VIII. -PRAKTIKUM 7— Konversi Simbol ke Biner

VIII.1 PENDAHULUAN

Pada PRAKTIKUM 6 yang lalu, telah dihasilkan besaran f1, drift1, shift1 dan sync1 yang sudah mengalami proses rafinasi berkali-kali. Bermodalkan besaran-besaran yang sudah dirafinasi ini, kemudian dimulai proses deteksi dengan metoda korelasi dengan memanfaatkan f1 sebagai acuan pembangkitan osilator lokal, drift1 sebagai acuan pemodelan drift osilator, shift1 sebagai jitter offset. Namun sebelumnya masih dicoba sekali lagi *time alignment* dengan *angka alignment* lebih halus dan dengan acuan koefisien korelasi-paket sync1. Jika ternyata didapatkan harga shift1 yang baru dengan melihat koefisien korelasi paket sync1 lebih besar, maka yang akan dipakai adalah shift1 yang ter-update tersebut.

Selanjutnya adalah tahap *vernier/fine timing alignment* ini sebenarnya sama dengan tahap alignment frekuensi dan alignment pewaktu pada tema PRAKTIKUM 6, namun dengan unit pewaktu (jumlah geseran sampel) lebih kecil. Ketika alignment pewaktu dianggap sudah cukup (dengan acuan metrik koefisien korelasi sync1), maka dilakukan korelasi tahap akhir yang diikuti dengan konversi dari kode simbol ke kode biner-terkuantisasi.

Setelah melalui proses serial yang panjang untuk sampai pada proses yang ditunjukkan pada PRAKTIKUM 7 ini, saat ini kita masuk pada proses dengan outputnya adalah deretan sampel power sinyal dimana setiap sampel akan dikuantisasi menjadi bilangan integer 0 hingga 256. Proses konversi serial data analog ke serangkaian nilai biner-terkuantisasi dilakukan dengan menggunakan pendekatan statistik. Para mahasiswa harus melacak dan mengamati sintaksis pemrograman dari kode sumber yang tersedia, bagaimana bentuk sintaksis pemrograman dari pendekatan statistik tersebut, agar supaya menjadi perangkat yang berfungsi sesuai dengan perencanaannya.

VIII.2 Vernier/fine Time Alignment (alignment pewaktu halus - offset pewaktu halus)

Proses vernier/fine time alignment dilaksanakan dengan cara memanggil fungsi:

`noncoherent_sequence_detection()`

Metrik untuk menentukan *shift1* yang paling optimum tetap sama dengan yang dilaksanakan oleh fungsi `sync_and_demodulate()` yakni dengan menghitung korelator-simbol di domain waktu, untuk selanjutnya dipakai sebagai dasar perhitungan **korelator-simbol**. Proses pembangkitan sinusoidal sebagai acuan proses korelasi sama dengan yang dipakai pada fungsi `sync_and_demodulate()`, perbedaan ada pada cara menghitung **korelator-paket** yang akan dibahas berikut ini.

Perhitungan korelator-simbol masih sama dengan yang dilaksanakan pada fungsi `sync_and_demodulate()sbb`,

```
is[0][i]=0.0; qs[0][i]=0.0;
is[1][i]=0.0; qs[1][i]=0.0;
```

```
    is[2][i]=0.0; qs[2][i]=0.0;
    is[3][i]=0.0; qs[3][i]=0.0;


    for (j=0; j<256; j++) {
        k=lag+i*256+j;
        if( (k>0) && (k<np) ) {
            is[0][i]=is[0][i] + id[k]*c0[j] + qd[k]*s0[j];
            qs[0][i]=qs[0][i] - id[k]*s0[j] + qd[k]*c0[j];
            is[1][i]=is[1][i] + id[k]*c1[j] + qd[k]*s1[j];
            qs[1][i]=qs[1][i] - id[k]*s1[j] + qd[k]*c1[j];
            is[2][i]=is[2][i] + id[k]*c2[j] + qd[k]*s2[j];
            qs[2][i]=qs[2][i] - id[k]*s2[j] + qd[k]*c2[j];
            is[3][i]=is[3][i] + id[k]*c3[j] + qd[k]*s3[j];
            qs[3][i]=qs[3][i] - id[k]*s3[j] + qd[k]*c3[j];
        }
  }
}
```

Selanjutnya **korelator-paket** dihitung dengan sintaksis sbb,

```
for (i=0; i<162; i=i++) {
  for (j=0;j<2;j++) {
  xi[j]=0.0; xq[j]=0.0;

  b=j;
  itone=pr3[i]+2*b;
  xi[j]=xi[j]+is[itone][i] ; //nilai korelasi infase
  xq[j]=xq[j]+qs[itone][i] ; //nilai korelasi quadratur

  p[j]=xi[j]*xi[j]+xq[j]*xq[j];
  p[j]=sqrt(p[j]);
  }

  /************************************************/
  /* hipotesis --> fsymb[i]== besar == 1; kecil== 0 */
  fsymb[i]=p[1]-p[0];
  /************************************************/
}
for (i=0; i<162; i++) {              //Normalize the soft symbols
  fsum=fsum+fsymb[i]/162.0;
  f2sum=f2sum+fsymb[i]*fsymb[i]/162.0;
}
fac=sqrt(f2sum-fsum*fsum);
for (i=0; i<162; i++) {
  fsymb[i]=symfac*fsymb[i]/fac;
  if( fsymb[i] > 127) fsymb[i]=127.0;
  if( fsymb[i] < -128 ) fsymb[i]=-128.0;
  symbols[i]=fsymb[i] + 128;
  }
```

VIII.3 Konversi Simbol ke Digital Biner-terkuantisasi

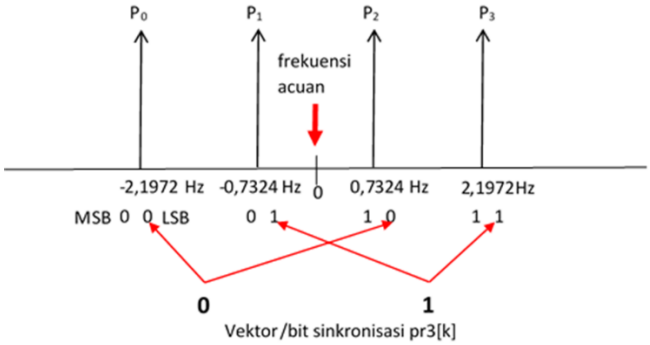
Konversi simbol ke besaran digital biner-terkuantisasi dilaksanakan dengan metode korelasi antar sinyal input dengan 4 sinyal sinusoidal yang dibangkitkan dari sistem osilator lokal. Bit sinkronisasi dipakai untuk mempersempit pilihan pada proses pengambilan keputusan. Manfaat bit sinkronisasi secara lebih jelas dapat diamati pada hasil printf dari program ./wsprd, dimana nampak jelas bahwa jika bit-sinkronisasi = 0 maka korelator-simbol yang diperhatikan hanya korelasi dengan P_0 dan P_2, sedangkan jika bit-sinkronisasi = 1 maka korelator-simbol yang diperhatikan hanya korelasi dengan P_1 dan P_3. Jika dilihat dari Gambar VIII-1, nampak bahwa bit-bit-sinkronisasi dipasang untuk memperlebar sebaran frekuensi FSK yang dibangkitkan oleh modulator di sisi pemancar WSPR, sekaligus sebagai pedoman saat proses demodulasi di sisi sistem penerimanya.

Tabel kebenaran untuk proses konversi dari dari besaran koefisien korelasi-simbol menjadi menjadi besaran digital biner dapat dilihat pada Tabel VIII-1 dibawah ini.

Tabel VIII-1: Konversi dari koefisien korelasi-simbol ke digital biner

Diketahui	**Jika bit message**	**Indeks frekuensi**	**Koefisien korelasi-simbol**	Hipotesis **fsymb[i] =**
pr3[i]	b	Itone=pr3[i]+2*b	p[b]=P_{itone}	p[1]-p[0]
0	0	0	p[0]=P_0	Jika kecil -> “0”
0	1	2	p[1]=P_2	Jika besar -> “1”
1	0	1	p[0]=P_1	Jika kecil -> “0”
1	1	3	p[1]=P_3	Jika besar -> “1”

Dimana P_{itone} adalah spektrum FSK WSPR seperti pada sketsa spektrum dibawah ini, yang sudah beberapa kali disebutkan pada praktikum-praktikum sebelumnya, sbb,



Gambar VIII-1: Sketsa sebaran spektrum frekuensi FSK pada sistem modulasi yang dipakai WSPR

Setelah didapat nilai hipotesis fsymb[i], selanjutnya dilaksanakan proses normalisasi nilai fsymb[i] agar nilainya dapat diakomodasi pada skala unsigned char 8 bit dengan cara sbb,

Nilai rerata

$$fsum = \frac{1}{162}\sum_{i=0}^{161} fsymb[i]$$

Nilai power rerata

$$f2sum = \frac{1}{162}\sum_{i=0}^{161} (fsymb[i])^2$$

Nilai standar deviasi

$$fac = \sqrt{f2sum - (fsum)^2}$$

Nilai fsymb[i] yang dinormalisasi,

$$fsymb[i] = symfac \times \left(\frac{fsymb[i]}{fac}\right)$$

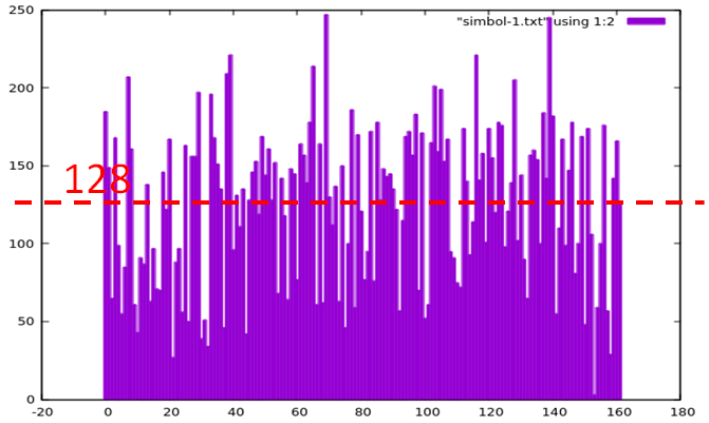
Dimana symfac adalah faktor normalisasi soft-symbol yang oleh desainernya ditetapkan harga 50. Selanjutnya masih dilakukan penyesuaian nilai agar lebih memudahkan konversi menjadi besaran digital biner,

$$fsymb[i] = \begin{cases} 127, & jika\ fsymb[i] > 127 \\ -128, & jika\ fsymb[i] < -128 \end{cases}$$

$$symbols[i] = fsymb[i] + 128$$

$$i = 0, 1, 2, \ldots\ldots, 161$$

Dibawah ini adalah gambar contoh deretan simbol yang sudah dinormalisasi sebelum dikonversi menjadi besaran digital biner-ternormalisasi, yang dipresentasikan menjadi bilangan integer 8 bit tanpa tanda, atau dengan kata lain bahwa nilai deretan output biner-terkuantisasi berkisar antara 0 s/d 255, yang berfluktiasi sepanjang 162 segmen waktu yang mereprentasi digit biner yang dikirimkan oleh pemancar. Nilai ouput tegangan ini kemudian akan dikonversi menjadi nilai loglikelihood untuk keperluan proses dekode konvolusi dengan metoda Fano.



Gambar VIII-2: Deretan 162 simbol yang sudah dinormalisasi sebelum dikonversi menjadi besaran digital biner.

VIII.4 Proses Deinterleave

Dalam paper-paper mengenai aplikasi kode konvolusi, telah banyak disebut bahwa kode konvolusi memang sangat handal untuk mengoreksi eror bit tunggal, namun sangat rentan terhadap eror burst, oleh sebab itu dalam praktek pemanfaatan kode konvolusi ini selalu didahului dengan proses interleave yang dimaksudkan untuk menjauhkan bit-bit penting pada posisi yang berdampingan/bertetangga.

Algoritma *Interleave* dan *Deinterleave* mempunyai struktur yang sama, yang artinya bahwa jika suatu sekuen data dikenakan proses interleave untuk keperluan mengacak posisi bit-bit nya, maka dengan cara mengenakannya pada proses deinterleave, posisi acak bit-bit tersebut akan kembali kepada deretan normal seperti sedia kala.

Pada Bab I sudah dijelaskan algoritma perihal interleave, dimana bagian terpentingnya adalah proses *bit reverse* dari angka indeks posisi deretan bit-bit. Proses bit reverse adalah proses-proses diaras bit (bitwise) yang masih dapat dilakukan oleh pemorograman C. Banyak sekali turunan algoritma bit reverse, namun untuk mengoptimasi kecepatan, algoritma harus dipilih dengan memperhatikan panjang bit bit data dari prosesor.

Rumus bit reverse yang dipakai pada fungsi Interleave/Deinterleave program WSPR adalah sbb,

```
 unsigned char i; // reverse this byte
unsigned char j; // reversed byte

j = ((i * 0x80200802ULL) & 0x0884422110ULL) * 0x0101010101ULL >> 32;
```

Algoritma bit reverse ini diambil dari:

http://graphics.stanford.edu/~seander/bithacks.html#ReverseByteWith32Bits

Untuk mengamati input/output dari proses Interleave/Deinterleave, jalankan program “inter.c”. Program ini dengan memakai algoritma bit reverse yang sama dengan WSPRD, mencoba untuk

melaksanakan simulasi pengacakan deretan 162 array integer 8 bit (char). Percobaan pengacakan data ini akan dilaksanakan pada sesi percobaan dibawah ini.

VIII.5 LANGKAH PERCOBAAN

1. Siapkan modul Raspberry Pi, nyalakan dan buka satu atau beberapa terminal window.
2. Masuk ke direktori “PRAKTIKUM-NONC”
3. Siapkan beberapa file WSPR hasil perekaman yang berasal dari beacon laboratorium dan yang berasal dari tempat yang jauh, dan letakkan pada direktori ini.
4. Kompilasi program “./wsprd” dengan menjalankan “Makefile”
   %> make -k
5. Jalankan program “./wsprd”
   %> ./wsprd xxxxxx.WAV \r atau
   %> ./wsprd xxxxxx.raw \r
6. Amatilah dari print out deretan koefisien koelasi-simbol yang tercetak. Apakah perbedaan antara bit-sinkronisasi = 0 dan bit-sinkronisasi =1.
7. Kompilasi “./inter.c” dan jalankan program “./inter”
   %> ./inter \r
8. Amatilah print out dari hasil eksekusi program “./inter” tersebut.

VIII.6 Laporan Praktikum

1. Pada fungsi noncoherent_sequence_detection(), jelaskan maksud parameter itone,pr3[i],b, is[itone][i], qs[itone][i], xi[j], xq[j], p[j].
2. Tunjukkan manfaat dari bit-sinkronisasi.
3. Tunjukkan deretan simbol yang siap untuk dikonversi menjadi besaran digital biner.
4. Tunjukkan sintaksis untuk implementasi proses normalisasi nilai fsymb[i] agar dapat disimpan pada unsigned char 8 bit.
5. Jelaskan maksud dari simulasi program “./inter”.
6. Laporan praktikum dikumpulkan sebelum PRAKTIKUM 8 dilaksanakan.