Appendix A
MATLAB Codes for Adaptive Resonance Theory Algorithms

In this Appendix we include some simple "*home-made*" MATLAB codes which help to illustrate and understand the Adaptive Resonance Theory algorithms described in Chapter 1. There are four programs (ART1, ARTMAP, Fuzzy-ART, and Fuzzy-ARTMAP) each of which applies one of the algorithms to a specific problem. The codes are included in this Appendix but can be retrieved also from `http://www.imse.cnm.es/~bernabe`

A.1 MATLAB CODE EXAMPLE FOR ART1

The *art1* routine generates a sequence of np input patterns of length $n1 \times n2$ and clusters them using the ρ and L values provided. From the MATLAB prompt this routine is called using the command

```
>> art1(n1,n2,rho,L,np)
```

The program then displays in a graphic window the present and previous status of *Input Pattern* (on the left side) and weight templates $\mathbf{z}_j$ (on the right side). Patterns are drawn as rectangular boxes of $n1 \times n2$ pixels. The weight template that has been chosen for update ($\mathbf{z}_J$) is drawn surrounded by a yellow line. For example, the command

```
>> art1(5,5,0.4,5,10)
```

would generate a similar sequence than the one shown in Fig. 1.3 of Chapter 1. The program stops after each pattern presentation and corresponding learning, and waits for the user to hit any key before presenting the next pattern. Input patterns are sequentially and iteratively provided until there is no weight update for a complete iteration of input patterns presentation. The program uses the auxiliary functions "*draw_status*" and "*art1plot*" given at the end of this Appendix. The MATLAB code for this program is:

File: "art1.m"

```
function art1(n1,n2,rho,L,np)

clf
n=n1*n2;
z=ones(n,1);
zold=z;
zprev=z;
Jold=1;
Mold=1;
M=1;
first=1;
I='';
for pattern=1:np
  I=[I,round(rand(n,1))];
end
iter=0;
learning=1;
while learning==1
  iter=iter+1;
  for pattern=1:np
    ok=0;
    for j=1:M
      T(j)=norm(min(I(:,pattern),z(:,j)),1)/(L-1+norm(z(:,j),1));
    end
    while ok==0
      [maxT,J]=max(T);
      if rho*norm(I(:,pattern),1) <= norm(min(I(:,pattern),z(:,J)),1)
        ok=1;
      else
        T(J) = -1;
      end
    end
    z(:,J)=min(I(:,pattern),z(:,J));
    if J==M
      M=M+1;
      z=[z ones(n,1)];
    end
    draw_status(M,Mold,pattern,np,J,Jold,I,z,zprev,n1,n2,first,iter)
    first=0;
    Jold=J;
    Mold=M;
    zprev=z;
    pause
  end
  if size(z)==size(zold)
    if z==zold
      learning=0;
    end
  end
  zold=z;
end
```

A.2 MATLAB CODE EXAMPLE FOR ARTMAP

This ARTMAP routine learns to recognize the fonts for letters 'A', 'B', and 'C' shown in Fig. 1.6 of Chapter 1. The system is first trained with these exemplars, then noisy versions of the input patterns are given, and the system classifies them. The program provides a graphic output showing the noisy test pattern and indicating whether it has been classified into *Class* 1 (letter 'A'), *Class* 2 (letter 'B'), *Class* 3, (letter 'C'), or *Class "Don't Know"*.

The program is called from the MATLAB prompt using the command

```
>> artmap(rhobar,L,noise)
```

where *rhobar* corresponds to parameter $\overline{\rho_a}$ discussed in Section 1.3, and noise is a number between '0' and '1' which controls how much the noisy input patterns are degraded for the test stage. If *noise* $= 0$ no degradation appears and if *noise* $= 1$ input patterns are pure noise.

The main program consists of the routines "*artmap*" and "*test_artmap*" whose code is given next, and the auxiliary functions "*initI*", "*initb*", "*degrade*", "*msq*", and "*art1plot*" given at the end of this Appendix.

File: "artmap.m"

```
function artmap(rhobar,L,noise)

epsilon=0.0001;
no=100;
n=200;
np=18;
Mb=3;
M=1;
Io=initI;
Ic=1-Io;
I=[Io;Ic];
b=initb;
z=ones(n,1);
w=ones(Mb,1);
zold=z;
wold=w;
learning=1;
iterations=0;
while learning==1
  iterations=iterations+1;
  for pattern=1:np
    rho=rhobar;
    ok=0;
    for j=1:M
      T(j)=norm(min(I(:,pattern),z(:,j)),1)/(L-1+norm(z(:,j),1));
    end
    while ok==0
      [maxT,J]=max(T);
      if rho*norm(I(:,pattern),1) <= norm(min(I(:,pattern),z(:,J)),1)
        [maxW,K]=max(b(:,pattern));
```

```
        if w(K,J) == 1
          ok=1;
        else
     rho=norm(min(I(:,pattern),z(:,J)),1)/norm(I(:,pattern),1)+epsilon;
          T(J) = -1;
          ok=0;
        end
      else
        T(J) = -1;
      end
    end
    z(:,J)=min(I(:,pattern),z(:,J));
    w(:,J)=min(b(:,pattern),w(:,J));
    if J==M
      M=M+1;
      z=[z ones(n,1)];
      w=[w ones(Mb,1)];
    end
  end
  if size(z)==size(zold)
    if z==zold
      if size(w)==size(wold)
        if w==wold
          learning=0;
        end
      end
    end
  end
  zold=z;
  wold=w;
end
iterations
categories=M
test_artmap(Io,z,w,rhobar,L,10,10,noise);
```

File: "test_artmap.m"

```
function test_artmap(Io,z,w,rhobar,L,n1,n2,noise)

Io=degrade(Io,noise);
Ic=1-Io;
I=[Io;Ic];
rho=rhobar;
[n np]=size(I);
[Mb M]=size(w);
  for pattern=1:np
    ok=0;
    for j=1:M
      T(j)=norm(min(I(:,pattern),z(:,j)),1)/(L-1+norm(z(:,j),1));
    end
    while ok==0
      [maxT,J]=max(T);
      if rho*norm(I(:,pattern),1) <= norm(min(I(:,pattern),z(:,J)),1)
        ok=1;
      else
        T(J) = -1;
      end
    end
    [maxW,K]=max(w(:,J));
    clf
```

```
    art1plot(msq(n1,n2,Io(:,pattern))');
    if J==M
      ss=sprintf('Pattern Number:  %d, Predicted Class is:  Do not
know',pattern);
      text(0,0,ss)
    else
      ss=sprintf('Pattern Number:  %d, Predicted Class is:
%d',pattern,K);
      text(0,0,ss)
    end
    pause
  end
```

A.3 MATLAB CODE EXAMPLE FOR FUZZY-ART

This program performs the example discussed in Fig. 1.14 of Chapter 1. It takes as input a set of points $(x, y) \in \mathcal{R}^2$ inside the unit square and clusters them into categories depending on the values given for parameters ρ and α. After training is completed the program shows in a graphic window the boxes for each resulting category, and plots the input points using a different character depending on the category it has been assigned to.

Before running this Fuzzy-ART program one must run the program "*genpun*" which generates random training points and saves them in a file 'points.mat'. Program "*genpun*" is called from the MATLAB prompt using

```
>> genpun(np,nc)
```

where *np* is the total number of random points to be generated, and *nc* is the number of centers around which the random points should be generated. For example, for the training set shown in Fig. 1.13 there were $np = 100$ total points (circles) around $nc = 4$ center points (crosses). Once the training set is generated, one can run the command

```
>> fart(rho,alpha)
```

The starting routine is "*fart*" which calls the main Fuzzy-ART routine "*fuzzy*". The code for these two routines is given next, while the auxiliary functions "*genpun*", "*aug*", and "*rectangle*" are given at the end of this Appendix.

File: "fart.m"

```
function fart(rho,alpha)

aa=fuzzy(rho,alpha);
load wij;
nc=size(wij);
ncat=nc(1)-1;
load points;
np=size(points);
```

```
npunt=np(1);
sym=sprintf('.ox sdv^<>ph');
ns=size(sym);
nsym=ns(2);
col=sprintf('ymcrgbwk');
clf
h=gcf;
set(h,'name','Fuzzy ART');
hold;
for cnp=1:1:npunt
  if aa(cnp) < 13
   plot(points(cnp,1),points(cnp,2),sym(aa(cnp)));
  else
   plot(points(cnp,1),points(cnp,2),'x');
  end
end
for cc=1:1:ncat
   rectang(wij(cc,1),wij(cc,3),1-wij(cc,2),1-wij(cc,4));
end
hold;
axis('equal');
```

File: "fuzzy.m"

```
function map=fuzzy(rho,alpha)

F1=4;
CAT=0;
wij = ones(1,F1);
wijold=1;
load points;
ns=size(points);
np=ns(1);
niter=1
while 1>0,
 for point=1:1:np
    a(1)=points(point,1);
    a(2)=1-points(point,1);
    a(3)=points(point,2);
    a(4)=1-points(point,2);
    for j=1:CAT+1
      T(j) = norm(min(a,wij(j,:)),1)/(alpha+norm(wij(j,:),1));
    end
    while 1>0,
          [Tmax,Jmax]=max(T);
        if norm(min(a,wij(Jmax,:)),1) >= rho*norm(a,1)
           map(point)=Jmax;
           break;
        end
        T(Jmax)=0;
    end
    wij(Jmax,:)=min(a,wij(Jmax,:));
    if Jmax==CAT+1
      CAT=CAT+1;
      wij = aug(wij,1);
   end
end
if size(wij)==size(wijold)
  if wij == wijold
```

```
            break;
      end
   end
   wijold=wij;
   ss=sprintf('niter=%d CAT=%d',niter,CAT)
   niter = niter+1;
   end
   save wij wij -ascii
   save map map -ascii
```

A.4 MATLAB CODE EXAMPLE FOR FUZZY-ARTMAP

This program executes the example discussed in Fig. 1.17 known as the problem "*Learning to Tell Two Spirals Apart*". It first generates the 194 training points of the two spirals using eqs. (1.52) and (1.53), and then trains the Fuzzy-ARTMAP system using the values for parameters $\overline{\rho_a}$ and α. Once training is complete the $\mathcal{R}^2$ unit square is partitioned into a $tnp \times tnp$ grid, and the center point of each square in this grid is classified as either belonging to the region represented by the first or the second spiral. The program provides a graphic output in which each square is assigned a color depending on the region it has been classified and draws on top the set of training points of the two spirals. The results shown in Fig. 1.17 were obtained using this program.

The program is run from the MATLAB prompt using the command

```
>> fartmap(tnp,rho,alpha)
```

Program "*fartmap*" is a starting routine which calls the main routine "*trainfam*". The code for these two routines is given next, while the code for the auxiliary functions "*spitrain*" and "*aug*" is given at the end of this Appendix.

File: "fartmap.m

```
function fartmap(tnp,rho,alpha)

figure(1)
aa=trainfam(tnp,rho,alpha);
xx=0:1/tnp:1;
yy=0:1/tnp:1;
clf
h=gcf;
axis([0 1 0 1]);
set(h,'colormap',cool);
set(h,'name','Fuzzy ARTMAP');
image(xx,yy,aa);
view(2);
d=1/(2*tnp);
axis([0-d 1+d 0-d 1+d]);
axis('equal');
axis('manual');
hold;
spiral=spitrain;
spi=size(spiral);
```

```
npi=spi(1);
for nsp=1:2:npi
 plot(spiral(nsp,1),spiral(nsp,2),'o')
 plot(spiral(nsp+1,1),spiral(nsp+1,2),'*')
end
axis off
hold;
```

File: "trainfam.m"

```
function map=trainfam(TNP,rho_init,alpha)

F1=4;
F2max=10000;
F2b=2;
CAT=0;
epsilon=0.01;
rho_ab=0.8;
wij = ones(1,F1);
wjk = ones(1,F2b);
wijold=1;
wjkold=1;
spiral=spitrain;
strain=size(spiral);
ntrain=strain(1);
niter=1
while 1>0,
   for nt=1:ntrain;
    a(1)=spiral(nt,1);
    a(2)=1-a(1);
    a(3)=spiral(nt,2);
    a(4)=1-a(3);
    b(1)=spiral(nt,3);
    b(2)=1-b(1);
    rhoa = rho_init;
    for j=1:CAT+1
      T(j) = norm(min(a,wij(j,:)),1)/(alpha+norm(wij(j,:),1));
    end
    while 1>0,
      while 1>0,
        [Tmax,Jmax]=max(T);
        if norm(min(a,wij(Jmax,:)),1) >= rhoa*norm(a,1)
          break;
        end
        T(Jmax)=0;
      end
      if norm(min(b,wjk(Jmax,:)),1) >= rho_ab
        break;
      end
      rhoa = norm(min(a,wij(Jmax,:)),1)/(F1/2)+epsilon;
      T(Jmax)=0;
    end
    wij(Jmax,:)=min(a,wij(Jmax,:));
    wjk(Jmax,:)=min(b,wjk(Jmax,:));
    if Jmax==CAT+1
      CAT=CAT+1;
      wij = aug(wij,1);
      wjk = aug(wjk,1);
```

```
      end
      rhoa=rho_init;
   end
   if size(wij)==size(wijold)
    if wij == wijold
      if size(wjk)==size(wjkold)
        if wjk == wjkold
          break;
        end
      end
    end
  end
  wijold=wij;
  wjkold=wjk;
  ss=sprintf('niter=%d CAT=%d',niter,CAT)
  niter = niter+1;
  end
  wij
  wjk
  CAT,niter
  for jy=1:TNP+1
    for ix=1:TNP+1
      a(1)=(ix-0.5)/TNP;
      a(2)=1-a(1);
      a(3)=(jy-0.5)/TNP;
      a(4)=1-a(3);
      for j=1:CAT
        T(j) = norm(min(a,wij(j,:)),1)/(alpha+norm(wij(j,:),1));
      end
      T(CAT+1)= -1;
      [Tmax,Jmax]=max(T);
      [bmax,Kmax]=max(wjk(Jmax,:));
      if Kmax==2
        map(jy,ix)=0;
      else
        map(jy,ix)=100;
      end
    end
  end
```

A.5 AUXILIARY FUNCTIONS

File: "draw_status.m"

```
function draw_status(M,Mold,pattern,np,J,Jold,I,z,zold,n1,n2,first,iter)

  clf
  subplot(2,M+2,M+2+1)
  axis('equal')
  axis off
  x='';
  for(i=1:n1)
    x=[x,I(n2*(i-1)+1:n2*i,pattern)];
  end
  art1plot(x);
  ss=sprintf('Iteration:  %d,  Pattern:  %d',iter,pattern);
  text(0,-0.5,ss)
  for(j=1:M)
```

```
    subplot(2,M+2,M+2+j+2)
    x='';
    for i=1:n1
      x=[x,z(n2*(i-1)+1:n2*i,j)];
    end
    art1plot(x);
    if j==J
      hold on
      plot([0.5,0.5],[0.5,n2+0.5],'y')
      plot([0.5,n1+0.5],[n2+0.5,n2+0.5],'y')
      plot([n1+0.5,n1+0.5],[n2+0.5,0.5],'y')
      plot([n1+0.5,0.5],[0.5,0.5],'y')
      hold off
    end
  end
  if first==1
    break;
  end
  pold=pattern-1;
  if pold==0
    pold=np;
  end
  subplot(2,M+2,1)
  axis('equal')
  axis off
  x='';
  for(i=1:n1)
    x=[x,I(n2*(i-1)+1:n2*i,pold)];
  end
  art1plot(x);
  for(j=1:Mold)
    subplot(2,M+2,j+2)
    x='';
    for i=1:n1
      x=[x,zold(n2*(i-1)+1:n2*i,j)];
    end
    art1plot(x);
    if j==Jold
      hold on
      plot([0.5,0.5],[0.5,n2+0.5],'y')
      plot([0.5,n1+0.5],[n2+0.5,n2+0.5],'y')
      plot([n1+0.5,n1+0.5],[n2+0.5,0.5],'y')
      plot([n1+0.5,0.5],[0.5,0.5],'y')
      hold off
    end
  end
```

File: "art1plot.m"

```
function art1plot(x)

[n,m]=size(x);
x=(1-x)*100;
image(x)
axis('off')
axis('equal')
axis([0.5 m+0.5 0.5 n+0.5])
colormap('gray')
```

File: "initI.m"

```
function I=initI

I='';
x=[0 0 0 0 0 1 0 0 0 0  0 0 0 0 1 1 1 0 0 0  0 0 0 0 1 0 1 0 0 0
   0 0 0 1 1 0 1 1 0 0  0 0 0 1 0 0 0 1 0 0  0 0 1 1 0 0 0 1 1 0
   0 0 1 0 0 0 0 0 1 0  0 1 1 1 1 1 1 1 1 1  0 1 0 0 0 0 0 0 0 1
   0 1 0 0 0 0 0 0 0 1];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 1 1  0 0 0 0 0 0 0 1 0 1  0 0 0 0 0 0 1 0 0 1
   0 0 0 0 0 1 0 0 0 1  0 0 0 0 1 0 0 0 0 1  0 0 0 1 0 0 0 0 0 1
   0 0 1 0 0 0 0 0 0 1  0 1 1 1 1 1 1 1 1 1  0 1 0 0 0 0 0 0 0 1
   1 1 0 0 0 0 0 0 0 1];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0
   0 0 0 0 1 1 0 1 0 0  0 0 0 1 0 0 1 1 0 0  0 0 1 0 0 0 0 1 0 0
   0 0 1 0 0 0 0 1 0 0  0 0 1 0 0 0 0 1 0 0  0 0 0 1 0 0 1 1 0 0
   0 0 0 0 1 1 0 1 1 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 0 0 0 0
   0 0 0 1 0 0 1 0 0 0  0 0 0 0 0 0 0 1 0 0  0 0 0 0 1 1 0 1 0 0
   0 0 0 1 0 0 1 1 0 0  0 0 1 0 0 0 0 1 0 0  0 0 0 1 0 0 1 1 0 0
   0 0 0 0 1 1 0 1 1 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0.0 0 0 0 0 0
   0 0 0 1 1 0 0 0 0 1  0 0 1 0 0 1 0 0 1 0  0 1 0 0 0 0 1 1 0 0
   0 1 0 0 0 0 1 0 0 0  0 1 0 0 0 0 1 1 0 0  0 0 1 0 0 1 0 0 1 0
   0 0 0 1 1 0 0 0 0 1];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 1  0 0 0 0 0 0 0 0 1 1  0 0 0 0 0 0 0 1 1 1
   0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 1 1 1 1 1  0 0 0 0 1 1 1 1 1 1
   0 0 0 1 1 1 1 1 1 1  0 0 1 1 1 1 1 1 1 1  0 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1];
I=[I,x'];
x=[0 1 1 1 1 1 1 1 0 0  0 0 1 0 0 0 0 0 1 0  0 0 1 0 0 0 0 0 1 0
   0 0 1 0 0 0 0 0 1 0  0 0 1 1 1 1 1 1 0 0  0 0 1 0 0 0 0 0 1 0
   0 0 1 0 0 0 0 0 0 1  0 0 1 0 0 0 0 0 0 1  0 0 1 0 0 0 0 0 1 0
   0 1 1 1 1 1 1 1 0 0];
I=[I,x'];
x=[0 0 0 1 1 1 1 1 0 0  0 0 0 0 1 0 0 0 1 0  0 0 0 0 1 0 0 0 1 0
   0 0 0 1 0 0 0 0 1 0  0 0 0 1 1 1 1 1 0 0  0 0 1 0 0 0 0 0 1 0
   0 0 1 0 0 0 0 0 0 1  0 1 0 0 0 0 0 0 0 1  0 1 0 0 0 0 0 0 1 0
   1 1 1 1 1 1 1 1 0 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 1 0 0 0 0 0 0 0  0 0 1 0 0 0 0 0 0 0
   0 0 1 0 0 0 0 0 0 0  0 0 1 0 1 1 0 0 0 0  0 0 1 1 0 0 1 0 0 0
   0 0 1 0 0 0 0 1 0 0  0 0 1 0 0 0 0 1 0 0  0 0 1 1 0 0 1 0 0 0
   0 0 1 0 1 1 0 0 0 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0
   0 0 0 1 0 0 0 0 0 0  0 0 1 1 1 1 0 0 0 0  0 0 1 0 0 0 1 0 0 0
   0 0 1 0 0 0 0 1 0 0  0 1 1 0 0 0 0 1 0 0  0 1 0 1 0 0 1 0 0 0
   0 1 0 0 1 1 0 0 0 0];
I=[I,x'];
x=[0 0 0 1 1 0 0 0 0 0  0 0 1 0 0 1 0 0 0 0  0 1 0 0 0 0 1 0 0 0
   0 1 0 0 0 0 1 0 0 0  0 1 0 1 1 1 0 0 0 0  0 1 0 0 0 0 1 0 0 0
   0 1 0 0 0 0 1 0 0 0  0 1 1 0 0 1 0 0 0 0  0 1 0 1 1 0 0 0 0 0
   0 1 0 0 0 0 0 0 0 0];
I=[I,x'];
x=[1 1 1 1 0 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0
```

```
   1 1 1 1 0 0 0 0 0 0  1 1 1 1 1 1 0 0 0 0  1 1 1 1 1 1 1 1 1 0
   1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 0  1 1 1 1 1 1 0 0 0 0
   1 1 1 0 0 0 0 0 0 0];
I=[I,x'];
x=[0 0 0 0 1 1 1 1 0 1  0 0 0 1 0 0 0 0 1 1  0 0 1 0 0 0 0 0 0 1
   0 1 0 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0
   0 1 0 0 0 0 0 0 0 0  0 0 1 0 0 0 0 0 0 1  0 0 0 1 0 0 0 0 1 1
   0 0 0 0 1 1 1 1 0 1];
I=[I,x'];
x=[0 0 0 0 1 1 1 0 0 1  0 0 0 1 0 0 0 1 1 1  0 0 0 1 0 0 0 0 0 1
   0 0 1 0 0 0 0 0 0 0  0 0 1 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 0
   0 1 0 0 0 0 0 0 0 0  0 1 0 0 0 0 0 0 0 1  0 1 1 0 0 0 0 1 1 1
   0 0 0 1 1 1 1 0 0 1];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 0 0 0  0 0 0 0 1 0 0 1 0 0
   0 0 0 1 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0  0 0 0 0 1 0 0 1 0 0
   0 0 0 0 0 1 1 0 0 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 0 0 0  0 0 0 1 0 0 0 1 0 0
   0 0 0 1 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0  0 0 0 1 0 0 0 1 0 0
   0 0 0 0 1 1 1 0 0 0];
I=[I,x'];
x=[0 0 0 1 0 0 1 0 0 0  0 0 0 0 1 1 0 0 0 0  0 0 0 0 1 1 0 0 0 0
   0 0 0 1 0 0 1 0 0 0  0 0 0 1 0 0 1 0 0 0  0 0 1 0 0 0 0 1 0 0
   0 0 1 0 0 0 0 1 0 0  0 0 0 1 0 0 1 0 0 0  0 0 0 1 0 0 1 0 0 0
   0 0 0 0 1 1 0 0 0 0];
I=[I,x'];
x=[0 0 0 0 0 0 0 1 1 1  0 0 0 0 0 1 1 1 1 1  0 0 0 0 1 1 1 1 1 1
   0 0 0 1 1 1 1 1 1 1  0 0 1 1 1 1 1 1 1 1  0 0 1 1 1 1 1 1 1 1
   0 0 0 1 1 1 1 1 1 1  0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1
   0 0 0 0 0 0 0 1 1 1];
I=[I,x'];
```

File: "initb.m"

```
function b=initb

b=[1 0 0;1 0 0;1 0 0;1 0 0;1 0 0;1 0 0;
   0 1 0;0 1 0;0 1 0;0 1 0;0 1 0;0 1 0;
   0 0 1;0 0 1;0 0 1;0 0 1;0 0 1;0 0 1];
b=b';
```

File: "degrade.m"

```
function a=degrade(I,noise)

if noise<0
  Error('Noise must be between 0 and 1')
end
if noise>1
  Error('Noise must be between 0 and 1')
end
[n np]=size(I);
for i=1:n
```

```
  for j=1:np
    if rand(1,1)<noise
        a(i,j)=round(rand(1,1));
    else
      a(i,j)=I(i,j);
    end
  end
end
```

File: "msq.m"

```
function x=msq(n1,n2,I)

  if size(I)  = n1*n2
    Error('Error in "msq.m"');
  end
  x='';
  for i=1:n1
    x=[x,I((i-1)*n2+1:i*n2,1)];
  end
```

File: "genpun.m"

```
function genpun(np,nc)

clusx=rand(nc,1);
clusy=rand(nc,1);
figure(1)
for point=1:1:np
   ac=rand(1);
   dcx=0.1*randn(1);
   dcy=0.1*randn(1);
   for cc=1:1:nc
      if ac< cc/nc
         if ac>(cc-1)/nc
            punt(point,1)=clusx(cc,1)+dcx;
            punt(point,2)=clusy(cc,1)+dcy;
         end
      end
   end
end
ma=max(max(punt));
mi=min(min(punt));
punt=(punt-mi)/(ma-mi);
clusx=(clusx-mi)/(ma-mi);
clusy=(clusy-mi)/(ma-mi);
plot(clusx,clusy,'+')
save points punt -ascii
hold
plot(punt(:,1),punt(:,2),'o')
hold
```

File: "aug.m"

```
function aa=aug(a,n)

[lines cols]=size(a);
u=ones(1,cols);
for i=1:n
  a = [a;u];
end
aa = a;
```

File: "rectang.m"

```
function rectang(x1,y1,x2,y2)

line([x1,x2],[y1,y1]);
line([x2,x2],[y1,y2]);
line([x2,x1],[y2,y2]);
line([x1,x1],[y2,y1]);
```

File: "spitrain.m"

```
function spiral=spitrain

for n=1:97
   alpha(n)=pi*(n-1)/16;
   r(n)=0.4*((105-n)/104);
   a(2*n-1,1)=r(n)*sin(alpha(n))+0.5;
   a(2*n-1,2)=r(n)*cos(alpha(n))+0.5;
   b(2*n-1,1)=1;
   a(2*n,1)=1-a(2*n-1,1);
   a(2*n,2)=1-a(2*n-1,2);
   b(2*n,1)=0;
   spiral=[a,b];
end
```

Appendix B
Computational Equivalence of the Original ART1 and the Modified ART1m Models

In the original ART1 paper [Carpenter, 1987] the architecture is mathematically described as sets of Short Term Memory (STM) and Long Term Memory (LTM) time domain nonlinear differential equations. The STM differential equations describe the evolution and interactions between processing units or neurons of the system, while the LTM differential equations describe how the interconnection weights change in time as a function of the state of the system. The time constants associated to the LTM differential equations are much slower than those associated to the STM differential equations. A valid assumption is to make the STM differential equations settle instantaneously to their corresponding steady state, and consider only the dynamics of the LTM differential equations. In this case, the STM differential equations must be substituted by nonlinear algebraic equations that describe the corresponding steady state of the system. Furthermore, Carpenter and Grossberg also introduced the *Fast Learning* mode of the ART1 architecture, in which the LTM differential equations are also substituted by their corresponding steady-state nonlinear algebraic equations. Thus, the ART1 architecture originally modeled as a dynamically evolving collection of neurons and synapses governed by time-domain differential equations, can be behaviorally modeled as the sequential application of nonlinear algebraic equations: an input pattern is given, the corresponding STM steady state is computed through the STM algebraic equations, and the system weights are updated using the corresponding LTM algebraic equations. At this point three different levels of ART1 implementations (in both software or hardware) can be distinguished:

- **Type-1, Full Model Implementation:** Both STM and LTM time-domain differential equations are realized. This implementation is the most expensive, and requires a large amount of computational power.

- **Type-2, STM Steady-State Implementation:** Only the LTM time-domain differential equations are implemented. The STM behavior is governed by nonlinear algebraic equations. This implementation requires less resources than the previous one. However, proper sequencing of STM events must be introduced artificially, which is architecturally implicit in the Type-1 implementation.

- **Type-3, Fast Learning Implementation:** This implementation is computationally the least expensive. In this case, STM and LTM events must be artificially sequenced.

Throughout the original ART1 paper [Carpenter, 1987], Carpenter and Grossberg provide rigorous demonstrations of the computational properties of the ART1 architecture. Some of these properties are concerned with *Type*-1 and *Type*-2 operations of the architecture, but most refer to the *Type*-3 model operation. From a functional point of view, i.e., when looking at the ART1 system as a black box regardless of the details of its internal operations, the system level computational properties of ART1 are fully contained in its *Fast-Learning* or *Type*-3 model. The theorems and demonstrations given by Carpenter and Grossberg [Carpenter, 1987] relating to *Type*-1 and *Type*-2 models of the system only ensure proper *Type*-3 behavior. The purpose of this Appendix is to demonstrate that the modified *Type*-3 model (ART1m) developed in Chapter 2 preserves all the *Type*-3 computational properties of the original ART1 architecture. The only functional difference between ART1 and ART1m, is the way the terms T_j are computed before competing in the Winner-Takes-All block. Therefore, the original properties and demonstrations that are not affected by the terms T_j will be automatically preserved. Such properties are, for example, the *Self-Scaling* property and the *Variable Coarseness* property tuned by the *Vigilance Parameter.* But there are other properties which are directly affected by the way the terms T_j are computed. In the remainder of this Appendix we will show that these properties remain in the ART1m architecture.

Let us define a few concepts before demonstrating that the original computational properties are preserved.

1. *Direct Access*: an input pattern $\mathbf{I}$ is said to have *Direct Access* to a learned category j if this category is the first one selected by the Winner-Takes-All *F2* layer and is accepted by the *vigilance subsystem*, so that no reset occurs.

2. *Subset Template*: an input pattern $\mathbf{I}$ is said to be a *Subset Template* of a learned category $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Nj})$ if $\mathbf{I} \subset \mathbf{z}_j$. Formally,

$$\begin{aligned} z_{ij} = 0 \Rightarrow I_i = 0 \quad \forall i = 1, \ldots, N \\ I_i = 1 \Rightarrow z_{ij} = 1 \quad \forall i = 1, \ldots, N \end{aligned} \tag{B.1}$$

there might be values of i such that $I_i = 0$ and $z_{ij} = 1$

3. *Superset Template*: an input pattern $\mathbf{I}$ is said to be a *Superset Template* of a learned category j if $\mathbf{z}_j \subset \mathbf{I}$.

4. *Mixed Template*: $\mathbf{z}_j$ and $\mathbf{I}$ are said to be mixed templates if neither $\mathbf{I} \subset \mathbf{z}_j$ nor $\mathbf{z}_j \subset \mathbf{I}$ are satisfied, and $\mathbf{I} \neq \mathbf{z}_j$.

5. *Uncommitted node*: an *F2* node j is said to be uncommitted if all its weights $z_{ij}(i = 1, \ldots, N)$ preserve their initial value ($z_{ij} = 1$), i.e., node j has not yet been selected to represent any learned category.

B.1 DIRECT ACCESS TO SUBSET AND SUPERSET PATTERNS

Suppose that a learning process has produced a set of categories in the *F2* layer. Each category j is characterized by the set of weights that connect node j in the *F2* layer to all nodes in the *F1* layer, i.e., $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Nj})$. Suppose that two of these categories, j_1 and j_2, are such that $\mathbf{z}_{j_1} \subset \mathbf{z}_{j_2}$ ($\mathbf{z}_{j_1}$ is a subset template of $\mathbf{z}_{j_2}$). Now consider two input patterns $\mathbf{I}^{(1)}$ and $\mathbf{I}^{(2)}$ such that,

$$\begin{aligned} \mathbf{I}^{(1)} &= \mathbf{z}_{j_1} \equiv (z_{1j_1}, z_{2j_1}, \ldots, z_{Nj_1}), \\ \mathbf{I}^{(2)} &= \mathbf{z}_{j_2} \equiv (z_{1j_2}, z_{2j_2}, \ldots, z_{Nj_2}). \end{aligned} \tag{B.2}$$

The *Direct Access to Subset and Superset* property assures that input $\mathbf{I}^{(1)}$ will have *Direct Access* to category j_1 and that input $\mathbf{I}^{(2)}$ will have *Direct Access* to category j_2. The proofs for this are as follows.

Original ART1:

Let us compute the values of T_{j_1} and T_{j_2} when the input patterns $\mathbf{I}^{(1)}$ and $\mathbf{I}^{(2)}$ are presented at the input of the system. For pattern $\mathbf{I}^{(1)}$ we will have,

$$\begin{aligned} T_{j_1} &= \frac{L|\mathbf{I}^{(1)} \cap \mathbf{z}_{j_1}|}{L - 1 + |\mathbf{z}_{j_1}|} = \frac{L|\mathbf{I}^{(1)}|}{L - 1 + |\mathbf{I}^{(1)}|} \\ T_{j_2} &= \frac{L|\mathbf{I}^{(1)} \cap \mathbf{z}_{j_2}|}{L - 1 + |\mathbf{z}_{j_2}|} = \frac{L|\mathbf{I}^{(1)}|}{L - 1 + |\mathbf{I}^{(2)}|} \end{aligned} \tag{B.3}$$

Since $|\mathbf{I}^{(1)}| < |\mathbf{I}^{(2)}|$, it is obvious that $T_{j_1} > T_{j_2}$ (remember that $L > 1$) and therefore category j_1 will become the active one. On the other hand, if input pattern $\mathbf{I}^{(2)}$ is presented at the input,

$$T_{j_1} = \frac{L|\mathbf{I}^{(2)} \cap \mathbf{z}_{j_1}|}{L-1+|\mathbf{z}_{j_1}|} = \frac{L|\mathbf{I}^{(1)}|}{L-1+|\mathbf{I}^{(1)}|}$$
$$T_{j_2} = \frac{L|\mathbf{I}^{(2)} \cap \mathbf{z}_{j_2}|}{L-1+|\mathbf{z}_{j_2}|} = \frac{L|\mathbf{I}^{(2)}|}{L-1+|\mathbf{I}^{(2)}|} \tag{B.4}$$

Since the function $Lx/(L-1+x)$ is an increasing function of x, it results that now $T_{j_2} > T_{j_1}$ and category j_2 will be chosen as the winner.

Modified ART1:

If pattern $\mathbf{I}^{(1)}$ is given as the input pattern we will have

$$\begin{aligned} T_{j_1} &= L_A|\mathbf{I}^{(1)} \cap \mathbf{z}_{j_1}| - L_B|\mathbf{z}_{j_1}| + L_M = L_A|\mathbf{I}^{(1)}| - L_B|\mathbf{I}^{(1)}| + L_M \\ T_{j_2} &= L_A|\mathbf{I}^{(1)} \cap \mathbf{z}_{j_2}| - L_B|\mathbf{z}_{j_2}| + L_M = L_A|\mathbf{I}^{(1)}| - L_B|\mathbf{I}^{(2)}| + L_M \end{aligned} \tag{B.5}$$

Since $|\mathbf{I}^{(1)}| < |\mathbf{I}^{(2)}|$, it follows that (remember that $L_B > 0$) $T_{j_1} > T_{j_2}$. In the case pattern $\mathbf{I}^{(2)}$ is presented at the input of the network it would be,

$$\begin{aligned} T_{j_1} &= L_A|\mathbf{I}^{(2)} \cap \mathbf{z}_{j_1}| - L_B|\mathbf{z}_{j_1}| + L_M = L_A|\mathbf{I}^{(1)}| - L_B|\mathbf{I}^{(1)}| + L_M \\ T_{j_2} &= L_A|\mathbf{I}^{(2)} \cap \mathbf{z}_{j_2}| - L_B|\mathbf{z}_{j_2}| + L_M = L_A|\mathbf{I}^{(2)}| - L_B|\mathbf{I}^{(2)}| + L_M \end{aligned} \tag{B.6}$$

In order to guarantee that $T_{j_2} > T_{j_1}$ the condition

$$L_A > L_B \tag{B.7}$$

has to be assured.

B.2 DIRECT ACCESS BY PERFECTLY LEARNED PATTERNS (THEOREM 1 OF ORIGINAL ART1)

This theorem, adapted to a *Type*-3 implementation, states the following

> An input pattern $\mathbf{I}$ has direct access to a node J which has perfectly learned the input pattern $\mathbf{I}$.

The proofs are as follows.

Original ART1:

In order to prove that $\mathbf{I}$ has direct access to J, we need to demonstrate that the following properties hold: (i) J is the first node to be chosen, (ii) J is accepted by the vigilance subsystem and (iii) J remains active as learning takes place. To prove property (i) we have to show that, at the start of each trial $T_J > T_j \; \forall j \neq J$. Since $\mathbf{I} = \mathbf{z}_J$,

$$T_J = \frac{L|\mathbf{I}|}{L - 1 + |\mathbf{I}|} \tag{B.8}$$

and

$$T_j = \frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{z}_j|} \tag{B.9}$$

Since $\frac{Lw}{L-1+w}$ is an increasing function of w (because $L > 1$) and $|\mathbf{I}|, |\mathbf{z}_j| > |\mathbf{I} \cap \mathbf{z}_j|$, we can state,

$$T_J = \frac{L|\mathbf{I}|}{L - 1 + |\mathbf{I}|} > \frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{I} \cap \mathbf{z}_j|} > \frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{z}_j|} = T_j. \tag{B.10}$$

So, property (i) is always fulfilled.

Property (ii) is directly verified since $|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{I}| \geq \rho|\mathbf{I}| \; \forall \rho \in [0, 1]$. Property (iii) is always verified because after node J is selected as the winning category, its weight template $\mathbf{z}_J$ will remain unchanged (because $\mathbf{z}_J|_{new} = \mathbf{I} \cap \mathbf{z}_J|_{old} = \mathbf{I} = \mathbf{z}_J|_{old}$), and consequently the inputs to the *F2* layer T_j will remain unchanged.

Modified ART1:

In order to demonstrate that $\mathbf{I}$ has direct access to J, we have only to prove that property (i) is verified for the modified algorithm, as the proof of properties (ii) and (iii) is identical to the case of the original algorithm. To prove property (i), we have to demonstrate that

$$T_J = L_A|\mathbf{I}| - L_B|\mathbf{I}| + L_M > L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M = T_j \tag{B.11}$$

Since $L_A w - L_B w + L_M$ is an increasing function of w ($L_A > L_B$), and $|\mathbf{I}|, |\mathbf{z}_j| > |\mathbf{I} \cap \mathbf{z}_j|$,

$$\begin{aligned} T_J &= L_A|\mathbf{I}| - L_B|\mathbf{I}| + L_M > L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{I} \cap \mathbf{z}_j| + L_M > \\ &> L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M = T_j \end{aligned} \quad (B.12)$$

B.3 STABLE CHOICES IN STM (THEOREM 2 OF ORIGINAL ART1)

Whenever an input pattern **I** is presented for the first time to the ART1 system, a set of T_j values is formed that compete in the Winner-Takes-All *F2* layer. The winner may be reset by the *vigilance subsystem*, and a new winner appears that may also be reset, and so on until a final winner is accepted. During this search process, the T_j values that led to earlier winners are set to zero. Let us call O_j the values of T_j at the beginning of the search process, i.e., before any of them is set to zero by the vigilance subsystem. Theorem 2 of the original ART1 architecture states:

> Suppose that an *F2* node J is chosen for STM storage instead of another node j because $O_J > O_j$. Then read-out of the top-down template preserves the inequality $T_J > T_j$ and thus confirms the choice of J by the bottom-up filter.

This theorem has only sense for a *Type*-1 implementation, because there, as a node in the *F2* layer activates, the initial values of T_j (immediately after presenting an input pattern I) may be altered through the top-down *'feed-back'* connections. In a *Type*-3 description (see Fig. 2.1) the initial terms T_j remain unchanged, independently of what happens in the *F2* layer. Therefore, this theorem is implicitly satisfied.

B.4 INITIAL FILTER VALUES DETERMINE SEARCH ORDER (THEOREM 3 OF ORIGINAL ART1)

Theorem 3 of the original ART1 architecture states that (page 92 of [Carpenter, 1987]):

> The Order Function ($O_{j_1} > O_{j_2} > O_{j_3} > \dots$) determines the order of search no matter how many times *F2* is reset during a trial.

The proof is the same for the original ART1 and the modified ART1 (both *Type*-3) implementation[1]. If T_{j_1} is reset by the *vigilance subsystem*, the values of $T_{j_2}, T_{j_3}, \dots$ will not change. Therefore, the new order sequence is $O_{j_2} > O_{j_3} > \dots$ and the original second largest value O_{j_2} will be selected as the winner. If T_{j_2} is now set to zero, O_{j_3} is the next winner, and so on.

[1] However, note that the resulting ordering $\{j_1, j_2, j_3, \dots\}$ can be different for the original and for the modified architecture

This Theorem, although trivial in a *Type*-3 implementation, has more importance in a *Type*-1 description where the process of selecting and shutting down a winner has the consequence of altering T_j values.

B.5 LEARNING ON A SINGLE TRIAL (THEOREM 4 OF ORIGINAL ART1)

This theorem (page 93 of [Carpenter, 1987]) states the following, assuming a *Type*-3 implementation is being considered[2]:

> Suppose that an *F2* winning node J is accepted by the vigilance subsystem. Then the LTM traces z_{ij} change in such a way that T_J increases and all other T_j remain constant, thereby confirming the choice of J. In addition, the set $\mathbf{I} \cap \mathbf{z}_J$ remains constant during learning, so that learning does not trigger reset of J by the vigilance subsystem.

The proofs are as follows.

Original ART1:

According to eq. (2.3), if J is the winning category accepted by the vigilance subsystem, we have that

$$T_J = \frac{L|\mathbf{I} \cap \mathbf{z}_J|}{L - 1 + |\mathbf{z}_J|} \tag{B.13}$$

This is the T_J value before learning takes place. After updating the weights (see Fig. 2.1(b)),

$$\mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{z}_J(old) \tag{B.14}$$

and the new T_J value is given by,

$$\begin{aligned} T_J(new) &= \frac{L|\mathbf{I} \cap \mathbf{z}_J(new)|}{L - 1 + |\mathbf{z}_J(new)|} = \frac{L|\mathbf{I} \cap \mathbf{I} \cap \mathbf{z}_J(old)|}{L - 1 + |\mathbf{I} \cap \mathbf{z}_J(old)|} \geq \\ &\geq \frac{L|\mathbf{I} \cap \mathbf{z}_J(old)|}{L - 1 + |\mathbf{z}_J(old)|} = T_J(old) \end{aligned} \tag{B.15}$$

[2] In the original ART1 paper[Carpenter, 1987] a more sophisticated demonstration for this theorem is provided. The reason is that there the demonstration is performed for a *Type*-1 description of ART1.

Note that by eq. (B.14),

$$\mathbf{I} \cap \mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{I} \cap \mathbf{z}_J(old) = \mathbf{I} \cap \mathbf{z}_J(old) \tag{B.16}$$

Since the only weights that are updated are those connected to the winning (and accepted) node J, all other $T_j|_{j \neq J}$ values remain unchanged. Therefore, it can be concluded, by eq. (B.15), that learning confirms the choice of J and that, by eq. (B.16), the set $\mathbf{I} \cap \mathbf{z}_J$ remains constant.

Modified ART1:

In this case, if J is the winning category accepted by the *vigilance subsystem*, by eq. (2.4) we have that

$$T_J = L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M \tag{B.17}$$

The update rule is the same as before (see Fig. 2.1(b)), therefore

$$\mathbf{z}_J(new) = \mathbf{I} \cap \mathbf{z}_J(old) \tag{B.18}$$

and the new T_J value is given now by,

$$\begin{aligned} T_J(new) &= L_A|\mathbf{I} \cap \mathbf{z}_J(old)| - L_B|\mathbf{I} \cap \mathbf{z}_J(old)| + L_M \geq \\ &\geq L_A|\mathbf{I} \cap \mathbf{z}_J(old)| - L_B|\mathbf{z}_J(old)| + L_M = T_J(old) \end{aligned} \tag{B.19}$$

Like before, learning confirms the choice of J, and by eq. (B.18) the set $\mathbf{I} \cap \mathbf{z}_J$ remains constant as well.

B.6 STABLE CATEGORY LEARNING (THEOREM 5 OF ORIGINAL ART1)

Suppose an arbitrary list (finite or infinite) of binary input patterns is presented to an ART1 system. Each template set $\mathbf{z}_j \equiv (z_{1j}, z_{2j}, \ldots, z_{Nj})$ is updated every time category j is selected by the Winner-Takes-All *F2* layer and accepted by the vigilance subsystem. Some of these times template $\mathbf{z}_j$ might be changed, and some others it might stay unchanged. Let us call the times $\mathbf{z}_j$ suffers a change $t_1^{(j)} < t_2^{(j)} < \cdots < t_{r_j}^{(j)}$. Since vector (or template) $\mathbf{z}_j$ has N components (initially set to '1'), and by eq. (B.14), each component can only change from

'1' to '0' but not from '0' to '1', it follows that template $\mathbf{z}_j$ can, at the most, suffer $N-1$ changes[3],

$$r_j \leq N \tag{B.20}$$

Since template $\mathbf{z}_j$ will remain unchanged after time $t_{r_j}^{(j)}$, it is concluded that the complete LTM memory will suffer no change after time

$$t_{learn} = \max_j \{t_{r_j}^{(j)}\} \tag{B.21}$$

If there is a finite number of nodes in the *F2* layer t_{learn} has a finite value, and thus learning completes after a finite number of time steps.

All this is true for both, the original and the modified ART1 architecture, and therefore the following theorem (page 95 of [Carpenter, 1987]) is valid for the two algorithms:

> In response to an arbitrary list of binary input patterns, all LTM traces $z_{ij}(t)$ approach limits after a finite number of learning trials. Each template set $\mathbf{z}_j$ remains constant except for at most N times $t_1^{(j)} < t_2^{(j)} < \cdots < t_{r_j}^{(j)}$ at which it progressively loses elements, leading to the
>
> $$\text{Subset Recoding Property: } \mathbf{z}_j(t_1^{(j)}) \supset \mathbf{z}_j(t_2^{(j)}) \supset \cdots \supset \mathbf{z}_j(t_{r_j}^{(j)}). \tag{B.22}$$
>
> The LTM traces $z_{ij}(t)$ such that $i \notin \mathbf{z}_j(t_{r_j}^{(j)})$ decrease to zero. The LTM traces $z_{ij}(t)$ such that $i \in \mathbf{z}_j(t_{r_j}^{(j)})$ remain always at '1'. The LTM traces such that $i \in \mathbf{z}_j(t_k^{(j)})$ but $i \notin \mathbf{z}_j(t_{k+1}^{(j)})$ stay at '1' for times $t \leq t_k^{(j)}$ but will change to and stay at '0' for times $t \geq t_{k+1}^{(j)}$.

B.7 DIRECT ACCESS AFTER LEARNING SELF-STABILIZES (THEOREM 6 OF ORIGINAL ART1)

Assuming *F2* has a finite number of nodes, the present theorem (page 98 of [Carpenter, 1987]) states the following:

> After recognition learning has self-stabilized in response to an arbitrary list of binary input patterns, each input pattern **I** either has direct access to the node j

[3]As mentioned at the end of Chapter 2, the empty template is not valid for the original ART1 system. Therefore, eq. (B.20) can be changed to $r_j \leq N-1$ for ART1, but not for ART1m.

which possesses the largest subset template with respect to $\mathbf{I}$, or $\mathbf{I}$ cannot be coded by any node of *F2*. In the latter case, *F2* contains no uncommitted nodes.

Since learning has already stabilized $\mathbf{I}$ can be coded only by a node j whose template $\mathbf{z}_j$ is a subset template with respect to $\mathbf{I}$. Otherwise, after j becomes active, the set $\mathbf{z}_j$ would contract to $\mathbf{z}_j \cap \mathbf{I}$, thereby contradicting the hypothesis that learning has already stabilized. Thus if $\mathbf{I}$ activates any node other than one with a subset template, that node must be reset by the *vigilance subsystem*. For the remainder of the proof, let J be the first *F2* node activated by $\mathbf{I}$. We need to show that if $\mathbf{z}_J$ is a subset template, then it is the subset template with the largest O_J; and if it is not a subset template, then all subset templates activated on that trial will be reset by the vigilance subsystem. To proof these two steps we need to differentiate between the original ART1 and the modified one.

Original ART1:

If J and j are nodes with subset templates with respect to $\mathbf{I}$, then

$$O_j = \frac{L|\mathbf{z}_j|}{L-1+|\mathbf{z}_j|} < O_J = \frac{L|\mathbf{z}_J|}{L-1+|\mathbf{z}_J|} \tag{B.23}$$

Since $\frac{L|\mathbf{z}_j|}{L-1+|\mathbf{z}_j|}$ is an increasing function of $|\mathbf{z}_j|$,

$$|\mathbf{z}_j| < |\mathbf{z}_J| \tag{B.24}$$

and,

$$R_j = \frac{|\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{I}|} = \frac{|\mathbf{z}_j|}{|\mathbf{I}|} < R_J = \frac{|\mathbf{I} \cap \mathbf{z}_J|}{|\mathbf{I}|} = \frac{|\mathbf{z}_J|}{|\mathbf{I}|} \tag{B.25}$$

Once activated, a node k will be reset if $R_k < \rho$. Therefore, if J is reset ($R_J < \rho$), then all other nodes with subset templates will be reset as well ($R_j < \rho$).

Now suppose that J, the first activated node, does not have a subset template with respect to $\mathbf{I}$ ($|\mathbf{I} \cap \mathbf{z}_J| < |\mathbf{z}_J|$), but that another node j with a subset template is activated in the course of search. We need to show that $|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}|$, so that j is reset. We know that,

$$O_j = \frac{L|\mathbf{z}_j|}{L-1+|\mathbf{z}_j|} < O_J = \frac{L|\mathbf{I} \cap \mathbf{z}_J|}{L-1+|\mathbf{z}_J|} < \frac{L|\mathbf{z}_J|}{L-1+|\mathbf{z}_J|} \tag{B.26}$$

which implies that $|\mathbf{z}_j| < |\mathbf{z}_J|$. Since J cannot be chosen, it has to be reset by the *vigilance subsystem*, which means that $|\mathbf{I} \cap \mathbf{z}_J| < \rho|\mathbf{I}|$. Therefore,

$$\frac{|\mathbf{z}_j|}{L-1+|\mathbf{z}_j|} < \frac{|\mathbf{I} \cap \mathbf{z}_J|}{L-1+|\mathbf{z}_J|} < \frac{\rho|\mathbf{I}|}{L-1+|\mathbf{z}_J|} < \frac{\rho|\mathbf{I}|}{L-1+|\mathbf{z}_j|} \tag{B.27}$$

which implies that,

$$|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}| \tag{B.28}$$

Modified ART1:

If J and j are nodes with subset templates with respect to $\mathbf{I}$, then

$$O_j = L_A|\mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M < O_J = L_A|\mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M \tag{B.29}$$

Since $(L_A - L_B)|\mathbf{z}_j|$ is an increasing function of $|\mathbf{z}_j|$,

$$|\mathbf{z}_j| < |\mathbf{z}_J| \tag{B.30}$$

and,

$$R_j = \frac{|\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{I}|} = \frac{|\mathbf{z}_j|}{|\mathbf{I}|} < R_j = \frac{|\mathbf{I} \cap \mathbf{z}_J|}{|\mathbf{I}|} = \frac{|\mathbf{z}_J|}{|\mathbf{I}|} \tag{B.31}$$

Therefore, if J is reset ($R_J < \rho$), then all other nodes with subset templates will be reset as well ($R_j < \rho$).

Now suppose that J, the first activated node, does not have a subset template with respect to $\mathbf{I}$ ($|\mathbf{I} \cap \mathbf{z}_J| < |\mathbf{z}_J|$), but that another node j with a subset template is activated in the course of search. We need to show that $|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}|$, so that j is reset. We know that,

$$\begin{aligned} O_j &= (L_A - L_B)|\mathbf{z}_j| + L_M < O_J = L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| + L_M < \\ &< (L_A - L_B)|\mathbf{z}_J| + L_M \end{aligned} \tag{B.32}$$

which implies that $|\mathbf{z}_j| < |\mathbf{z}_J|$. Since J cannot be chosen, it has to be reset by the *vigilance subsystem*, which means that $|\mathbf{I} \cap \mathbf{z}_J| < \rho|\mathbf{I}|$. Therefore,

$$\begin{aligned} L_A|\mathbf{z}_j| - L_B|\mathbf{z}_j| &< L_A|\mathbf{I} \cap \mathbf{z}_J| - L_B|\mathbf{z}_J| < L_A\rho|\mathbf{I}| - L_B|\mathbf{z}_J| < \\ &< L_A\rho|\mathbf{I}| - L_B|\mathbf{z}_j| \end{aligned} \tag{B.33}$$

which implies that,

$$|\mathbf{I} \cap \mathbf{z}_j| = |\mathbf{z}_j| < \rho|\mathbf{I}| \tag{B.34}$$

B.8 SEARCH ORDER(THEOREM 7 OF ORIGINAL ART1)

The original Theorem 7 (page 100 of [Carpenter, 1987]) states the following:

Suppose that input pattern satisfies

$$L - 1 \leq \frac{1}{|\mathbf{I}|} \tag{B.35}$$

and

$$|\mathbf{I}| \leq N - 1 \tag{B.36}$$

Then $F2$ nodes are searched in the following order, if they are searched at all.

Subset templates with respect to $\mathbf{I}$ are searched first, in order of decreasing size. If the largest subset template is reset, then all subset templates are reset. If all subset templates have been reset and if no other learned templates exist, then the first uncommitted node to be activated will code $\mathbf{I}$. If all subset templates are searched and if there exist learned superset templates but no mixed templates, then the node with the smallest superset template will be activated next and will code $\mathbf{I}$. If all subset templates are searched and if both superset templates $\mathbf{z}_J$ and mixed templates $\mathbf{z}_j$ exist, then j will be searched before J if and only if

$$|\mathbf{z}_j| < |\mathbf{z}_J| \quad and \quad \frac{|\mathbf{I}|}{|\mathbf{z}_J|} < \frac{|\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_j|} \tag{B.37}$$

If all subset templates are searched and if there exist mixed templates but no superset templates, then a node j with a mixed template will be searched before an uncommitted node J if and only if

$$\frac{L|\mathbf{I} \cap \mathbf{z}_j|}{L - 1 + |\mathbf{z}_j|} > T_J(\mathbf{I}, t = 0). \tag{B.38}$$

Where $T_J(\mathbf{I}, t=0) = (L \sum I_i z_{iJ}(0))/(L - 1 + \sum z_{iJ}(0))$. The conditions expressed in eqs. (B.35)-(B.38) have to be changed in order to adapt this theorem to the modified ART1 architecture. The original proof will not be reproduced here, because it differs drastically from the one we will provide for the modified theorem. The modified theorem is identical to the original one, except for eqs. (B.35)-(B.38). It states the following:

Suppose that

$$\frac{L_A}{L_B} < \frac{N}{N-1} \tag{B.39}$$

and input pattern satisfies

$$|\mathbf{I}| \leq N - 1 \tag{B.40}$$

Then *F2* nodes are searched in the following order, if they are searched at all.

Subset templates with respect to $\mathbf{I}$ are searched first, in order of decreasing size. If the largest subset template is reset, then all subset templates are reset. If all subset templates have been reset and if no other learned templates exist, then the first uncommitted node to be activated will code $\mathbf{I}$. If all subset templates are searched and if there exist learned superset templates but no mixed templates, then the node with the smallest superset template will be activated next and will code $\mathbf{I}$. If all subset templates are searched and if both superset templates $\mathbf{z}_J$ and mixed templates $\mathbf{z}_j$ exist, then j will be searched before J if and only if

$$|\mathbf{z}_j| < |\mathbf{z}_J| \quad \text{and} \quad \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} < \frac{L_B}{L_A} \tag{B.41}$$

If all subset templates are searched and if there exist mixed templates but no superset templates, then a node j with a mixed template will be searched before an uncommitted node J if and only if

$$L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M > T_J(\mathbf{I}, t=0). \tag{B.42}$$

Where $T_J(\mathbf{I}, t=0) = L_A \sum I_i z_{iJ}(0) - L_B \sum z_{iJ}(0) + L_M$. The proof has several parts:

1. First we show that a node J with a subset template ($\mathbf{I} \cap \mathbf{z}_J = \mathbf{z}_J$) is searched before any node j with a non subset template. In this case,

$$\begin{aligned} O_j &= L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M = \\ &= |\mathbf{I} \cap \mathbf{z}_j|(L_A - L_B\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|}) + L_M \end{aligned} \tag{B.43}$$

Now, note that

$$\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|} > \frac{N}{N-1} \tag{B.44}$$

because[4]

$$\frac{|\mathbf{z}_j|}{|\mathbf{I} \cap \mathbf{z}_j|}|_{min} = \frac{|\mathbf{z}_j|}{|\mathbf{z}_j| - 1}|_{min} = \frac{N-1}{N-2} > \frac{N}{N-1} \tag{B.45}$$

From eqs. (2.4), (B.39) and (B.44), it follows that

$$O_j < |\mathbf{I} \cap \mathbf{z}_j|L_B(\frac{L_A}{L_B} - \frac{N}{N-1}) + L_M < L_M \tag{B.46}$$

On the other hand,

$$O_J = (L_A - L_B)|\mathbf{z}_J| + L_M > L_M \tag{B.47}$$

Therefore,

$$O_J > O_j \tag{B.48}$$

2. Subset templates are searched in order of decreasing size:
   Suppose two subset templates of $\mathbf{I}$, $\mathbf{z}_J$ and $\mathbf{z}_j$ such that $|\mathbf{z}_J| > |\mathbf{z}_j|$. Then

$$O_J = (L_A - L_B)|\mathbf{z}_J| + L_M > (L_A - L_B)|\mathbf{z}_j| + L_M = O_j \tag{B.49}$$

   Therefore node J will be searched before node j. By eq. (B.31), if the largest subset template is reset, then all other subset templates are reset as well.

[4] We are assuming that j is not an uncommitted node ($|\mathbf{z}_j| < N$).

3. Subset templates J are searched before an uncommitted node j:

$$\begin{aligned} O_j &= L_A|\mathbf{I}| - L_B N + L_M \leq L_A(N-1) - L_B N + L_M = \\ &= L_B(\frac{L_A}{L_B}(N-1) - N) + L_M < L_B(\frac{N}{N-1}(N-1) - N) + L_M = \\ &= L_M < (L_A - L_B)|\mathbf{z}_J| + L_M = O_J \end{aligned} \quad \text{(B.50)}$$

Therefore now, if all subset templates are searched and if no other learned template exists, then an uncommitted node will be activated and code the input pattern.

4. If all subset templates have been searched and there exist learned superset templates but no mixed templates, the node with the smallest superset template J will be activated (and not an uncommitted node j) and code $\mathbf{I}$:

$$O_J = L_A|\mathbf{I}| - L_B|\mathbf{z}_J| + L_M > L_A|\mathbf{I}| - L_B N + L_M = O_j \quad \text{(B.51)}$$

If there are more than one superset templates, the one with the smallest $|\mathbf{z}_J|$ will be activated. Since $|\mathbf{I} \cap \mathbf{z}_J| = |\mathbf{I}| \geq \rho|\mathbf{I}|$ there is no reset, and $\mathbf{I}$ will be coded.

5. If all subset templates have been searched and there exist a superset template J and a mixed template j , then $O_j > O_J$ if and only if eq. (B.41) holds:

$$O_j - O_J = L_A(|\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{I}|) + L_B(|\mathbf{z}_J| - |\mathbf{z}_j|) \quad \text{(B.52)}$$

(a) if eq. (B.41) holds:

$$O_j - O_J = L_A(\frac{L_B}{L_A} - \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|})(|\mathbf{z}_J| - |\mathbf{z}_j|) > 0 \quad \text{(B.53)}$$

(b) if $O_j > O_J$:
Assume first that $|\mathbf{z}_J| - |\mathbf{z}_j| < 0$. Then, by eq. (B.53), it has to be

$$\frac{L_B}{L_A} < \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} \quad \text{(B.54)}$$

Since $L_A > L_B > 0$ it had to be $|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j| < 0$, which is false. Therefore, it must be $|\mathbf{z}_J| - |\mathbf{z}_j| > 0$ and

$$\frac{L_B}{L_A} > \frac{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|}{|\mathbf{z}_J| - |\mathbf{z}_j|} \quad \text{(B.55)}$$

6. If all subset templates are searched and if there exist mixed templates but no superset templates, then a node j with a mixed template ($O_j = L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M$) will be searched before an uncommitted node J ($O_J = L_A|\mathbf{I}| - L_B N + L_M$) if and only if eq. (B.42) holds:

$$O_j - O_J = L_A(|\mathbf{I} \cap \mathbf{z}_j| - |\mathbf{I}|) - L_B(|\mathbf{z}_j| - N) > 0 \Leftrightarrow \tag{B.56}$$
$$\Leftrightarrow L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M > L_A|\mathbf{I}| - L_B N + L_M = T_J(\mathbf{I}, t = 0)$$

   This completes the proof of the modified Theorem 7 for the modified ART1 architecture.

B.9 BIASING THE NETWORK TOWARDS UNCOMMITTED NODES

In the original ART1 architecture, choosing L large increases the tendency of the network to choose uncommitted nodes in response to unfamiliar input patterns $\mathbf{I}$. In the modified ART1 architecture, the same effect is observed when choosing $\alpha = L_A/L_B$ large. This can be understood through the following reasoning.

When an input pattern $\mathbf{I}$ is presented, an uncommitted node is chosen before a coded node j if

$$L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| < L_A|\mathbf{I}| - L_B N \tag{B.57}$$

This inequality is equivalent to

$$\frac{L_A}{L_B} > \frac{N - |\mathbf{z}_j|}{|\mathbf{I}| - |\mathbf{I} \cap \mathbf{z}_j|} \tag{B.58}$$

As the ratio $\alpha = L_A/L_B$ increases it is more likely that eq. (B.58) is satisfied, and hence that uncommitted nodes are chosen before coded nodes, regardless of the *vigilance parameter* value ρ.

B.10 EXPANDING PROOFS TO FUZZY–ART

All properties, theorems and proofs in this Appendix are directly applicable to Fuzzy–ART by simply substituting the intersection operator ($\cap$) by the fuzzy minimum operator ($\wedge$). Note that the subset and superset concepts used in ART1 also expand to the fuzzy subset and superset concepts in Fuzzy–ART by doing this simple substitution. In ART1 pattern $\mathbf{a}$ is said to be a subset of pattern $\mathbf{b}$ (or $\mathbf{b}$ a superset of $\mathbf{a}$), and is denoted as $\mathbf{a} \subset \mathbf{b}$, if

$$\mathbf{a} \cap \mathbf{b} = \mathbf{a} \tag{B.59}$$

In Fuzzy–ART pattern $\mathbf{a}$ is said to be a fuzzy subset of pattern $\mathbf{b}$ (or $\mathbf{b}$ a fuzzy superset of $\mathbf{a}$), and is also denoted as $\mathbf{a} \subset \mathbf{b}$, if [Zadeh, 1965]

$$\mathbf{a} \wedge \mathbf{b} = \mathbf{a} \tag{B.60}$$

B.11 REMARKS

Even though in this Appendix we have shown that the computational properties of the original ART1 system are preserved in the modified ART1 system, the response of both systems to an arbitrary list of training patterns will not be exactly the same. The main underlying reason for this difference in behavior is that the initial ordering

$$O_{j_1} > O_{j_2} > O_{j_3} > \ldots \tag{B.61}$$

is not always exactly the same for both architectures. In Chapter 2 we tried to study the differences in behavior between the two ART1 systems. As we saw, for most cases the behavior is identical, although in a few cases a different behavior results.

Appendix C
Systematic Width-and-Length Dependent CMOS Transistor Mismatch Characterization

Precise analog CMOS circuit design requires availability of confident transistor mismatch models during the design and simulation stages. During the design phase of an analog VLSI circuit, designers face many constraints imposed by the design specifications, such as speed, bandwidth, noise, precision, power consumption, area consumption, which need to be traded off for optimum overall performance. Designers must rely on accurate simulation tools in order to achieve a well optimized final design, specially if performance is pushed to the limits allowed by a given fabrication process. Simulation tools are reliable as long as they are based on good models and confident characterization techniques. If good and well characterized models are embedded in a reliable simulator, circuit designers can confidently test different circuit topologies and optimize each one of them by optimally sizing their transistors. Automatic design tools are available that by interacting with a simulator are able to obtain transistor sizes for close-to-optimum performance for a given circuit topology and a set of design constraints [Medeiro, 1994].

Many times it is not possible to simulate properly the precision limits that can be achieved by a certain circuit topology in a given fabrication process because VLSI circuit manufacturers rarely provide transistor mismatch information, and, if they do, its dependence on transistor size (width and length, independently[1]) is not known. In this Appendix we provide a very simple and cheap methodology to characterize transistor mismatch as a function of transistor width and length, and how to use this information to predict mismatch effects in circuit simulators.

[1] Sometimes manufactures provide mismatch information as a function of transistor area, but this information has been obtained for (almost) square transistors [Pelgrom, 1989].

In the specialized literature transistor mismatch is usually characterized by providing the standard deviation of a set of transistor electrical parameters such as the threshold voltage V_{TO}, the current gain factor $\beta = \mu C_{ox} W/L$ (μ is mobility, C_{ox} is gate oxide capacitance density, W is transistor width, and L is transistor length), and bulk threshold parameter γ. Table C.1 shows a few examples [Pelgrom, 1989], [Lakshmikumar, 1986], [Bastos, 1995], [Bastos, 1997] on what dependencies of $\sigma^2_{(\Delta\beta/\beta)}$, $\sigma^2_{(\Delta V_{TO})}$, and $\sigma^2_{(\Delta\gamma)}$ on transistor sizes ($x = 1/W$, $y = 1/L$, W is transistor width, L is transistor length) and distance D have been postulated. A very nice study [Michael, 1992] based on BSIM transistor models is also available in the literature.

In the present paper we provide an experimental method to obtain a relatively high number of samples (of $\sigma^2_{(\Delta\beta/\beta)}$, $\sigma^2_{(\Delta V_{TO})}$, $\sigma^2_{(\Delta\gamma)}$ and) in the $\{x, y\}$ space. Then we will fit these measured samples to a function

$$\begin{aligned} \sigma^2_{(\Delta P)} &= C_{00} + C_{10}x + C_{01}y + C_{20}x^2 + C_{11}xy + C_{02}y^2 + \cdots = \\ &= \sum_{n,m} C_{nm} x^n y^m \end{aligned} \tag{C.1}$$

where ΔP is the observed mismatch of a certain electrical parameter (like $\Delta\beta/\beta$, ΔV_{TO}, or $\Delta\gamma$). Note that we are not interested in discovering the physical meaning of coefficients C_{nm}, but just in obtaining a good approximation for the function $\sigma^2_{(\Delta P)} = f(x, y)$ in order to use it confidently in a circuit simulator. Note also that the $\{x, y\}$ space limits are $x_{max} = 1/W_{min}$, $y_{max} = 1/L_{min}$, $x_{min} = 0$, $y_{min} = 0$. Measuring a reasonable high number of sample points in this $\{x, y\}$ space provides sufficient information to interpolate the functions $\sigma^2_{(\Delta P)}$, which are fairly smooth in this space. Next Section describes the mismatch characterization chip used to obtain all characterization data.

C.1 MISMATCH CHARACTERIZATION CHIP

According to Table C.1 the mismatch in parameter P between two identical transistors is statistically characterized by a quadratic deviation whose general form is

$$\sigma^2_{(\Delta P)} = f(x, y) + S_P^2 D^2 \tag{C.2}$$

where $x = 1/W$, $y = 1/L$, W and L are the transistor width and length, and D is the distance between them. The two terms in eq. (C.2) indicate that there are two physical causes producing transistor mismatch. The first term is produced by the fact that **device physical parameters** (like doping concentrations, junctions depth, implants depth, oxide thicknesses, ...) are not

Table C.1. Examples of Mismatch Models in the Literature ($x = 1/W$, $y = 1/L$)

	$\sigma^2_{(\Delta\beta/\beta)}$	$\sigma^2_{(\Delta V_{TO})}$	$\sigma^2_{(\Delta\gamma)}$
[Pelgrom, 1989]	$A_s^2xy + A_w^2x^2y+$ $+A_L^2xy^2 + S_\beta^2D^2$	$A_{V_{TO}}^2xy + S_{V_{TO}}^2D^2$	A_γ^2xy+ $+S_\gamma^2D^2$
[Lakshmikumar, 1986]	$A_{\beta 1}xy+$ $+A_{\beta 2}(x^2+y^2)$	$A_{VO}xy$	-
[Bastos, 1995]	A_β^2xy	$A_{V1}^2xy + A_{V2}^2xy^2-$ $-A_{V3}^2x^2y$	-

exactly constant but suffer from noise-like perturbations along a die. By increasing transistor areas the **device electrical parameters** P (like threshold voltage V_{TO}, current gain factor β, or bulk threshold parameter γ) will become less sensitive to the noisy nature of the **device physical parameters**. The second term in eq. (C.2), characterized by parameter S_P, is produced by the fact that the **device physical parameters** present a certain gradient variation along the dies. Usually, the gradients present in small and medium size dies can be approximated by planes. Statistical characterization of these planes (which means obtaining S_P) can be performed with a small number of transistors per die and measuring many dies. On the other hand, the transistor mismatch induced by the **device physical parameters** noisy nature, changes little from die to die. Consequently, its statistical characterization can be done by putting many transistors in a single die and measuring a reduced number of dies. This is very convenient for a 'do-it-yourself' working style, since circuit designers can easily have a small number of samples of a prototype chip at a reasonable cost. This Appendix thus concentrates on the characterization of size dependent mismatch terms, and a wide range of transistor sizes will be characterized. On the contrary, note that characterization of distance terms (like S_P) is less critical for circuit designers because gradient-induced mismatches can be compensated through layout techniques (like common centroids, for example).

With all this in mind we designed a special purpose chip intended to characterize the 'noise- induced-terms' of CMOS transistor mismatches, as a function of transistor size. As shown in Fig. C.1, the chip consists of an array of identical cells. Each cell contains 30 NMOS and 30 PMOS transistors, each of a different size. Sizes are such that widths are $W = 40\mu m$, $20\mu m$, $10\mu m$, $5\mu m$, $2.5\mu m$, $1.25\mu m$, and lengths are $L = 40\mu m$, $10\mu m$, $4\mu m$, $2\mu m$, $1\mu m$. Digital decoding/selection circuitry is included in each cell and outside the array. Elements in the chip are arranged in a way such that all NMOS transistors have their Drains connected to chip pin DN, all PMOS transistors have their

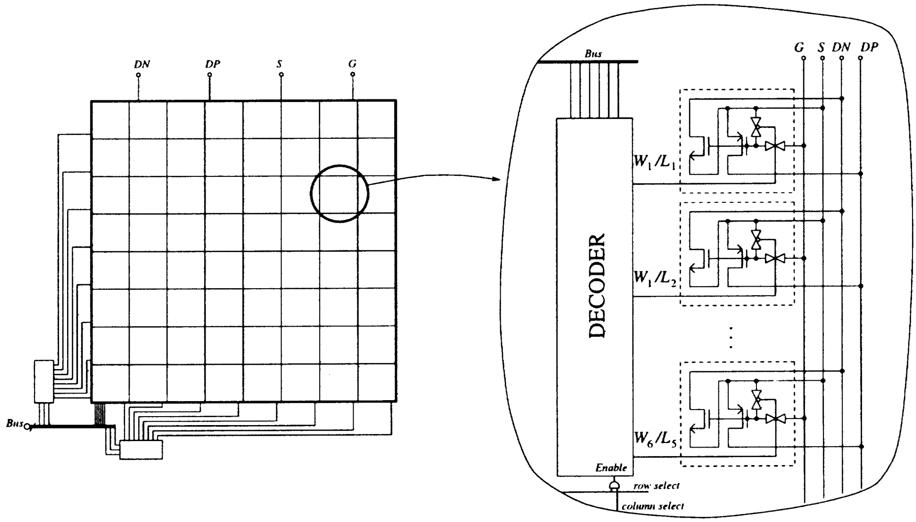

Figure C.1. Mismatch Characterization Chip Simplified Schematic

Drains connected to chip pin DP, all NMOS and PMOS transistors have their sources connected to chip pin S, all NMOS and PMOS transistors have their Gates short-circuited to their sources, except for one NMOS-PMOS pair which has their Gates connected to chip pin G. The digital bus and the internal decoding/selection circuitry selects one cell in the array and, inside this cell, one pair of NMOS and PMOS transistors, connecting their Gates to chip pin G. A chip with an 8×8 cell array has been fabricated in a digital $1.0\mu m$ CMOS process which occupies an area of $4.0mm \times 3.5mm$, and uses 18 pins (12 for the decoding/selection Bus, DN, DP, G, S, V_{dd} , and Gnd). Some transistors in the periphery cells presented large systematic deviations with respect to those in the inside cells. Consequently, statistical computations were performed only on inner cells transistors, thus rendering an effective cell array of 6×6.

The experimental characterization set-up consists of a host computer controlling the decoding/selection bus and a DC curves measuring instrument (like the HP4145). This instrument is connected to pins DN, DP, S, G, and chip substrate. The host computer selects one NMOS-PMOS pair and the instrument measures first the NMOS transistor (putting connection DP into high-impedance and measuring through pins S, G, and DN) and then the PMOS transistor (putting connection DN into high-impedance and measuring through pins S, G, and DP). A simple software program sequentially selects and measures all transistors in the chip. Next Section describes the DC curves that were measured for each transistor, how electrical parameter mismatches were extracted from these curves, and how their statistical characterization was performed.

C.2 MISMATCH PARAMETER EXTRACTION AND STATISTICAL CHARACTERIZATION

Transistor parameter mismatches were obtained by measuring pairs of identical transistors located in adjacent cells of the same rows. Since in the chip there are 6 × 6 effective cells, there are 6 rows, each of which provides 5 pairs of adjacent cells. This results in 30 adjacent transistor pairs (for each transistor size and type). The statistical significance of 30 measurements to determine a standard deviation is as follows: assuming a normal distribution, if 30 samples are available to compute a standard deviation $\sigma_{Computed}$, it can be assured that the 95% confidence interval for the real standard deviation σ_{Real} is [Rade, 1990]

$$0.7964 \times \sigma_{Computed} \leq \sigma_{Real} \leq 1.344 \times \sigma_{Computed}. \tag{C.3}$$

For each transistor pair, two curves were measured while operating in the ohmic region (always in strong inversion). These curves are[2]

$$\textbf{Curve 1:} \quad \begin{array}{l} I_{DS}(V_{GS}),\ V_{SB} = 0V, \\ V_{DS} = 0.1V,\ V_{GS} \in [V_{GS_{min}}, V_{GS_{max}}] \end{array} \tag{C.4}$$

$$\textbf{Curve 2:} \quad \begin{array}{l} I_{DS}(V_{SB}),\ V_{GS} = 3.0V, \\ V_{DS} = 0.1V,\ V_{SB} \in [V_{SB_{min}}, V_{SB_{max}}] \end{array} \tag{C.5}$$

Care must be taken in order to keep current levels sufficiently small so that mismatch introduced by series resistances (contact resistances, variable length routing wires, ...) is negligible. The following strong inversion ohmic region transistor model was assumed,

$$I_{DS} = \beta \frac{V_{GS} - V_T(V_{SB}) - \frac{1}{2}V_{DS}}{1 + \theta(V_{GS} - V_T(V_{SB}))} V_{DS} \tag{C.6}$$

$$V_T(V_{SB}) = V_{TO} + \gamma \left[\sqrt{\Phi + V_{SB}} - \sqrt{\Phi}\right] \tag{C.7}$$

[2] All voltages and currents are taken in absolute value, so that the same expressions are valid for NMOS and PMOS transistors.

which renders the following current mismatch for each transistor pair:

$$\begin{aligned} \frac{\Delta I_{DS}}{I_{DS}} &= \frac{\Delta\beta}{\beta} - \frac{1+\frac{1}{2}\theta V_{DS}}{V_{GS} - V_T(V_{SB}) - \frac{1}{2}V_{DS}}\Delta V_T - \\ &- \frac{V_{GS} - V_T(V_{SB})}{1+\theta(V_{GS} - V_T(V_{SB}))}\Delta\theta \end{aligned} \tag{C.8}$$

$$\Delta V_T = \Delta V_{TO} + \Delta\gamma\left[\sqrt{\Phi + V_{SB}} - \sqrt{\Phi}\right] \tag{C.9}$$

The drain and source series resistances (due to contacts, diffusion resistance, and metal routing lines) have the effect of changing the extracted value of the mobility reduction parameter θ in eq. (C.6) [Pelgrom, 1989],

$$\theta = \theta_{Real} + \beta R_{DS} \tag{C.10}$$

where θ_{Real} is the real mobility reduction parameter of the transistor and R_{DS} is the sum of the series resistances at drain and source. The mismatch contribution of $\Delta(\beta R_{DS})$ can be of the order or higher than that of $\Delta\theta_{Real}$, but both contribute very little to $\Delta I_{DS}/I_{DS}$.

For each transistor pair, the measurement/extraction procedure was as follows:

- **Curve 1** (eq. (C.4)) was measured for both transistors. Using the Levenberg-Marquardt nonlinear curve fitting technique, the first curve was fitted to eq. (C.6) and parameters β, V_{TO}, and θ were obtained. Using the two measured curves, the curve $\Delta I_{DS}/I_{DS}$ was computed and fitted to eq. (C.8) obtaining $\Delta\beta/\beta$, ΔV_{TO}, and $\Delta\theta$ for this transistor pair.

- **Curve 2** (eq. (C.5)) was measured for both transistors, and curve $\Delta I_{DS}/I_{DS}$ was computed. According to eqs. (C.8), (C.9) it must fit to

$$\begin{aligned} \frac{\Delta I_{DS}}{I_{DS}} &= \frac{\Delta\beta}{\beta} - \frac{1+\frac{1}{2}\theta V_{DS}}{V_x - \frac{1}{2}V_{DS}}\left[\Delta V_{TO} + \Delta\gamma(\sqrt{\Phi + V_{SB}} - \sqrt{\Phi})\right] - \\ &- \frac{V_x}{1+\theta V_x}\Delta\theta, \end{aligned} \tag{C.11}$$

  where $V_x = V_{GS} - V_T(V_{SB})$. For this transistor pair $\Delta\beta/\beta$ and ΔV_{TO} are already known. The values for $V_x = V_{GS} - V_T(V_{SB})$ can be obtained from eq. (C.6)

$$V_x = \frac{I_{DS} + \frac{\beta}{2} V_{DS}^2}{\beta V_{DS} - \theta I_{DS}} \quad \text{(C.12)}$$

since β, θ, and V_{DS} are already known, and the I_{DS} values are those just measured at Curve 2. Consequently, $\Delta\gamma$ is the only unknown parameter in eq. (C.11), which can be obtained for this pair by fitting eq. (C.11), after substituting eq. (C.12) into it.

This measurement/extraction procedure is repeated for the $N_T = 30$ transistor pairs. For each extracted mismatch parameter ΔP ($\Delta\beta/\beta$, ΔV_{TO}, $\Delta\gamma$) its standard deviation

$$\sigma^2_{(\Delta P)} = \frac{1}{N_T} \sum_{n=1}^{N_T} (\Delta P_n - \overline{\Delta P})^2 \quad \text{(C.13)}$$

is computed. For each fabricated chip, eq. (C.13) should be obtained for each transistor size and type (NMOS or PMOS).

C.3 CHARACTERIZATION RESULTS

A mismatch characterization chip was fabricated in a digital double-metal single-poly $1.0\mu m$ CMOS process. The die area of the chip is $3.5mm \times 4.0mm$. Ten samples were delivered by the foundry, eight of which were fault free. For each die, transistor size, and transistor type the following parameters were extracted, following the procedure described in the previous Section,

$$\sigma_{(\Delta\beta/\beta)} \quad , \quad \sigma_{(\Delta V_{TO})} \quad , \quad \sigma_{(\Delta\gamma)} \quad \text{(C.14)}$$

Table C.2 shows these parameters for the NMOS transistors, averaged over all dies and indicating the spread from die to die. Table C.3 shows the average over all dies of these parameters for the PMOS transistors. Each cell in Tables C.2 and C.3 indicates

$$\overline{\sigma_{(\Delta P)}} \pm 3\sigma(\sigma_{(\Delta P)}) \quad \text{(C.15)}$$

where,

$$\overline{\sigma_{(\Delta P)}} = \frac{1}{N_{Dies}} \sum_{n_d=1}^{N_{Dies}} \sigma_{(\Delta P)}(n_d) \tag{C.16}$$

$$\sigma(\sigma_{(\Delta P)}) = \frac{1}{N_{Dies}} \sum_{n_d=1}^{N_{Dies}} (\sigma_{(\Delta P)}(n_d) - \overline{\sigma_{(\Delta P)}})^2$$

and N_{Dies} is the total number of fault-free dies.

Looking at the deviations $\sigma_{(\Delta P)}$ in Tables C.2 and C.3 and at their $\pm 3\sigma$ inter-chip spread, one can see that the 3σ spread is of the order of $\pm 50\%$ of the average deviation. This shows that the deviations $\sigma_{(\Delta P)}$ of transistor mismatch parameters have a fairly stable behavior from chip to chip.

In order to measure the precision of the extracted deviation values, one of the dies was measured without sweeping the transistor pairs: Curves 1 and 2 (eqs. (C.4) and (C.5)) were measured 30 times for the same pair. The resulting deviations indicate the measurement set-up and extraction procedure precision. This is given in Table C.4 for the NMOS transistors and in Table C.5 for the PMOS transistors. Units in Table C.4 and C.5 are the same than for Tables C.2 and C.3, except for a 10^{-3} factor.

The extracted data for each column in Tables C.2 and C.3 can be considered to be sample points of a two dimensional surface whose independent variables are $x = 1/W$ and $y = 1/L$,

$$\sigma_{(\Delta P)} = f(x, y) \tag{C.17}$$

As mentioned before, several functionals have been attributed to eq. (C.17). We will assume the following polynomial dependency

$$\begin{aligned} \sigma^2_{(\Delta P)_{fit}}(x, y, C_{nm}) &= C_{00} + C_{10}x + C_{01}y + C_{20}x^2 + C_{11}xy + C_{02}y^2 + \cdots = \\ &= \sum_{n,m} C_{nm} x^n y^m \end{aligned} \tag{C.18}$$

The optimum set of coefficients C_{nm} were obtained by fitting eq. (C.18) (using Least Mean Squares) to the experimental data $\sigma_{(\Delta P)}(x_i, y_i, n_d)$ for all sizes and dies. For this, the following error term was defined,

$$\epsilon = \sum_{n_d=1}^{N_{Dies}} \sum_{i=1}^{N_{Sizes}} w_i \left[\sigma^2_{(\Delta P)}(x_i, y_i, n_d) - \sigma^2_{(\Delta P)_{fit}}(x_i, y_i, C_{nm})\right]^2 \tag{C.19}$$

Table C.2. NMOS Mismatch Characterization Parameters averaged over all measured dies and indicating the $\pm 3\sigma$ spread from die to die

sizes	$\sigma_{(\Delta\beta/\beta)}(\times 10^{-3})$	$\sigma_{(\Delta V_{TO})}(mV)$	$\sigma_{(\Delta\gamma)}(mV^{1/2})$
40/40	1.18 ± 0.78	0.60 ± 0.35	0.49 ± 0.15
40/10	2.21 ± 1.71	1.21 ± 0.59	0.57 ± 0.22
40/4	3.36 ± 1.71	1.27 ± 0.52	0.86 ± 0.36
40/2	5.92 ± 1.41	1.94 ± 0.76	1.29 ± 0.71
40/1	9.01 ± 2.93	5.23 ± 1.97	3.93 ± 1.28
20/40	1.87 ± 0.66	0.66 ± 0.25	0.48 ± 0.27
20/10	2.43 ± 0.90	1.14 ± 0.40	0.73 ± 0.34
20/4	4.17 ± 1.75	1.67 ± 0.56	1.14 ± 0.46
20/2	9.18 ± 4.74	2.58 ± 1.84	1.80 ± 1.07
20/1	12.52 ± 7.10	6.67 ± 1.95	4.72 ± 1.19
10/40	1.77 ± 0.51	0.66 ± 0.24	0.62 ± 0.49
10/10	4.87 ± 1.45	1.37 ± 0.41	0.92 ± 0.43
10/4	6.83 ± 1.94	2.03 ± 0.88	1.39 ± 0.37
10/2	7.89 ± 2.23	3.70 ± 1.69	2.27 ± 1.29
10/1	13.41 ± 7.36	8.60 ± 2.30	5.49 ± 3.23
5/40	2.97 ± 1.26	0.82 ± 0.61	0.76 ± 0.25
5/10	4.53 ± 2.28	2.01 ± 0.68	1.35 ± 0.56
5/4	6.77 ± 1.90	3.29 ± 1.04	2.02 ± 1.24
5/2	10.10 ± 4.28	4.76 ± 2.21	3.13 ± 1.14
5/1	14.80 ± 4.00	11.66 ± 3.10	6.15 ± 2.54
2.5/40	6.71 ± 2.88	1.27 ± 0.86	1.01 ± 0.40
2.5/10	8.84 ± 5.01	2.46 ± 1.26	1.66 ± 0.78
2.5/4	9.09 ± 3.69	4.14 ± 1.43	2.98 ± 1.50
2.5/2	13.85 ± 5.46	6.68 ± 3.32	4.30 ± 1.05
2.5/1	21.56 ± 10.77	15.93 ± 8.49	8.51 ± 3.28
1.25/40	11.40 ± 4.56	1.67 ± 0.87	1.79 ± 0.56
1.25/10	12.50 ± 4.96	3.52 ± 1.92	2.69 ± 1.34
1.25/4	10.71 ± 2.58	5.24 ± 2.06	3.75 ± 1.53
1.25/2	15.26 ± 8.25	9.65 ± 4.65	5.69 ± 2.66
1.25/1	21.69 ± 9.06	21.94 ± 6.02	11.06 ± 4.29

where w_i is a weighting term that depends on the spread of $\sigma_{(\Delta P)}(x_i, y_i, n_d)$ from die to die, and N_{Sizes} is the total number of transistor sizes available in the chip. If $\sigma_{(\Delta P)}$, for size $x_i = 1/W_i$, $y_i = 1/L_i$ has a large spread from die to die then weight w_i will be smaller than for another size whose spread is smaller. If for a given size (x_i, y_i) the spread from die to die is $\sigma(\sigma_{(\Delta P)}(x_i, y_i))$ then the weight w_i is defined as

Table C.3. PMOS Mismatch Characterization Parameters averaged over all measured dies and indicating the $\pm 3\sigma$ spread from die to die

sizes	$\sigma_{(\Delta\beta/\beta)}(\times 10^{-3})$	$\sigma_{(\Delta V_{TO})}(mV)$	$\sigma_{(\Delta\gamma)}(mV^{1/2})$
40/40	1.41 ± 0.67	1.38 ± 0.77	0.45 ± 0.18
40/10	2.20 ± 1.35	1.89 ± 1.54	0.53 ± 0.20
40/4	3.10 ± 1.41	1.87 ± 0.73	0.64 ± 0.16
40/2	6.22 ± 2.70	2.73 ± 0.69	1.08 ± 0.41
40/1	11.43 ± 6.39	5.78 ± 2.33	3.65 ± 1.70
20/40	1.75 ± 0.49	1.06 ± 0.87	0.54 ± 0.20
20/10	2.97 ± 1.20	1.80 ± 0.70	0.58 ± 0.19
20/4	3.77 ± 1.56	2.27 ± 1.09	0.83 ± 0.21
20/2	10.06 ± 4.41	3.56 ± 0.80	1.59 ± 0.89
20/1	13.55 ± 4.92	7.60 ± 2.61	3.81 ± 1.83
10/40	1.64 ± 0.74	1.19 ± 0.50	0.49 ± 0.22
10/10	4.76 ± 2.06	2.18 ± 1.10	0.73 ± 0.32
10/4	6.75 ± 2.03	2.78 ± 0.72	1.00 ± 0.52
10/2	9.03 ± 7.84	4.86 ± 3.14	1.93 ± 0.78
10/1	20.70 ± 6.41	10.66 ± 2.77	5.18 ± 1.83
5/40	2.89 ± 1.44	1.43 ± 0.56	0.55 ± 0.30
5/10	5.21 ± 1.60	2.54 ± 0.78	0.91 ± 0.51
5/4	6.86 ± 1.67	4.33 ± 1.90	1.35 ± 0.47
5/2	11.23 ± 7.76	7.56 ± 4.29	2.22 ± 0.52
5/1	17.99 ± 8.99	12.81 ± 4.41	4.98 ± 2.17
2.5/40	6.04 ± 1.94	1.73 ± 0.69	0.58 ± 0.23
2.5/10	9.44 ± 6.28	3.40 ± 0.77	1.12 ± 0.50
2.5/4	11.82 ± 4.01	5.86 ± 1.37	1.94 ± 0.82
2.5/2	14.39 ± 8.64	9.08 ± 4.18	3.14 ± 1.54
2.5/1	25.44 ± 8.31	17.63 ± 5.36	6.60 ± 2.40
1.25/40	12.22 ± 8.05	2.65 ± 1.73	0.95 ± 0.40
1.25/10	14.32 ± 7.29	4.50 ± 2.46	1.71 ± 0.70
1.25/4	12.46 ± 5.98	6.70 ± 3.00	2.49 ± 0.94
1.25/2	19.33 ± 10.56	10.76 ± 4.12	3.80 ± 1.49
1.25/1	28.54 ± 9.41	21.05 ± 7.97	7.18 ± 3.89

$$w_i = \frac{e^{-\Omega_i}}{\Omega_i^2} \quad , \quad \Omega_i = \frac{\sigma(\sigma_{(\Delta P)}(x_i, y_i))}{\sigma_{(\Delta P)}(x_i, y_i)}. \tag{C.20}$$

Fig. C.2(a) shows the resulting fitted surface for $\sigma_{(\Delta\beta/\beta)_{fit}}(1/W, 1/L)$, for NMOS transistors. Also shown in Fig. C.2(a) (with diamonds) are the experimental measured values for $\sigma_{(\Delta\beta/\beta)}(1/W_i, 1/L_i, n_d)$ for each transistor size

Table C.4. NMOS Precision Measurements for the data in Table C.2

sizes	$\sigma_{(\Delta\beta/\beta)}$	$\sigma_{(\Delta V_{TO})}(V)$	$\sigma_{(\Delta\gamma)}(V^{1/2})$
40/40	2.99e-04	1.70e-04	2.64e-04
40/10	4.29e-04	2.54e-04	3.58e-04
40/4	3.84e-04	2.09e-04	4.36e-04
40/2	4.57e-04	2.88e-04	4.69e-04
40/1	6.50e-04	4.15e-04	6.62e-04
20/40	2.64e-04	1.40e-04	3.14e-04
20/10	2.80e-04	1.59e-04	3.29e-04
20/4	3.56e-04	1.92e-04	3.07e-04
20/2	4.35e-04	2.24e-04	4.98e-04
20/1	5.58e-04	3.92e-04	6.02e-04
10/40	3.92e-04	2.46e-04	3.10e-04
10/10	3.87e-04	2.39e-04	4.17e-04
10/4	4.51e-04	3.04e-04	4.00e-04
10/2	4.20e-04	2.44e-04	3.53e-04
10/1	6.11e-04	4.20e-04	7.70e-04
5/40	3.75e-04	1.71e-04	4.45e-04
5/10	4.38e-04	2.57e-04	3.71e-04
5/4	2.91e-04	1.80e-04	2.94e-04
5/2	3.49e-04	2.53e-04	3.76e-04
5/1	4.60e-04	2.47e-04	7.99e-04
2.5/40	2.80e-04	1.25e-04	3.68e-04
2.5/10	3.49e-04	2.32e-04	4.20e-04
2.5/4	3.64e-04	2.13e-04	4.77e-04
2.5/2	3.52e-04	2.07e-04	4.06e-04
2.5/1	4.20e-04	3.29e-04	4.97e-04
1.25/40	2.72e-04	1.76e-04	2.60e-04
1.25/10	3.54e-04	1.69e-04	3.24e-04
1.25/4	3.68e-04	2.27e-04	3.24e-04
1.25/2	4.20e-04	2.70e-04	4.53e-04
1.25/1	4.86e-04	2.55e-04	4.63e-04

and for each die, for NMOS transistors. Fig. C.2(b) shows the fitted surface and experimental data for the threshold voltage deviation $\sigma_{(\Delta V_{TO})}$, and Fig. C.2(c) does it for the bulk threshold parameter deviation $\sigma_{(\Delta\gamma)}$, for NMOS transistors. The coefficients C_{nm} that result from the fitting procedure are given in Tables C.6 and C.7 for all deviations for NMOS and PMOS transistors, respectively. The units of the coefficients are such that if x and y are expressed in μm^{-1} then the deviations σ have the units used in Tables C.2, C.3 (without the 10^{-3} factor), and Tables C.4, C.5.

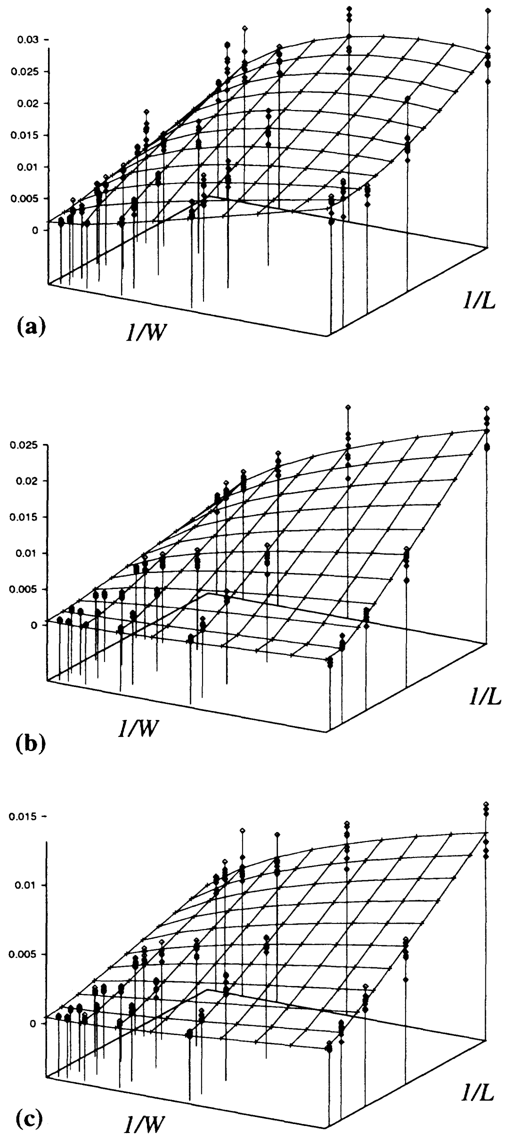

Figure C.2. Experimentally measured/extracted mismatch data (diamonds) as a function of transistor size, for NMOS transistors. Also shown are the interpolated surfaces. (a) Results for $\sigma_{(\Delta\beta/\beta)}$, (b) for $\sigma_{(\Delta V_{TO})}$, (c) and for $\sigma_{(\Delta\gamma)}$

Table C.5. PMOS Precision Measurements for the data in Table C.3

sizes	$\sigma_{(\Delta\beta/\beta)}$	$\sigma_{(\Delta V_{TO})}(V)$	$\sigma_{(\Delta\gamma)}(V^{1/2})$
40/40	4.99e-04	2.95e-04	4.10e-04
40/10	4.01e-04	1.77e-04	4.70e-04
40/4	3.70e-04	1.60e-04	3.73e-04
40/2	4.23e-04	1.56e-04	4.06e-04
40/1	6.46e-04	2.67e-04	6.66e-04
20/40	4.67e-04	1.89e-04	3.70e-04
20/10	4.57e-04	2.24e-04	3.74e-04
20/4	3.77e-04	1.21e-04	4.60e-04
20/2	3.70e-04	1.98e-04	3.83e-04
20/1	5.53e-04	2.53e-04	4.98e-04
10/40	4.80e-04	1.97e-04	3.78e-04
10/10	4.22e-04	2.37e-04	3.86e-04
10/4	4.64e-04	1.80e-04	4.03e-04
10/2	5.29e-04	2.01e-04	4.74e-04
10/1	4.16e-04	2.33e-04	3.71e-04
5/40	3.43e-04	1.75e-04	3.51e-04
5/10	5.26e-04	1.74e-04	4.38e-04
5/4	4.02e-04	2.32e-04	3.76e-04
5/2	4.21e-04	2.05e-04	3.57e-04
5/1	4.85e-04	2.64e-04	4.81e-04
2.5/40	4.52e-04	2.81e-04	3.45e-04
2.5/10	4.43e-04	1.60e-04	3.91e-04
2.5/4	3.63e-04	1.26e-04	3.22e-04
2.5/2	3.40e-04	1.06e-04	4.04e-04
2.5/1	4.02e-04	2.37e-04	5.01e-04
1.25/40	4.26e-04	2.50e-04	5.03e-04
1.25/10	2.05e-04	6.80e-05	2.76e-04
1.25/4	3.34e-04	1.08e-04	3.08e-04
1.25/2	4.84e-04	1.41e-04	4.87e-04
1.25/1	6.41e-04	3.23e-04	5.65e-04

Correlations among deviations of different parameters can also be easily obtained. However, according to our data, no definite conclusions can be made. We observed that for some dies there were high correlation coefficients for some parameter pairs, while for other dies the same correlation coefficient could change significantly (we observed changes from almost +1 correlation factor for one die to almost −1 for another die). If these measurements are correct this would mean that deviation correlations could change significantly from one area of the wafer to another. However, before making any conclusions about

Table C.6. NMOS Resulting coefficients for the fitting functions

	C_{00}	C_{11}	C_{20}	C_{02}	C_{21}	C_{12}	C_{22}
$\sigma^2_{(\Delta\beta/\beta)}$	1.7e-06	8.9e-04	2.0e-04	5.1e-05	-1.2e-03	4.2e-04	0
$\sigma^2_{(\Delta V_{TO})}$	3.4e-07	-4.3e-05	8.1e-06	1.6e-05	0	6.2e-04	0
$\sigma^2_{(\Delta\gamma)}$	2.0e-07	-7.5e-06	5.1e-06	1.3e-05	3.3e-05	1.6e-04	-5.7e-05

Table C.7. PMOS Resulting coefficients for the fitting functions

	C_{00}	C_{11}	C_{20}	C_{02}	C_{21}	C_{12}	C_{22}
$\sigma^2_{(\Delta\beta/\beta)}$	1.4e-06	7.1e-04	2.6e-04	1.5e-04	-6.0e-04	9.1e-04	-6.5e-04
$\sigma^2_{(\Delta V_{TO})}$	2.5e-06	1.7e-04	0	0	-1.3e-04	8.4e-04	-4.5e-04
$\sigma^2_{(\Delta\gamma)}$	1.7e-07	-2.0e-05	0	1.0e-05	5.5e-05	1.3e-04	-1.3e-04

this possible "wafer-level" behavior it would be necessary to make intensive measurements using many dies per wafer and for many wafers. In our case, since we only used 8 dies from one run we can only make approximate conclusions regarding the measured standard deviation values, but not much can be said about their correlations.

References

Allen, P. E. and Holberg, D. R. (1987). *CMOS Analog Design.* Holt Rinehart and Winston Inc., New York.

Andreou, A. G., Boahen, K. A., O., P. P., Pavasovic, A., Jenkins, R. E., and Strohbehn, K. (1991). Current-mode subthreshold MOS circuits for analog VLSI neural systems. *IEEE Transactions on Neural Networks*, 2(2):205–213.

Andreou, A. G., Meitzler, R. C., Strohbehn, K., and Boahen, K. A. (1995). Analog VLSI neuromorphic image acquisition and pre-processing systems. *Neural Networks*, 8(7/8):1323–1347.

Arreguit, X. (1989). *Compatible Lateral Bipolar Transistors in CMOS Technology: Model and Applications.* PhD thesis, Ecole Polytechnique Federale de Lausanne.

Bachelder, I. A., Waxman, A. M., and Seibert, M. (1993). A neural system for mobile robot visual place learning and recognition. *Proceedings of the World Congress on Neural Networks (WCNN'93)*, pages 512–517.

Baloch, A. A. and Waxman, A. M. (1991). Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN. *Neural Networks*, 4:271–302.

Baraldi, A. and Parmiggiani, F. (1995). A neural network for unsupervised categorization of multivalued input patterns, an application of satellite image clustering. *IEEE Transactions on Geoscience and Remote Sensing*, 33:305–316.

Bastos, J., Steyaert, M., Pergoot, A., and Sansen, W. (1997). Mismatch characterization of submicron MOS transistors. *Analog Integrated Circuits and Signal Processing*, 12:95–106.

Bastos, J., Steyaert, M., Roovers, R., Kinger, P., Sansen, W., Graindourze, B., Pergoot, A., and Janssens, E. (1995). Mismatch characterization of small size

MOS transistors. *Proc. IEEE 1995 Int. Conf. Microelectronic Test Structures*, 8:271–276.

Bernardon, A. M. and Carrick, J. E. (1995). A neural system for automatic target learning and recognition applied to bare and camouflaged SAR targets. *Neural Networks*, 8:1103–1108.

Bezdec, J. C. and Pal, S. K. (1992). *Fuzzy Models for Pattern Recognition.* IEEE Press.

Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum, New York.

Bult, K. and Geelen, G. J. G. M. (1991). The CMOS gain–boosting technique. *Analog Integrated Circuits and Signal Processing*, 1:99–135.

Bult, K. and Wallinga, H. (1987). A class of analog CMOS circuits based on the square–law characteristic of a MOS transistor in saturation. *IEEE Journal of Solid-State Circuits*, SC–22:357–365.

Carpenter, G. A. (1997). Distributed learning, recognition, and prediction by ART and ARTMAP neural networks. *Neural Networks*, 10(8):1473–1494.

Carpenter, G. A. and Gjaja, M. N. (1994). Fuzzy-ART choice functions. *Proceedings of the 1994 World Congress on Neural Networks (WCNN'94)*, 1:713–722.

Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self–organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115.

Carpenter, G. A. and Grossberg, S. (1991a). *Pattern Recognition by Self Organizing Neural Networks.* MIT Press, Cambridge, MA.

Carpenter, G. A., Grossberg, S., Markuzon, N., and Reynolds, J. H. (1992). Fuzzy-ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713.

Carpenter, G. A., Grossberg, S., and Reynolds, J. H. (1991b). ARTMAP: Supervised real–time learning and classification of nonstationary data by a self–organizing neural network. *Neural Networks*, 4:759–771.

Carpenter, G. A., Grossberg, S., and Rosen, D. B. (1991c). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771.

Caudell, T. P. and Healy, M. J. (1994a). Adaptive resonance theory networks in the encephalon autonomous vision system. *Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN'94)*, pages 1235–1240.

Caudell, T. P., Smith, S. D. G., Escobedo, S. D. G., and Anderson, M. (1994b). NIRS: Large scale ART1 neural architectures for engineering design retrieval. *Neural Networks*, 7:1339–1350.

Cauwenberghs, G. (1995a). A micropower CMOS algorithmic A/D/A converter. *IEEE Transactions on Circuits and Systems, Part I*, 42(11):913–919.

Cauwenberghs, G. (1997). Analog VLSI stochastic perturbative learning architectures. *International Journal of Analog Integrated Circuits and Signal Processing*, 13(1/2):195–209.

Cauwenberghs, g. and Pedroni, V. (1995b). A charge-based CMOS parallel analog vector quantizer. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, pages 779–786. MIT Press, Cambridge, MA.

Cauwenberghs, G. and Yariv, A. (1992). Fault-tolerant dynamic multi-level storage in analog VLSI. *IEEE Transactions on Circuits and Systems, Part II*, 41(12):827–829.

Choi, J. and Sheu, B. J. (1993). A high–precision VLSI winner–take–all circuit for self organizing neural networks. *IEEE Journal of Solid–State Circuits*, 28.

Christodoulou, C. G., Huang, J., Georgiopoulos, M., and Liou, J. J. (1995). Design of gratings and frequency selective surfaces using Fuzzy ARTMAP neural networks. *Journal of Electromagnetic Waves and Applications*, 9:17–36.

Chu, L. C. (1991). Fault–tolerant model of neural computing. *IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pages 122–125.

Cohen, M., Abshire, P., and Cauwenberghs, G. (1998). Mixed-mode VLSI implementation of fuzzy ART. *Proceedings of the 1998 IEEE International Symposium on Circuits and systems (ISCAS'98).*

Degrauwe, M. G., Rijmenants, J., Vittoz, E. A., and De Man, H. J. (1982). Adaptive biasing CMOS amplifiers. *IEEE Journal of Solid–State Circuits*, SC–17:522–528.

Dubes, R. and Jain, A. (1988). *Algorithms that Cluster Data.* Prentice Hall, Englewood Cliffs, NJ.

Dubrawski, A. and Crowley, J. L. (1994). Learning locomotion reflexes: A self-supervised neural system for a mobile robot. *Robotics and Autonomous Systems*, 12:133–142.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* Wiley, New York.

Elias, S. A. and Grossberg, S. (1975). Pattern formation, contrast control, and oscillations in the short term memory of shunting on–center off–surround networks. *Biological Cybernetics*, 20:69–98.

Fahlman, S. E. (1989). Faster-learning variations on back–propagation: An empirical study. In Touretzky, D. S., Hinton, H., and Sejnowski, T., editors, *Proc. 1988 Connectionist Summer School*, pages 38–51. Morgan–Kaufmann, San Mateo, CA.

Gan, K. W. and Lua, K. T. (1992). Chinese character classification using adaptive resonance network. *Pattern Recognition*, 25:877–888.

Gersho, A. and Gray, R. M. (1992). *Vector Quantization and Signal Compression.* Kluwer, Norwell, MA.

Gilbert, B. (1975). Translinear circuits: A proposed classification. *Electronics Letters*, 11(1):14–16.

Gilbert, B. (1990a). Current–mode circuits from a translinear viewpoint: A tutorial in analog IC design: The current–mode approach. *IEE Circuits and Systems Series 2*, 2:11–91.

Gilbert, B. (1990b). Current-mode circuits from a translinear viewpoint, a tutorial. In Toumazou, C., Lidgey, F. J., and G., H. D., editors, *In Analogue IC Design: the current-mode approach*, pages 11–91. IEE Circuits and Systems Series 2.

Gjerdingen, R. O. (1990). Categorization of musical patterns by self-organizing neuronlike networks. *Music Perception*, 7:339–370.

Gopal, S., Sklarew, D. M., and Lambin, E. (1994). Fuzzy neural networks in multi-temporal classification of landcover change in the sahel. *Proceedings of the DOSES Workshop on New Tools for Spatial Analysis*, pages 55–68.

Grasmann, U. (1997). Process and apparatus for monitoring a vehicle interior. *US Patent number 5,680,096.*

Gregor, R. W. (1992). On the relationship between topography and transistor matching in an analog CMOS technology. *IEEE Transactions on Electron Devices*, 39(2):275–282.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134.

Grossberg, S. (1980). How does the brain build a cognitive code? *Psychological Review*, 87:151.

Ham, F. M. and Cohen, G. M. (1996a). Determination of concentrations of biological substances using raman spectroscopy and artificial neural network discriminator. *US Patent number 5,553,616.*

Ham, F. M. and Han, S. (1996b). Classification of cardiac arrhythmias using Fuzzy ARTMAP. *IEEE Transactions on Biomedical Engineering*, 43(4):425–430.

Hartigan, J. (1975). *Clustering Algorithms.* Wiley, New York.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* IEEE Press and Macmillan College Publishing Co.

He, Y., Ciringiroglu, U., and Sánchez-Sinencio, E. (1993). A high-density and low-power charge-based hamming network. *IEEE Transactions on VLSI Systems*, 1:56–62.

Himmelbauer, W., Furth, P. M., Pouliquen, P. O., and Andreou, A. G. (1996). Log-domain filters in subthreshold MOS. Technical Report 96-03, The Johns Hopkins University, Electrical and Computer Engineering Dept.

Hochet, B., Peiris, V., Abdot, S., and Declercq, M. J. (1991). Implementation of a learning kohonen neuron based on a new multilevel storage technique. *IEEE Journal of Solid State Circuits*, 26:262–267.

Jones, S., Sammut, K., Nielsen, C., and Staunstrup, J. (1991). Toroidal neural network: Architecture and processor granularity issues. In Ramacher, U. and Rueckert, U., editors, *VLSI Design of Neural Networks*, pages 229–254. Kluwer Academic Publishers, Dordrecht, Netherlands.

Kalkunte, S. S., Kumar, J. M., and Patnaik, L. M. (1992). A neural network approach for high resolution fault diagnosis in digital circuits. *Proceedings of the International Joint Conference on Neural Networks*, pages 83–88.

Kasperkiewicz, J., Racz, J., and Dubrawski, A. (1995). HPC strength prediction using artificial neural network. *Journal of Computing in Civil Engineering*, 9:279–284.

Keulen, E., Colak, S., Withagen, H., and Hegt, H. (1994). Neural network hardware performance criteria. *Proceedings of the 1994 Int. Conf. on Neural Networks (ICNN'94)*, pages 1885–1888.

Kim, J. H., Lursinsap, C., and Park, S. (1991). Fault-tolerant artificial neural networks. *Int. Joint Conf. on Neural Networks (IJCNN'91)*, 2:951.

Kim, J. W., Jung, K. C., Kim, S. K., and Kim, H. J. (1995). Shape classification of on-line chinese character strokes using ART1 neural network. *Proceedings of the World Congress on Neural Networks (WCNN'95)*, pages 191–194.

Koch, M. W., Moya, M. M., Hostetler, L. D., and Fogler, R. J. (1995). Cueing, feature discovery, and one-class learning for synthetic aperture radar automatic target recognition. *Neural Networks*, 8:1081–1102.

Kohonen, T. (1989). *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, Germany.

Kosko, B. (1987). Adaptive bidirectional associative memories. *Applied Optics*, 26:4947–4960.

Kumar, N., Himmelbauer, W., Cauwenberghs, G., and Andreou, A. G. (1997). An analog VLSI chip with asynchronous interface for auditory feature extraction. *Proceedings of the 1997 IEEE International Conference on Circuits and Systems*, 1:553–556.

Lakshmikumar, K. R., Hadaway, R. A., and Copeland, M. A. (1986). Characterization and modeling of mismatch in MOS transistors for precision analog design. *IEEE Journal of Solid-State Circuits*, SC–21(6):1057–1066.

Lang, K. J. and Wittbrock, M. J. (1989). Learning to tell two spirals apart. In Touretzky, D. S., Hinton, H., and Sejnowski, T., editors, *Proc. 1988 Connectionist Summer School*, pages 52–59. Morgan–Kaufmann, San Mateo, CA.

Lazzaro, J., Ryckebush, R., Mahowald, M. A., and Mead, C. (1989). Winner–take–all networks of O(n) complexity. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, volume 1, pages 703–711. Morgan–Kaufmann, Los Altos, CA.

Lin, D., Dicarlo, L. A., and Jenkins, J. M. (1988). Identification of ventricular tachycardia using intracavitary electrograms: analysis of time and frequency domain patterns. *Pacing and Clinical Electrophysiology*, 11:1592–1606.

Linares-Barranco, B., Sánchez-Sinencio, E., Rodríguez-Vázquez, A., and Huertas, J. L. (1992). A modular T-mode design approach for analog neural network hardware implementations. *IEEE Journal of Solid–State Circuits*, 27(5):701–713.

Linares-Barranco, B., Sánchez-Sinencio, E., Rodríguez-Vázquez, A., and Huertas, J. L. (1993). A CMOS analog adaptive BAM with on–chip learning and weight refreshing. *IEEE Transactions on Neural Networks*, 4(3):445–455.

Lubkin, J. and Cauwenberghs, G. (1998). A micropower learning vector quantizer for parallel analog-to-digital data compression. *Proceedings of the 1998 IEEE International Symposium on Circuits and systems (ISCAS'98)*.

Ly, S. and Choi, J. J. (1994). Drill conditioning monitoring using ART1. *Proceedings of the IEEE International Conference on Neural Networks (ICNN'94)*, pages 1226–1229.

Lyon, R. F. and Schediwy, R. R. (1987). CMOS static memory with a new four–transistor memory cell. In Losleben, P., editor, *Advanced Research in VLSI*, pages 110–132. MIT Press, Cambridge, MA.

Mauduit, N., Duranton, M., Gobert, J., and Sirat, J. A. (1992). Lneuro 1.0: A piece of hardware lego for building neural network systems. *IEEE Transactions on Neural Networks*, 3(3):414–422.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison Wesley, Reading, MA.

Medeiro, F., Rodríguez-Macías, R., Fernández, F. V., Domínguez-Castro, R., and Rodríguez-Vázquez, A. (1994). Global analogue cell design and optimization using statistical techniques. *Analog Integrated Circuits and Signal Processing*, 6:179–195.

Mehta, B. V., Vij, L., and Rabelo, L. C. (1993). Prediction of secondary structures of proteins using Fuzzy ARTMAP. *Proceedings of the World Congress on Neural Networks (WCNN'93)*, pages 228–232.

Michael, C. and Ismail, M. (1992). Statistical modeling of device mismatch for analog MOS integrated circuits. *IEEE Journal of Solid-State Circuits*, 27(2):154–166.

Moore, B. (1989). ART1 and pattern clustering. In Touretzky, D. S., Hinton, H., and Sejnowski, T., editors, *Proc. 1988 Connectionist Summer School*, pages 174–185. Morgan–Kaufmann, San Mateo, CA.

Murshed, N. A., Bortozzi, F., and Sabourin, R. (1995). Off-line signature verification, without a priori knowledge of class $\omega 2$. A new approach. *Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95).*

Nairn, D. G. and Salama, C. A. T. (1990a). Current–mode algorithmic analog–to–digital converters. *IEEE Journal of Solid-State Circuits*, 25:997–1004.

Nairn, D. G. and Salama, C. A. T. (1990b). A ratio–independent algorithmic analog–to–digital converter combining current mode and dynamic techniques. *IEEE Transactions on Circuits and Systems*, 37:319–325.

Neti, C., Schneider, M. H., and Young, E. D. (1992). Maximally fault tolerant neural networks. *IEEE Transactions on Neural Networks*, 3(1):14–23.

Pan, T. W. and Abidi, A. A. (1989). A 50-dB variable gain amplifier using parasitic bipolar transistors in CMOS. *IEEE Journal of Solid-State Circuits*, 24(4):951–961.

Pao, Y. H. (1989). *Adaptive Recognition and Neural Networks.* Addison Wesley, Reading, MA.

Pelgrom, M. J. M., Duinmaijer, A. C. J., and Welbers, A. P. G. (1989). Matching properties of MOS transistors. *IEEE Journal of Solid-State Circuits*, 24:1433–1440.

Pouliquen, P. O., Andreou, A. G., and Strohbehn, K. (1997). Winner–takes–all associative memory: A hamming distance vector quantizer. *Journal of Analog Integrated Circuits and Signal Processing*, 13(1/2):211–222.

Rabiner, L. and Juang, B. H. (1993). *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ.

Racz, J. and Dubrawski, A. (1995). Artificial neural network for mobile robot topological localization. *Robotics and Autonomous Systems*, 16:73–80.

Rade, L. and Westergren, B. (1990). *BETA Mathematics Handbook.* CRC Press.

Ramacher, U., Beichter, J., Raab, W., Anlauf, J., Bruels, N., Hachmann, U., and Wesseling, M. (1991). Design of a 1st generation neurocomputer. In Ramacher, U. and Rueckert, U., editors, *VLSI Design of Neural Networks*, pages 271–310. Kluwer Academic Publishers, Dordrecht, Netherlands.

Riedmiller, M. (1994). Advanced supervised learning in multi–layer perceptrons - from backpropagation to adaptive learning algorithms. *Int. Journal of Computer Standards and Interfaces*, 5.

Rodríguez-Vázquez, A., Domínguez-Castro, R., Medeiro, F., and Delgado-Resti-tuto, M. (1995). High resolution CMOS current comparators: Design and application to current–mode function generation. *International Journal of Analog Integrated Circuits and Signal Processing*, 7:149–165.

Rodríguez-Vázquez, A., Espejo, S., Domínguez-Castro, R., Huertas, J. L., and Sánchez-Sinencio, E. (1993). Current–mode techniques for the implementation of continous- and discrete–time cellular neural networks. *IEEE Transactions on Circuits and Systems II*, 40(3):132–146.

Roulston, D. J. (1990). *Bipolar Semiconductor Devices.* McGraw–Hill, New York.

Sackinger, D. and Guggenbuhl, W. (1990). A high–swing, high–impedance MOS cascode circuit. *IEEE Journal of Solid–State Circuits*, 25:289–298.

Sánchez-Sinencio, E. and Lau, C. (1992). *Artificial Neural Networks: Paradigms, Applications, and Hardware Implementations.* IEEE Press.

Sánchez-Sinencio, E., Ramírez-Angulo, J., Linares-Barranco, B., and Rodríguez-Vázquez, A. (1989). Operational transconductance amplifier–based nonlinear function syntheses. *IEEE Journal of Solid–State Circuits*, 24:1576–1586.

Schlimmer, J. S. (1987). *Concept Adquisition through Representational Adjustment (Technical Report 87–19).* PhD thesis, Department of Information and Computer Sciences, University of California at Irvine.

Seevinck, E. and Wiegerink, J. (1991). Generalized translinear circuit principle. *IEEE Journal of Solid–State Circuits*, SC–26(8):1198–1102.

Seibert, M. and Waxman, A. M. (1992). Adaptive 3D object recognition from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:107–124.

Seibert, M. and Waxman, A. M. (1993). An approach to face recognition using saliency maps and caricatures. *Proceedings of the World Congress on Neural Networks (WCNN'93)*, pages 661–664.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1994). The active–input regulated–cascode current mirror. *IEEE Transactions on Circuits and Systems, Part I: Fundamental Theory and Applications*, 41(6):464–467.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1995). A modular current–mode high–precision winner–take–all circuit. *IEEE Transactions on Circuits and Systems, Part II*, 42(2):132–134.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1996a). A modified ART1 algorithm more suitable for VLSI implementations. *Neural Networks*, 9(6):1025–1043.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1996b). A real–time clustering microchip neural engine. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(2):195–209.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1997). An ART1 microchip and its use in multi-ART1 systems. *IEEE Transactions on Neural Networks*, 8(5):1184–1194.

Serrano-Gotarredona, T. and Linares-Barranco, B. (1998a). A high–precision current–mode WTA–MAX circuit with multi–chip capability. *IEEE Journal of Solid–State Circuits*, 33(2):280–286.

Serrano-Gotarredona, T., Linares-Barranco, B., and Andreou, A. G. (1998b). Voltage clamping current mirrors with 13-decades gain adjustment range suitable for low power MOS/bipolar current mode signal processing circuits. *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98)*.

Sheingold, D. H. (1976). *Nonlinear Circuits Handbook.* Analog Devices Inc., Norwood, MA.

Shyu, J. B., Temes, G. C., and Krummenacher, F. (1984). Random error effects in matched MOS capacitors and current sources. *IEEE Journal of Solid–State Circuits*, SC–19(6):948–955.

Soliz, P. and Donohoe, G. W. (1996). Adaptive resonance theory neural network for fundus image segmentation. *Proceedings of the 1996 World Congress on Neural Networks (WCNN'96)*, pages 1180–1183.

Srinivasa, N. and Sharma, R. (1996). A self-organizing invertible map for active vision applications. *Proceedings of the World Congress on Neural Networks (WCNN'96)*, pages 121–124.

Strader, N. R. and Harden, J. C. (1989). Architectural yield optimization. In Schwartzlander, E. E., editor, *Wafer Scale Integration*, pages 57–118. Kluwer Academic Publishers, Boston.

Tarng, Y. S., Li, T. C., and Chen, M. C. (1994). Tool failure monitoring for drilling processes. *Proceedings of the Third International Conference on Fuzzy Logic, Neural Nets and Soft Computing, Iizuka, Japan*, pages 109–111.

Tse, P. and Wang, D. D. (1996). A hybrid neural networks based machine condition forecaster and classifier by using multiple vibration parameters. *Proceedings of the 1994 IEEE International Conference on Neural Networks*, pages 2096–2100.

Tsividis, Y. (1988). *Operation and Modeling of the MOS Transistor.* McGraw–Hill, New York.

Wand, Y. F., Cruz, J. B., and Mulligan, J. H. (1990). On multiple training for bidirectional associative memory. *IEEE Transactions on Neural Networks*, 1:275–276.

Waxman, A. M., Seibert, M. C., Gove, A., Fay, D. A., Bernardon, A. M., Lazott, C., Steele, W. R., and Cunningham, R. K. (1995). Neural processing of targets in visible, multispectral IR and SAR imagery. *Neural Networks*, 8:1029–1051.

Wienke, D. (1994). Neural resonance and adaptation - towards nature's principles in artificial pattern recognition. In Buydens, L. and Melssen, W., editors, *Chemometrics: Exploring and Exploting Chemical Information.* University Press, Nijmegen, NL.

Yuille, A. L. and Geiger, D. (1995). Winner–take–all mechanisms. In Arbib., M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 1056–1060. MIT Press.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Index