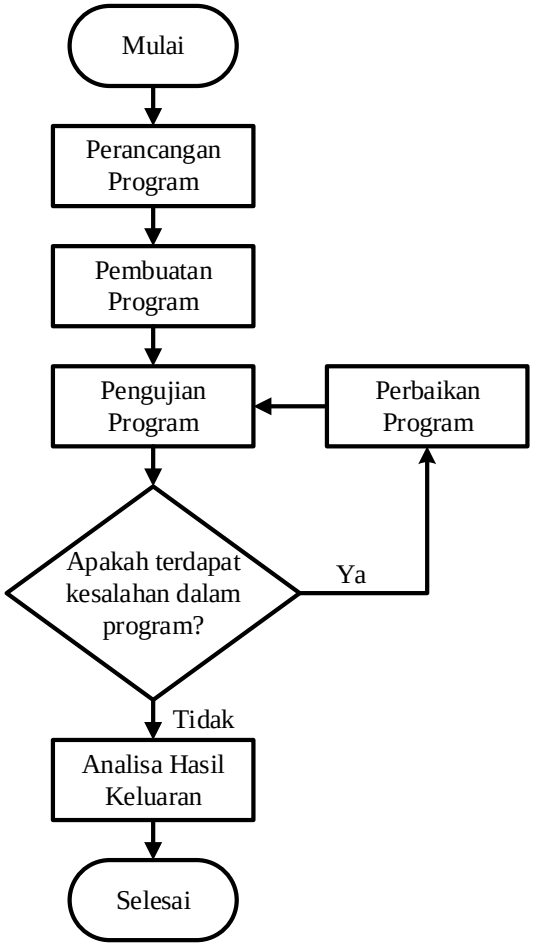
BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

Penelitian ini menggunakan suatu pemodelan program simulasi dalam menganalisis unjukkerja sistem FBMC menggunakan modulasi OQAM pada komunikasi SISO. Model simulasi yang diimplemetasikan dalam penelitian ini, menggunakan sebuah program MATLAB R2016a.

3.2 ALUR PENELITIAN

Rancangan proses penyelesaian skripsi, mengenai simulasi program FBMC OQAM pada sistem SISO dengan menggunakan Matlab, dijabarkan dalam diagram alir (*flowchart*) yang ditunjukkan pada Gambar 3.1.



Gambar 3. 1 *Flowchart* Rancangan Simulasi Program

3.3 PARAMETER SIMULASI

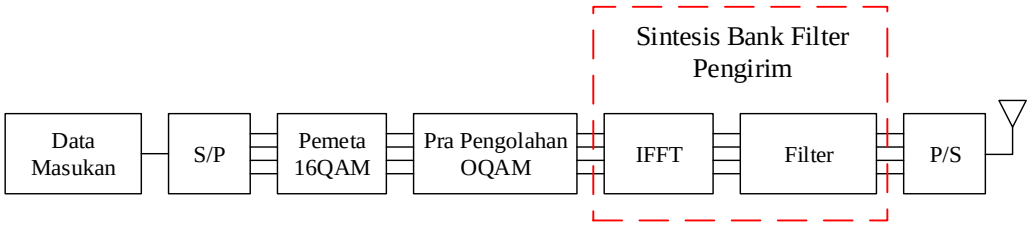
Parameter yang digunakan dalam simulasi ini, sebagaimana tertera pada Tabel 3.1.

Tabel 3. 1 Parameter Simulasi

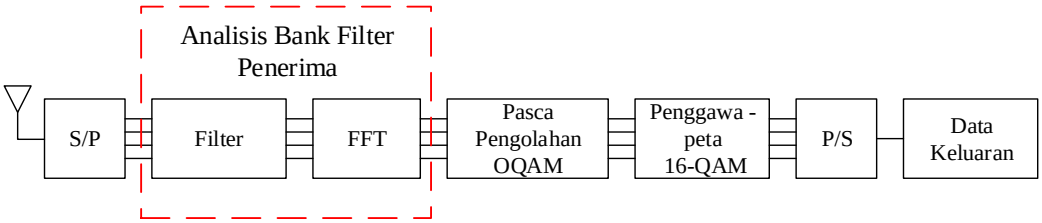
Simbol	Parameter	Nilai
ntx	Jumlah antena pengirim	1
nrx	Jumlah antena penerima	1
M	Modulasi Dasar	16 QAM
ml	Jumlah level modulasi	4
fs	Frekuensi cuplik	22050 Hz

3.4 Pemodelan Sistem SISO FBMC OQAM

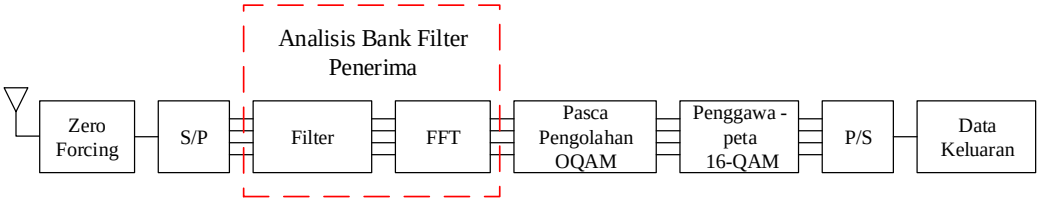
Pemodelan pengirim dan penerima sistem SISO FBMC OQAM dijelaskan pada blok diagram berikut:



Gambar 3. 2 Bagan Diagram SISO FBMC OQAM di sisi Pengirim



Gambar 3. 3 Bagan SISO FBMC OQAM di sisi penerima



Gambar 3. 4 Bagan SISO FBMC OQAM di sisi penerima dengan *Zero Forcing*

Berikut ini merupakan penjelasan mengenai blok diagram SISO FBMC OQAM.

3.4.1 Pengiriman

3.4.1.1 Data Masukan

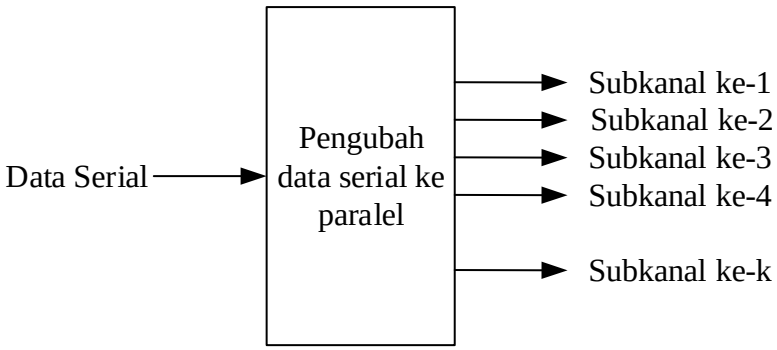
Data masukan pada simulasi ini berupa file audio dengan format file .wav. Sinyal audio pada simulasi ini, memiliki frekuensi cuplik sebesar 22050 Hz. Untuk dapat mengirimkan dan mengolah sinyal audio tersebut, perlu adanya perubahan terlebih dahulu dari sinyal audio ke sinyal diskrit atau bilangan biner. Sinyal audio memiliki nilai amplitudo ternormalisasi dari -1 volt sampai dengan 1 volt. Sehingga untuk melakukan proses pengubahan sinyal audio menjadi sinyal diskrit, amplitudonya perlu ditambah 1 agar menjadi prositif. Setelah itu dikalikan dengan 2^{n-1} untuk dapat mengubahnya menjadi bilangan desimal, dengan n merupakan jumlah bit biner pada setiap titik sinyal audio (n = 8 bit biner). Setelah didapat nilai desimalnya maka baru dapat diubah menjadi bentuk binernya. Adapun kode program untuk mengubah sinyal audio menjadi bilangan biner sebagai berikut:

```
%import audio to matlab and convert to binary sequences
auname = fullfile(pname, fname);
[auori, fs] = audioread(auname);
au0 = auori(:, 1);
awal = 20000;
akhir = (awal + N/8) - 1;
au0 = au0(awal : akhir);
% make sure they are already normalized to - 1 to +1 v
au0 = au0 ./ max(abs(au0));
au0 = round(au0 * 100)/100;
% lift them up to be >= 0 v
[rr, cc] = find(au0 < 0);
au = au0;
au(rr, 1) = au0(rr, 1) + 1;
% audio properties
info = audioinfo(auname);
% convert them to binary sequence
nbits = 8;
```

```
audec = floor(au .* (2^(nbits - 1)));
aubin = dec2bin(audec, nbits)';
aubin_dou = aubin(:) - '0';
data_input = aubin_dou';
```

3.4.1.2 Pengubah Data Seri ke Paralel (S/P)

Pada bagan pengubah bit seri ke paralel ini berfungsi untuk mengubah bentuk, dari bentuk data bit seri ke dalam bentuk paralel dengan mengelompokkan bit-bit tersebut sesuai dengan level modulasi yang digunakan. Seperti yang ditunjukkan pada Gambar 3.5.



Gambar 3. 5 Pengubahan data seri ke paralel

Dan program berikut, dapat digunakan untuk mengubah data seri menjadi paralel.

```
sp_tx = reshape(data_input, ml, length(data_input)/ml);
sp_tx = sp_tx';
```

Bit biner yang berasal dari sinyal audio terdiri dari 1024 bit untuk sekali pengiriman. Maka ketika telah berubah menjadi bentuk paralel menjadi 256 simbol dengan masing-masing simbol terdiri dari empat bit (modulasi 16 QAM, 1 simbol terdiri dari 4 bit).

3.4.1.3 Pemeta 16-QAM

Hasil keluaran dari pengubah data seri ke paralel, selanjutnya dipetakan dengan modulasi 16 QAM. Pemetaan 16 QAM dilakukan untuk mengubah data masukan biner menjadi bentuk bilangan kompleks $S_k = I_k + jQ_k$, dengan k adalah variabel simbol, I adalah *inphase* atau bilangan riil dan Q adalah *quadrature* atau bilangan imajiner. Modulasi dasar yang digunakan pada simulasi ini menggunakan

pemetaan 16 QAM. Maka, setiap simbol yang diterima pada pemetaan ini terdiri dari empat bit. Untuk proses pemetaan modulasi 16 QAM menggunakan standar pemetaan dari 3GPP yaitu pada persamaan 2.1. Berdasarkan persamaan tersebut, setiap simbol yang dikirimkan dari data S/P dapat dipetakan menggunakan kode program berikut ini:

```
for c=1:(length(sp_tx))
    qammod(c,:)=(1/sqrt(10))*((1-2*(sp_tx(c,4)))*...
        (2-(1-2*(sp_tx(c,2))))+sqrt(-1)*...
        (1-2*(sp_tx(c,3)))*(2-(1-2*(sp_tx(c,1)))));
end
```

“qammod” merupakan variabel baru untuk hasil keluaran pemetaan 16 QAM. “sp_tx” merupakan hasil keluaran dari proses seri ke paralel. Dan proses perulangan pada “c=1(length:(sp_tx))” digunakan untuk mengulang proses pengubahan bit biner menjadi bilangan kompleks yang berada di dalamnya sebanyak panjang data dari variabel sp_tx. Sehingga setiap simbol hasil pemeta 16 QAM dalam bentuk kompleks seperti ditunjukkan pada Tabel 3.2.

Tabel 3. 2 Pemetaan 16-QAM

No	Biner 4 bit				Bilangan Kompleks (χ)
	Bit-4	Bit-3	Bit-2	Bit-1	
1	0	0	0	0	0,3162 + 0,3162 *i*
2	0	0	0	1	- 0,3162 + 0,3162 *i*
3	0	0	1	0	0,3162 - 0,3162 *i*
4	0	0	1	1	-0,3162 - 0,3162 *i*
5	0	1	0	0	0,9486 + 0,3162 *i*
6	0	1	0	1	- 0,9486 + 0,3162 *i*
7	0	1	1	0	0,9486 - 0,3162 *i*
8	0	1	1	1	- 0,9486 - 0,3162 *i*
9	1	0	0	0	0,3162 + 0,9486 *i*
10	1	0	0	1	- 0,3162 + 0,9486 *i*
11	1	0	1	0	0,3162 - 0,9486 *i*

Tabel 3. 2 Pemetaan 16-QAM (Lanjutan)

No	Biner 4 bit				Bilangan Kompleks (χ)
	Bit-4	Bit-3	Bit-2	Bit-1	
12	1	0	1	1	- 0,3162 - 0,9486 *i*
13	1	1	0	0	0,9486 + 0,9486 *i*
14	1	1	0	1	- 0,9486 + 0,9486 *i*
15	1	1	1	0	0,9486 - 0,9486 *i*
16	1	1	1	1	- 0,9486 - 0,9486 *i*

3.4.1.4 Pra Pengolahan OQAM

Pada proses pra pengolahan OQAM, simbol yang dikirimkan dipecah menjadi 2 yaitu simbol ganjil dan genap. Simbol ganjil genap didapat dari urutan simbol yang dikirimkan. Simbol tersebut diolah di dalam pra pengolahan OQAM yang terdiri dari 2 operasi yaitu:

1. Proses pengubahan bilangan kompleks menjadi bilangan riil.
2. Proses perkalian dengan $\theta_{k,n}$.

Pada operasi pertama simbol ganjil genap yang terdiri dari bilangan riil dan imajiner masing – masing akan dipisahkan. Lalu masing-masing bilangan riil dan imajiner mengalami peningkatan jumlah (*upsampling* sebesar 2 kali) yang diikuti dengan pergeseran setengah fasa sebesar 90^o (1 fasa = 180^o). Untuk lebih jelasnya proses pra pengolahan OQAM dapat dilihat pada Gambar 2.8.

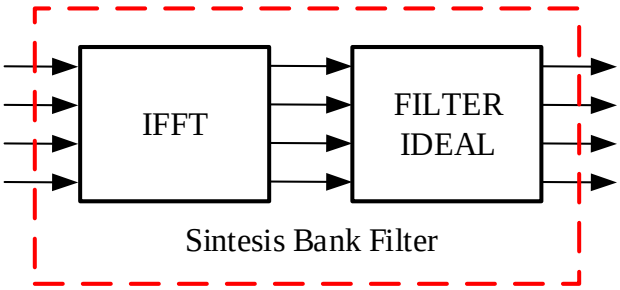
Adapun kode program dari proses tersebut, sebagai berikut:

```
for n = 1 : length(qammod_atas)
   if (rem(n, 2) == 1) %ganjil
   oqam_pre(n,1)=imag(qammod_atas(n))*((j).^(n));
   oqam_pre(n,2)=real(qammod_atas(n))*((j).^(n));
   else %genap
   oqam_pre(n,1)=real(qammod_atas(n))*((j).^(n));
   oqam_pre(n,2)=imag(qammod_atas(n))*((j).^(n));
   end
end
```

Namun, yang membedakan proses di dalam pra pengolahan OQAM pada simbol ganjil dan genap, terletak pada tahap pergeseran setengah fasa. Setiap simbol genap, pergeseran fasa terjadi pada bilangan imajiner. Sedangkan, pergeseran setengan fasa pada bilangan riil terjadi pada setiap simbol ganjil. Proses pertama pada pra pengolahan OQAM bertujuan untuk membuat simbol baru. Pada operasi kedua simbol baru yang telah didapat dikalikan dengan $\theta_{k,n}$ (persamaan 2.2), untuk mendapatkan nilai riilnya.

3.4.1.5 Sintesis Bank Filter

Proses Sintesis Bank Filter seperti pada Gambar 3.6, terdiri dari 2 proses yaitu proses kebalikan transformasi *fourier* dan proses filter atau bank filter.



Gambar 3. 6 Proses Sintesis Bank Filter

Adapaun kode program pada sintesis bank filter, sebagai berikut:

```
%Transformation Block
ifft_out = ifft(oqam_pre);
%Filter
z=1; %contoh filter = 1
for x=1:length(ifft_out)
    ppn_sfb(x,:)=ifft_out(x)*z;
end
```

“oqam pre” merupakan variabel untuk menjelaskan simbol baru hasil dari keluaran pra pengolahan OQAM. Simbol baru tersebut, masing-masing dikalikan dengan kebalikan proses transformasi fourier yang disimbolkan dengan variabel “ifft_out”. Kemudian setiap simbol keluaran IFFT dengan nama variabel “ppn_sfb” akan dikenakan dengan filter. Filter pada simulasi ini diasumsikan menggunakan filter ideal atau filter kotak. Sehingga simbol yang ada, akan langsung diteruskan tanpa adanya data yang dihilangkan.

3.4.1.6 Pengubah Paralel ke Seri (P/S)

Pengubah paralel ke seri di sisi pengirim, berfungsi untuk mengubah keluaran dari sintesis bank filter yang semula bentuknya data paralel menjadi bentuk seri, untuk dapat dikirimkan menjadi 1 baris. Pengubahan bentuk dari paralel menjadi seri dapat menggunakan program berikut:

```
ps_tx = reshape(ppn_sfb,1,[]);
```

"ps_tx" merupakan hasil pengubahan bentuk paralel menjadi seri. "ppn_sfb" merupakan variabel data masukan yang berasal dari keluaran proses analisis bank filter.

3.4.1.7 Pemodelan Kanal Transmisi

Kanal transmisi yang digunakan pada simulasi ini adalah kanal AWGN. Pada kanal AWGN diasumsikan memiliki derau yang terdistribusi normal (*Gaussian*). Derau AWGN terdistribusi normal dengan nilai rata-rata adalah nol. Derau ini bersifat acak dan bersifat menambahkan sinyal asli seperti yang telah dijelaskan pada persamaan 2.5. Kode program yang digunakan untuk mendapatkan sinyal yang diterima yaitu:

```
rx = (hn * source)+noise;
```

"source" merupakan simbol yang dikirim dari antena pengirim, "hn" merupakan kanal AWGN dengan Panjang data yang telah disamakan dengan panjang data dari data source dan derau AWGN berasal dari data acak yang telah disesuaikan dengan panjang data "source".

3.4.2 Penerima

3.4.2.1 Deteksi Simbol Pemaksaan Nol (*Zero Forcing*)

Proses deteksi simbol dilakukan dengan mengolah sinyal secara digital untuk mendapatkan nilai sinyal aslinya. Pengolahan sinyal ini dapat dilakukan dengan berbagai macam algoritma deteksi simbol seperti, *Zero Forcing* (ZF), *Least Square* (LS), *Minimum Mean Square Error* (MMSE), dan lain- lain. Pada simulasi ini difokuskan pada algoritma yang paling sederhana yaitu *Zero Forcing* (ZF). Algoritma ini cukup mudah digunakan, seperti pada persamaan 2.7 yaitu hanya dengan menginverskan kanal H dan dikalikan dengan sinyal kirim kemudian

ditambahkan dengan derau AWGN. Kode program untuk deteksi simbol pemaksaan nol, sebagai berikut:

```
rx2 = (1./h).*rx;
```

"rx" merupakan variabel yang digunakan untuk menampung data terima yang telah mendapatkan penambahan derau, "rx2" merupakan variabel baru yang digunakan untuk menampung data yang diterima dari antena penerima setelah melalui proses deteksi simbol. "h" merupakan kanal AWGN dengan panjang data yang telah disamakan dengan panjang data dari "data source" atau data yang dikirim. Dan derau AWGN berasal dari data acak yang telah disesuaikan dengan panjang data "source".

3.4.2.2 Pengubah Data Seri menjadi Paralel (S/P)

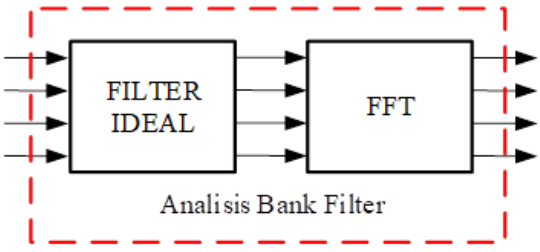
Pengubah data seri menjadi paralel di sisi penerima, berfungsi untuk mengubah keluaran dari antena penerima bentuk paralel. Untuk dapat dilanjutkan ke proses selanjutnya yaitu proses analisis bank filter. Kode program untuk mengubah data seri menjadi paralel, dapat menggunakan perintah "reshape" seperti berikut:

```
sp_rx2 = reshape(rx2,[],1);
```

"sp_rx2" merupakan hasil pengubahan data seri menjadi paralel. "reshape(rx2,[],1)" digunakan untuk mengubah data pada variabel "rx2" menjadi banyak baris (dilambangkan dengan tanda []) menjadi 1 kolom.

3.4.2.3 Analisis Bank Filter

Proses Analisis Bank Filter merupakan kebalikan dari proses sintesis bank filter. Analisis bank filter, terdiri dari 2 proses yaitu proses *transformasi* fourier dan proses filter atau bank filter seperti ditunjukkan pada Gambar 3.7



Gambar 3. 7 Proses Analisis Bank Filter

Untuk dapat melakukan proses tersebut dapat menggunakan kode program:

```
%% Filter
z=1; %contoh filter = 1
for x=1:length(sp_rx)
    ppn_afb(:,x)= sp_rx(x)*z;
end
%% Transformasi FFT
fft_out = fft(ppn_afb);
ppn_afb_out=reshape(fft_out,a,b);
```

Simbol baru hasil keluaran S/P yaitu "`sp_rx`", masing-masing simbol akan dikenakan dengan filter. Filter pada bagian penerima, diasumsikan juga menggunakan filter ideal atau filter kotak. Sehingga, simbol yang ada akan langsung diteruskan tanpa adanya data yang dihilangkan. Selanjutnya keluaran filter dengan variabel "ppn_afb" dikalikan dengan proses transformasi *fourier* (FFT). Kemudian setiap simbol keluaran FFT dengan variabel "ppn_afb_out", dikirimkan untuk diproses selanjutnya pada blok diagram pasca pengolahan OQAM.

3.4.2.4 Pasca Pengolahan OQAM

Pasca pengolahan OQAM merupakan kebalikan dari bagan pra pengolahan OQAM. Blok pasca pengolahan seperti yang ditunjukkan pada Gambar 2.9, yang memiliki dua struktur yang berbeda berdasarkan urutan kanal ganjil atau genap.

Proses pasca pengolahan OQAM dapat dilakukan dengan menggunakan kode program:

```
for n = 1 : length(ppn_afb_out)
if (rem(n, 2) == 1) %odd
   oqam_post=ppn_afb_out*((-1*j).^(n));
   oqam_post=real(oqam_post);
   oqam_post_atas(n,1)=oqam_post(n,2)+j*oqam_post(n,1);
else %even
   oqam_post=ppn_afb_out*((-1*j).^(n));
```

```
   oqam_post=real(oqam_post);
oqam_post_atas(n,1)=oqam_post(n,1)+j*oqam_post(n,2);
end
end
```

Operasi pertama adalah perkalian dengan kebalikan dari $\theta_{k,n,}$ untuk memperoleh nilai riil dari masing-masing simbol. Operasi kedua adalah pengubahan bilangan riil menjadi bilangan kompleks, di mana dua simbol bernilai riil berturut-turut dikalikan dengan *j* agar membentuk simbol bernilai kompleks. Konversi riil ke kompleks ini bertujuan mengembalikan ke posisi simbol semula dengan cara merurunkan jumlah (*downsampling* sebanyak 2).

3.4.2.5 Pengawa Peta 16 QAM

Pada bagian penerima, terjadi proses kebalikan yaitu dari pemeta 16 QAM, yaitu proses pengawa peta 16 QAM. Pada proses ini, setiap simbol akan dipetakan kembali untuk membentuk empat bit data kembali. Proses pengambilan keputusan pada pengawa peta 16 QAM, digunakan untuk menentukan simbol mana yang sebenarnya dikirimkan oleh antena pengirim. Pengambilan keputusan diperlukan karena adanya pengaruh kanal dan derau yang mengakibatkan simbol QAM di bagian penerima menjadi tidak asli dibandingkan keluaran pemeta 16 QAM pada pengirim. Hasil pengambilan keputusan ini, berupa komponen riil dan imajiner yang kemudian dikembalikan kembali menjadi bit-bit data. Adapun aturan pengambilan keputusan seperti tabel 3.3 dibawah ini:

Tabel 3. 3 Pengawa Petaan 16-QAM

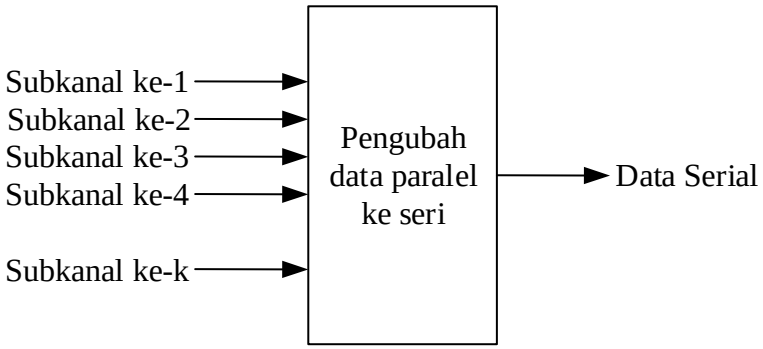
No	Riil (x)	Imajiner (y)	Keluaran
1	0,6324 > x > 0	0 < y < 0,6324	0 0 0 0
2	-0,6324 < x < 0	0 < y < 0,6324	0 0 0 1
3	0,6324 > x > 0	0 > y > -0,6324	0 0 1 0
4	-0,6324 < x < 0	0 > y > -0,6324	0 0 1 1
5	x > 0,6324	0 < y < 0,6324	0 1 0 0
6	x < -0,6324	0 < y < 0,6324	0 1 0 1

Tabel 3. 3 Pengawa Petaan 16-QAM (Lanjutan)

No	Riil (x)	Imajiner (y)	Keluaran
7	x > 0,6324	0 > y > -0,6324	0 1 1 0
8	x < -0,6324	-0,6324 < y < 0	0 1 1 1
9	0,6324 > x > 0	y > 0,6324	1 0 0 0
10	-0,6324 < x < 0	y > 0,6324	1 0 0 1
11	0,6324 > x > 0	y < -0,6324	1 0 1 0
12	-0,6324 < x < 0	y < -0,6324	1 0 1 1
13	x > 0,6324	y > 0,6324	1 1 0 0
14	x < -0,6324	y > 0,6324	1 1 0 1
15	x > 0,6324	y < -0,6324	1 1 1 0
16	x < -0,6324	y < -0,6324	1 1 1 1

3.4.2.6 Pengubah Paralel menjadi Serial (P/S)

Blok pengubah paralel menjadi seri ditunjukkan pada gambar 3.10. Fungsinya adalah untuk mengubah bentuk dari yang semula bentuk bit paralel menjadi bit seri, dengan mengelompokkan empat bit – empat bit biner keluaran dari pengawa petaan 16 QAM, menjadi satu baris sebagai keluaran dari proses pengawa-petaan 16-QAM ke data serial bit.



Gambar 3. 8 Pengubah Data Paralel menjadi Seri

3.4.2.7 Data Keluaran

Di sisi keluaran, bit-bit biner tersebut dikembalikan kembali ke bentuk semula yaitu sinyal audio. Hasil bit - bit biner keluaran sinyal tersebut, dapat digunakan untuk membandingkan bit keluaran dengan bit masukan untuk

mendapatkan nilai SNR terhadap BER. Dan kemudian bit biner tersebut dikembalikan lagi ke dalam bentuk semula yaitu sinyal audio untuk dapat diperoleh hasil dari perbaikan dari sistem tersebut. Untuk dapat mengubah bit keluaran menjadi sinyal audio dapat menggunakan kode program berikut:

```
aurecbin = reshape(char(data_output2 +
'0'),nbits,numel(data_output)/nbits);
aurec = bin2dec(aurecbin');
aurecvol = (aurec ./ (2^(nbits - 1)));
aurecvol(rr, 1) = aurecvol(rr, 1) - 1;
rec_audio{i, 1} = aurecvol
```