ;mem #1=bkj
15    ;resu dans [r10|r11|r12]
16
17    .begin
18
19            set_idle        #0
20            movei   #15 , r2        ;r2=compteur
21            move    r0 , r3         ;r3=aik initialise a aij
22            move    r1 , r4         ;r4=bkj initialise a bij
23            move    r14 , r5        ;r5=pointeur aik initialise a j
24            move    r15 , r6        ;r6=pointeur bkj initialise a i
25            store   #0 , r3 , r5    ;stocke aij
26            store   #1 , r4 , r6    ;stocke bij
27
28    label0  movei   #%00000100 , r13        ;r13=dir EST
29            wait_b4_send    r13
30            send    r13 , r3 , r3   ;envoie aik a droite
31            movei   #%00000010 , r13        ;r13=dir SUD
32            wait_b4_send    r13
33            send    r13 , r4 , r4   ;envoie bkj en bas
34            movei   #%00000001 , r13        ;r13=dir OUEST
35            wait_b4_get     r13
36            get     r13 , r3 , r3   ;recoit ai,k-1 de gauche
37            subi    #1 , r5
38            andi    #%00001111 , r5 ;r5=(r5-1)%16
39            store   #0 , r3 , r5    ;stocke ai,k-1
40            movei   #%00001000 , r13        ;r13=dir NORD
41            wait_b4_get     r13
42            get     r13 , r4 , r4   ;recoit bk-1,j du haut
43            subi    #1 , r6
44            andi    #%00001111 , r6 ;r6=(r6-1)%16
45            store   #1 , r4 , r6    ;stocke bk-1,j
46            subi    #1 , r2
47            brnz    #label0 , r2    ;boucle
48
49            movei   #0 , r2         ;r2=compteur
50            movei   #0 , r7
51            movei   #0 , r10        ;[r10|r11|r12]=resu
52            movei   #0 , r11
53            movei   #0 , r12
54    label1  movei   #0 , r5         ;r5=signe resu
55            load    #0 , r3 , r2    ;r3=aik
56            move    r3 , r6
57            andi    #%10000000 , r6 ;r3 negatif ?
58            brz     #label2 , r6    ;si oui:
59            sub     r7 , r3 , r3    ;r3=-r3
60            xori    #1 , r5         ;changement de signe
61    label2  load    #1 , r4 , r2    ;r4=bkj
62            move    r4 , r6
63            andi    #%10000000 , r6 ;r4 negatif ?
64            brz     #label3 , r6    ;si oui:
65            sub     r7 , r4 , r4    ;r4=-r4
66            xori    #1 , r5         ;changement de signe
67    label3  mulu    r3 , r4 , r8    ;[r8|r9]=abs(aik*bkj)
68            brnz    #label4 , r5    ;si produit positif:
69            add     r9 , r12 , r12
70            sltu    r12 , r9 , r6   ;r6=carry
71            add     r6 , r8 , r8    ;debordement impossible
```

Nov 7 1996 11:14:16 **produit.asm** Page 2

```
72            add     r8 , r11 , r11
73            sltu    r11 , r8 , r6   ;r6=carry
74            add     r6 , r10 , r10
75            bra     #label5
76    label4  sltu    r12 , r9 , r6   ;r6=carry
77            sub     r12 , r9 , r12
78            add     r6 , r8 , r8    ;debordement impossible
79            sltu    r11 , r8 , r6   ;r6=carry
80            sub     r11 , r8 , r11
81            sub     r10 , r6 , r10
82    label5  addi    #1 , r2
83            andi    #%00001111 , r2
84            brnz    #label1 , r2    ;boucle jusqu'a 15
85
86            set_idle        #1
87            wait_b4_get     r7      ;blocage
88
89    .end
```

Apr 29 1997 14:51:31 **recons_iter.asm** Page 1

```
1     ;
2     ; reconstruction positive
3     ; version iterative
4     ;
5     ; A UTILISER AVEC reseau5.c
6     ;
7
8     ; r0=reg_x
9     ; r1=reg_y
10    ; r2=reg_bord = 0 ssi PE de bord
11    ; mem #0 et #1 reserves I/O
12    ; mem #2=image de reference
13    ; mem #3=reconstruction courante,
14    ;       supposee initialisee avec le resultat de l'erosion
15
16    .begin
17            set_idle        #0
18            movei   #0 , r15        ;r15=compteur numero bloc
19            movei   #1 , r11        ;r11=increment bloc
20            movei   #0 , r10        ;r10=compteur iteration image (pour mesure)
21    label5  movei   #0 , r14        ;r14=flag de modif
22            addi    #1 , r10
23    label4  movei   #16 , r9        ;r9=compteur iteration bloc
24            load    #3 , r3 , r15   ;r3=pixel courant
25            move    r3 , r6
26            load    #2 , r8 , r15   ;r8=valeur ref
27
28    label2  movei   #%00001111 , r4 ;r4=dir NESO
29            wait_b4_send    r4
30            brz     #label0 , r2    ;traitement PE de bord
31    label6  send    r4 , r6 , r6    ;envoi du pixel
32
33    label1  movei   #%00001111 , r4 ;r4=dir NESO
34            wait_b4_get     r4
35            get     r4 , r5 , r5    ;r5=valeur voisine
36            max     r5 , r6 , r6
37            wait_b4_get     r4
38            get     r4 , r5 , r5    ;r5=valeur voisine
39            max     r5 , r6 , r6
40            wait_b4_get     r4
41            get     r4 , r5 , r5    ;r5=valeur voisine
42            max     r5 , r6 , r6
43            wait_b4_get     r4
44            get     r4 , r5 , r5    ;r5=valeur voisine
45            max     r5 , r6 , r6
46            min     r6 , r8 , r6
47            subi    #1 , r9
48            brnz    #label2 , r9    ;iteration bloc suivante
49            sltu    r3 , r6 , r7    ;val precedente<val courante?
50            brz     #label3 , r7    ;sinon bloc suivant
51            store   #3 , r6 , r15   ;mise a jour pixel
52            movei   #1 , r14        ;flag de modif actif
53    label3  add     r11, r15 , r15
54            brnz    #label4 , r15   ;bloc suivant
55            xori    #%11111110 , r11        ;changement de sens
56            brnz    #label5 , r14   ;iteration image suivante
57
58            set_idle        #1
59            movei   #%00001111 , r4
60            wait_b4_get     r4      ;attente d'un voisin
61            set_idle        #0      ;si pas fini
62            bra     #label5
63
64    ;traitement des send pour les PE de bord
65    label0  brnz    #label7 , r1    ;traitement des PE du haut:saut si y!=0
66            movei   #0 , r12
67            move    r15 , r13       ;r13=no_bloc
68            andi    #%11110000 , r13
69            subi    #%11110000 , r13        ;y_bloc=15 ?
70            brz     #label10 , r13  ;si oui, envoi d'un zero
71            move    r15 , r13
```

Apr 29 1997 14:51:31 **recons_iter.asm** Page 2

```
72            addi    #16 , r13       ;new_no_bloc=no_bloc+16
73            load    #3, r12 , r13
74    label10 movei   #%00001000 , r13        ;r13=dir NORD
75            send    r13 , r12 , r12 ;envoi du pixel
76            andi    #%00000111 , r4 ;supprime dir NORD
77            bra     #label8         ;saut vers le traitement gauche/droite
78    label7  move    r1 , r13        ;traitement des PE du bas
79            subi    #15 , r13
80            brnz    #label8 , r13   ;saut si y!=15
81            movei   #0 , r12
82            move    r15 , r13       ;r13=no_bloc
83            andi    #%11110000 , r13        ;y_bloc=0 ?
84            brz     #label11 , r13  ;si oui, envoi d'un zero
85            move    r15 , r13
86            subi    #16 , r13       ;new_no_bloc=no_bloc-16
87            load    #3 , r12 , r13
88    label11 movei   #%00000010 , r13        ;r13=dir SUD
89            send    r13 , r12 , r12 ;envoi du pixel
90            andi    #%00001101 , r4 ;supprime dir SUD
91    label8  brnz    #label9 , r0    ;traitement des PE de gauche:saut si x!=0
92            move    #0 , r12
93            move    r15 , r13       ;r13=no_bloc
94            andi    #%00001111 , r13
95            subi    #15 , r13       ;x_bloc=15 ?
96            brz     #label12 , r13  ;si oui, envoi d'un zero
97            move    r15 , r13
98            addi    #1 , r13        ;new_no_bloc=no_bloc+1
99            load    #3 , r12 , r13
100   label12 movei   #%00000001 , r13        ;r13=dir OUEST
101           send    r13 , r12 , r12 ;envoi du pixel
102           andi    #%00001110 , r4 ;supprime dir OUEST
103           bra     #label6         ;saut vers envois normaux restants
104   label9  move    r0 , r13        ;traitement des PE de droite
105           subi    #15 , r13
106           brnz    #label6 , r13   ;saut si x!=15
107           movei   #0 , r12
108           move    r15 , r13       ;r13=no_bloc
109           andi    #%00001111 , r13        ;x_bloc=0 ?
110           brz     #label13 , r13  ;si oui, envoi d'un zero
111           move    r15 , r13
112           subi    #1 , r13        ;new_no_bloc=no_bloc-1
113           load    #3 , r12 , r13
114   label13 movei   #%00000100 , r13        ;r13=dir EST
115           send    r13 , r12 , r12 ;envoi du pixel
116           andi    #%00001011 , r4 ;supprime dir EST
117           bra     #label6
118   .end
```

Jul 10 1997 17:21:26 **recons_bloc.asm** Page 1

```
1     ;
2     ; reconstruction positive
3     ; version dataflow avec partitionnement LSGP
4     ;
5     ; A UTILISER AVEC reseau6b.c
6     ;
7
8     ; r0=reg_x
9     ; r1=reg_y
10    ; mem #0 et #1 reserves I/O
11    ; mem #2=image de reference
12    ; mem #3=reconstruction courante,
13    ;       supposee initialisee avec le resultat de l'erosion
14    ; mem #4=directions ou il faut envoyer le pixel
15    ; mem #5=tag = 1 ssi pixel a envoyer
16    ;       initialise a 1 pour les pixels de bord des maxima regionaux ou locaux
17
18    .begin
19            set_idle        #0
20            movei   #0 , r5         ;r5=reg_zero
21            search  #4 , r5 , r5    ;selectionne tous les pixels
22            movei   #$ff , r4       ;r4=reg_mask
23            write   #4 , r5 , r4    ;direction=0
24            movei   #1 , r6         ;r6=reg_one
25            search  #5 , r6 , r4    ;selectionne les pixels a envoyer
26            movei   #%00001111 , r3 ;r3=reg_NESO
27            write   #4 , r3 , r4    ;direction=NESO
28            count   r7 , r5         ;r7=reg_test , r5=reg_all
29            brnz    #label6 , r7
30            brz     #label18 , r5   ;saut si pas de pixel a envoyer
31
32    label6  read    #3 , r8 , r9    ;r8=valeur courante, r9=no_pixel
33            read    #4 , r10 ,r9    ;r10=direction
34
35            move    r10 , r5
36            andi    #%00001000 , r5
37            brz     #label7 , r5    ;saut si pas NORD
38            move    r9 , r7
39            andi    #%11110000 , r7
40            brnz    #label8 , r7    ;saut si pas pixel du haut
41            brz     #label9 , r1    ;saut si PE du haut
42            move    r9 , r7
43            ori     #%11110000 , r7 ;r7=new_no_pixel
44            send    r5 , r8 , r7
45            brnz    #label7 , r5    ;saut si envoi echoue
46            bra     #label9
47    label8  move    r9 , r7         ;traitement pas pixel du haut
48            subi    #16 , r7        ;r7=no_pixel NORD
49            load    #3 , r11 , r7   ;r11=val pixel voisin
50            load    #2 , r12 , r7   ;r12=val ref voisin
51            sltu    r11 , r8 , r13  ;val voisin<val courante ?
52            brz     #label9 , r13   ;saut sinon
53            sltu    r11 , r12 , r13 ;val voisin<val ref voisin ?
54            brz     #label9 , r13   ;saut sinon
55            min     r8 , r12 , r11  ;mise a jour voisin
56            store   #3 , r11 , r7   ;ecriture valeur
57            movei   #%00001101 , r5 ;dir voisin=NEO
58            store   #4 , r5 , r7
59            store   #5 , r6 , r7    ;active tag
60    label9  andi    #%11110111 , r10        ;dir NORD effacee
61
62    label7  move    r10 , r5
63            andi    #%00000010 , r5
64            brz     #label10 , r5   ;saut si pas SUD
65            move    r9 , r7
66            andi    #%11110000 , r7
67            subi    #%11110000 , r7
68            brnz    #label11 , r7   ;saut si pas pixel du bas
69            move    r1 , r7
70            subi    #15 , r7
71            brz     #label12 , r7   ;saut si PE du bas
```

Jul 10 1997 17:21:26 **recons_bloc.asm** Page 2

```
72            move    r9 , r7
73            andi    #%00001111 , r7 ;r7=new_no_pixel
74            send    r5 , r8 , r7
75            brnz    #label10 , r5   ;saut si envoi echoue
76            bra     #label12
77    label11 move    r9 , r7         ;traitement pas pixel du bas
78            addi    #16 , r7        ;r7=no_pixel SUD
79            load    #3 , r11 , r7   ;r11=val pixel voisin
80            load    #2 , r12 , r7   ;r12=val ref voisin
81            sltu    r11 , r8 , r13  ;val voisin<val courante ?
82            brz     #label12 , r13  ;saut sinon
83            sltu    r11 , r12 , r13 ;val voisin<val ref voisin ?
84            brz     #label12 , r13  ;saut sinon
85            min     r8 , r12 , r11  ;mise a jour voisin
86            store   #3 , r11 , r7   ;ecriture valeur
87            movei   #%00000111 , r5 ;dir voisin=ESO
88            store   #4 , r5 , r7
89            store   #5 , r6 , r7    ;active tag
90    label12 andi    #%11111101 , r10        ;dir SUD effacee
91
92    label10 move    r10 , r5
93            andi    #%00000001 , r5
94            brz     #label13 , r5   ;saut si pas OUEST
95            move    r9 , r7
96            andi    #%00001111 , r7
97            brnz    #label14 , r7   ;saut si pas pixel de gauche
98            brz     #label15 , r0   ;saut si PE de gauche
99            move    r9 , r7
100           ori     #%00001111 , r7 ;r7=new_no_pixel
101           send    r5 , r8 , r7
102           brnz    #label13 , r5   ;saut si envoi echoue
103           bra     #label15
104   label14 move    r9 , r7         ;traitement pas pixel de gauche
105           subi    #1 , r7         ;r7=no_pixel OUEST
106           load    #3 , r11 , r7   ;r11=val pixel voisin
107           load    #2 , r12 , r7   ;r12=val ref voisin
108           sltu    r11 , r8 , r13  ;val voisin<val courante ?
109           brz     #label15 , r13  ;saut sinon
110           sltu    r11 , r12 , r13 ;val voisin<val ref voisin ?
111           brz     #label15 , r13  ;saut sinon
112           min     r8 , r12 , r11  ;mise a jour voisin
113           store   #3 , r11 , r7   ;ecriture valeur
114           movei   #%00001011 , r5 ;dir voisin=NSO
115           store   #4 , r5 , r7
116           store   #5 , r6 , r7    ;active tag
117   label15 andi    #%11111110 , r10        ;dir OUEST effacee
118           bra     #label13
119
120   label2  search  #5 , r6 , r4    ;cherche un pixel a envoyer
121           count   r7 , r5
122           brnz    #label6 , r7
123           brnz    #label6 , r5    ;saut vers l'envoi s'il reste des pixels
124           set_idle        #1      ;le PE a potentiellement fini
125           wait_b4_get     r10     ;attente d'un message, dir=NESO
126           set_idle        #0      ;le PE redevient actif
127   label18 bra     #label0         ;saut vers la reception du message
128
129   label13 move    r10 , r5
130           andi    #%00000100 , r5
131           brz     #label5 , r5    ;saut si pas EST
132           move    r9 , r7
133           andi    #%00001111 , r7
134           subi    #%00001111 , r7
135           brnz    #label16 , r7   ;saut si pas pixel de droite
136           move    r0 , r7
137           subi    #15 , r7
138           brz     #label17 , r7   ;saut si PE de droite
139           move    r9 , r7
140           andi    #%11110000 , r7 ;r7=new_no_pixel
141           send    r5 , r8 , r7
142           brnz    #label5 , r5    ;saut si envoi echoue
```

```
143             bra     #label17
144     label16 move    r9 , r7         ;traitement pas pixel de droite
145             addi    #1 , r7         ;r7=no_pixel EST
146             load    #3 , r11 , r7   ;r11=val pixel voisin
147             load    #2 , r12 , r7   ;r12=val ref voisin
148             sltu    r11 , r8 , r13  ;val voisin<val courante ?
149             brz     #label17 , r13  ;saut sinon
150             sltu    r11 , r12 , r13 ;val voisin<val ref voisin ?
151             brz     #label17 , r13  ;saut sinon
152             min     r8 , r12 , r11  ;mise a jour voisin
153             store   #3 , r11 , r7   ;ecriture valeur.
154             movei   #%00001110 , r5 ;dir voisin=NES
155             store   #4 , r5 , r7
156             store   #5 , r6 , r7    ;active tag
157     label17 andi    #%11111011 , r10        ;dir EST effacee
158
159     label5  store   #4 , r10 , r9   ;stocke les directions restantes
160             brz     #label1 , r10
161             movei   #1 , r10
162     label1  store   #5 , r10 , r9   ;mise a jour du tag
163     label0  move    r3 , r10        ;reception d'un message, r10=dir NESO
164             get     r10 , r8 , r9   ;s'il y a un message,r8=valeur,r9=no_pixel
165             sub     r10 , r3 , r7
166             brz     #label2 , r7    ;saut s'il n'y a pas de message
167             load    #2 , r11 , r9   ;r11=valeur de reference
168             load    #3 , r12 , r9   ;r12=valeur courante
169             sltu    r12 , r8 , r7   ;valeur recue>valeur courante ?
170             brz     #label0 , r7    ;saut sinon (message ignore)
171             sltu    r12 , r11 , r7  ;valeur courante<valeur de reference ?
172             brz     #label0 , r7    ;saut sinon (message ignore)
173             min     r11 , r8 , r8   ;mise a jour du pixel (dilat. geodesique)
174             store   #3 , r8 , r9    ;stocke la nouvelle valeur
175             bra     #label5         ;saut vers suite de la mise a jour
176     .end
```

Jul 10 1997 17:21:20 **recons_new.asm** Page 1

```
1     ;
2     ; reconstruction positive
3     ; version dataflow avec entrelacement
4     ;
5     ; A UTILISER AVEC reseau6a.c
6     ;
7
8     ; r0=reg_x
9     ; r1=reg_y
10    ; r2=reg_bord = 0 ssi PE de bord
11    ; mem #0 et #1 reserves I/O
12    ; mem #2=image de reference
13    ; mem #3=reconstruction courante,
14    ;       initialisee avec le resultat de l'erosion
15    ; mem #4=directions ou il faut envoyer le pixel
16    ; mem #5=tag = 1 ssi pixel a envoyer
17    ;       initialise a 1 pour les pixels de bord des maxima regionaux ou locaux
18
19    .begin
20            set_idle        #0
21            movei   #0 , r5         ;r5=reg_zero
22            search  #4 , r5 , r5    ;selectionne tous les pixels
23            movei   #$ff , r4       ;r4=reg_mask
24            write   #4 , r5 , r4    ;direction=0
25            movei   #1 , r6         ;r6=reg_one
26            search  #5 , r6 , r4    ;selectionne les pixels a envoyer
27            movei   #%00001111 , r3 ;r3=reg_NESO
28            write   #4 , r3 , r4    ;direction=NESO
29            count   r7 , r5         ;r7=reg_test , r5=reg_all
30            brnz    #label10 , r7
31            brz     #label0 , r5    ;saut si pas de pixel a envoyer
32
33    label10 read    #3 , r8 , r9    ;r8=valeur , r9=no_bloc
34            read    #4 , r10 ,r9    ;r10=direction
35            brz     #label6 , r2    ;traitement des PE de bord
36            send    r10 , r8 , r9   ;envoie le pixel:valeur et no_bloc
37    label5  store   #4 , r10 , r9   ;stocke les directions restantes
38            brz     #label1 , r10
39            movei   #1 , r10
40    label1  store   #5 , r10 , r9   ;mise a jour du tag
41    label0  move    r3 , r10        ;reception d'un message, r10=dir NESO
42            get     r10 , r8 , r9   ;s'il y a un message,r8=valeur,r9=no_bloc
43            sub     r10 , r3 , r7
44            brz     #label2 , r7    ;saut s'il n'y a pas de message
45            load    #2 , r11 , r9   ;r11=valeur de reference
46            load    #3 , r12 , r9   ;r12=valeur courante
47            sltu    r12 , r8 , r7   ;valeur recue>valeur courante ?
48            brz     #label0 , r7    ;saut sinon (message ignore)
49            sltu    r12 , r11 , r7  ;valeur courante<valeur de reference ?
50            brz     #label0 , r7    ;saut sinon (message ignore)
51            min     r11 , r8 , r8   ;mise a jour du pixel (dilat. geodesique)
52            store   #3 , r8 , r9    ;stocke la nouvelle valeur
53            bra     #label5         ;saut vers suite de la mise a jour
54    label2  search  #5 , r6 , r4    ;cherche un pixel a envoyer
55            count   r7 , r5
56            brnz    #label10 , r7
57            brnz    #label10 , r5   ;saut vers l'envoi s'il reste des pixels
58
59            set_idle        #1      ;le PE a potentiellement fini
60            wait_b4_get     r10     ;attente d'un message, dir=NESO
61            set_idle        #0      ;le PE redevient actif
62            bra     #label0         ;saut vers la reception du message
63
64    ;traitement des send pour les PE de bord
65    label6  movei   #0 , r5         ;r5=directions restantes
66            brnz    #label7 , r1    ;traitement des PE du haut:saut si y!=0
67            move    r10 , r7
68            andi    #%00001000 , r7 ;envoi vers le NORD ?
69            brz     #label8 , r7    ;saut sinon
70            andi    #%11110111 , r10        ;direction NORD traitee
71            move    r9 , r7
```

Jul 10 1997 17:21:20 **recons_new.asm** Page 2

```
72            andi    #%11110000 , r7 ;y_bloc=0 ?
73            brz     #label8 , r7    ;si oui, message elimine
74            move    r9 , r7
75            subi    #16 , r7        ;new_no_bloc=no_bloc-16
76            movei   #%00001000 , r13        ;r13=dir NORD
77            send    r13 , r8 , r7   ;propage le pixel (valeur,new_no_bloc)
78            or      r13 , r5 , r5   ;modif des dir. restantes si envoi a echoue
79            bra     #label8         ;saut vers le traitement gauche/droite
80    label7  move    r1 , r7         ;traitement des PE du bas
81            subi    #15 , r7
82            brnz    #label8 , r7    ;saut si y!=15
83            move    r10 , r7
84            andi    #%00000010 , r7 ;envoi vers le SUD ?
85            brz     #label8 , r7    ;saut sinon
86            andi    #%11111101 , r10        ;direction SUD traitee
87            move    r9 , r7
88            andi    #%11110000 , r7
89            subi    #%11110000 , r7 ;y_bloc=15 ?
90            brz     #label8 , r7    ;si oui, message elimine
91            move    r9 , r7
92            addi    #16 , r7        ;new_no_bloc=no_bloc+16
93            movei   #%00000010 , r13        ;r13=dir SUD
94            send    r13 , r8 , r7   ;propage le pixel (valeur,new_no_bloc)
95            or      r13 , r5 , r5   ;modif des dir. restantes si envoi a echoue
96    label8  brnz    #label9 , r0    ;traitement des PE de gauche:saut si x!=0
97            move    r10 , r7
98            andi    #%00000001 , r7 ;envoi vers l'OUEST ?
99            brz     #label11 , r7   ;saut sinon
100           andi    #%11111110 , r10        ;direction OUEST traitee
101           move    r9 , r7
102           andi    #%00001111 , r7 ;x_bloc=0 ?
103           brz     #label11 , r7   ;si oui, message elimine
104           move    r9 , r7
105           subi    #1 , r7         ;new_no_bloc=no_bloc-1
106           movei   #%00000001 , r13        ;r13=dir OUEST
107           send    r13 , r8 , r7   ;propage le pixel (valeur,new_no_bloc)
108           or      r13 , r5 , r5   ;modif des dir. restantes si envoi a echoue
109           bra     #label11        ;saut vers envois normaux restants
110   label9  move    r0 , r7         ;traitement des PE de droite
111           subi    #15 , r7
112           brnz    #label11 , r7   ;saut si x!=15
113           move    r10 , r7
114           andi    #%00000100 , r7 ;envoi vers l'EST ?
115           brz     #label11 , r7   ;saut sinon
116           andi    #%11111011 , r10        ;direction EST traitee
117           move    r9 , r7
118           andi    #%00001111 , r7
119           subi    #15 , r7        ;x_bloc=15 ?
120           brz     #label11 , r7   ;si oui, message elimine
121           move    r9 , r7
122           addi    #1 , r7         ;new_no_bloc=no_bloc+1
123           movei   #%00000100 , r13        ;r13=dir EST
124           send    r13 , r8 , r7   ;propage le pixel (valeur,new_no_bloc)
125           or      r13 , r5 , r5   ;modif des dir. restantes si envoi a echoue
126   label11 send    r10 , r8 , r9   ;envoi dans les dir. normales restantes
127           or      r5 , r10 , r10  ;modif des dir. restantes si envoi a echoue
128           bra     #label5         ;saut vers mise a jour direction/tag
129   .end
```

Jul 4 1997 14:16:04 **rotate.asm** Page 1

```
1     ;
2     ; rotation autour de x=128,y=128
3     ; angle compris dans [-pi/2,0[ U ]0,pi/2]
4     ;
5     ; l'objet doit etre dans le carre (65,65)-(191,191)
6     ; objet = pixels differents de 0 pour simplifier
7     ; sinon utiliser le masque binaire
8     ;
9     ; A UTILISER AVEC reseau3.c
10    ;
11
12    ;r14=reg_x
13    ;r15=reg_y
14    ;mem #0 et #1 reserves I/O
15    ;mem #2 et #3=table tan(t/2) et sin(t)
16    ;mem #4=source (=dest a la fin)
17    ;mem #5=dest
18    ;mem #6=tag_objet|tag_envoi|000000
19    ;mem #7=translations
20
21    .begin
22            set_idle        #0
23            movei   #3 , r9         ;r9=nb de passes
24                                    ;commentaires valables pour passes 1,3
25            movei   #%00010000 , r0
26            wait_b4_get     r0      ;attend un message de C pour theta
27            get     r0 , r1 , r3    ;r3=theta
28            brz     #label20 , r1   ;si r1!=0 (angle negatif):
29            ori     #%10000000 , r9 ;marque r9
30
31    label20 move    r9 , r10
32            andi    #%00000001 , r10
33            brnz    #label18 , r10  ;si passe 2:
34            movei   #%00001010 , r11        ;r11=dir NORD|SUD
35            movei   #%00000010 , r12        ;r12=dir SUD
36            move    r15 , r13       ;r13=y_PE
37            load    #3 , r2 , r3    ;r2=sin(theta)
38            bra     #label19
39    label18 movei   #%00000101 , r11        ;r11=dir EST|OUEST
40            movei   #%00000100 , r12        ;r12=dir EST
41            move    r14 , r13       ;r13=x_PE
42            load    #2 , r2 , r3    ;r2=tan(theta/2)
43
44    label19 movei   #0 , r0
45            search  #4 , r0 , r0    ;selectionne tous les pixels
46            movei   #Sff , r1
47            write   #5 , r0 , r1    ;init dest
48            movei   #%10000000 , r0
49            write   #6 , r0 , r1    ;tag_objet = 1
50            movei   #0 , r0
51            search  #4 , r0 , r1    ;selectionne les pixels a 0
52            write   #6 , r0 , r1    ;tag_objet = 0
53
54            movei   #0 , r4         ;init r4=y_bloc<<4
55            movei   #0 , r0
56            move    r10 , r1
57            xori    #%00000001 , r1
58            shli    #7 , r1
59            xor     r9 , r1 , r1
60            andi    #%10000000 , r1 ;r1!=0 si inversion a faire
61    label2  move    r4 , r5
62            brnz    #label22 , r10  ;si passe 2:
63            or      r14 , r5 , r5
64            bra     #label23
65    label22 or      r15 , r5 , r5   ;r5=y_ligne
66    label23 subi    #128 , r5       ;centrage sur 128
67            brz     #label4 , r1    ;saut si pas d'inversion
68            sub     r0 , r5 , r5    ;y=-y
69    label4  brnz    #label3 , r2    ;si coef=0:
70            move    r5 , r6         ;resu=y (coef=256)
71            bra     #label1
```

Jul 4 1997 14:16:04 **rotate.asm** Page 2

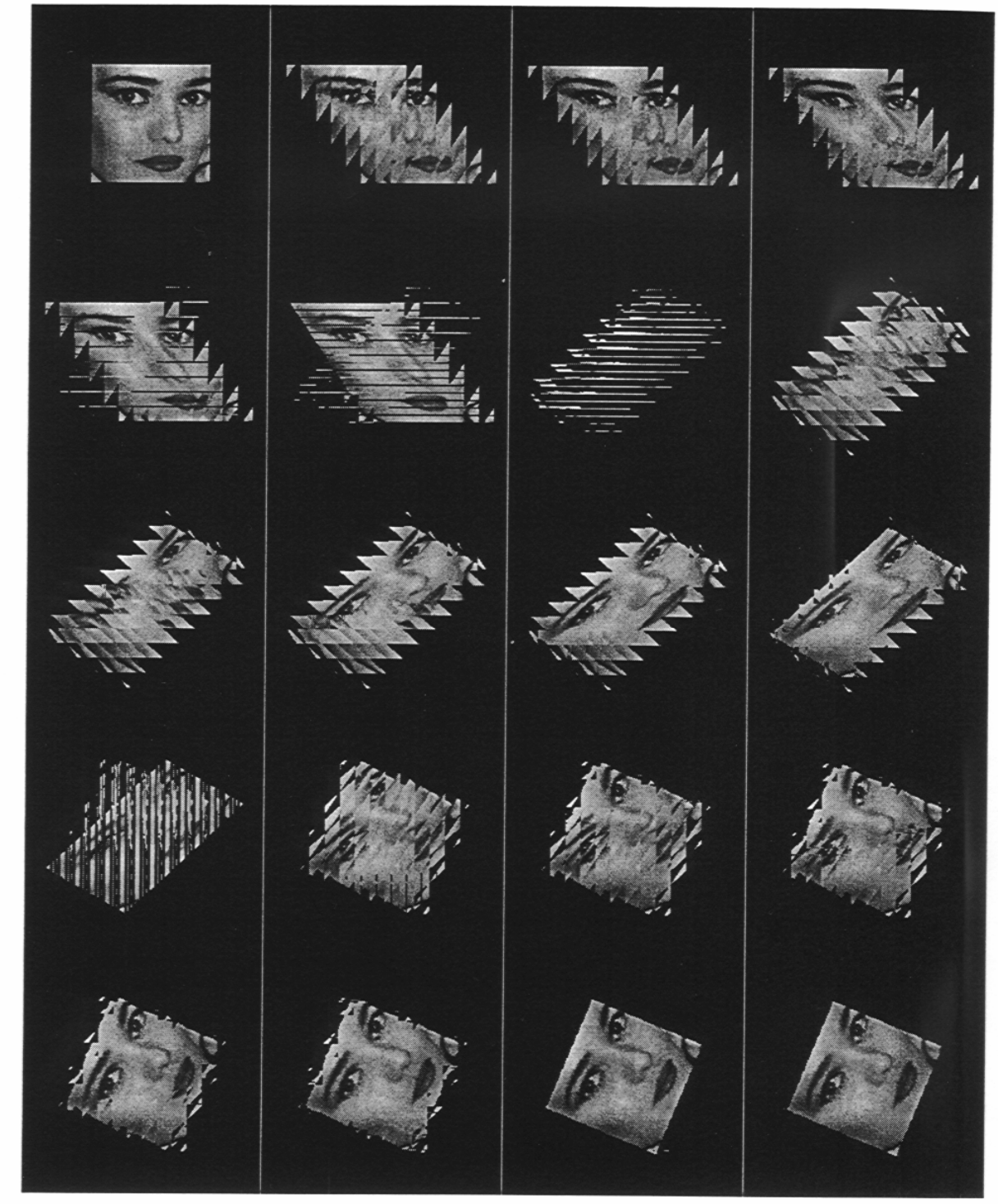
```
72    label3  move    r5 , r8
73            andi    #%10000000 , r8
74            brz     #label0 , r8    ;saut si y positif
75            sub     r0 , r5 , r5    ;y=-y
76    label0  mulu    r2 , r5 , r6    ;[r6|r7]=tan(theta/2)*abs(y)
77            andi    #%10000000 , r7
78            brz     #label5 , r7
79            addi    #1 , r6         ;arrondi
80    label5  brz     #label1 , r8    ;saut si y positif
81            sub     r0 ,r6 ,r6      ;y=-y
82    label1  store   #7 , r6 , r4    ;memorise le deplacement en (y_bloc<<4)
83            andi    #%00001111 , r6 ;r6 = deplacement modulo 16
84            move    r6 , r7
85            andi    #%00001000 , r7
86            brz     #label6 , r7    ;saut si r6 dans [0,7] = a droite
87            movei   #16 , r7
88            sub     r7 , r6 , r6
89            ori     #%10000000 , r6 ;sinon r6=128+(16-r6) = a gauche
90    label6  addi    #1 , r4
91            store   #7 , r6 , r4    ;memorise le deplacement en (y_bloc<<4)+1
92            addi    #15 , r4        ;r4=(y_bloc++)<<4
93            brnz    #label2 , r4
94
95            movei   #%10000000 , r0
96    label9  search  #6 , r0 , r0    ;cherche un pixel de l'objet
97            count   r1 , r2
98            brnz    #label28 , r1
99            brz     #label7 , r2
100   label28 read    #4 , r1 , r2    ;r1=valeur pixel , r2=no_bloc
101           brnz    #label26 , r10  ;si passe 2:
102           move    r2 , r4
103           andi    #%11110000 , r4
104           or      r13 , r4 ,r4
105           move    r2 , r5
106           andi    #%00001111 , r5
107           shli    #4 , r5
108           load    #7 , r6 , r5
109           add     r4 , r6 , r4
110           move    r4 , r6
111           andi    #%11110000 , r6
112           shrui   #4 , r5
113           or      r6 , r5 , r5
114           bra     #label27
115   label26 move    r2 , r4
116           andi    #%00001111 , r4 ;r4=x_bloc
117           shli    #4 , r4
118           or      r13 , r4 ,r4    ;r4=x_pixel
119           move    r2 , r5
120           andi    #%11110000 , r5 ;r5=y_bloc<<4
121           load    #7 , r6 , r5    ;r6=deplacement
122           add     r6 , r4 , r4    ;r4=new x_pixel
123           move    r4 , r6
124           shrui   #4 , r6         ;r6=new x_bloc
125           or      r6 , r5 , r5    ;r5=new no_bloc
126   label27 load    #6 , r7 , r2    ;r7=ancien tag du pixel(old no_bloc)
127           andi    #%01111111 , r7
128           store   #6 , r7 , r2    ;efface tag_objet
129           andi    #%00001111 , r4 ;r4=x_pixel%16
130           sub     r13 , r4 , r4
131           brz     #label8 , r4    ;si le pixel ne reste pas dans le meme PE:
132           load    #6 , r7 , r5    ;r7=ancien tag du pixel(new no_bloc)
133           ori     #%01000000 , r7
134           store   #6 , r7 , r5    ;positionne tag_envoi
135   label8  store   #5 , r1 , r5    ;ecrit le pixel dans dest
136           bra     #label9
137
138   label17 move    r12 , r6        ;r6=dir EST
139           ori     #%01100000 , r6 ;message DIFFUSE avec tag 011
140           wait_b4_send    r6
141           send    r6 , r6 , r6    ;envoie un message DIFFUSE au suivant
142           xor     r12 , r11 , r6  ;r6=dir OUEST
```

Jul 4 1997 14:16:04 **rotate.asm** Page 3

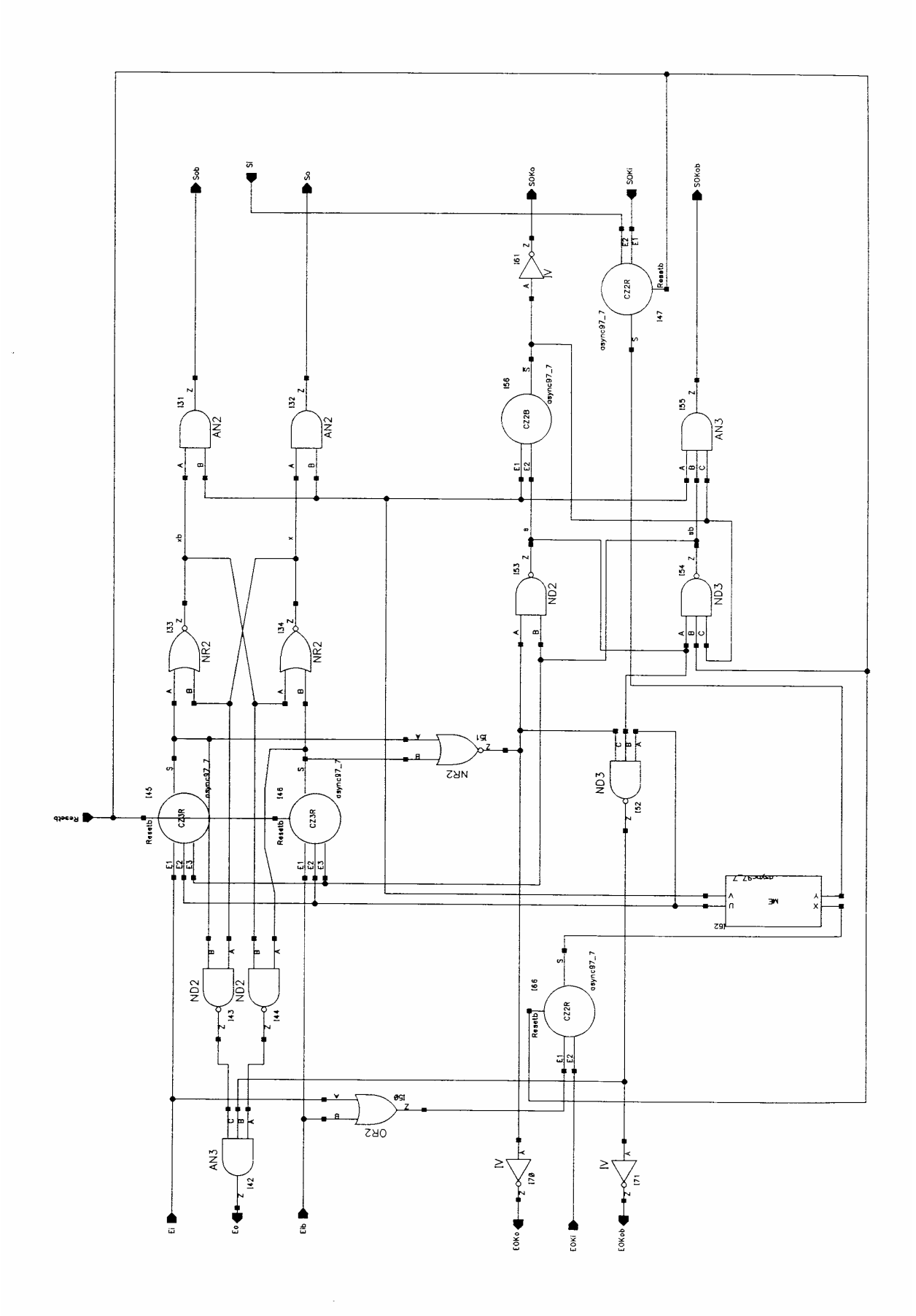
```
143             wait_b4_get     r6
144             get     r6 , r6 , r6    ;attend le retour du message DIFFUSE
145
146     label30 movei   #%00000000 , r0         ;copie dest vers source
147             movei   #%11111111 , r1
148             movei   #%00000001 , r2
149             search  #4 , r0 , r0    ;selectionne tout
150             write   #4 , r0 , r1    ;initialise tout a 0
151     label21 search  #5 , r1 , r2    ;selectionne les 1 de la colonne indiquee par r2
152             write   #4 , r1 , r2    ;recopie les 1 dans la meme colonne du plan 4
153             shli    #1 , r2         ;position suivante
154             brnz    #label21 , r2
155
156             subi    #1 , r9
157             move    r9 , r10
158             andi    #%00000011 , r10
159             brnz    #label20 , r10  ;passe suivante
160             bra     #label10
161
162     label7  brnz    #label10 , r13  ;saut si x_PE!=0
163     label15 movei   #1 , r0         ;sinon PE "maitre" de la ligne
164     label11 movei   #%01000000 , r4
165             search  #6 , r4 , r4    ;cherche un pixel a envoyer
166             count   r1 , r2
167             brnz    #label29 , r1
168             brz     #label12 , r2
169     label29 read    #5 , r1 , r2    ;r1=valeur pixel , r2=no_bloc
170             move    r12 , r4
171             ori     #%00100000 , r4 ;r4=dir DIFFUSE|EST
172             wait_b4_send    r4
173             send    r4 , r2 , r2    ;envoie no_bloc a toute la ligne de PE
174             xor     r12 , r11 , r4  ;r4=dir OUEST
175             wait_b4_get     r4
176             get     r4 , r4 , r4    ;attend le retour du message DIFFUSE
177
178     label14 movei   #0 , r4
179             store   #6 , r4 , r2    ;efface le tag du pixel
180             move    r2 , r5
181             brnz    #label24 , r10  ;si passe 2:
182             andi    #%00001111 , r5
183             shli    #4 , r5
184             bra     #label25
185     label24 andi    #%11110000 , r5
186     label25 ori     #%00000001 , r5 ;r5=(y_bloc<<4)+1
187             load    #7 , r4 , r5    ;r4=deplacement
188             move    r4 , r5
189             andi    #%00001111 , r4 ;r4=deplacement dans [0,8]
190             move    r11 , r7        ;r7=dir EST|OUEST
191             move    r12 , r6        ;r6=dir EST
192             andi    #%10000000 , r5
193             brz     #label13 , r5   ;saut si deplacement a droite
194             xor     r7 , r6 , r6    ;r6=dir OUEST
195     label13 move    r6 , r8
196             wait_b4_send    r8
197             send    r8 , r1 , r1    ;envoie pixel
198             move    r7 , r8
199             wait_b4_get     r8
200             get     r8 , r1 , r1    ;recoit pixel
201             subi    #1 , r4
202             brnz    #label13 , r4   ;boucle de deplacement
203             store   #5 , r1 , r2    ;stocke le pixel
204             brz     #label16 , r0   ;retour PE non maitre
205             bra     #label11        ;retour PE maitre
206
207     label12 move    r13 , r4
208             subi    #15 , r4
209             brz     #label17 , r4   ;fin de la passe si x_PE=15
210             move    r12 , r6        ;r6=dir EST
211             wait_b4_send    r6
212             send    r6 , r6 , r6    ;envoie un message!=DIFFUSE au suivant
213
```

Jul 4 1997 14:16:04 **rotate.asm** Page 4

```
214     label10 movei   #0 , r0         ;PE non maitre
215     label16 move    r11 , r6        ;r6=dir EST|OUEST
216             set_idle        #1
217             wait_b4_get     r6      ;attend un message
218             set_idle        #0
219             get     r6 , r2 , r2    ;r2=no_bloc si message DIFFUSE
220             move    r6 , r4
221             andi    #%01100000 , r4 ;message de type DIFFUSE ?
222             brz     #label15 , r4   ;si non, PE devient maitre de la ligne
223             wait_b4_send    r6      ;r6=dir DIFFUSE|dir opposee a la reception
224             send    r6 , r2 , r2    ;propage le message DIFFUSE
225             andi    #%01000000 , r4 ;changement de passe ?
226             brnz    #label30 , r4
227             load    #5 , r1 , r2    ;r1=pixel(no_bloc)
228             bra     #label14
229     .end
```



Visualisation du plan mémoire destination au cours de l'exécution de l'algorithme de rotation


Sob
Si
So
SOKo
SOKi
SOKob
I31 AN2
I32 AN2
I55 AN3
I56 C22B
async97_7
I61 IV
I47 C22R
Resetb
I53 ND2
I54 ND3
I33 NR2
I34 NR2
I51 NR2
I45 C23R
I46 C23R
I52 ND3
I62 ME
I43 ND2
I44 ND2
I66 C22R
I50 OR2
I42 AN3
I70 IV
I71 IV
Ei
Eo
Eib
EOKo
EOKi
EOKob