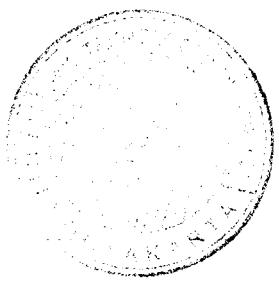
BAB III

PERANCANGAN SISTEM

3.1. Perhitungan Pesat Galat Bit

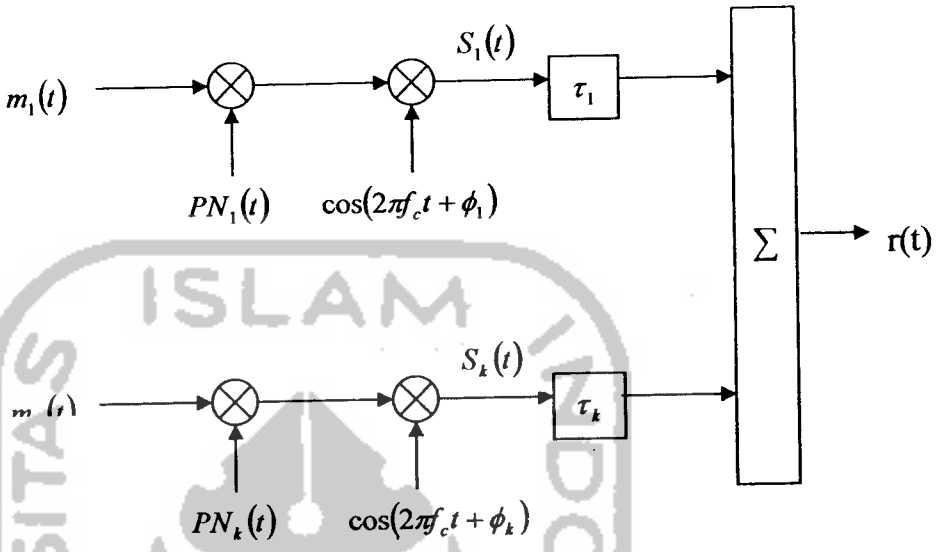
Sebenarnya besar laju galat bit pada *CDMA* yang mengalami pudaran lintasan jamak sangat sulit untuk ditentukan secara matematis. Hal ini karena faktor penyebab pudaran jenis ini sangat bervariasi dan tergantung kondisi lingkungan sekitar lingkungan sinyal dari pengirim ke penerima. Perbedaan kondisi lingkungan pada tiap –tiap daerah menyebabkan besarnya nilai laju galat bit yang diperoleh di suatu daerah belum tentu sama dengan didaerah lainnya.

Penghitungan nilai pesat galat bit pada sistem *CDMA* dapat dilakukan dengan cara pendekatan *Gauss (Gauss Approximation)*. Namun untuk keadaan yang melibatkan pudaran lintasan jamak diperlukan perhitungan yang melibatkan faktor lintasan jamak dan jumlah pencabangan penerima *RAKE*. Hal ini dikarenakan kedua parameter ini cukup menentukan besarnya kecilnya nilai pesat galat bit yang muncul dan keduanya mempunyai parameter serta perhitungan yang berdiri sendiri. Namun karena keterbatasan program dan juga rumus perhitungan yang ada, maka analisis matematika digunakan hanya dengan pendekatan *Gauss*. Dan hasil ini akan digunakan sebagai pembanding dengan hasil keluaran program simulasi.

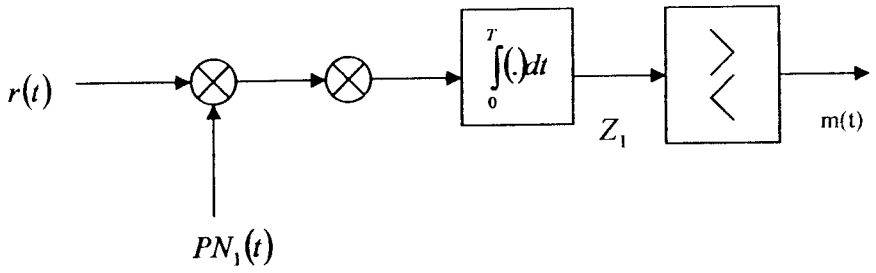


3.1.1. Perhitungan dengan pendekatan *Gauss*

Suatu sistem spektrum tersebar runtun langsung (DS-SS) dengan pengguna jamak sebanyak K dapat digambarkan seperti gambar berikut 3.1



Gambar 3.1
Model pengguna sebanyak k dalam sistem spektrum tersebar *CDMA*



Gambar 3.2
Struktur penerima untuk pengguna ke-1

Dengan asumsi bahwa tiap –tiap pengguna mempunyai runtun kode *PN* dengan N chip tiap simbol dan periode berdurasi T, sehingga $NT_c = T$

Persamaan sinyal yang dikirimkan oleh tiap –tiap pengguna sistem *DS-CDMA* dapat dituliskan sebagai berikut

$$S_k(t) = \sqrt{\frac{2E_s}{T_s}} m_k(t) P_k(t) \cos(2\pi f_c t + \phi_k) \qquad (3.1)$$

Ket :

$S_k(t)$ = sinyal yang dkirimkan oleh pengguna

$P_k(t)$ = runtun kode *PN* untuk pengguna ke-k

$m_k(t)$= runtun data dari pengguna ke-k.

Sinyal yang diterima akan berisi penjumlah dari k sinyal yang berbeda-beda (sebuah pengguna yang diinginkan dan k-1 pengguna yang tak diinginkan). Penangkapan sinyal yang diinginkan dilakukan dengan cara mengkorelasi sinyal yang diterima dengan runtun kode *PN* yang sesuai sehingga menghasilkan sebuah variable keputusan. Variable keputusan untuk bit terkirim ke-1 dari pengguna 1 adalah

$$Z_i^{(1)} = \int_{(i-1)T+\tau_i}^{iT+\tau_i} r(t)p_1(t-\tau_1)\cos[2\pi.f_c(t-\tau_1)+\phi_1]dt \qquad (3.2)$$

jika $m_{i,j}$=-1maka bit yang diterima akan menjadi salah saat $Z_1^{(1)} > 0$. sehingga kemungkinan terjadi pesat galat dapat dihitung sebagai $\Pr\left[Z_1^{(1)} > 0 \middle| m_{i,j} = -1\right]$. Karenakan sinyal yang diterima r(t) adalah suatu kombinasi dari beberapa sinyal, maka persamaan 3.3. Dapat ditulis sebagai berikut

$$Z_i^{(1)} = I_1 + \sum_{k=2}^{K} I_k + \varepsilon \qquad (3.3)$$

Ket :

K =jumlah pengguna

dengan

$$I_1 = \int_0^T S_1(t)P_1(t)\cos(2\pi.f_c t)dt = \sqrt{\frac{E_s T}{2}} \qquad (3.4)$$

adalah tanggapan penerima terhadap sinyal yang diinginkan dari pengguna1.

$$\varepsilon = \int_0^T n(t)p_1(t)\cos(2\pi.f_c t)dt \qquad (3.5)$$

adalah variable acak *Gauss* yang mewakili derau rata –rata nol dan varians

$$E[\varepsilon^2] = \frac{N_0 T}{4} \tag{3.6}$$

dan

$$I_k = \int_0^T S_k(t-\tau_k) p_1(t) \cos(2\pi . f_c t) dt \tag{3.7}$$

Mewakili interferensi masupan jamak dari pengguna ke-k. asumsikan bahwa I_k adalah efek kumulatif dari N chip –chip acak dari penginterferensi ke-k melalui pengintegralan selama periode T dari satu bit. Sesuai dengan teorema batas pusat *(Central Limit Theorem,CLT),* yang menjelaskan bahwa fungsi rapat probabilitas *(Probability Density Function)* dari jumlah N buah variable acak bebas akan semakin mendekati rapat *Gaussian* seiring dengan meningkatkannya nilai N, penjumlah dari efek – efek ini akan cenderung berupa suatu distribusi *Gauss*. Dengan mengganggap terdapat K-1 pengguna yang dapat dianggap sebagai penginterferensian yang terdistribusi, maka besarnya interferensi masupan jamak total

$$I = \sum_{k-2}^{K} I_k \tag{3.8}$$

dapat diperkirakan denga variable acak *Gauss*. Pendekatan *Gauss* menghasilkan persamaan untuk probabilitas rata –rata galat bit sebagai berikut

$$P_e = Q\left(\frac{1}{\sqrt{\frac{K-1}{3N} + \frac{N_0}{2E_b}}} \right) \tag{3.9}$$

untuk keadaan jumlah pemakai hanya satu, K=1, maka persamaan 3.9 ,menjadi

$$P_{e,K-1} = Q\left(\frac{1}{\sqrt{\frac{N_0}{2E_b}}}\right) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.10}$$

dengan E_b adalah energi tiap bit, dan N_0 adalah rapat spektral derau. Nilai E_b / N_0 dapat ditulis sebagai berikut

$$\frac{E_b}{N_0} = \frac{P.SFT}{P_n.T_c} = \frac{P.SF}{P_n} = SF.SNR \text{ (1)} \tag{3.11}$$

Ket:

P = daya sinyal

SF= *Spreading* faktor atau perolehan pengolahan

P_n = daya derau

T_c = periode chip.

Sedangkan Q adalah fungsi Q yang merupakan fungsi rapat probabilitas normal *Gauss* untuk variable acak *Gauss*. Fungsi Q ditulis sebagai berikut

$$Q(z) = \int_z^{\infty} \frac{1}{\sqrt{2\pi}} e^{-y^2/2}\, dy \tag{3.12}$$

dengan

$$z \frac{x_0 - m}{\sigma} \tag{3.13}$$

Untuk dapat diterapkan dalam bahasa pemograman, bentuk integral dalam persamaan (3.12) harus disederhanakan. Dengan menggunakan penjabaran *Bagby (1995),* maka diperoleh bentuk fungsi Q sebagai berikut:

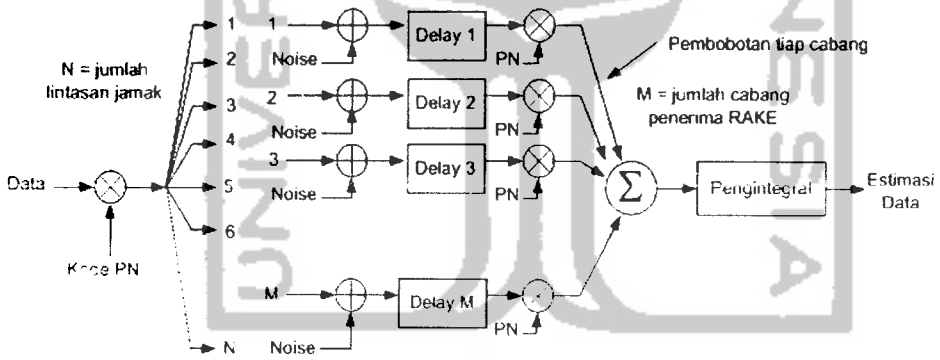
$$Q(z)=\frac{1}{2}-\frac{1}{2}\left\{1-\frac{1}{30}\left[7e^{-z^2/2}+16e^{-z^2(2-\sqrt{2})}+\left(7+\frac{1}{4}\pi z^2\right)e^{-z^2}\right]\right\}^{1/2} \qquad (3.14)$$

3.2. Penjelasan Program Simulasi

3.2.1. Gambaran Umum

Program simulasi dibuat dengan bahasa pemograman *Delphi 5 enterprise*. Program ini dibuat untuk mensimulasikan kinerja system DS-CDMA dengan memperhitungkan faktor pudaran lintasan jamak dan pencabangan pada sisi penerima *RAKE*. Hasil keluaran dari program ini berupa nilai pesat galat bit *(BER)* yang dapat diperoleh dengan simulasi system *DS-CDMA* maupun dengan pendekatan perhitungan matematis. Perhitungan matematis yang dibuat adalah dengan pendekatan *Gauss* seperti yang telah dijelaskan sebelumnya.

Skema program simulasi dibuat seperti gambar 3.3 berikut



Gambaran 3.3. sistem secara umum

Beberapa asumsi yang digunakan dalam program simulasi adalah:

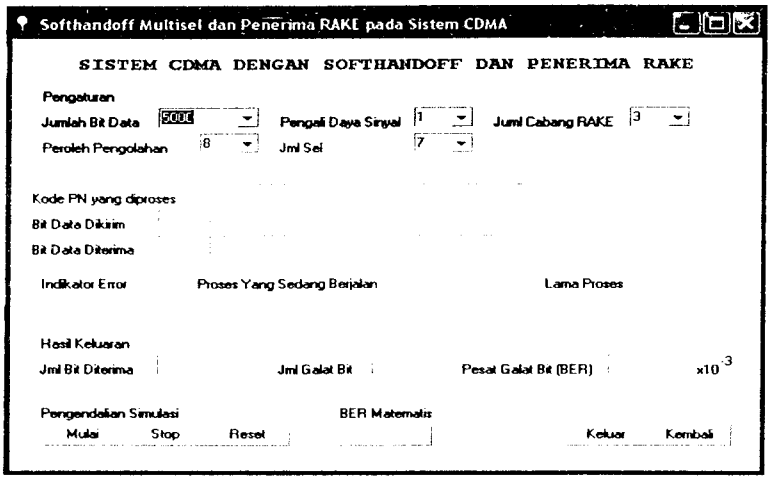
1. Jumlah pemakai dalam sistem *CDMA* ini adalah satu dengan maksud agar galat yang muncul bukan diakibatkan oleh interferensi pengguna lain.

   Daya sinyal yang ditangkap dipenerima mengalami penyusutan selama proses perambatan dari pengirim ke penerima dan besar susutan adalah acak dan mempunyai batas maksimum tertentu.

2. Pengaruh derau *(Noise)* pada masing –masing lintasan jamak bersifat independent satu sama lain. Besarnya tunda waktu dan fase yang muncul pada masing –masing lintasan jamak adalah bervariasi dan disesuaikan menurut daya sinyal sesudah mengalami penyusutan dan membentuk pudaran *Rayleigh.*

3. Penerima *RAKE* dianggap dapat mendeteksi tunda waktu dan kuat sinyal yang masuk ke masing- masing pencabangan dan sempurna. Proses peraupan *(Despreading)* dilakukan sebelum proses penggabungan sinyal dari tiap-tiap pencabangan *RAKE.* Sedangkan proses pengintegralan dilakukan setelah proses penggabungan nilai *BER* dengan perhitungan secara matematis menggunakan pendekatan *Gauss* dengan jumlah pengguna satu.

Program simulasi mempunyai dua tampilan utama yaitu satu merupakan tampilan simulasi sistem *DS-CDMA* secara real-time dan juga secara matematis, dan satu lagi merupakan representasi grafis hubungan antara pesat galat bit dengan beberapa parameter-parameter yang sudah ada. Kedua gambar tampilan utama program adalah sebagai berikut ditunjukan pada gambar 3.4.



Gambar 3.4. Tampilan Program

Pada tampilan program seperti ditunjukan pada gambar 3.4 Memiliki beberapa pengaturan yang dapat diubah –ubah. Pengaturan ini meliputi jumlah bit data, perolehan pengolahan, daya sinyal yang dikirim, jumlah lintasan jamak yang terjadi, dan jumlah cabang penerima *RAKE*. Tampilan ini akan mensimulasikan sistem *DS-CDMA* secara *Real Time* dan juga matematis. Data yang dikirimkan diberikan secara acak dan kontiyu. Proses akan terus berjalan sampai ditekan tombol "stop" untuk menghentikan simulasi atau juga jumlah bit data yang diterima telah mencapai jumlah bit data yang dikirim. Selama proses berlangsung, program akan terus memantau bit data yang diterima dan jumlah galat yang terjadi. Hasilnya kemudian digunakan untuk menghitung *BER* sesaat sistem *DS-CDMA* yang disimulasikan

Pada tampilan program seperti pada gambar 3.4. tidak ada pengaturan yang dapat dilakukan. Tampilan ini hanya menampilkan secara grafis hasil- hasil uji simulasi dengan program ini. Hasil grafis hubungan pesat galat bit dengan

beberapa parameter ini diperoleh dengan variasi beberapa nilai pesan galat bit sesaat, namun merupakan nilai pesan galat bit yang muncul pada saat mengirimkan sejumlah bit data yang sudah ditentukan sebelumnya. Tampilan nilai pesat galat bit sesaat tidak dibuat karena nilai pesat galat bit yang dibutuhkan adalah nilai pesat galat bit variasi dari nilai beberapa parameter yang bila dibuat real time akan membutuhkan waktu yang cukup lama untuk menghasilkan satu tampilan grafik. Dan grafik hasil- hasil uji ditampilkan pada gambar 20 meliputi hasil uji dengan simulasi sistem *DS-CDMA* secara *Real Time* dan juga secara perhitungan matematis.

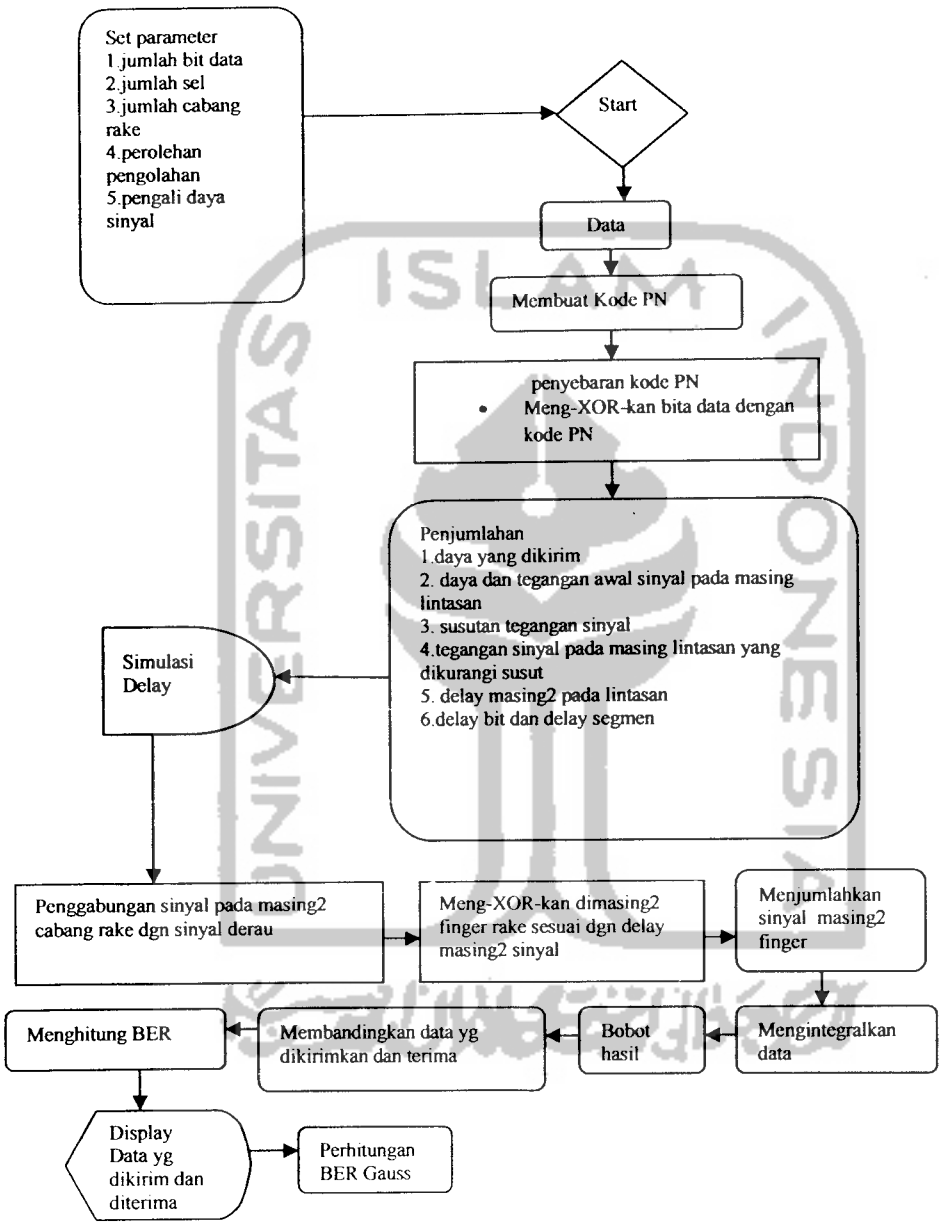
Struktur Program Simulasi

Dalam simulasi ini, Data masukan dari satu pengguna dibangkitkan secara acak. Masing- masing bit data acak dari pengguna kemudian disebarkan menggunakan runtunh *PN*. Runtun *PN* yang digunakan dalam simulasi ini adalah sandi *Walsh*, yang merupakan baris –baris dalam matrik *Hadamard-64*. Bit –bit hasil pengaburan dengan sandi *Walsh* dari semua pengguna kemudian di jumlahkan.

Dipenerima, data yang diterima merupakan data yang berasal dari pengguna. Untuk memperoleh data bagi pengguna tertentu, dilakukan proses peraupan untuk pengguna yang diinginkan. Untuk memperoleh data aslinya, maka hasil perkalian harus dintergalkan selama satu periode bit. Setelah dilakukan proses pengintegralan ini, maka data yang dikirimkan telah dipulihkan kembali. D ata yang telah dipulihkan kembali. Data yang telah dipulihkan kemudian dibandingkan dengan data asalnya untuk mengetahui apakah terjadi kesalahan

atau tidak. Banyaknya kesalahan (galat) yang terjadi direkam, dan digunakan untuk melakukan perhitungan *BER*.

Selain dilakukan perhitungan *BER* secara eksperimental, juga dilakukan perhitungan *BER* secara matematis, yaitu metode *Aproksimasi Gauss*.



Gambar 3.5. FlowChart Simulasi

3.2.2. Penjelasan Struktur Program

Program simulasi ini mempunyai data masukan hanya untuk satu pengguna. Pembuatan data untuk masukan dilakukan secara acak dengan membangkitkan runtun data biner acak sesuai dengan penggalan program ini membuat data acak

```
BitData:=random(1000);
if BitData<=250 then BitData:=0
else if (BitData>250) and (BitData<500) then BitData:=1
else if (BitData>500) and (BitData<750) then BitData:=0
else BitData:=1;
```

Sedangkan runtun bit kode *PN* dihasilkan dengan penggalan program berikut

```
for i:=1 to Gain do
 begin
 BitPN:=random(1000);
 if BitPN<=250 then BitPN:=0
 else if (BitPN>250) and (BitPN<500) then BitPN:=1
 else if (BitPN>500) and (BitPN<750) then BitPN:=0
 else BitPN:=1;
 PN[i]:=BitPN;
 KodePN:=KodePN+inttostr(PN[i]);
 Edit8.Text:=KodePN;
 end
end
```

Dari penggalan program diatas dapat terlihat bahwa runtun bit *PN* yang dihasilkan adalah acak dan panjangnya sesuai dengan besar perolehan pengolahan yang diinginkan.

Tiap bit data yang telah dibangkitkan kemudian disebarkan menggunakan runtun bit *PN* dengan penggalan program berikut

```
if BitData=0 then data_a:=-1 else data_a:=1;
for i:=1 to Gain do
 begin
 if PN[i]  0 then PN[i]:=-1 else PN[i]:=1;
 cdmax[i]:=X_OR(data_a,PN[i]);
 end;
```

Penggalan program diatas akan memangil fungsi *X-OR* untuk berfungsi untuk meng-*XOR*-kan bit data dengan bit kode *PN*

```
function X_OR(a,b:integer):integer;
begin
 if ((a>0) and (b<0)) or ((a<0) and (b>0)) then X_OR:=abs(a)
 else if ((a>0) and (b>0)) or ((a<0) and (b<0)) then X_OR:=-1*abs(a)
 else X_OR:=0;
end;
```

Bit –bit hasil penyebaran dengan kode *PN* masih bernilai1 dan-1. nilai bit ini kemudian dibobot sesuai dengan daya pancar sinyal yang diinginkan. Besarnya pengali daya pancar sinyal ini dapat ditentukan pada bagian pengaturan program simulasi. Dengan memperbesar nilai pengali daya sinyal, maka akan menyebabkan besarnya amplitude bit-bit sinyal juga semakin besar.

Pada saat transmisi sinyal diasumsikan terjadi pudaran lintasan jamak sehingga muncul beberapa lintasan sinyal yang besarnya dibuat bervariasi. Besarnya daya sinyal pada masing-masing lintasan jamak dibuat sedemikian sehingga saling berbeda dan terjadi perbedaan besar daya yang linear menurun

dari yang terbesar sampai ke yang terkecil. Dan penentuan besar daya sinyal ini masih mengikuti prinsip total besar daya dari tiap –tiap lintasan jamak adalah sama dengan besar daya sinyal yang terpancar. Sehingga besar daya sinyal ditiap lintasan jamak dapat dirumuskan sebagai berikut

$$P_i = \frac{i}{\sum_{i=1}^{N} i} x P_{total} = \frac{N.i}{N\sum_{i=1}^{i} i} x P_{total} \quad \text{dengan N=jumlah lintasan jamak} \qquad (3.15)$$

persamaan diatas dapat diaplikasikan dalam penggalan program sebagai berikut

```
pembagi:=lintas;
for i:=2 to lintas do pembagi:=pembagi+lintas*i;
for i:=1 to lintas do DayaLintasx[i]:=((lintas-i+1)*lintas*daya)/pembagi;
```

Sedangkan besarnya tegangan sinyal yang dipancarkan adalah akar kuadrat dari daya yang terpancar sesuai dengan penggalan program berikut

```
for i:=1 to lintas do VoltLintasx[i]:=sqrt(DayaLintasx[i]*2);
```

Selain munculnya lintasan jamak, program simulasi juga memperhitungkan adanya susutan sinyal selama perambatan. Besarnya susutan sinyal yang terjadi pada masing-masing –masing lintasan jamak adalah acak dan mempunyai nilai maksimal tertentu, dan pada program daya susut maksimal dibuat setengah dari daya sinyal awalnya.

Untuk mensimulasikan adanya tunda waktu dan beda fase pada masing – masing lintasan jamak, maka perlu diperkirakan bahwa sinyal akhir dipenerima berupa sinyal pudaran lintasan jamak yang mengikuti distribusi *Rayleigh*. Sinyal dengan distribusi seperti ini dapat dibangkitkan dengan menggabungkan dua buah sinyal acak yang mengikuti distribusi *Gauss*, dengan salah satu komponen sinyal dengan komponen lainnya menempati sumbu kordinat yang berbeda. Kemudian

sinyal resultan dari kedua komponen sinyal tadi akan membentuk sinyal dengan distribusi *Rayleigh*. Dan sudut yang dibentuk oleh sinyal resultan terhadap salah satu sumbu dapat dianggap mewakili tunda fase sinyal tersebut terhadap sinyal awal sewaktu dikirimkan. Sehingga dengan menentukan batas beda fase maksimal dapat dibuat simulasi beda fase masing –masing sinyal lintasan jamak. Dan sinyal pada lintasan jamak pertama dianggap mempunyai beda fase yang paling kecil, sedangkan sinyal pada lintasan berikutnya dianggap mempunyai beda fase yang paling kecil berikutnya.

Proses diatas dapat aplikasikan dengan penggalan program berikut

```
if length(edit5.text)=1 then begin
for i:=1 to lintas do begin
 repeat
  Aix[i]:=VoltLintasx[i]*(1+Gauss);
  until Aix[i]<>0;
 Ajx[i]:=VoltLintasx[i]*(1+Gauss);
 sudutx[i]:=(arctan(Ajx[i]/Aix[i])*180/pi);
 table2.append;
 Table2.FieldValues['Sudut']:=sudutx[i];
 end;
Table2.post;
end;
Table2.first;
for i:=1 to lintas do begin
 sudutx[i]:=Table2.FieldValues['Sudut'];Table2.next;
 delayx[i]:=round(sudutx[i]/90*Gain*20); {jauhnya delay maks=2 bit}
 end;
```

Selanjutnya sinyal pada masing –masing lintasan jamak akan mengalami penyesuaian sesuai dengan tunda waktu yang kurang dari satu chip data maka tiap- tiap satu chip data dibagi lagi menjadi beberapa segmen yang dalam program simulasi dibuat sepuluh segmen. Proses ini ditunjukan pada penggalan program berikut:

```
segmen:=10;
for i:=1 to lintas do begin
 DelayBitx[i]:=delayx[i] div (segmen*gain);
 DelaySegmenx[i]:=delayx[i] mod (segmen*gain);
 end;
n:=1;
for i:=1 to gain do begin
 for j:=1 to lintas do begin
  z:=DelayBitx[j]+1;
  for k:=0 to segmen-1 do begin
   if DataKirim <= DelayBitx[j] then VoltSegmenx[j,n+k+DelaySegmenx[j]]:=0
   else if DataKirim=DelayBitx[j]+1 then begin
    if (n+k+DelaySegmenx[j])>(segmen*gain) then VoltSegmenx[j,n+k+DelaySegmenx[j]-
segmen*gain]:=0
    else begin
     m:=(n+k+segmen-1) div segmen;
     val(edit5.text[z],data,kode);
     if data=0 then data_a:=-1 else data_a:=1;
     VoltSegmenx[j,n+k+DelaySegmenx[j]]:=X_OR(data_a,PN[m])*abs(VoltLintasx[j]);
     end;
    end{akhir if DataKirim=DelayBitx[j]+1}
   else begin {DataKirim>DelayBitx[j]}
```

```
m:=(n+k+segmen-1) div segmen;
if (n+k+DelaySegmenx[j])>(segmen*gain) then begin
 val(edit5.text[z+1],data,kode);
 if data=0 then data_a:=-1 else data_a:=1;
 VoltSegmenx[j,n+k+DelaySegmenx[j]-
segmen*gain]:=X_OR(data_a,PN[m])*abs(VoltLintasx[j]);
 end
 else begin {(n+k+DelaySegmenx[j])<=(segmen*gain)}
 val(edit5.text[z],data,kode);
 if data=0 then data_a:=-1 else data_a:=1;
 VoltSegmenx[j,n+k+DelaySegmenx[j]]:=X_OR(data_a,PN[m])*abs(VoltLintasx[j]);
 end;
 end;
 end; {akhir segmen}
 end;{akhir lintas}
n:=n+segmen;
end;{akhir gain dan keseluruhan segmen dan lintasan}
```

Kemudian masing –masing sinyal lintasan jamak tadi akan mendapatkan gangguan derau yang tidak tergantung *(Independent)* antara satu dengan lainnya. Pembangkitan sinyal derau dibuat dengan penggalan program fungsi sebagai berikut

```
for i:=1 to Gain*segmen do begin
 for j:=1 to jumrake do begin
  deraux[j,i]:=BuatDerau;
  VoltRakex[j,i]:=VoltRakex[j,i]+deraux[j,i];
  end;
 end;
```

dan proses penggabungan sinyal *CDMA* dengan sinyal derau ditunjukan oleh penggalan program berikut

```
for i:=1 to Gain*segmen do begin
DayaOutSegmenx[i]:=0;
for j:=1 to jumrake do begin
DayaOutSegmenx[i]:=DayaOutSegmenx[i]+OutXORx[j,i];
end;
end;
```

Pada bagian penerima, sinyal dari lintasan jamak akan ditangkap oleh tiap –tiap cabang penerima *RAKE*. Sinyal yang pertama sampai dan mempunyai tunda waktu terkecil berikutnya akan diterima di pecabangan pertama dan sinyal yang mempunyai tunda waktu terkecil berikutnya akan diterima dipecabangan berikutnya. Daya sinyal lintasan jamak yang diterima di pecabangan pertama dan sinyal yang mempunyai tunda waktu terkecil berikutnya akan diterima dipecabangan berikutnya. Daya sinyal lintasan jamak yang diterima dipecabangan berikutnya. Daya sinyal lintasan jamak yang diterima dicabang *RAKE* pertama adalah yang terkuat, dan sinyal yang terkuat berikutnya menempati cabang *RAKE* yang berikutnya.

Selanjutnya sinyal yang ada masing –masing cabang penerima *RAKE* akan mengalami proses peraupan *(Despreading)* dengan cara meng-*XOR*kan kembali sinyal tersebut dengan kode *PN* yang sama pada saat pengiriman data

```
for j:= 1 to jumrake do begin
 for i:=1 to gain*segmen do begin
  m:=(i+segmen-1-DelaySegmenx[j]) div segmen;
mesti dipilah antara yg ada delaybit dgn yg tdk ada}
```

```
 if (i>DelaySegmenx[j]) and (length(edit2.text)>DelayBitx[j]) then begin
  if i<=DelaySegmenx[j] then m:=(i+segmen-1-DelaySegmenx[j]+Gain*segmen) div segmen;
  outXORx[j,i]:=X_OR2(VoltRakex[j,i],PN[m]);
  end;
 end;
end;
```

Sinyal hasil peraupan dari masing –masing cabang *RAKE* kemudian dijumlahkan di penggabung sesuai dengan penggalan program berikut

```
for i:=1 to Gain*segmen do begin
 DayaOutSegmenx[i]:=0;
 for j:=1 to jumrake do begin
  DayaOutSegmenx[i]:=DayaOutSegmenx[i]+OutXORx[j,i];
 end;
end;
```

Agar sinyal hasil penggabungan dapat dipulihkan kembali menjadi sinyal bit data aslinya, maka sinyal tersebut harus diintegralkan selama satu periode bit. Proses ini ditunjukkan penggalan program berikut

```
for i:=1 to Gain*segmen do IntegralPerSegmen[i]:=0;
IntegralTotal:=0;
for i:=1 to Gain*segmen do begin
 if i=1 then IntegralPerSegmen[i]:=DayaOutSegmenx[i]
 else IntegralPerSegmen[i]:= IntegralPerSegmen[i-1]+DayaOutSegmenx[i];
 IntegralTotal:=IntegralTotal+IntegralPerSegmen[i];
 end;
{membobot hasil integral}
RerataIntegral:=IntegralTotal/(Gain*segmen);
if RerataIntegral>0 then OutIntegrator:=1
```

```
else OutIntegrator:=0;
```

Setelah dilakukan proses pengintegralan ini, maka data yang dikirimkan telah dipulihkan kembali.

Data yang telah dipulihkan kemudian dibandingkan dengan data asalnya untuk mengetahui apakah terjadi kesalahan atau tidak. Banyaknya kesalahan (galat) yang terjadi dihitung, dan digunakan untuk melakukan perhitungan *BER* dalam penggalan program berikut

```
BER:=error/DataKirim*1000; {BER aslinya dibagi 1000}
```

Pemakaian pengali seribu dikarenakan untuk memudahkan menampilkan besarnya *BER* pada program simulasi, dan berarti *BER* aslinya adalah dibagi seribu atau dikali $x10^{-3}$

Sedangkan untuk perhitungan secara matematis, perhitungan dengan pendekatan *Gauss* dilakukan oleh penggalan program berikut

```
EnergiBit:=daya*Gain;
psdDerau:=dayaDerau;
z:=sqrt(2*EnergiBit/psdDerau);
BERGauss:=fungsiQ(z)*1000; {BERgauss aslinya dibagi 1000}
Edit3.text:=formatfloat('#.###E-',BERGauss)
```