**BÁO CÁO BÀI TEST FFT256**

# I. Tổng quan các bước:

B1. Tạo tín hiệu x[] bằng hàm Gen\_sig().

B2. Biến đổi FFT với hàm FFT() và lưu vào X[], tính thời gian chạy.

B3. Tính toán lại độ lớn của X[] và lưu vào P[].

B4. Lưu kết quả vào file FFT\_For.txt nhằm trực quan hóa dữ liệu.

1. Gen\_sig(x);
3. // Perform FFT and measure run time
4. start = clock();
5. FFT(x, X);
6. end = clock();

9. time\_consume = (double)(end - start)/CLOCKS\_PER\_SEC;
10. printf("this function take %lf seconds", time\_consume);


14. // Nomarlize the amplitude
15. int N\_oneside = N/2;
16. \_Complex X\_oneside[N/2];
18. for (int i = 0; i < N\_oneside; ++i)
19. {
20. X\_oneside[i] = X[i] / N\_oneside;
21. }

24. // Amplitude of X
25. for(int i = 0; i<N; i++) P[i] = cabs(X\_oneside[i]);

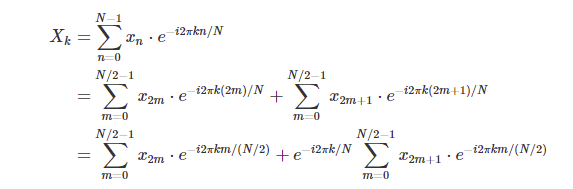

29. // write to .txt file
30. FILE \*f = fopen("FFT\_For.txt", "w");
32. for (int i = 0; i < N; ++i)
33. {
34. fprintf(f, "%f,", x[i]);
35. }
36. fprintf(f, "\n");
37. for (int k = 0; k < N\_oneside ; k++) fprintf(f,"%f,", P[k]);
39. fclose(f);

# II. Chi tiết thực hiện:

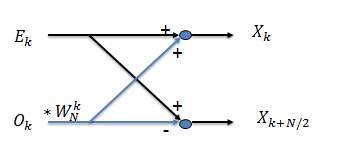
B1: Tạo tín hiệu x[] là tổng 2 tín hiệu sin với:

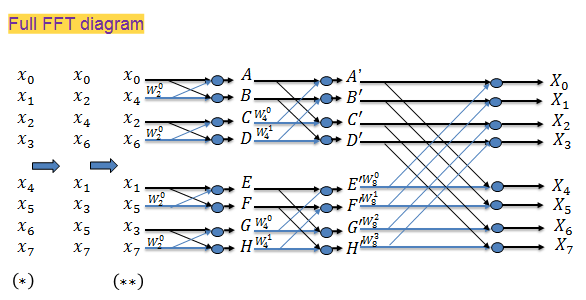
* Tần số freq1 = 10, freq2 = 5.
* Biên độ lần lượt là 3 và 5.
* Thời gian t được chia đều theo số mẫu N.

1. void Gen\_sig(double \*x)
2. {
3. double t, freq1 = 10, freq2 = 5;
5. for (int n=0 ; n<N ; n++)
6. {
7. t = (double)n \* (1/(double)N);
9. x[n] = 3\*sin(freq1\*PIx2\*t) + 5\*sin(freq2\*PIx2\*t);
11. }
13. }

B2: Ta có công thức: 

Và sơ đồ :





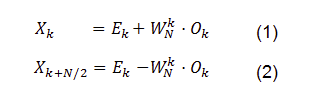
* Ta thực hiện tách chuỗi tín hiệu vào ra 2 thành phần vị trí chẵn và lẻ, đến khi chỉ còn 1 thành phần thì giá trị đó chính bằng biến đổi FFT của nó. Do đó ta gán tín hiệu FFT X[] bằng tín hiệu vào x[].

1. for(i = 0; i < N; ++i){
2. X[i] = x[i];
3. }

* Vì vị trí các giá trị trong tín hiệu vào đã bị thay đổi sau khi chia thành phần chẵn lẽ nên ta cần đổi lại thứ tự trong X[] bằng hàm reBit(), hàm này thực hiện đảo bit và trả về giá trị (index) tương ứng đúng theo thứ tự cần có.

1. for(i = 0; i < N; ++i)
2. {
3. j = revBit(logN, i);
5. if(i < j)
6. {
7. temp = X[j];
8. X[j] = X[i];
9. X[i] = temp;
10. }
11. }

* Từ đây trở đi, X được tính theo công thức ở trên:



Với E: phần chẵn

O: phần lẻ

k: hệ số tương ứng tần số

1. for (int step = 1 ; step < N; step<<=1)
2. {
3. \_Complex w = 1 ,
4. w\_base = cexp(-I \* PIx2 / (step\*2));
6. for(k = 0; k < step; ++k)
7. {
8. W[k] = w;
9. w \*= w\_base;
10. }
12. even\_index = 0
13. odd\_index = even\_index + step;
15. while(even\_index < N)
16. {
17. for (i=0 ; i < step ; ++i)
18. {
19. X\_next[even\_index + i] = X[even\_index + i] + W[i] \* X[odd\_index + i];
20. X\_next[even\_index + i + step] = X[even\_index + i] - W[i] \* X[odd\_index+ i];
22. }
23. even\_index += 2 \* step;
24. odd\_index = even\_index + step;
25. }
26. for(i = 0; i < N; ++i)
27. {
28. X[i] = X\_next[i];
29. }
30. }

* Chia việc tính các cột như trên sơ đồ thành các bước(step). Như vậy có tất cả log2(N) step trừ đi step đầu tiên khi gán x[] với X[]. Tạo một vòng lặp với step chạy từ 1 đến log2(N), step bằng 1/2 số điểm tại bước tương ứng (vd: 1,2,4,8,….).
* Nhận thấy W có thể tính toán mà chỉ cần phị thuộc vào k và N nên ta tính các hệ số này trước (dòng 6 đến 10).
* Dòng 12 đến 25 thực hiện lặp để tính lần lượt các phần tử theo công thức ở trên sử dụng X[] ở bước trước, lưu vào X\_next[] và lưu ngược trở lại X[] để thực hiện bước tiếp theo.

B3: Do X[] lặp lại từ N/2 nên ta chỉ lấy tần số từ 0 đến N/2-1, tính độ lớp bằng hàm cabs() rồi lưu vào P[].

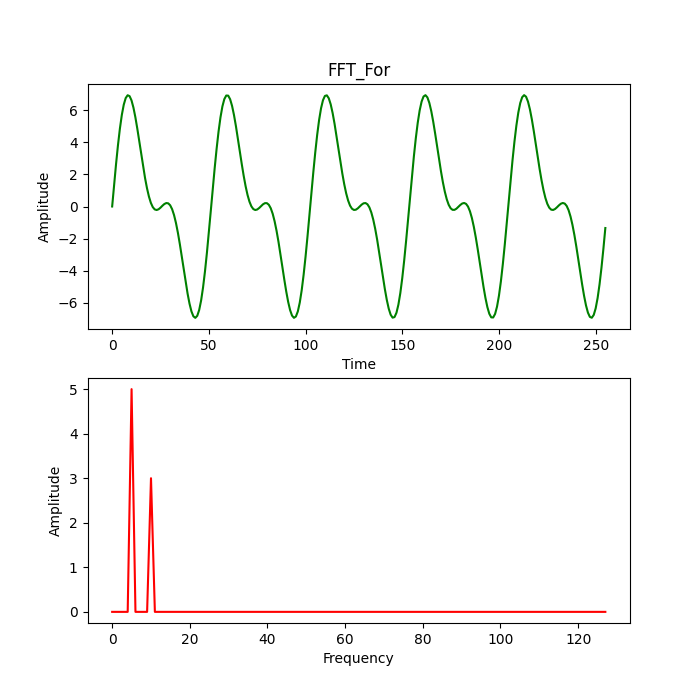
1. // Nomarlize the amplitude
2. int N\_oneside = N/2;
3. \_Complex X\_oneside[N/2];
5. for (int i = 0; i < N\_oneside; ++i)
6. {
7. X\_oneside[i] = X[i] / N\_oneside;
8. }

11. // Amplitude of X
12. for(int i = 0; i<N; i++) P[i] = cabs(X\_oneside[i]);

B4: Lưu file gồm tín hiệu vào x[] và độ lớn của tín hiệu FFT X[].

# III. Kết quả:

* Biểu đồ:



* Thời gian chạy(4096 điểm):

1. DFT: 1,5 – 1,6s
2. FFT đệ quy: 0,003 - 0,008s
3. FFT không đệ quy: 0,001 – 0,005s