디지털통신시스템

기말 프로젝트

융합전자공학과 2024247094 조한승

1. 63 비트 길이의 PN 부호 생성과 run, balance, shift and add 특성
2. Preferred sequence 2 개를 이용하여 63 비트 길이의 Gold 부호 2 개 발생
3. 발생된 Gold 부호의 auto-correlation 과 cross-correlation 특성
4. 발생된 Gold 부호에 "0" 비트를 하나 더한 부호의 auto-correlation 과 cross-correlation 특성

```
clc; clear;
SR=[1 1 1 1 1 1];
%Polynomial=x^6+x+1 PNcode 생성
PN(1)=SR(6);
for i=1:2^length(SR)-2
   mem=mod(SR(6)+SR(1),2);
   SR=circshift(SR,1);
   SR(1)=mem;
   PN(i+1)=SR(6);
end
PNcode=2*PN-1;
shift=0:128;
for i=1:length(shift)
    autocorr(i)=sum(PNcode.*circshift(PNcode,shift(i)));
end
figure(1)
stem(shift,autocorr/length(PNcode)) %PNcode 길이로 정규화
title("PNcode Autocorrelation")
%run property
l=1;
run=zeros(1,length(SR));
PNcode(length(PNcode)+1)=2;
for i=2:length(PN)+1
    if PNcode(i) == PNcode(i-1)
        l=l+1;
    else
        run(l)=run(l)+1;
        l=1;
    end
end
PNcode=PNcode(1:length(PNcode)-1);
%balance property
num1=sum(PN==1);
num0=sum(PN==0);
figure(2)
subplot(2,1,1)
stem(0:1,[num0 num1])
title("PNcode Balance property")
xlim([-1 2])
subplot(2,1,2)
stem(run/sum(run))
title("PNcode Run property")
%shift-and-add property (shift=3)
shifted_PN=circshift(PN,3);
shifted_add_PN=bitxor(PN,shifted_PN);
for i=0:length(PN)
    mem=circshift(shifted_add_PN,i);
    if mem == PN
        same_shift=i;
    end
```

```
end
figure(3)
subplot(3,1,1)
stem(PN)
title("Original PN code")
subplot(3,1,2)
stem(shifted_add_PN)
title("Shift-and-add PN code(shift=3)")
subplot(3,1,3)
stem(circshift(shifted_add_PN,same_shift));
title("Shift-and-add PN code(shift=3) with 31 shift")

%Gold code Simulation
pref_SR=[1 1 1 1 1 1];
%Polynomial=x^6+x^5+x^2+x+1 PNcode 생성

pref_PN(1)=pref_SR(6);
for i=1:2^length(pref_SR)-2
   mem=mod(pref_SR(6)+pref_SR(5)+pref_SR(2)+pref_SR(1),2);
   pref_SR=circshift(pref_SR,1);
   pref_SR(1)=mem;
   pref_PN(i+1)=pref_SR(6);
end
pref_PNcode=2*pref_PN-1;
for i=1:length(PNcode)+1
    pref_autocorr(i)=sum(PNcode.*circshift(pref_PNcode,i-1));
end
figure(4)
stem(0:length(pref_autocorr)-1,pref_autocorr/length(pref_PN)) %PNcode 길이로 정규화
title("preferred PN code Autocorrelation(x^6+x+1, x^6+x^5+x^2+x+1),
Crosscorrelation(-1/63(-0.015), -17/63(-0.269), 15/63(0.238))")

Gold(1,:)=PN;
Gold(2,:)=pref_PN;
for i=1:length(PNcode)
    Gold(i+2,:)=mod(PN+circshift(pref_PN,i-1),2);
end
Goldcode=2*Gold-1;

for i=1:length(PNcode)
    Gold_autocorr(i)=sum(Goldcode(6,:).*circshift(Goldcode(6,:),i-1));
end
figure(5)
stem(0:length(Gold_autocorr)-1,Gold_autocorr/length(Gold_autocorr)) %PNcode 길이로
정규화
title("Gold code Autocorrelation (-1/63(-0.015), -17/63(-0.269), 15/63(0.238),
63/63(1))")

for i=1:length(PNcode)
    Gold_crosscorr(i)=sum(Goldcode(6,:).*circshift(Goldcode(10,:),i-1));
end
figure(6)
stem(0:length(Gold_crosscorr)-1,Gold_crosscorr/length(Gold_crosscorr)) %PNcode
길이로 정규화
title("Gold code Crosscorrelation (-1/63(-0.015), -17/63(-0.269), 15/63(0.238))")

%Gold code added with 0
```

```
for i=1:length(PNcode)
    Gold_added0_autocorr(i)=sum([Goldcode(6,:) 0].*circshift([Goldcode(6,:) 0],i-
1));
end
figure(7)
stem(0:length(Gold_added0_autocorr)-
1,Gold_added0_autocorr/length(Gold_added0_autocorr)) %PNcode 길이로 정규화
title("Gold code added with 0 Autocorrelation (-1/63(-0.015), -17/63(-0.269),
15/63(0.238), 63/63(1))")

for i=1:length(PNcode)
    Gold_added0_crosscorr(i)=sum([Goldcode(6,:) 0].*circshift([Goldcode(10,:)
0],i-1));
end
figure(8)
stem(0:length(Gold_added0_crosscorr)-
1,Gold_added0_crosscorr/length(Gold_added0_crosscorr)) %PNcode 길이로 정규화
title("Gold code added with 0 Crosscorrelation (-1/63(-0.015), -17/63(-0.269),
15/63(0.238))")
```

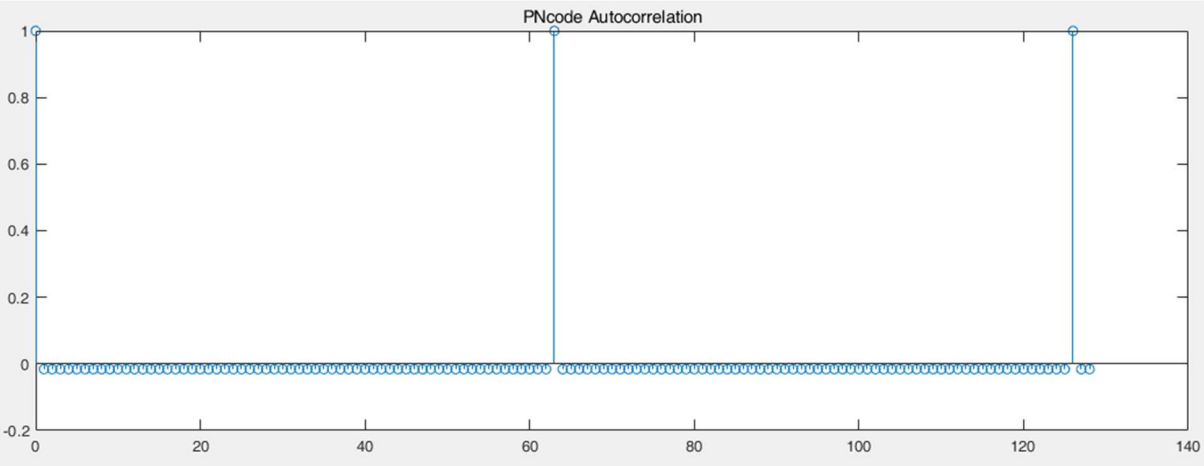

그림 1. PN code의 Autocorrelation 특성

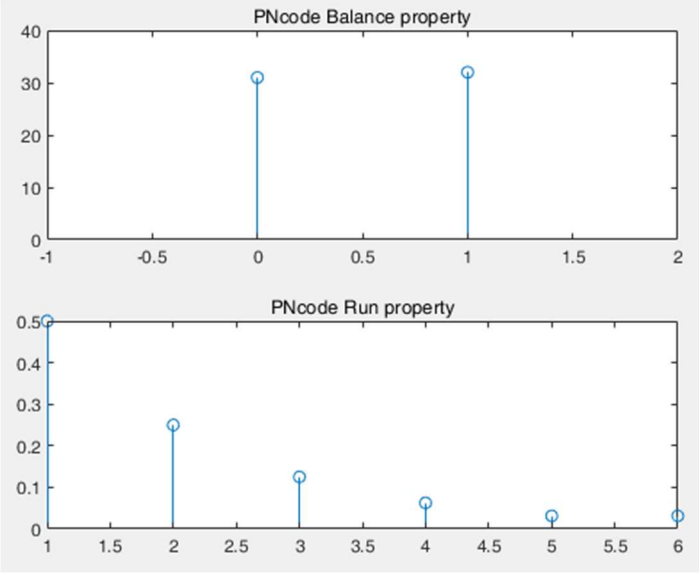

그림 2. PN code의 Balance와 Run 특성

Balance property의 경우 1은 32개 0은 31개로 나타났으며 이론적인 값과 일치했다. Run property의 경우에도 0.5 0.25 0.125 0.0625 0.0313 0.0313으로 run의 값이 커짐에 따라 빈도가 1/2배가 되는 모습을 확인할 수 있었다.

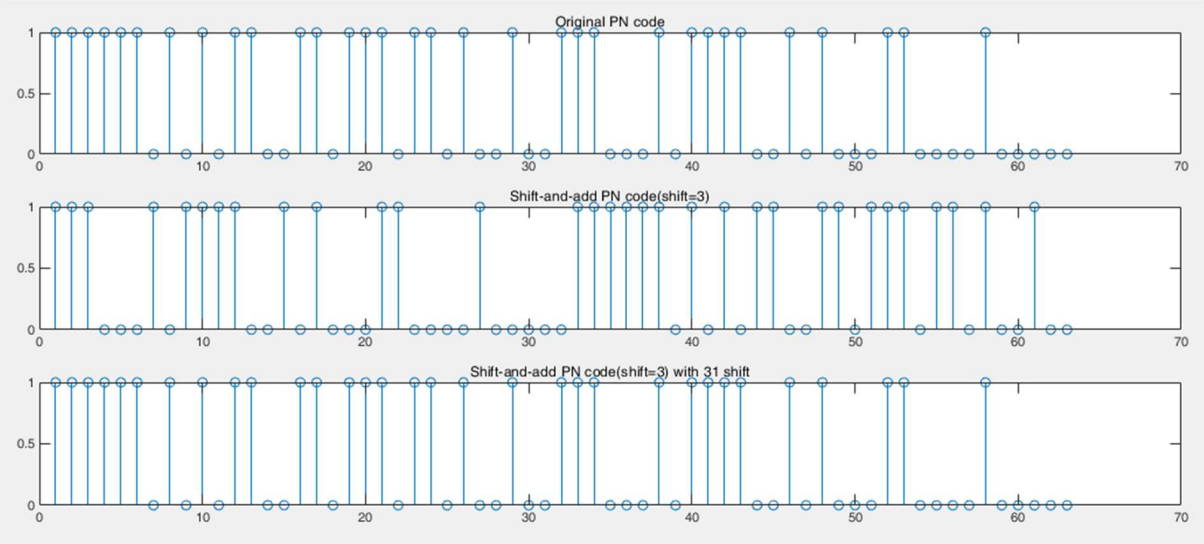

그림 3. PN code의 Shift-and-add(shift=3) 특성

Original PN code에 3만큼 shift를 한 code를 생성한 후 original PN code과 XOR연산을 진행했다. 그 결과 동일한 phase만 다른 동일한 PN code를 얻을 수 있었고 31 shift를 할 경우 Original PN code와 일치함을 확인했다.

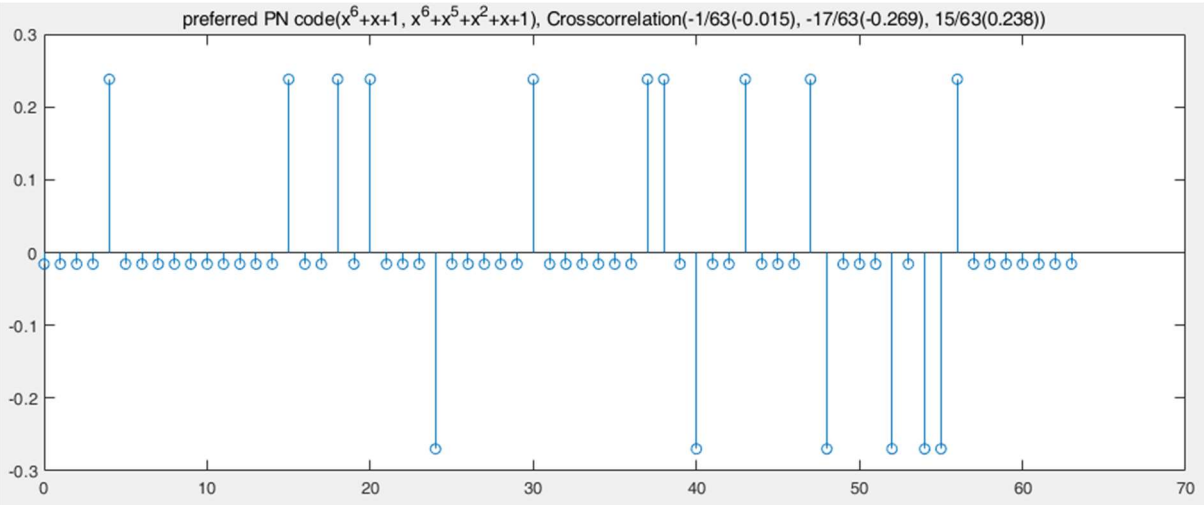

그림 4. Preferred sequence (x^6+x+1, $x^6+x^5+x^2+x+1$)의 cross-correlation

여러 Polynomial을 조합하며 preferred sequence를 찾았고 x^6+x+1과 $x^6+x^5+x^2+x+1$일 경우에 preferred sequence임을 확인했다. 위 그림에서 cross-correlation 값이 -1/63, -17/63, 15/63 세가지 값으로 이루어진 것을 확인할 수 있다.

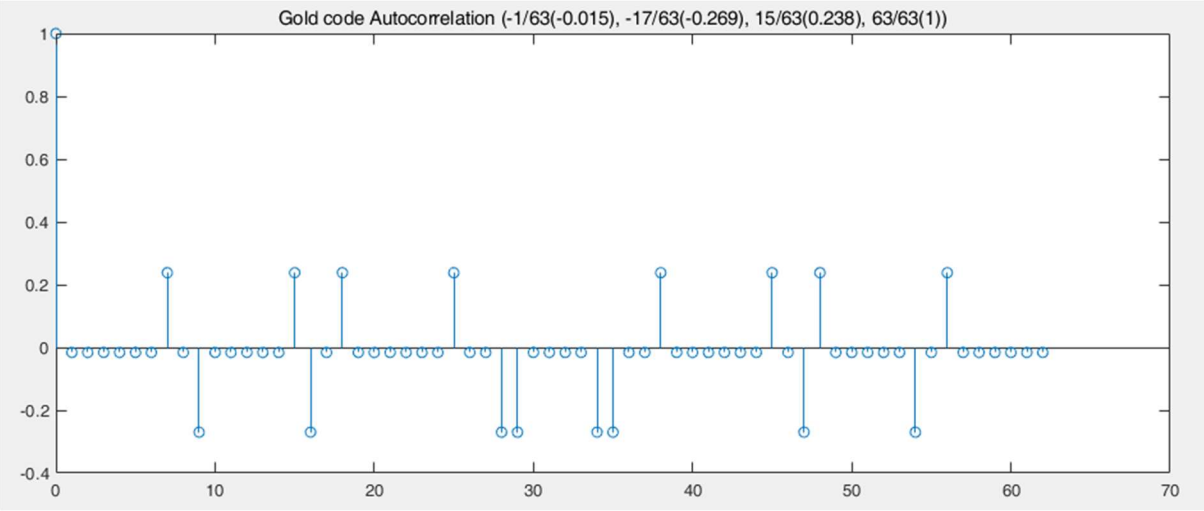

그림 5. Gold code의 auto-correlation 특성

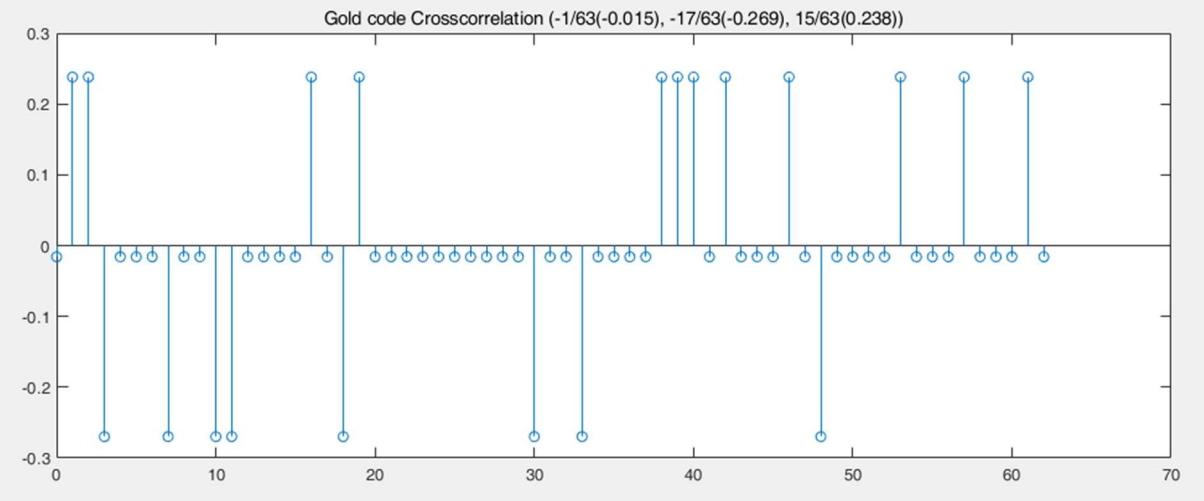

그림 6. Gold code의 cross-correlation 특성

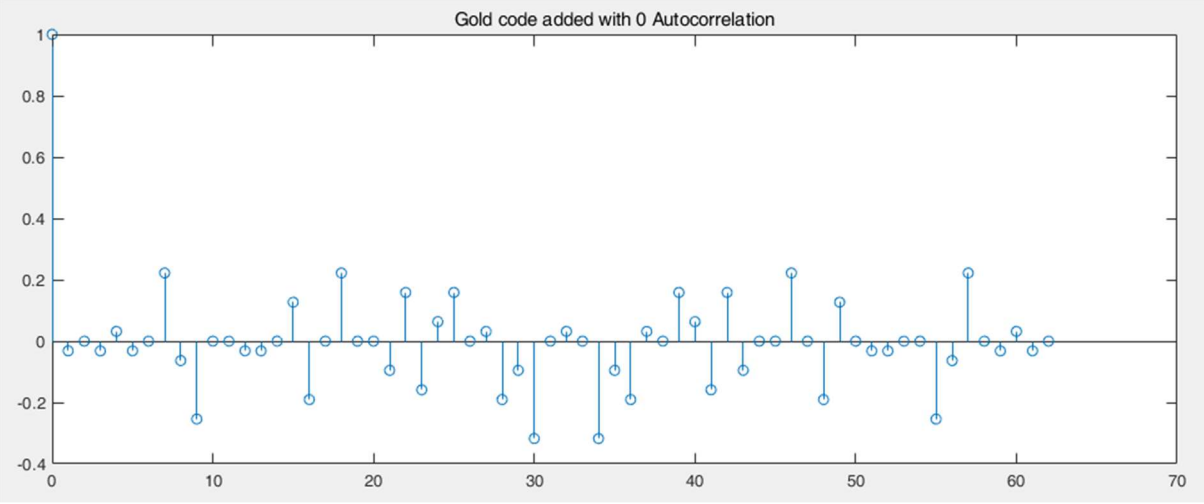

그림 7. 비트 "0"이 추가된 Gold code의 auto-correlation 특성

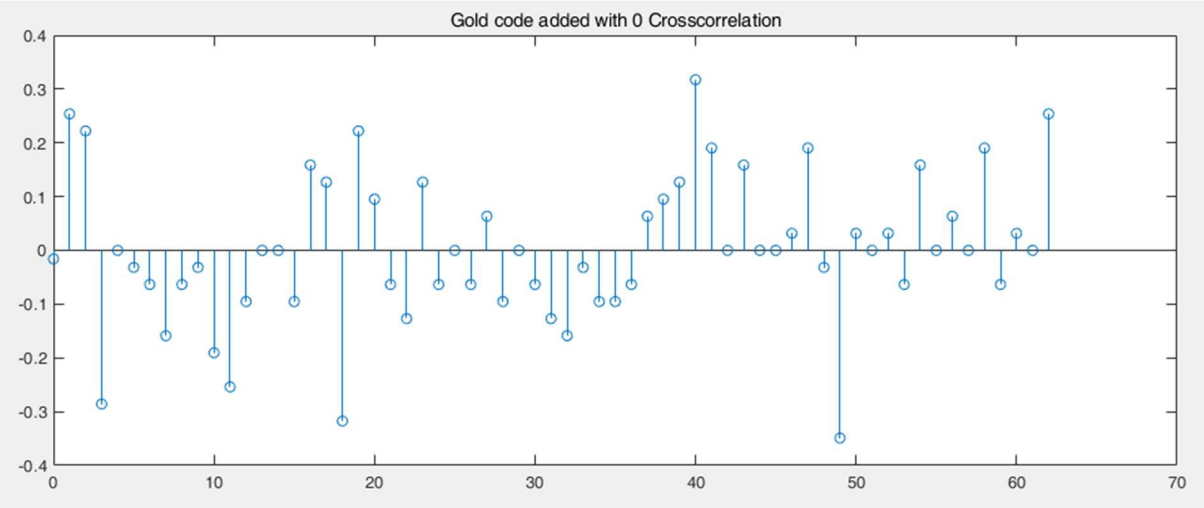

그림 8. 비트 "0"이 추가된 Gold code의 cross-correlation 특성

Gold코드의 auto-correlation은 shift가 0일 때 1인 것을 제외하고 모든 지점에서 -1/63, -17/63, 15/63 세가지 값을 가지는 것을 확인했다. Cross-correlation의 경우 -1/63, -17/63, 15/63 세가지 값만 나타나는 것을 확인했다. 비트 "0"이 추가된 경우 Gold code의 Gold code의 correlation 특성이 나타나지 않음을 확인했다.

1. 43,890 비트길이의 JPL ranging sequence와 K5 sequence를 생성하고 auto-correlation과 transitions per chip, dc bias를 조사 및 비교

2. IS-95 CDMA의 short PN 부호(32,768 비트)의 auto-correlation과 transitions per chip, dc bias를 조사 및 비교

```
clc; clear; close all
%JPL & K5 generation
C1=[1 -1];
C2=[1 1 1 -1 -1 1 -1];
C3=[1 1 1 -1 -1 -1 1 -1 1 1 -1];
C4=[1 1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1];
C5=[1 1 1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 -1 -1];
D1=(C1+1)/2;
D2=(C2+1)/2;
D3=(C3+1)/2;
D4=(C4+1)/2;
D5=(C5+1)/2;

for i=1:43890
    if (D1(1)+D2(1)+D3(1)+D4(1)+D5(1)) >= 3
        JPL_D(i)=1;
    else
        JPL_D(i)=0;
    end
```

```
    D1=circshift(D1,-1);
    D2=circshift(D2,-1);
    D3=circshift(D3,-1);
    D4=circshift(D4,-1);
    D5=circshift(D5,-1);
end
JPL_C=2*JPL_D-1;
for i=1:43890
    K5_C(i)=sign(3*C1(1)-C2(1)+C3(1)+C4(1)-C5(1));
    C1=circshift(C1,-1);
    C2=circshift(C2,-1);
    C3=circshift(C3,-1);
    C4=circshift(C4,-1);
    C5=circshift(C5,-1);
end
K5_D=(K5_C+1)/2;
%Autocorrelation
shift=0:43890*2+1;
for i=1:length(shift)
    JPL_autocorr(i)=sum(JPL_C.*circshift(JPL_C,shift(i)));
end
for i=1:length(shift)
    K5_autocorr(i)=sum(K5_C.*circshift(K5_C,shift(i)));
end
figure(1)
subplot(2,1,1)
stem(shift,JPL_autocorr/length(JPL_C)) %PNcode 길이로 정규화
title("JPL Autocorrelation")
subplot(2,1,2)
stem(shift,K5_autocorr/length(K5_C)) %PNcode 길이로 정규화
title("K5 Autocorrelation")

%transition per chip
l=0;
for i=2:length(JPL_D)
    if JPL_D(i) ~= JPL_D(i-1)
        l=l+1;
    else
    end
end
JPL_TPC=l/length(JPL_D);
p=0;
for i=2:length(K5_D)
    if K5_D(i) ~= K5_D(i-1)
        p=p+1;
    else
    end
end
K5_TPC=p/length(K5_D);
%DC bias
JPL_DC=mean(JPL_C);
K5_DC=mean(K5_C);

%IS-95 PN code
%Inphase PN code polynomial: x^15+x^13+x^9+x^8+x^7+x^5+1
SRI=ones(1,15);
PNI(1)=SRI(15);
for i=1:2^length(SRI)-2
```

```
   memI=mod(SRI(15)+SRI(13)+SRI(9)+SRI(8)+SRI(7)+SRI(5),2);
   SRI=circshift(SRI,1);
   SRI(1)=memI;
   PNI(i+1)=SRI(6);
end
PNcodeI=2*PNI-1;
%Quadrature PN code polynomial: x^15+x^12+x^11+x^10+x^6+x^5+x^4+x^3+1
SRQ=ones(1,15);
PNQ(1)=SRQ(15);
for i=1:2^length(SRQ)-2
   memQ=mod(SRQ(15)+SRQ(12)+SRQ(11)+SRQ(10)+SRQ(6)+SRQ(5)+SRQ(4)+SRQ(3),2);
   SRQ=circshift(SRQ,1);
   SRQ(1)=memQ;
   PNQ(i+1)=SRQ(6);
end
PNcodeQ=2*PNQ-1;
%Autocorrelation
shift=0:length(PNcodeI)+1;
for i=1:length(shift)
    PNcodeI_autocorr(i)=sum(PNcodeI.*circshift(PNcodeI,shift(i)));
end
figure(2)
plot(shift, PNcodeI_autocorr/length(PNcodeI_autocorr))

shift=0:length(PNcodeQ)+1;
for i=1:length(shift)
    PNcodeQ_autocorr(i)=sum(PNcodeQ.*circshift(PNcodeQ,shift(i)));
end
figure(3)
plot(shift, PNcodeQ_autocorr/length(PNcodeQ_autocorr))

%transition per chip
l=0;
for i=2:length(PNcodeI)
    if PNcodeI(i) ~= PNcodeI(i-1)
        l=l+1;
    else
    end
end
PNcodeI_TPC=l/length(PNcodeI);
p=0;
for i=2:length(PNcodeQ)
    if PNcodeQ(i) ~= PNcodeQ(i-1)
        p=p+1;
    else
    end
end
PNcodeQ_TPC=p/length(PNcodeQ);
%DC bias
PNcodeI_DC=mean(PNcodeI);
PNcodeQ_DC=mean(PNcodeQ);
```

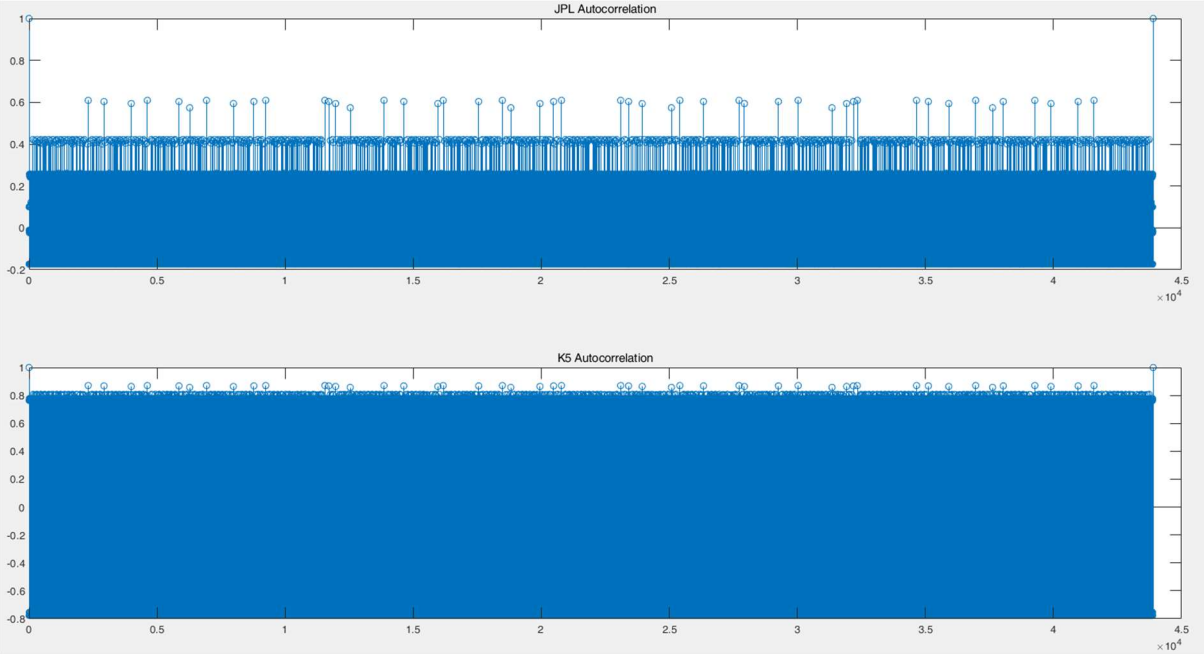

그림 9. JPL과 K5의 auto-correlation 특성

	JPL	K5
Transition per chip	0.5867	0.8881
DC bias	0.1321	0.0047

표 1. JPL과 K5의 transition per chip과 DC bias

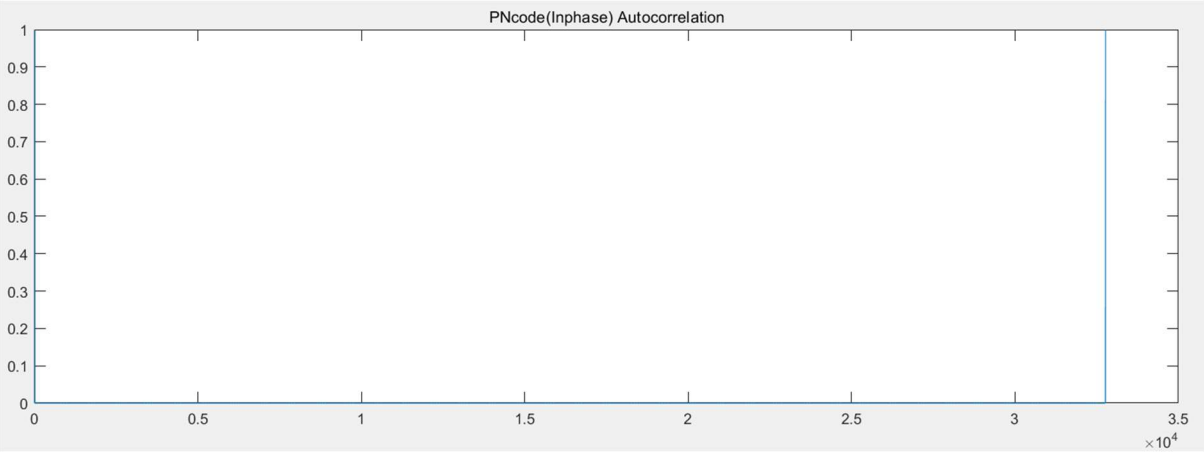

그림 10. IS-95 CDMA short PN code(Inphase)의 auto-correlation 특성

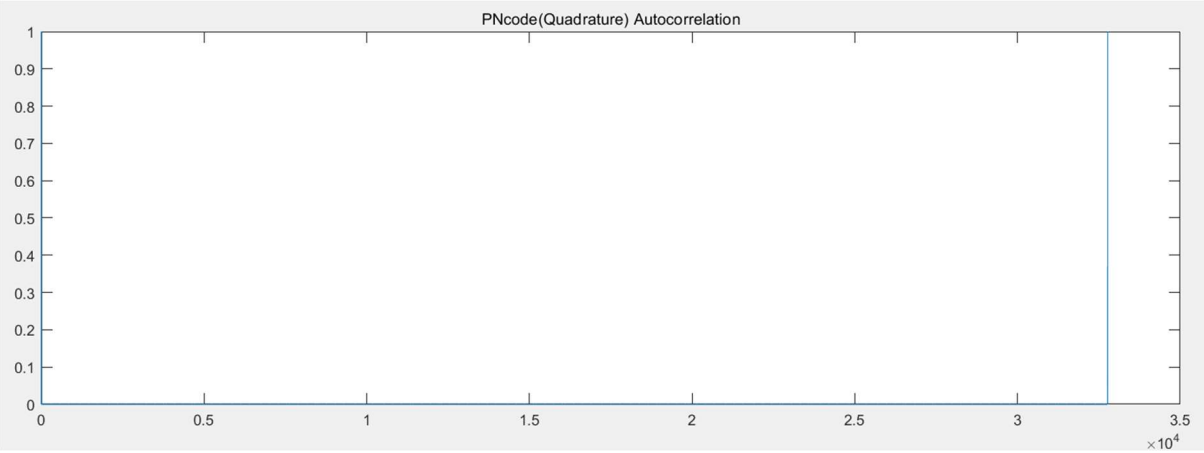

그림 11. IS-95 CDMA short PN code(Quadrature)의 auto-correlation 특성

	IS-95 Inphase	IS-95 Quadrature
Transition per chip	0.5	0.5
DC bias	3.0519e-5	3.0519e-5

표 2. IS-95 CDMA short PN code의 Transition per chip과 DC bias

1. K5와 JPL의 regenerative PN ranging 모의 실험

2. 43,890 비트 길이의 K5와 JPL sequence 생성

3. 임의의 delay를 가정하고 각 component code 별 locking 수행

4. regenerative 수행 후 SNR에 따라 성능 검증

```
%Tc=1/(10^7);
%JPL & K5 generation
clc; clear; close all
C1=[1 -1];
C2=[1 1 1 -1 -1 1 -1];
C3=[1 1 1 -1 -1 -1 1 -1 1 1 -1];
C4=[1 1 1 1 -1 -1 -1 1 -1 -1 1 1 -1 1 -1];
C5=[1 1 1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 -1 -1];
D1=(C1+1)/2;
D2=(C2+1)/2;
D3=(C3+1)/2;
D4=(C4+1)/2;
D5=(C5+1)/2;

for i=1:43890
```

```
    if (D1(1)+D2(1)+D3(1)+D4(1)+D5(1)) >= 3
        JPL_D(i)=1;
    else
        JPL_D(i)=0;
    end
    D1=circshift(D1,-1);
    D2=circshift(D2,-1);
    D3=circshift(D3,-1);
    D4=circshift(D4,-1);
    D5=circshift(D5,-1);
end
JPL_C=2*JPL_D-1;
for i=1:43890
    K5_C(i)=sign(3*C1(1)+C2(1)-C3(1)-C4(1)+C5(1));
    C1=circshift(C1,-1);
    C2=circshift(C2,-1);
    C3=circshift(C3,-1);
    C4=circshift(C4,-1);
    C5=circshift(C5,-1);
end
K5_D=(K5_C+1)/2;
%Local Generated Code
rep_C1=repmat(C1, 1, length(JPL_C)/length(C1));
rep_C2=repmat(C2, 1, length(JPL_C)/length(C2));
rep_C3=repmat(C3, 1, length(JPL_C)/length(C3));
rep_C4=repmat(C4, 1, length(JPL_C)/length(C4));
rep_C5=repmat(C5, 1, length(JPL_C)/length(C5));
snr=-35:-12;

N=10^4;
for m=1:length(snr) %SNR 에 따라 성능 검증
    JPL_detection=0;
    K5_detection=0;
    for n=1:N
        %랜덤한 딜레이를 추가하여 신호 생성
        delay=randi([1,length(JPL_C)]);
        Delayed_JPL_C=circshift(JPL_C, delay);
        Delayed_K5_C=circshift(K5_C, delay);
        Original_JPL_C=Delayed_JPL_C;
        Original_K5_C=Delayed_K5_C;
        %신호에 SNR 에 따른 AWGN 추가
        Delayed_JPL_C=awgn(Delayed_JPL_C, snr(m));
        Delayed_K5_C=awgn(Delayed_K5_C, snr(m));

        %각 branch 별로 Autocorrelation
        for i=1:length(C1)
            JPLC1_autocorr(i)=sum(Delayed_JPL_C.*circshift(rep_C1,i-1));
        end
        for i=1:length(C2)
            JPLC2_autocorr(i)=sum(Delayed_JPL_C.*circshift(rep_C2,i-1));
        end
        for i=1:length(C3)
            JPLC3_autocorr(i)=sum(Delayed_JPL_C.*circshift(rep_C3,i-1));
        end
        for i=1:length(C4)
            JPLC4_autocorr(i)=sum(Delayed_JPL_C.*circshift(rep_C4,i-1));
        end
        for i=1:length(C5)
```

```
            JPLC5_autocorr(i)=sum(Delayed_JPL_C.*circshift(rep_C5,i-1));
        end
        %각 branch 별로 autocorrelation 피크 지점 탐지로 phase 확보
        [v1,index1]=max(JPLC1_autocorr(1:length(C1)));
        [v2,index2]=max(JPLC2_autocorr(1:length(C2)));
        [v3,index3]=max(JPLC3_autocorr(1:length(C3)));
        [v4,index4]=max(JPLC4_autocorr(1:length(C4)));
        [v5,index5]=max(JPLC5_autocorr(1:length(C5)));
        index1=index1-1;%index 에서 1 을 빼서 0 점 보정
        index2=index2-1;
        index3=index3-1;
        index4=index4-1;
        index5=index5-1;
        %앞에서 찾은 phase 만큼 original C1~C5 코드를 딜레이 시킨후 JPL 코드 합성
        Offset_D1=circshift(D1,index1);
        Offset_D2=circshift(D2,index2);
        Offset_D3=circshift(D3,index3);
        Offset_D4=circshift(D4,index4);
        Offset_D5=circshift(D5,index5);
        for i=1:43890
            if (Offset_D1(1)+Offset_D2(1)+Offset_D3(1)+Offset_D4(1)+Offset_D5(1))
>= 3
                Regen_JPL_D(i)=1;
            else
                Regen_JPL_D(i)=0;
            end
            Offset_D1=circshift(Offset_D1,-1);
            Offset_D2=circshift(Offset_D2,-1);
            Offset_D3=circshift(Offset_D3,-1);
            Offset_D4=circshift(Offset_D4,-1);
            Offset_D5=circshift(Offset_D5,-1);
        end
        Regen_JPL_C=2*Regen_JPL_D-1;

        if sum(Original_JPL_C ~= Regen_JPL_C) == 0
            JPL_detection=JPL_detection+1;
        else

        end
        %K5 Autocorrelation
        %각 branch 별로 Autocorrelation
        for i=1:length(C1)
            K5C1_autocorr(i)=sum(Delayed_K5_C.*circshift(rep_C1,i-1));
        end
        for i=1:length(C2)
            K5C2_autocorr(i)=sum(Delayed_K5_C.*circshift(rep_C2,i-1));
        end
        for i=1:length(C3)
            K5C3_autocorr(i)=sum(Delayed_K5_C.*circshift(rep_C3,i-1));
        end
        for i=1:length(C4)
            K5C4_autocorr(i)=sum(Delayed_K5_C.*circshift(rep_C4,i-1));
        end
        for i=1:length(C5)
            K5C5_autocorr(i)=sum(Delayed_K5_C.*circshift(rep_C5,i-1));
        end
        %각 branch 별로 autocorrelation 피크 지점 탐지로 phase 확보
```

```
        [K5v1,K5index1]=max(K5C1_autocorr(1:length(C1)));
        [K5v2,K5index2]=max(K5C2_autocorr(1:length(C2)));
        [K5v3,K5index3]=max(abs(K5C3_autocorr(1:length(C3))));
        [K5v4,K5index4]=max(abs(K5C4_autocorr(1:length(C4))));
        [K5v5,K5index5]=max(K5C5_autocorr(1:length(C5)));
        K5index1=K5index1-1;%index 에서 1 을 빼면 각 코드의 timing offset
        K5index2=K5index2-1;
        K5index3=K5index3-1;
        K5index4=K5index4-1;
        K5index5=K5index5-1;
        K5Offset_C1=circshift(C1,K5index1);
        K5Offset_C2=circshift(C2,K5index2);
        K5Offset_C3=circshift(C3,K5index3);
        K5Offset_C4=circshift(C4,K5index4);
        K5Offset_C5=circshift(C5,K5index5);
        %앞에서 찾은 phase 만큼 original C1~C5 코드를 딜레이 시킨후 K5 코드 합성
        for i=1:43890
            Regen_K5_C(i)=sign(3*K5Offset_C1(1)+K5Offset_C2(1)-K5Offset_C3(1)-
K5Offset_C4(1)+K5Offset_C5(1));
            K5Offset_C1=circshift(K5Offset_C1,-1);
            K5Offset_C2=circshift(K5Offset_C2,-1);
            K5Offset_C3=circshift(K5Offset_C3,-1);
            K5Offset_C4=circshift(K5Offset_C4,-1);
            K5Offset_C5=circshift(K5Offset_C5,-1);
        end
        if sum(Original_K5_C ~= Regen_K5_C) == 0
            K5_detection=K5_detection+1;
        else

        end

    end
    JPL_SP(m)=JPL_detection/N;
    K5_SP(m)=K5_detection/N;
end
semilogy(snr,JPL_SP)
hold on
semilogy(snr,K5_SP)
legend("JPL Detection probability", "K5 Detection probability")
```

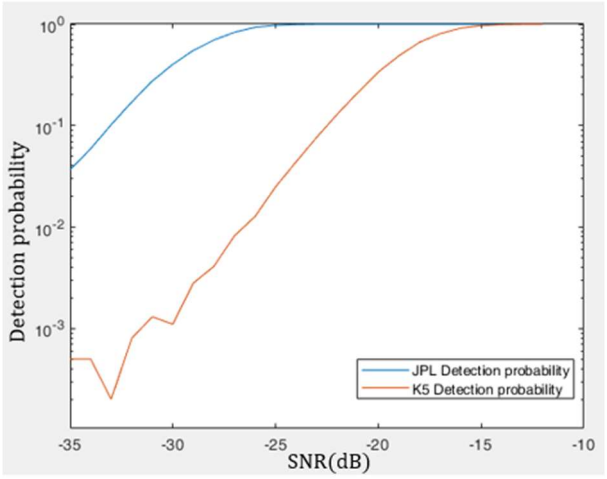

그림 12. JPL과 K5의 regenerative performance