**Trường Đại Học Bách Khoa**
**Khoa Điện – Điện Tử**
**Bộ Môn Viễn Thông**


# XỬ LÝ SỐ TÍN HIỆU

## Tài liệu thí nghiệm



**BỘ MÔN VIỄN THÔNG- 2015**

# Mục lục

# Mục lục hình vẽ

# Mục lục các bảng

# GIỚI THIỆU

Các bộ xử lý số tín hiệu được sử dụng trong rất nhiều ứng dụng thực tế, từ truyền thông và điều khiển cho đến xử lý tiếng nói và hình ảnh. Hầu hết các thiết bị gia dụng hiện nay cũng tích hợp các bộ xử lý số tín hiệu. Chúng được sử dụng trong điện thoại di động, máy ảnh số, HDTV, radio, truyền fax, các modem, máy in, máy trợ thính và nhiều thiết bị khác.

Hệ thống xử lý số tín hiệu cơ bản bao gồm một bộ biến đổi A/D để thu nhận tín hiệu vào. Sau đó, dạng biểu diễn số của tín hiệu vào sẽ được xử lý bởi một bộ xử lý số tín hiệu và tín hiệu ra được đưa qua bộ biến đổi D/A. Hệ thống cơ bản này cũng bao gồm một bộ lọc ngõ vào chống chồng lấn phổ và một bộ lọc ngõ ra để khôi phục tín hiệu đã xử lý. Trong các bài thí nghiệm này, chúng ta được trang bị kit TMS320C6713 của Texas Instruments. Kit TMS320C6713 là một công cụ mạnh với các phần cứng và phần mềm cần thiết cho xử lý tín hiệu thời gian thực. Nó là một hệ thống xử lý số tín hiệu hoàn chỉnh, bao gồm một bộ xử lý số dấu chấm động C6713 và bộ codec 32-bit stereo TLV320AIC23 (gọi tắt là AIC23) cho việc xuất nhập tín hiệu.

Qua các bài thí nghiệm này, hi vọng các bạn sinh viên sẽ hiểu rõ thêm các khái niệm đã học trong môn xử lý số tín hiệu, nắm được các bước cơ bản trong việc thực hiện một ứng dụng xử lý số tín hiệu lên một bộ xử lý số, như chip C6713 của Texas Instruments. Ngoài ra, các bạn sinh viên có thể ứng dụng các lý thuyết xử lý số tín hiệu để thực hiện các ứng dụng xử lý tín hiệu âm thanh và hình ảnh.

1

# GIỚI THIỆU KIT DSP VÀ MATLAB



## 1. MỤC ĐÍCH THÍ NGHIỆM

- Giới thiệu tổng quan về kit xử lý tín hiệu kit C6713 DSK.
- Biết được các bước thực hiện giải thuật trên kit C6713 DSK.
- Giới thiệu tổng quan về Matlab và SPTool để thiết kế bộ lọc.
- Giới thiệu sinh viên thực thi chương trình trên Matlab với các lệnh cơ bản về xử lý âm thanh, hình ảnh.

## 2. THIẾT BỊ THÍ NGHIỆM

| STT | Tên thiết bị | Số lượng |
|---|---|---|
| 01 | Máy vi tính | 01 |
| 02 | Kit C6713 DSK | 01 |

## 3. GIỚI THIỆU KIT XỬ LÝ SỐ C6713 DSK (DSP STARTER KIT)

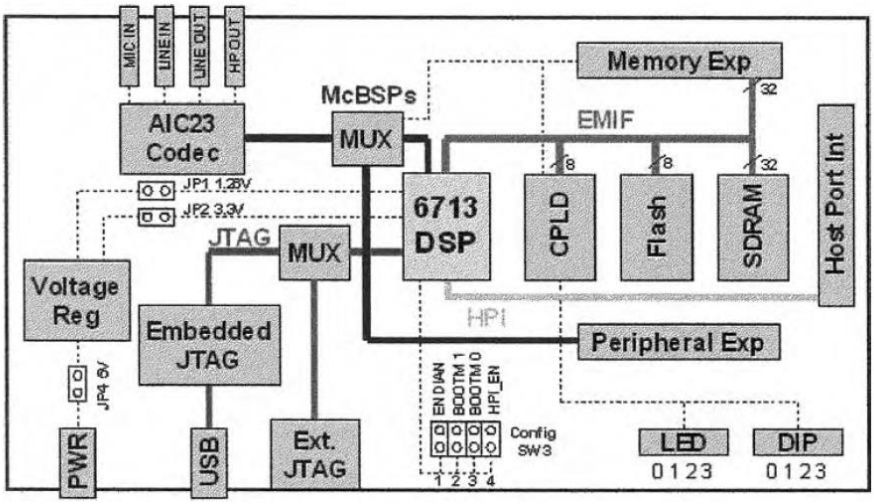
### 3.1. Kit C6713 DSK

Kit DSK là một hệ thống DSP hoàn chỉnh. Board DSK bao gồm bộ xử lý số dấu chấm động C6713 và bộ codec 32 bit stereo TLV320AIC23 (AIC23) để xuất nhập. Bộ codec onboard AIC23 sử dụng kỹ thuật sigma – delta để biến đổi A/D và D/A. Nó được kết nối với một đồng hồ hệ thống 12 – MHz. Tần số lấy mẫu có thể thay đổi từ 8 đến 96 KHz.

Board DSK bao gồm 16MB SDRAM và 256kB Flash memory. Bốn jack cắm trên boad cho phép xuất nhập: MIC IN (microphone input), LINE IN (line input), LINE OUT (line output) và HEADPHONE (headphone output). Trạng thái của 4 dip switch trên DSK có thể đọc được từ chương trình. DSK hoạt động ở tần số 225 MHz. Trên board DSK cũng bao gồm các ổn áp cung cấp 1.26V cho nhân C6713 và 3.3V cho bộ nhớ và các ngoại vi.

Bộ xử lý TMS320C6713 dựa trên kiến trúc VLIW (very-long-instruction-word), phù hợp cho các giải thuật nặng về tính toán số. Bộ nhớ chương trình nội được tổ chức để mỗi chu kỳ có thể nạp 8 lệnh (instruction), mỗi instruction dài 32 bit.

Các bộ xử lý C67xx (ví dụ C6701, C6711 và C6713) thuộc về họ các bộ xử lý C6x dấu chấm động, trong khi đó C62xx và C64xx thuộc về họ các bộ xử lý C6x dấu chấm tĩnh. C6713 có thể xử lý cả dấu chấm động và dấu chấm tĩnh.




**Hình 1. Sơ đồ khối của DSK**

### 3.2. Code Composer Studio (CCS)

CCS là một môi trường phát triển tích hợp (IDE). CCS cung cấp các công cụ sinh mã, như một bộ biên dịch C, một chương trình assembler và một chương trình linker. Nó có khả năng đồ họa và hỗ trợ real-time debug. Nó cung cấp một công cụ phần mềm thuận tiện cho việc xây dựng và sửa lỗi chương trình.

Trình dịch C sẽ dịch chương trình nguồn viết bằng C (tập tin có kiểu .c) để tạo thành một tập tin nguồn assembly (kiểu .asm). Trình assembler sẽ tạo ra các tập tin đối tượng ngôn ngữ máy (.obj) từ các tập tin .asm. Trình linker sẽ kết hợp các tập tin đối tượng và các thư việc đối tượng để tạo ra một tập tin thực thi với kiểu .out. Tập tin thực thi này có thể được nạp và chạy trực tiếp trên bộ xử lý C6713.

Để tạo một project, người dùng có thể thêm vào các tập tin phù hợp. Các tùy chọn về compiler/linker có thể xác định dễ dàng. Một số tính năng debug có sẵn, như đặt các breakpoint và xem các biến; xem bộ nhớ, các thanh ghi và trộn C với assembly code; các kết quả đồ họa; và theo dõi thời gian thực thi.

Chúng ta sẽ làm việc với một vài kiểu tập tin khác nhau, bao gồm:

1. file.pjt: để tạo và xây dựng một project có tên là “file”
2. file.c: chương trình nguồn viết bằng C
3. file.asm: chương trình nguồn bằng ngôn ngữ assembly, được tạo bởi người dùng hoặc bởi bộ dịch C.
4. file.h: tập tin header
5. file.lib: tập tin thư viện
6. file.cmd: tập tin lệnh của linker, ánh xạ các section vào bộ nhớ

7. file.obj: tập tin đối tượng được tạo ra bởi assembler
8. file.out: tập tin thực thi được tạo ra bởi linker để nạp và chạy trên bộ xử lý C6713

### 3.2.1. Các tập tin hỗ trợ

Các tập tin hỗ trợ sau được chứa trong folder C:\CCStudio_v3.1\myprojects\source\support (trừ các tập tin thư viện) được sử dụng trong hầu hết các bài thí nghiệm.

1. *C6713dskinit.c*: chứa các hàm khởi động DSK, codec, các cổng nối tiếp và để xuất nhập. Tập tin này không được bao gồm với CCS.
2. *C6713dskinit.h*: tập tin header chứa các prototype của các hàm.
3. *C6713dsk.cmd*: Tập tin lệnh linker. Tập tin này có thể được sửa đổi khi sử dụng bộ nhớ ngoài thay cho bộ nhớ trong.
4. *vectors_intr.asm*: một tập tin vector bao gồm trong CCS đã được sửa đổi để quản lý ngắt. Có 12 ngắt, từ INT4 đến INT15, và ngắt 11 được chọn trong tập tin này. Chúng được dùng cho các chương trình có sử dụng ngắt.
5. *vectors_poll.asm*: tập tin vector cho các chương trình hỏi vòng (polling)
6. *rts6700.lib, dsk6713bsl.lib, csl6713.lib*: Tập tin thư viện hỗ trợ run – time, board và chip. Các tập tin này được cung cấp với CCS và được chứa trong các folder C6000\cgtools\lib, C6000\dsk6713\lib và C6000\csl\lib một cách tương ứng. (Thư mục C6000 nằm trong thư mục cài đặt của CCS, mặc định là C:\CCStudio_v3.1\)

### 3.2.2. Các ví dụ lập trình trên DSK

Sau đây là một số ví dụ lập trình để minh họa cho một số đặc tính của CCS và board DSK. Mục tiêu chính là để làm quen với các công cụ phần mềm và phần cứng.

**Ví dụ 1: Tạo tín hiệu dùng 8 điểm với điều khiển DIP Switch (sine8_LED)**

Ví dụ này tạo ra một tín hiệu hình sine bằng phương pháp tra bảng. Quan trọng hơn, nó minh họa vài đặc tính của CCS trong hiệu chỉnh, xây dựng một project, sử dụng các công cụ sinh mã và chạy một chương trình trên bộ xử lý C6713. Chương trình nguồn sine8_LED.c thực hiện việc tạo sóng sine có nội dung như dưới đây.

```
//Sine8_LED.c Sine generation with DIP switch control
#include "dsk6713_aic23.h" //support file for codec,DSK
Uint32 fs = DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
short loop = 0; //table index
short gain = 10; //gain factor
short sine_table[8]={0,707,1000,707,0,-707,-1000,-707};//sine values
void main()
{
    comm_poll(); //init DSK, codec, McBSP
    DSK6713_LED_init(); //init LED from BSL
    DSK6713_DIP_init(); //init DIP from BSL
    while(1) //infinite loop
    {
```

```
        if(DSK6713_DIP_get(0)==0) //=0 if switch #0 pressed
        {
            DSK6713_LED_on(0); //turn LED #0 ON
            output_sample(sine_table[loop]*gain);//output every Ts (SW0 on)
            if (++loop > 7) loop = 0; //check for end of table
        }
        else DSK6713_LED_off(0); //LED #0 off
    } //end of while (1)
}
```

**Giải thích chương trình**

Trong chương trình này, một bảng **sine_table** được tạo ra và chứa 8 điểm thể hiện giá trị của sin(t) tại t = 0, 45, 90, 135, 180, 225, 270 và 315 độ (tỷ lệ 1000). Trong hàm *main*(), một hàm khác, *comm_poll*, được gọi. Hàm này được chứa trong tập tin *c6713dskinit.c*. Nó khởi động DSK, bộ codec AIC onboard và các cổng nối tiếp trên bộ xử lý C6713.

Lệnh *while(1)* trong hàm *main* tạo một vòng lặp vô tận. Khi dip switch 0 được nhấn, LED 0 được bật lên và tín hiệu sine được tạo ra. Ngược lại, hàm *DSK6713_DIP_get(0)* sẽ là *false* và LED 0 sẽ tắt.

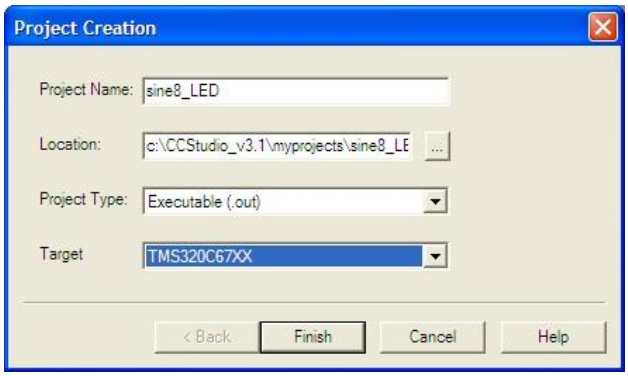
Hàm *output_sample*, chứa trong tập tin hỗ trợ *C6713dskinit.c*, được gọi để xuất giá trị dữ liệu đầu tiên trong bảng *sine_table[0]* = 0. Chỉ số lặp *loop* sẽ được tăng dần cho đến hết bảng và trở lại giá trị zero.

Mỗi chu kỳ lấy mẫu $T_s = 1/F_s = 1/8000 = 0.125$ms, giá trị của dip switch 0 được kiểm tra và một giá trị tiếp theo trong bảng *sine_table* (nhân với tỷ lệ *gain*) được xuất ra. Trong một chu kỳ tín hiệu, 8 giá trị dữ liệu ( cách nhau 0.125ms) được xuất ra để tạo một tín hiệu sine. Chu kỳ của tín hiệu sine thu được là $T = 8.(0.125\text{ms}) = 1$ms ứng với tần số $f = 1/T = 1$ kHz.

**Tạo project**

Phần này sẽ minh họa cách tạo một project mới, thêm các tập tin cần thiết để biên dịch project sine8_LED.

1. Trong CCS, chọn Project → New. Trong hộp thoại tạo Project, nhập tên project là sine8_LED, chọn Project Type là Executable (.out) và Target là TMSC67xx.




**Hình 2. Hộp thoại Project Creation**

CCS sẽ tự tạo ra một thư mục tên là *sine8_LED* trong thư mục *C:\CCStudio_v3.1\myprojects.*

2. Sau khi tạo project, cần chép các tập tin cần thiết vào trong thư mục *sine8_LED* vừa tạo trước khi thêm các tập tin này vào project.
   a. Chép các tập tin *sine8_LED.c* và *gain.gel* từ thư mục *myprojects\source\project_1* vào trong thư mục *sine8_LED* ở trên. Tập tin *sine8_LED.c* chính là tập tin nguồn chính được viết bằng C ở trên. Tập tin *gain.gel* được viết bằng một ngôn ngữ thông dịch gọi là GEL (General Extension Language), tập tin này sẽ tạo ra một giao diện có thanh trượt (slide) để cho phép thay đổi biến gain trong chương trình chính khi đang chạy chương trình trên kit.
   b. Chép các tập tin hỗ trợ: *c6713dskinit.h*, *c6713dskinit.c*, *vectors_poll.asm* và *c6713dsk.cmd* trong *myprojects\source\support* vào trong thư mục *sine8_LED*. Do chương trình này sử dụng cách lập trình hỏi vòng (polling) nên ta sử dụng tập tin hỗ trợ *vectors_poll.asm*. Trong trường hợp lập trình có ngắt, ta sẽ sử dụng tập tin *vectors_intr.asm.*
   c. Cuối cùng, chép các tập tin *dsk6713.h* và *dsk6713_aic23.h* trong *C6000\dsk6713\include* vào trong thư mục *sine8_LED*.
3. Sau khi đã chuẩn bị xong các tập tin cần thiết, trở về CCS để thêm các tập tin vào project. Để thêm tập tin vào project, chọn Project → Add Files to Project. Trong hộp thoại xuất hiện, vào thư mục *sine8_LED* ở trên.
   a. Chọn kiểu tập tin (*File of Types*) là *C Source Files* và chọn các tập tin *C6713dskinit.c* và *sine8_LED.c* rồi nhấn nút Open để thêm chúng vào project.
   b. Chọn kiểu tập tin là *ASM Source Files* và thêm tập tin *vectors_poll.asm* vào project.
   c. Chọn kiểu tập tin là *Linker Command File* và thêm tập tin *c6713dsk.cmd* vào project.
   d. Chọn kiểu tập tin là *Object and Library Files* để thêm các tập tin thư viện vào project. Thêm tập tin *rts6700.lib* (hỗ trợ kiến trúc C67x) nằm ở *C6000\cgtools\lib* vào project. Tương tự, thêm tập tin *dsk6713bsl.lib* (nằm ở *C6000\dsk6713\lib*) và tập tin *csl6713.lib* (nằm ở *C6000\csl\lib*) vào project.
   e. Chọn Project → Scan All File Dependencies để CCS tự thêm vào các tập tin header.

   Sau khi thực hiện các bước trên, cửa sổ Project View sẽ hiển thị các tập tin đã được thêm vào Project.

**Xác định các tùy chọn để biên dịch chương trình**

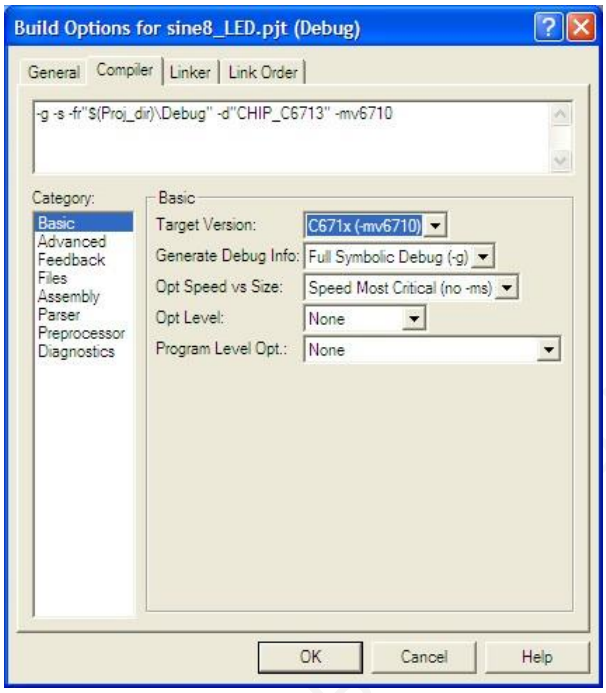
Ở bước này, các tùy chọn để CCS dịch chương trình nguồn thành tập tin thực thi sẽ được thiết lập.

***Các tùy chọn của Compiler***

1. Trong CCS, chọn Project → Build Options. Trong cửa sổ xuất hiện, chọn thẻ Compiler.
2. Chọn mục Basic (trong Category), và đặt các tùy chọn như sau: (xem hình 10)
   a. Target Version: C671x {-mv6710}
   b. Generate Debug Info: Full Symbolic Debug

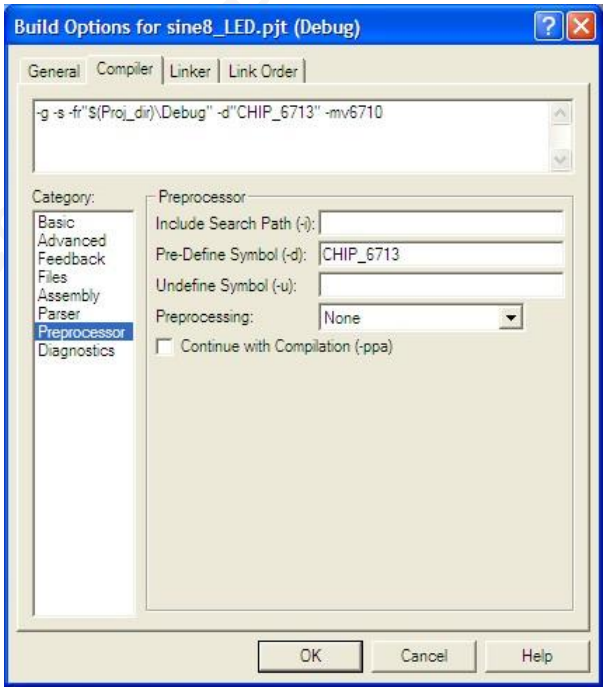
c. Opt Speed vs. Size: Speed most Critical

d. Opt Level and Program Level Opt. : None



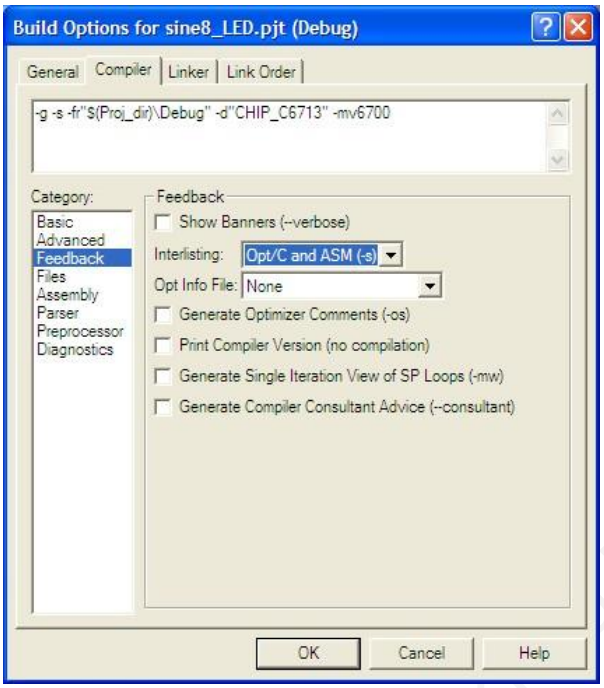

**Hình 3. Tùy chọn Compiler – Mục Basic**

3. Chọn mục Preprocessor và nhập vào Pre-Define Symbol (-d) là CHIP_6713



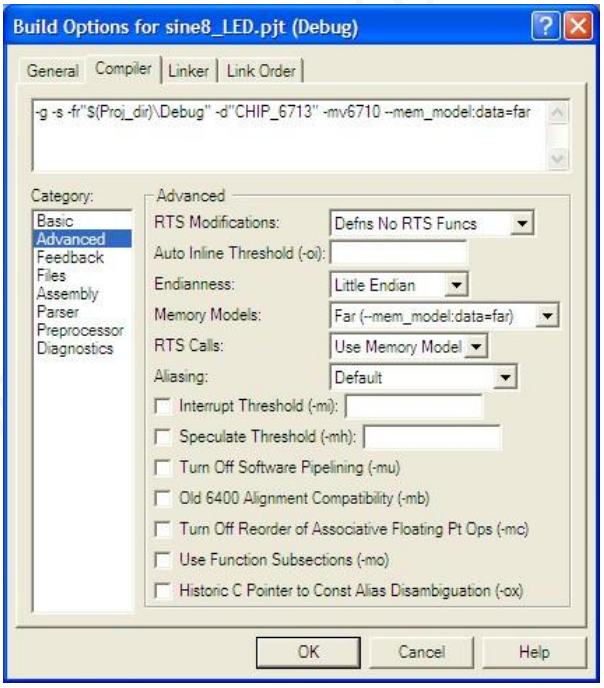

**Hình 4. Tùy chọn Compiler - Mục Preprocessor**

4. Chọn mục Feedback và chọn Interlisting là OPT/C and ASM {-s}




**Hình 5. Tùy chọn Compiler - Mục Feedback**

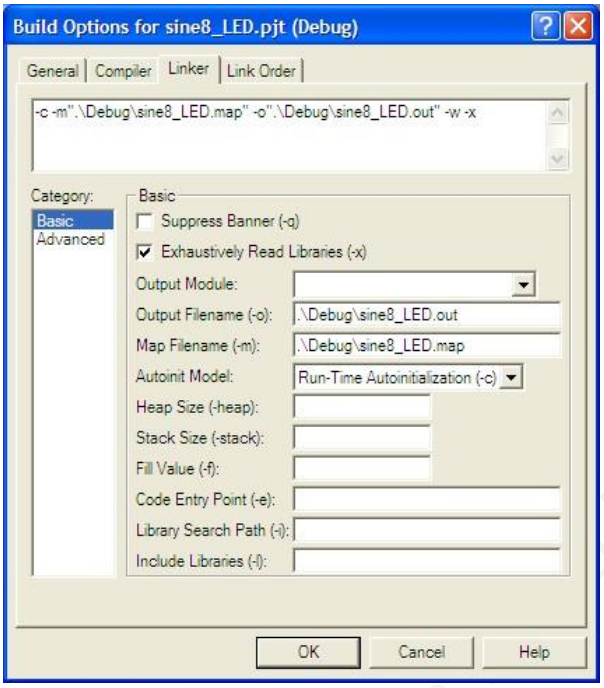
5. Chọn mục Advanced và chọn Memory Models là Far (--mem_model:data=far)




**Hình 6. Tùy chọn Compiler - Mục Advanced**

***Các tùy chọn của Linker***

1. Trong cửa sổ Build Options, chọn thẻ Linker để đặt các tùy chọn của Linker.
2. Mặc định CCS để Output Filename cùng tên với tên project là *sine8_LED.out*. Tạm thời vẫn giữ nguyên như vậy.
3. Chọn Autoinit Model là Run-time Autoinitialization.




**Hình 7. Các tùy chọn của Linker**

**Dịch và chạy chương trình**

Sau khi đã thiết lập các tùy chọn phù hợp cho Compiler và Linker, chúng ta hãy tiến hành biên dịch chương trình và nạp lên trên kit để chạy.

1. Chọn Project → Rebuild All hoặc nhấn nút có hình 3 mũi tên xuống trên toolbar. CCS sẽ dịch tất cả các tập tin C và Assembly. Các tập tin đối tượng tạo ra được liên kết với các tập tin thư viện. Cuối cùng, CCS tạo ra một tập tin thực thi *sine8_LED.out* có thể nạp lên kit để chạy.
2. Chọn File → Load Program, mở thư mục *Debug* trong thư mục *sine8_LED*, chọn tập tin *sine8_LED.out* để nạp nó lên trên kit. Sau đó, chọn Debug → Run để chạy chương trình.
3. Như đã giải thích ở trên, khi DIP Switch 0 ở vị trí ON, chương trình sẽ bật sáng LED 0 và xuất ra tín hiệu sine với tần số 1KHz. Để quan sát kết quả của chương trình, có thể thực hiện theo một trong các cách như sau:
   a. Dùng headphone và cắm vào ngõ ra HEADPHONE trên kit để nghe âm thanh.
   b. Quan sát dạng sóng sine trên máy dao động ký (oscilloscope): kết nối oscilloscope với ngõ ra LINE OUT của kit.
   c. Trong máy tính ở phòng thí nghiệm có sẵn phần mềm mô phỏng Oscilloscope có tên là DSP_Tool. Phần mềm này sẽ đọc dữ liệu từ soundcard của máy tính và hiển thị. Để sử dụng chương trình này, sử dụng cáp Audio (được cung cấp) kết nối ngõ ra LINE OUT trên DSK với ngõ vào LINE IN trên Soundcard của máy tính. Trong DSP_Tool, chọn File → Oscilloscope để hiển thị giao diện Oscilloscope rồi chọn File → Start get real data from soundcard để bắt đầu đọc dữ liệu vào. Ngoài ra, cũng có thể hiển thị phổ

bằng cách chọn File → Spectrum Analyzer. (Chú ý cần cấu hình để soundcard nhận dữ liệu vào từ ngõ Line In)

**Cửa sổ Watch window**

Cửa sổ Watch Window cho phép thay đổi giá trị của một thông số hoặc để theo dõi một biến. Trong khi chương trình đang chạy và DIP Switch 0 đang được nhấn. (Lưu ý dòng chữ DSP RUNNING trên thanh Status của CCS).

1. Chọn View → Quick Watch window. Thường cửa sổ này hiển thị ở phần bên dưới của CCS. Nhập *gain* và nhấn "Add to Watch". Giá trị *gain* bằng 10 (đã được đặt trong chương trình) sẽ xuất hiện trong cửa sổ Watch.
2. Thay đổi *gain* từ 10 thành 30 trong cửa sổ Watch rồi nhấn Enter. Dạng sóng quan sát sẽ thay đổi khi biến *gain* thay đổi giá trị.

**Sử dụng tập tin *gain.gel***

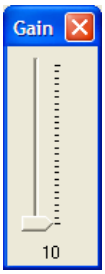
Trong phần trên chúng ta đã đề cập đến tập tin *gain.gel*. Tập tin này tạo một giao diện để cho phép thay đổi biến *gain* một cách tương tác khi chương trình đang chạy.

1. Trước hết, cần phải nạp tập tin này vào bằng cách chọn File → Load GEL và mở tập tin *gain.gel*. Nhấn đúp chuột lên tập tin này trong cửa sổ Project View để xem nội dung của nó.

```
/*gain.gel Create slider and vary amplitude (gain) of sinewave*/
menuitem "Sine Gain"
slider Gain(10,35,5,1,gain_parameter) /*incr by 5,up to 35*/
{
    gain = gain_parameter; /*vary gain of sine*/
}
```

   Nội dung của tập tin *gain.gel* như trên. Trong đó, hàm slider Gain được tạo ra để hiển thị thanh trượt. Thanh trượt này bắt đầu từ giá trị 10 và kết thúc ở giá trị 35 và mỗi mức tăng là 5 đơn vị.

2. Chọn GEL → Sinde Gain → Gain, cửa sổ sau sẽ xuất hiện cho phép thay đổi giá trị của biến *gain*.




**Hình 8. Cửa sổ slider cho phép thay đổi biến *gain***

3. Nhấn nút mũi tên hướng lên để tăng *gain* từ 10 đến 15 và quan sát dạng sóng sine tạo ra để thấy sự thay đổi.

**Thay đổi tần số của tín hiệu sine tạo ra**

Tần số của tín hiệu sine tạo ra có thể thay đổi bằng một trong các cách sau.

1. Thay đổi tần số lấy mẫu. Trong tập tin nguồn ở trên, tần số lấy mẫu fs được gán giá trị là DSK6713_AIC23_FREQ_8KHZ. Giá trị này là 1 hằng số nguyên đã được định nghĩa sẵn. Bộ codec AIC hỗ trợ các tần số lấy mẫu 8, 16, 24, 32, 44.1, 48 và 96kHz. Ví dụ, để có tín hiệu sine ra có tần số là 2kHz, cần tăng tần số lấy mẫu lên 16kHz bằng cách đặt fs=DSK6713_AIC23_FREQ_16KHZ.
2. Thay đổi số điểm ở trong bảng tra, ví dụ còn 4 điểm thay vì 8 điểm – ví dụ, {0, 1000, 0, -1000}. Khi đó cần thay đổi kích thước của mảng sine_table và giá trị biến loop. Hãy chứng minh rằng tần số tạo ra là f = fs/(số điểm).

Hai thanh trượt có thể được sử dụng để vừa thay đổi gain, vừa thay đổi tần số. Các tần số tín hiệu khác nhau có thể tạo ra bằng cách thay đổi biến loop trong chương trình (ví dụ như nhảy cách một điểm lấy một điểm).

Lưu ý rằng với chương trình trên sóng sine chỉ được tạo ra khi DIP Switch 0 được nhấn. Để sử dụng một DIP Switch khác, ví dụ DIP Switch 3, trong chương trình phải sử dụng các hàm *DSK6713_DIP_get(3)*, *DSK6713_LED_on(3)* và *DSK6713_LED_off(3)*.

**Ví dụ 2: Tạo tín hiệu sine và vẽ với CCS**

Ví dụ này cũng tạo ra một tín hiệu sine với 8 điểm như trong ví dụ 1 nhưng nó minh họa khả năng vẽ dạng sóng trong miền thời gian và miền tần số của CCS. Chương trình chính *sine8_buf.c* có nội dung như sau:

```
//sine8_buf Sine generation. Output buffer plotted within CCS
#include "dsk6713_aic23.h" //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
int loop = 0; //table index
short gain = 10; //gain factor
short sine_table[8]={0,707,1000,707,0,-707,-1000,-707};
short out_buffer[256]; //output buffer
const short BUFFERLENGTH = 256; //size of output buffer
int i = 0; //for buffer count
interrupt void c_int11() //interrupt service routine
{
    output_sample(sine_table[loop]*gain); //output sine values
    out_buffer[i] = sine_table[loop]*gain; //output to buffer
    i++; //increment buffer count
    if(i==BUFFERLENGTH) i=0;  //if @ bottom reinit count
    if (++loop > 7) loop = 0; //check for end of table
        return; //return from interrupt
}
void main()
{
    comm_intr(); //init DSK, codec, McBSP
    while(1); //infinite loop
```

```
}
```

Trong chương trình này, một vùng đệm *out_buffer* có kích thước 256 được sử dụng để lưu lại các dữ liệu xuất ra.

Trong hàm *main*, *comm_intr* được gọi. Hàm này nằm trong *c6713dskinit.c* để hỗ trợ chương trình có dùng ngắt. Phát biểu *while(1)* trong hàm main tạo một vòng lặp vô hạn để chờ ngắt xảy ra. Khi có xung lấy mẫu, ngắt 11 xảy ra và trình phục vụ ngắt (ISR – interrupt service routine) *c_int11* được gọi. Địa chỉ của ISR này được xác định trong tập tin *vectors_intr.asm* với một chỉ dẫn rẽ nhánh đến địa chỉ này, sử dụng vector ngắt INT11.

Trong ISR này, hàm *output_sample*, chứa trong tập tin *c6713dskinit.c*, được gọi để xuất ra dữ liệu đầu tiên trong *sine_table*. Chỉ số loop được tăng cho đến hết bảng rồi lặp lại từ 0. Một bộ đệm ra được tạo để giữ 256 (xác định bởi BUFFERLENGTH) mẫu tín hiệu sine xuất ra.

**Xây dựng chương trình**

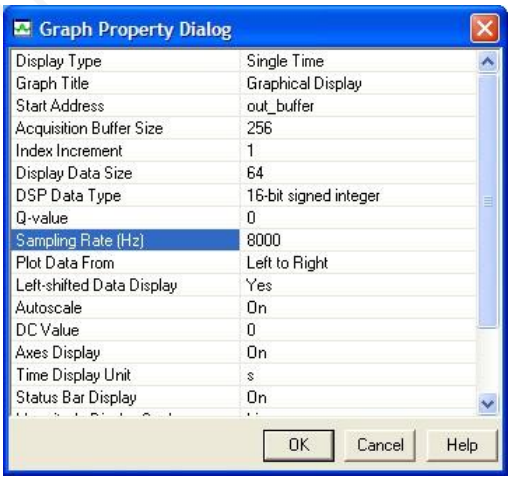
Hãy tạo project sine8_buf.pjt và thêm các tập tin cần thiết như trong ví dụ 1. Lưu ý rằng chương trình này sử dụng ngắt nên tập tin *vectors_intr.asm* được sử dụng thay cho tập tin *vectors_polls.asm*. Tập tin *sine8_buf.c* được chứa trong thư mục *C:\CCStudio_v1.3\myprojects\source\project_2*.

Xác lập các tùy chọn như trong ví dụ 1 và dịch chương trình. Nạp và chạy chương trình trên kit và kiểm tra rằng có một sóng sine 1KHz được tạo ra.

**Vẽ với CCS**

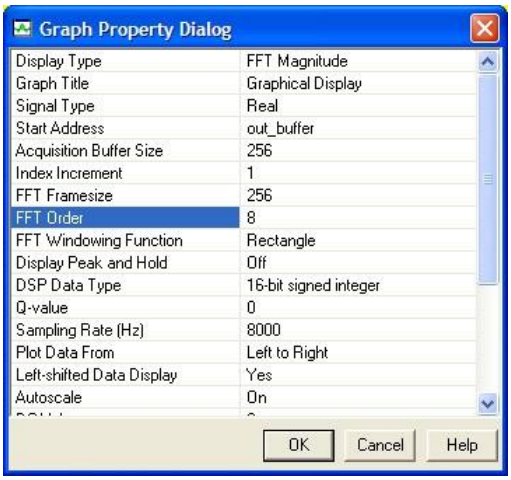
Bộ đệm ra được cập nhật liên tục mỗi 256 điểm. Sau đây CCS sẽ được sử dụng để vẽ dữ liệu ra hiện tại chứa trong bộ đệm *out_buffer*.

1. Chọn View → Graph → Time/Frequency. Thay đổi các tùy chọn trong cửa sổ Graph Property Dialog như sau để vẽ trong miền thời gian. Địa chỉ bắt đầu của bộ đệm chính là tên mảng *out_buffer* được nhập vào Start Address. Các tùy chọn khác có thể để như mặc định.



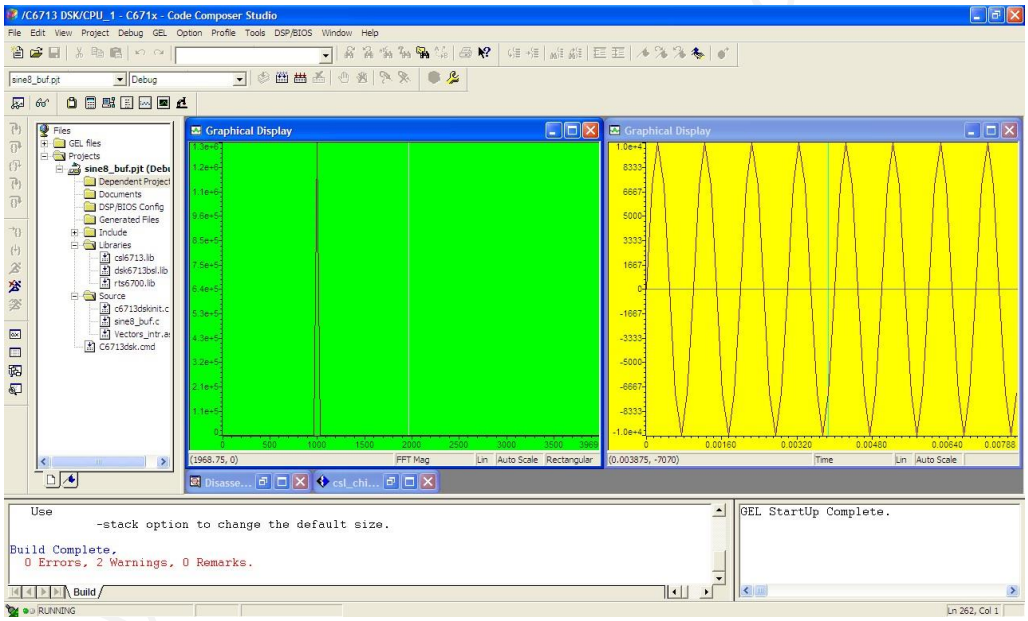

**Hình 9. Các tùy chọn để vẽ trong miền thời gian**

2. Để vẽ trong miền tần số, chọn các tùy chọn như trong hình sau. Chọn bậc của FFT (FFT Order ) sao cho FFT Framesize = $2^{order}$.




**Hình 10. Các tùy chọn để vẽ trong miền tần số**

Kết quả vẽ được thể hiện trên hình sau:




**Hình 11. Kết quả vẽ bằng CCS cả trong miền tần số và trong miền thời gian**

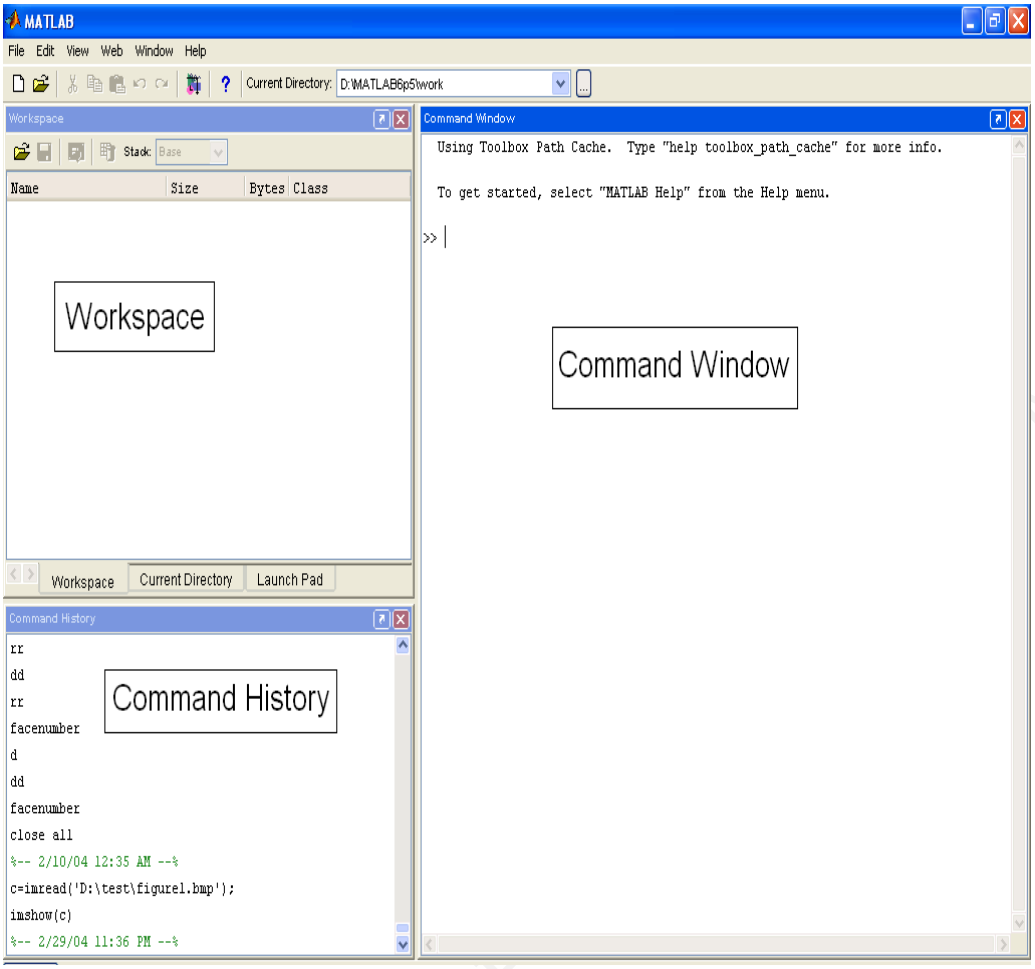
# 4. GIỚI THIỆU MATLAB VÀ CÔNG CỤ SPTOOL

## 4.1. Matlab

Matlab (Matrix Laboratory) là môi trường tính toán đa ứng dụng, được tính toán để thực hiện nhanh các phép toán ma trận. Matlab hỗ trợ nhiều hàm phục vụ cho nhiều lĩnh vực khác nhau. Matlab cho phép thực hiện dễ dàng các tính toán số và đồ họa.

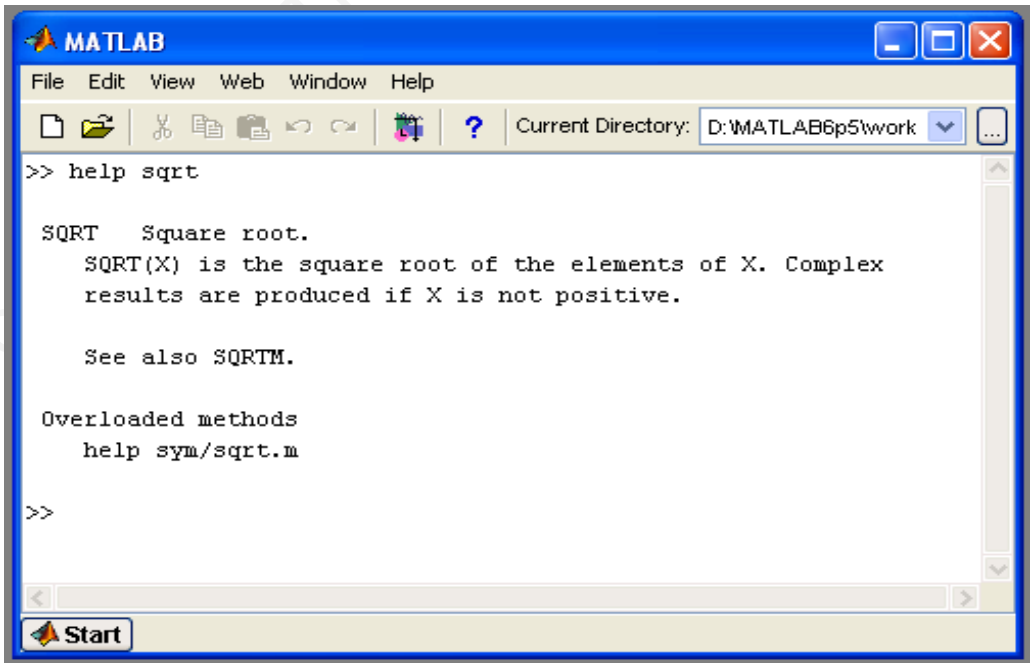
Khi khởi động Matlab, giao diên làm việc (command window) sẽ xuất hiện như hình:,

- **Cửa sổ lệnh** (command window): thực thi các lệnh.
- **Không gian biến** (workspace): hiện thị các biến được định nghĩa.
- **Các lệnh đã thực hiện** (command history): hiện thị các lệnh đã sử dụng.




**Hình 12. Các cửa sổ làm việc của Matlab**

Để hiểu rõ cách sử dụng một hàm, có thể sử dụng lệnh >>help và hàm tương ứng.




**Hình 13. Lệnh giúp đỡ (help)**

Matlab có thể thực thi các lệnh ở cửa sổ lệnh từ dấu nhắc của cửa sổ lệnh.

**Ví dụ: Cách lệnh vẽ đồ thị tín hiệu sin**

```
>> t = 0:0.01:2; % cho t thay giá trị từ 0 đến 2, mỗi giá trị cach nhau 0.01
>> x = sin(2*pi*t); % tính giá trị hàm (2 pi t) cho từng giá trị t
>> plot(t,x,'b'); % Vẽ đồ thị đường màu xanh
>> xlabel('t in sec'); ylabel('x(t)'); % Đặt tên trục x và trục y
>> title('Plot of sin(2\pi t)'); % Tiêu đề đồ thị
```

**Ví dụ: Đọc xuất tín hiệu âm thanh**

```
>> [road,fs]=wavread('road.wav'); % đọc file âm thanh, mãng road chứa dữ liệu âm thanh stereo, fs là tần số lấy mẫu
>> left=road(:,1); % Lấy dữ liệu kênh trái
>> right=road(:,2); % Lấy dữ liệu kênh phải
% Vẽ đồ thị 2000 mẫu tín hiệu kênh trái
>> time=(1/fs)*2000; % Thời gian 2000 mẫu
>> t=linspace(0,time,2000); % Chia thời gian từ 0 đến time có 2000 mẫu
>>plot(t,left(1:2000))   %  Vẽ dạng sóng
>> xlabel('time (sec)'); % Tên trục x
>>ylabel('relative signal strength') % Tên trục y
>>grid on % tạo lưới trên đồ thị
% Tạo hiệu ứng tiếng dội (echo)
>>Lenleft=length(left); % Tính số mẫu của tín hiệu
>> delay=1000;  % Cho số mẫu trễ mong muốn
>>a=0.5;       % Suy hao tín hiệu trễ
>>echo=left+a*[zeros(1, delay) left(1:end-delay)'];
>>soundsc(echo,fs); % Nghe tín hiệu echo
>>wavwrite(echo, fs, 'roadecho.wav'); % Lưu tín hiệu sau xử lý thành file
```

**Ví dụ: Đọc và xuất file ảnh**

```
% Đọc ảnh
>>img = imread('apple.jpg');
>>dim = size(img);
% Hiện thị ảnh
>>figure;
>>imshow(img);         % image(img)
% lưu file ảnh
>>imwrite(img, 'output.bmp', 'bmp');
```

Ngoài ra, các hàm thực hoặc chương trình con có thể thực hiện qua file. m

**Ví dụ: Tạo hàm m** cộng hai tín hiệu sin có tần số và biên độ khác nhau

```
function y=sumsin(A1, f1, A2, f2, Time, fs)
% Cong 2 tin hieu sin co bien do va tan so khac nhau
% Input
% A1, f1: bien do va tan so tin hieu sin 1
% A2, f2: bien do va tan so tin hieu sin 2
% Time (giay) khoang thoi gian can tinh
% fs (Hz): tan so lay mau
% Output: y(t)=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)
% Vi du: y=sumsin(1, 2, 2, 4, 1, 50)
N=round(Time*fs)   % So mau
n=0:N;
x1=A1*sin(2*pi*(f1/fs)*n);
x2=A2*sin(2*pi*(f2/fs)*n);
y=x1+x2;
figure;
subplot(3,1,1);
stem(n/fs, x1);
xlabel('time');
ylabel('x1(t)');
subplot(3,1,2);
stem(n/fs, x2);
xlabel('time');
ylabel('x2(t)');
subplot(3,1, 3);
stem(n/fs, y);
xlabel('time');
ylabel('y(t)=x1(t)+x2(t)');
end
```

lưu chương trình con với tên phải cùng tên với tên hàm, ví dụ trên ta lưu lại sumsin.m. Hàm sumsin() khi đó có thể được gọi từ chương trình khác hoặc có thể thực thi  từ của số lệnh, ví dụ

```
>> y=sumsin(1, 2, 2, 4, 1, 50)
```

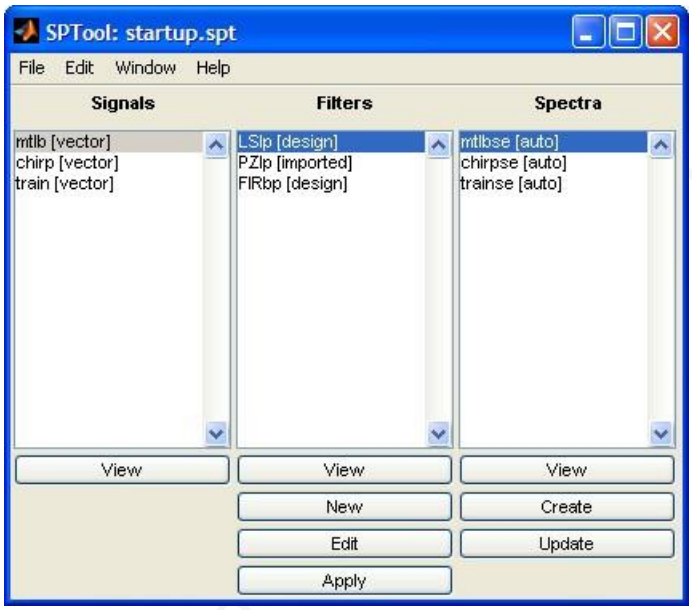
### 4.2. Thiết kế bộ lọc số bằng công cụ SPTool

SPTool là một công cụ có giao diện tương tác dùng cho xử lý số tín hiệu. Công cụ này có thể được sử dụng để phân tích tín hiệu, thiết kế các bộ lọc, phân tích các bộ lọc, lọc tín hiệu và phân tích phổ của tín hiệu.

Để khởi động SPTool, từ dấu nhắc lệnh của MATLAB, nhập lệnh

>> sptool

Khi đó, giao diện của SPTool sẽ xuất hiện như sau:




**Hình 14. Giao diện của SPTool**

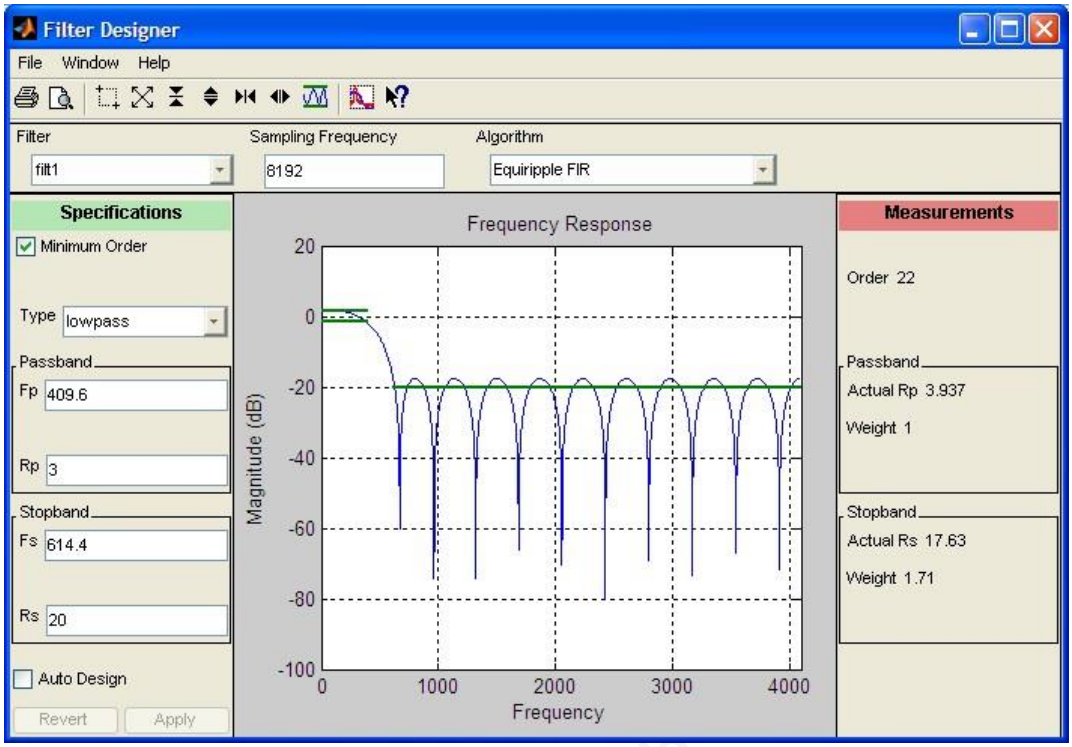
Khi mới mở SPTool, nó chứa một tập hợp các tín hiệu, bộ lọc và phổ mặc định. Trên giao diện của SPTool, có 3 cột: **Signals**, **Filters** và **Spectra**. Dưới mỗi cột có các nút sử dụng cho cột đó. Cột **Signals** hiển thị các tín hiệu, cột **Filters** hiển thị các bộ lọc và cột **Spectra** hiển thị các phổ trong workspace (vùng làm việc) của SPTool.

Các tín hiệu, bộ lọc hoặc phổ trong workspace của MATLAB có thể được đưa vào SPTool bằng lệnh **Import** trong menu **File** của SPTool. Các tín hiệu, bộ lọc hoặc phổ được tạo ra hoặc được import vào SPTool tồn tại dưới dạng các cấu trúc của MATLAB. Để lưu lại các tín hiệu, bộ lọc và phổ đã tạo ra hoặc chỉnh sửa trong SPTool, sử dụng lệnh **Export** trong menu **File**, chúng cũng sẽ được lưu lại dưới dạng các cấu trúc MATLAB.

Để bắt đầu thiết kế một bộ lọc mới, các bạn hãy nhấn vào nút **New** ngay dưới cột **Filter**. Khi đó, giao diện Filter Designer dùng để thiết kế bộ lọc như sau sẽ xuất hiện.

Filter Designer cung cấp một môi trường đồ họa tương tác để thiết kế các bộ lọc số IIR hoặc FIR dựa trên các tiêu chuẩn do người dùng xác định.

- Các loại bộ lọc có thể thiết kế: Thông thấp, thông cao, thông dải, chắn dải.
- Các phương pháp thiết kế bộ lọc FIR: Equiripple, Least squares, Window
- Các phương pháp thiết kế bộ lọc IIR: Butterworth, Chebyshev loại I, Chebyshev loại II, Elliptic.




**Hình 15. Giao diện Filter Designer**

**Ví dụ 1: Thiết kế một bộ lọc FIR chắn dải bằng SPTool**

Bộ lọc, được thiết kế bằng phương pháp cửa sổ Kaiser, với các thông số sau:

*Chiều dài của đáp ứng xung: N = 89 (MATLAB hiển thị bậc bộ lọc bằng 88)*

*Tần số trung tâm: 2700 Hz*

*Tần số cắt: 2500 Hz và 2900 Hz*

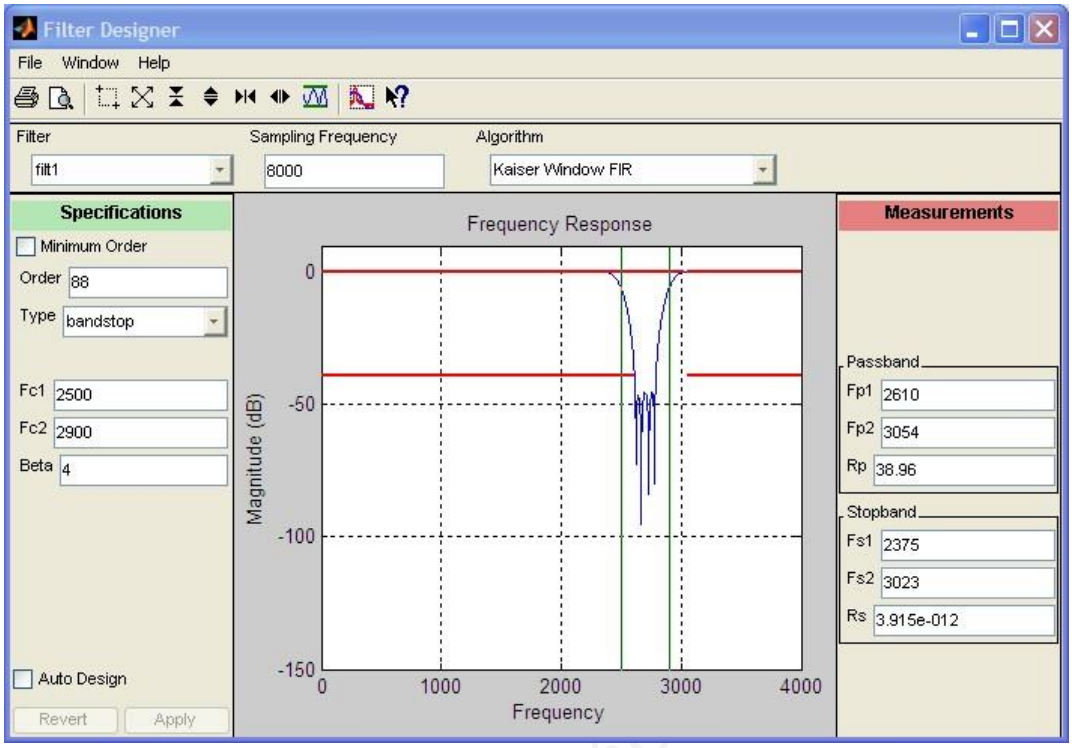
*Giá trị của $\beta = 4$*

*Tần số lấy mẫu 8000 Hz*

Các bước thiết kế như sau:

1. Khởi động SPTool. Dưới cột Filters, nhấn nút New để mở cửa sổ Filter Designer.
2. Trong giao diện của Filter Designer:
   a. Trong text box Filter: Tên bộ lọc được tự đặt (ở đây là **filt1**). Tên này có thể thay đổi sau này.
   b. Nhập các thông số thiết kế vào:
      i. Sampling Frequency = 8000
      ii. Algorithm: Kaiser Window FIR
      iii. Bỏ chọn ở check box Minimum Order. (nếu chọn thì sẽ thiết kế bộ lọc có bậc tối thiểu).
      iv. Filter Order = 88, Type = Bandstop, Fc1 = 2500, Fc2 = 2900, Beta = 4

c. Nhấn Apply. Khi đó đáp ứng tần số của bộ lọc thiết kế sẽ được hiển thị.

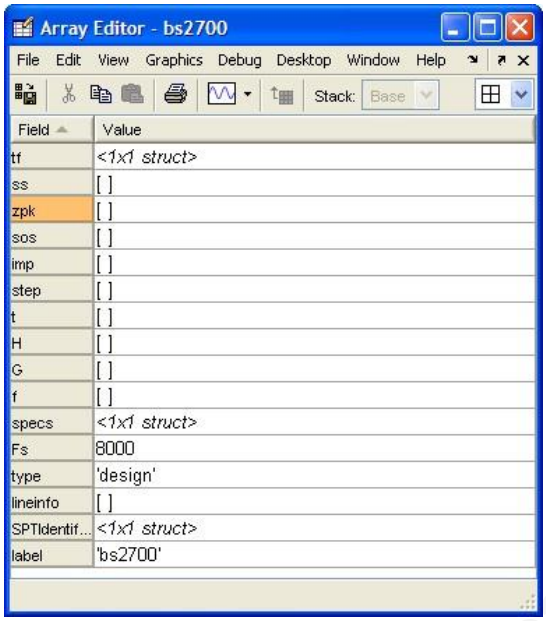



**Hình 16. Đáp ứng tần số của bộ lọc đã thiết kế**

3. Trở về cửa sổ SPTool, trong cột Filters sẽ xuất hiện thêm một dòng **filt1 [design]**. Đây chính là bộ lọc vừa thiết kế. Sau này, nếu muốn sửa đổi thiết kế, chọn lại tên bộ lọc và nhấn nút Edit ở phía dưới. Để dễ nhớ, ta sẽ thay đổi tên bộ lọc trên thành **bs2700** bằng cách chọn Edit → Name…→filt1 [design]. Trong cửa sổ mới xuất hiện, nhập tên mới.

Khi thiết kế một bộ lọc FIR như trên, kết quả mà ta cần nhận được sau khi thiết kế là các giá trị của vector đáp ứng xung **h** của bộ lọc thiết kế. Để lấy các giá trị của vector đáp ứng xung, ta thực hiện như sau:

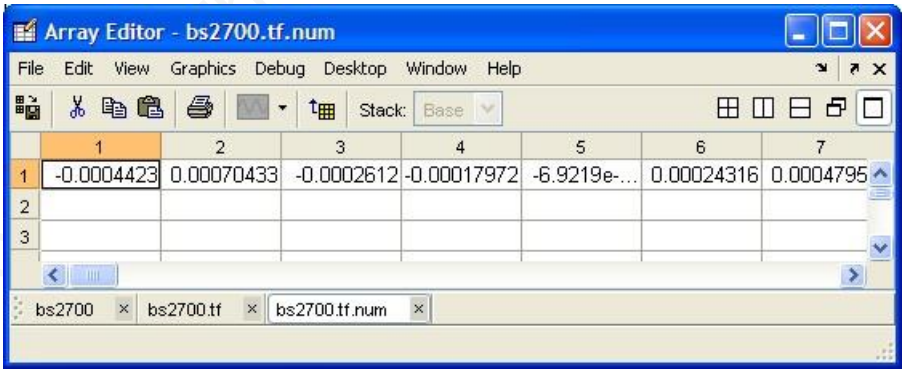
1. Từ cửa sổ SPTool, chọn File → Export… Trong Export list xuất hiện, chọn *Filter: bs2700 [design]* rồi nhấn nút **Export to workspace**
2. Đóng cửa sổ SPTool lại. Một thông báo xuất hiện hỏi có muốn lưu lại phiên làm việc hiện tại hay không. Nếu muốn lưu lại, chọn Save.
3. Mở cửa sổ Workspace của MATLAB, ta sẽ thấy trong workspace sẽ xuất hiện biến mới là **bs2700**. Đây chính là bộ lọc mà ta đã thiết kế trong SPTool và xuất ra workspace của MATLAB. Biến này được lưu dưới dạng một cấu trúc mô tả bộ lọc đã thiết kế. Nhấn đúp chuột vào tên biến bs2700 trong workspace, ta sẽ thấy được các field của cấu trúc này như sau:




**Hình 17. Các field của bs2700**

4. Trong các field này, field **tf** thể hiện hàm truyền của bộ lọc. Field này cũng là một cấu trúc gồm 2 field: **tf.num** và **tf.den** thể hiện tương ứng các hệ số của đa thức tử số và đa thức mẫu số. Đối với bộ lọc FIR, hàm truyền chỉ có tử số và các hệ số của tử số chính là đáp ứng xung của bộ lọc. Do đó, với bộ lọc trên, các giá trị của vector đáp ứng xung được lưu trong **bs2700.tf.num**. Trong cửa sổ Array Editor trên, lần lượt nhấn đúp vào field **tf** rồi nhấn đúp vào **num**, ta sẽ thấy các hệ số đáp ứng xung của bộ lọc. Để gán các hệ số này vào một vector h, trong MATLAB có thể dùng lệnh sau:

   >> h = bs2700.tf.num




**Hình 18. Vector đáp ứng xung của bộ lọc đã thiết kế**

Các giá trị thu được của vector đáp ứng xung sẽ được sử dụng để thực hiện bộ lọc số lên trên kit DSP.

**Ví dụ 2: Thiết kế bộ lọc IIR chắn dải bằng SPTool**

Sử dụng phương pháp Elliptic để thiết kế một bộ lọc IIR chắn dải bậc 10, tần số trung tâm 1750Hz. Chú ý rằng MATLAB hiển thị bậc bộ lọc là 5, biểu diễn số phần bậc 2 của bộ lọc. **(Điều này đúng với các bộ lọc IIR thông dải và chắn dải)**

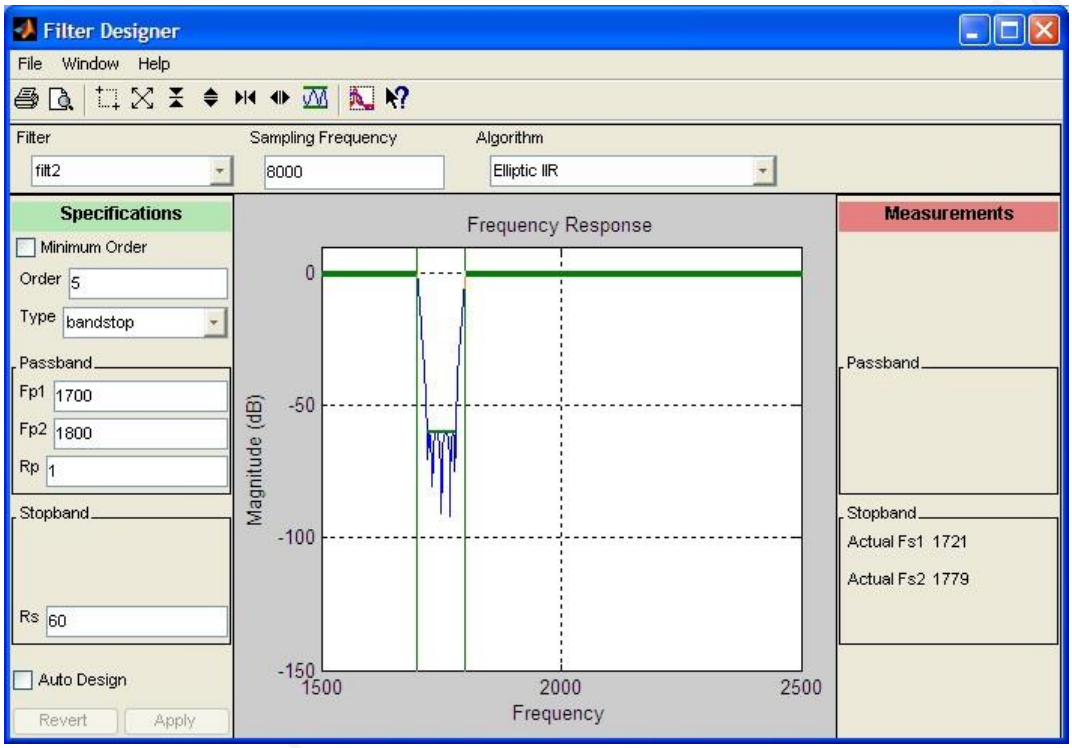
Các thông số của bộ lọc này như sau.

*Tần số cắt: 1700 Hz và 1800 Hz*

*Độ gợn dải thông và dải chắn tương ứng là 1 dB và 60 dB*

*Tần số lấy mẫu: 8000 Hz*

Thực hiện tương tự như ví dụ trên, lưu bộ lọc thiết kế với tên **bs1750** và xuất ra workspace. Trong workspace sẽ có một cấu trúc tên là bs1750. Các hệ số tử số và mẫu số của hàm truyền được lưu tương ứng trong các biến **bs1750.tf.num** và **bs1750.tf.den**.




**Hình 19. Đáp ứng tần số của bộ lọc IIR đã thiết kế**

Dạng cực – zero của một hàm truyền H(z) như sau:

$$H(z) = k\frac{(z-z_1)(z-z_2)...(z-z_n)}{(z-p_1)(z-p_2)...(z-p_m)}$$

Hàm truyền trên có thể được viết lại như sau:

$$H(z) = g\prod_{k=1}^{L} H_k(z) = g\prod_{k=1}^{L}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

Với L là số nguyên gần nhất lớn hơn cực đại của n/2 và m/2.

Trong MATLAB, các phần bậc 2 của H(z) được lưu trong 1 ma trận như sau:

$$sos = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

Từ các hệ số tử và mẫu ở trên, ta sẽ chuyển thành dạng các phần bậc hai bằng các lệnh sau:

```
>> [z,p,k] = tf2zp(bs1750.tf.num, bs1750.tf.den)
>> sos = zp2sos(z,p,k)
```

Các phần tử của ma trận sos này sẽ được sử dụng để thực hiện bộ lọc IIR này lên kit DSP.

2

# LẤY MẪU VÀ LƯỢNG TỬ HÓA TRÊN KIT C6713 DSK

**Họ và tên SV báo cáo 1: ……. MSSV: …….....**

**Họ và tên SV báo cáo 2: ……. MSSV: …….....**

**Họ và tên SV báo cáo 3: ……. MSSV: …….....**

**Họ và tên SV báo cáo 4: ……. MSSV: …….....**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: …….……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |
| | | | | |



## 1. MỤC ĐÍCH THÍ NGHIỆM

- Hiểu rõ quá trình lấy mẫu và lượng tử hóa tín hiệu trong bộ ADC.
- Hệ thống lại các lý thuyết đã học.
- Giúp sinh viên có cái nhìn trực quan về hiện tượng alising khi điều kiện lấy mẫu không thỏa.
- Giúp sinh viên hiểu được ảnh hưởng của việc tăng/giảm số bit để mã hóa một mẫu tín hiệu.

# 2. THIẾT BỊ THÍ NGHIỆM

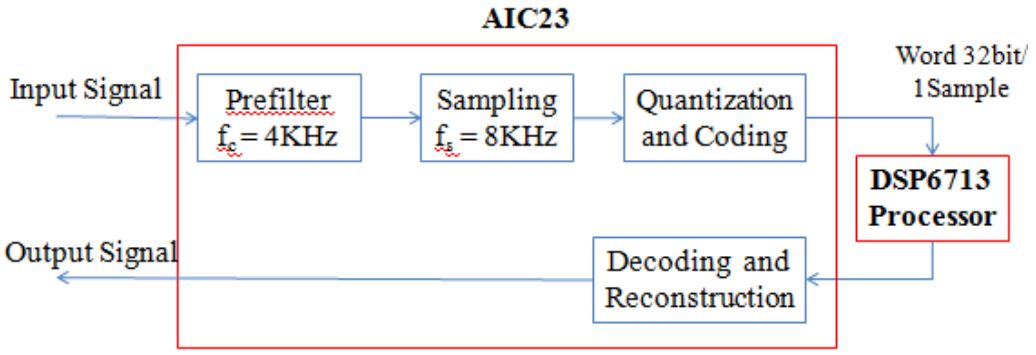
| STT | Tên thiết bị | Số lượng |
|---|---|---|
| 01 | Máy vi tính | 01 |
| 02 | Kit C6713 DSK | 01 |
| 03 | Máy phát sóng | 01 |
| 04 | Bộ dây nối tín hiệu | 01 |

# 3. CƠ SỞ LÝ THUYẾT

## 3.1. Giới thiệu

Xử lý tín hiệu số có rất nhiều ưu điểm so với xử lý các tín hiệu tương tự. Một hệ thống xử lý số tín hiệu bao gồm bộ ADC, bộ xử lý trung tâm và bộ DAC. Tín hiệu tương tự khi đi vào hệ thống, đi qua bộ ADC sẽ được lấy mẫu và mã hóa thành chuỗi bit nhị phân. Chuỗi bit này đi qua bộ xử lý trung tâm và sau đó đi qua bộ DAC. Bộ DAC sẽ biến chuỗi bit trở thành lại các tín hiệu tương tự đi ra ngoài.

KIT C6713 DSK sử dụng chip AIC23 để lấy mẫu và mã hóa tín hiệu. Tín hiệu tương tự được đưa vào AIC23 và lấy mẫu với tần số 8KHz. Để tránh hiện tượng aliasing, chip AIC23 có thêm một bộ tiền lọc với tần số 4KHz để đảm bảo tín hiệu vào có tần số lớn nhất là 4KHz. Các mẫu sau đó được lượng tử hóa và mã hóa thành chuỗi 8bit. Do bộ xử lý trung tâm 6713DSP hoạt động với các word 32 bit nên chuỗi bit mã hóa của các mẫu được thêm 24 số 0 vào sau cùng để trở thành các word 32 bit.



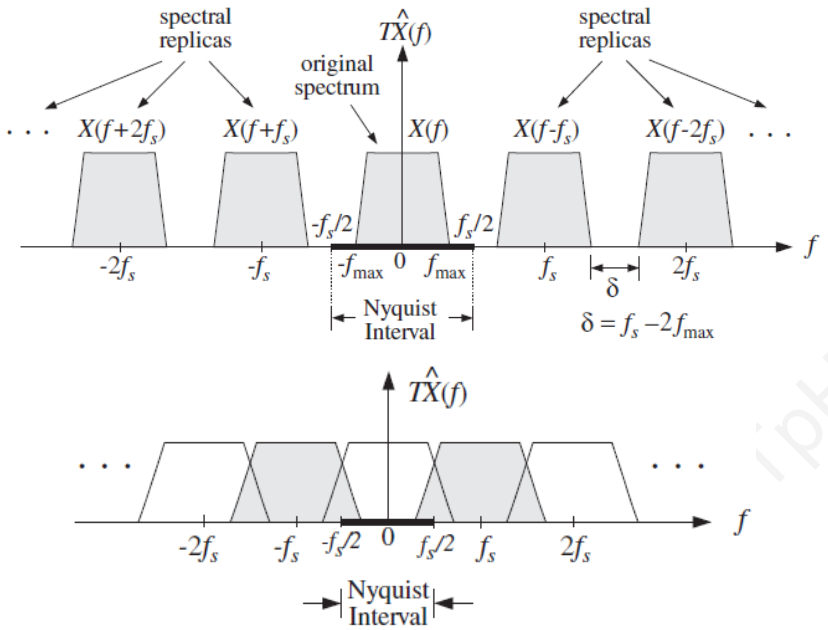

**Hình 20. Chip AIC23**

## 3.2. Lấy mẫu tín hiệu

Lấy mẫu là quá trình biến đổi tín hiệu liên tục về mặt thời gian thành một tín hiệu rời rạc về mặt thời gian. Tín hiệu x(t) sau khi lấy mẫu sẽ trở thành tín hiệu rời rạc về mặt thời gian x(n) = x(nT), với T là chu kỳ lấy mẫu. 02 tín hiệu liên tục khác nhau, nhưng khi được lấy mẫu với tần số fs, ta được x1(n) = x2(n), hiện tượng này được gọi là aliasing, và 2 tín hiệu đó được gọi là alias với nhau. Để tránh hiện tượng này, tín hiệu phải thõa điều kiện lấy mẫu.

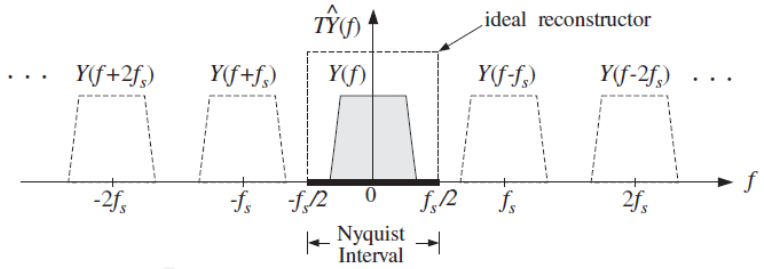
***Định lý lấy mẫu***: Tín hiệu được lấy mẫu phải có băng thông giới hạn, tần số lấy mẫu thấp nhất lớn hơn 2 lần tần số lớn nhất của tín hiệu fs >= 2fmax.

Tín hiệu sau khi lấy mẫu có phổ là phổ của tín hiệu ban đầu được lặp lại sau những khoảng fs. Hiện tượng aliasing xảy ra do sự chồng lấp của phổ tín hiệu sau những khoảng fs.




**Hình 21. Ảnh hưởng của tần số lấy mẫu**

Tín hiệu sau khi lấy mẫu được phục hồi lý tưởng. Bộ phục hồi lý tưởng là một bộ lọc thông thấp với tần số cắt bằng tới tần số Nyquist fs/2.

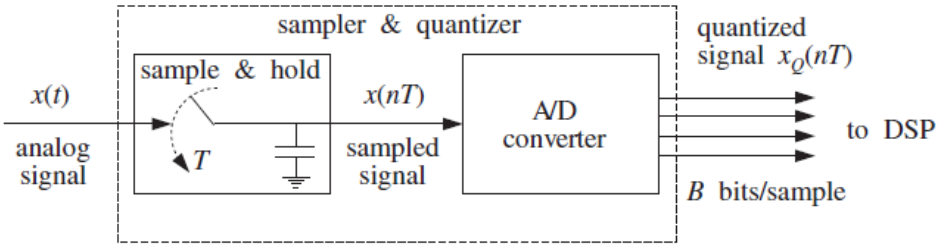



**Hình 22. Khôi phục tín hiệu tương tự bằng bộ lọc**

Nếu tín hiệu ban đầu có tần số f nằm trong khoảng Nyquist, tín hiệu sau khi lấy mẫu với tần số fs và phục hồi lý tưởng sẽ có tần số bằng với tần số tín hiệu ban đầu. Ngược lại, tín hiệu có tần số nằm ngoài khoảng Nyquist sau khi lấy mẫu và phục hồi lý tưởng sẽ có tần số fa = f mod(fs). Tín hiệu có tần số fa như thế được gọi là tín hiệu bị alias với tín hiệu có tần số f khi ta lấy mẫu với tần số fs.

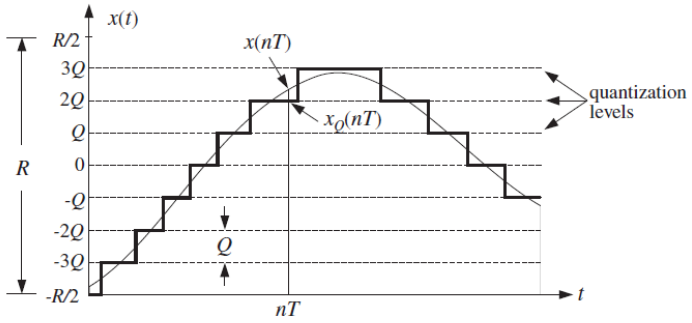
### 3.3. Lượng tử và mã hóa

Các tín hiệu sau khi lấy mẫu sẽ được lượng tử và mã hóa thành cách chuỗi B bit.




**Hình 23. Lượng tử và mã hóa**

Nếu tín hiệu có tầm toàn thang là R, độ phân giản lượng tử $Q = R/2^B$.




**Hình 24. Lượng tử tín hiệu**

Nếu coi quá trình lượng tử hóa giống như việc cộng thêm nhiễu lượng tử và tín hiệu, ta có tỉ số tín hiệu trên nhiễu sẽ là $SNR_{dB} = 6B$. Nếu số bit B mã hóa các mẫu tăng lên thì tỉ số tín hiệu trên nhiễu tăng lên, chất lượng tín hiệu sau khi phục hồi sẽ tăng lên, quá trình lượng tử hóa tín hiệu ít ảnh hưởng đến tín hiệu. Ngược lại, nếu số bit mã hóa tín hiệu giảm đi, chất lượng tín hiệu sau khi phục hồi sẽ giảm xuống do nhiễu lượng tử tăng lên, tín hiệu sau khi phục hồi sẽ bị sai dạng so với tín hiệu ban đầu.

## 4. CHUẨN BỊ THÍ NGHIỆM

1. Một hệ thống có tần số lấy mẫu fs = 8KHz. Xác định tần số cắt của bộ tiền lọc lý tưởng để không xảy ra hiện tượng aliasing. Giải thích.

2. Một tín hiệu $x(t) = 5\sin(6\pi t)$ (t: ms). Xác định tần số lấy mẫu thấp nhất để có thể phục hồi lại tín hiệu. Tín hiệu được lấy mẫu với tần số 4KHz. Sau đó tín hiệu được phục hồi lý tưởng. Xác định tín hiệu sau khi được phục hồi lý tưởng. Giải thích ngắn gọn.

3. Cho một tín hiệu có tầm toàn thang R = 10V. Xác định số bit B để mã hóa tín hiệu được sai số lượng tử hiệu dụng (rms) không quá 50microV.

| |
|---|
| |

4. Cho một tín hiệu lưỡng cực có tầm toàn thang là 16V, được mã hóa thành 4bit bằng phương pháp rounding. Các mẫu tín hiệu có giá trị: -7.9, -7.1,- 6.8,- 5.5,-3.1, 0, 1.3, 2.6, 5.8, 6.9.

   a. Xác định chuỗi bit cho các mẫu trên nếu mã hóa bằng bộ mã offset binary.

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

   b. Lặp lại câu trên với bộ mã bù bậc 2.

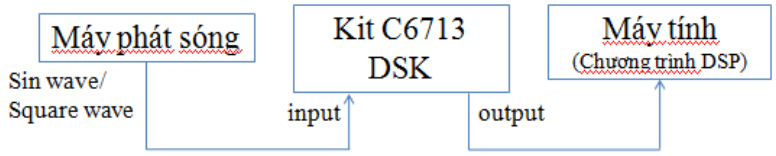
| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## 5. TIẾN HÀNH THÍ NGHIỆM

### Lấy mẫu tín hiệu

Trước hết, hãy thực hiện theo từng bước ví dụ đơn giản sau.

***Ví dụ :*** *Cho tín hiệu hình sin có tần số 3KHz đi qua bộADC của AIC32. Tín hiệu sau đó được cho đi ngược lại bộ DAC của AIC32. Quan sát tín hiệu được hiển thị trên máy tính.*



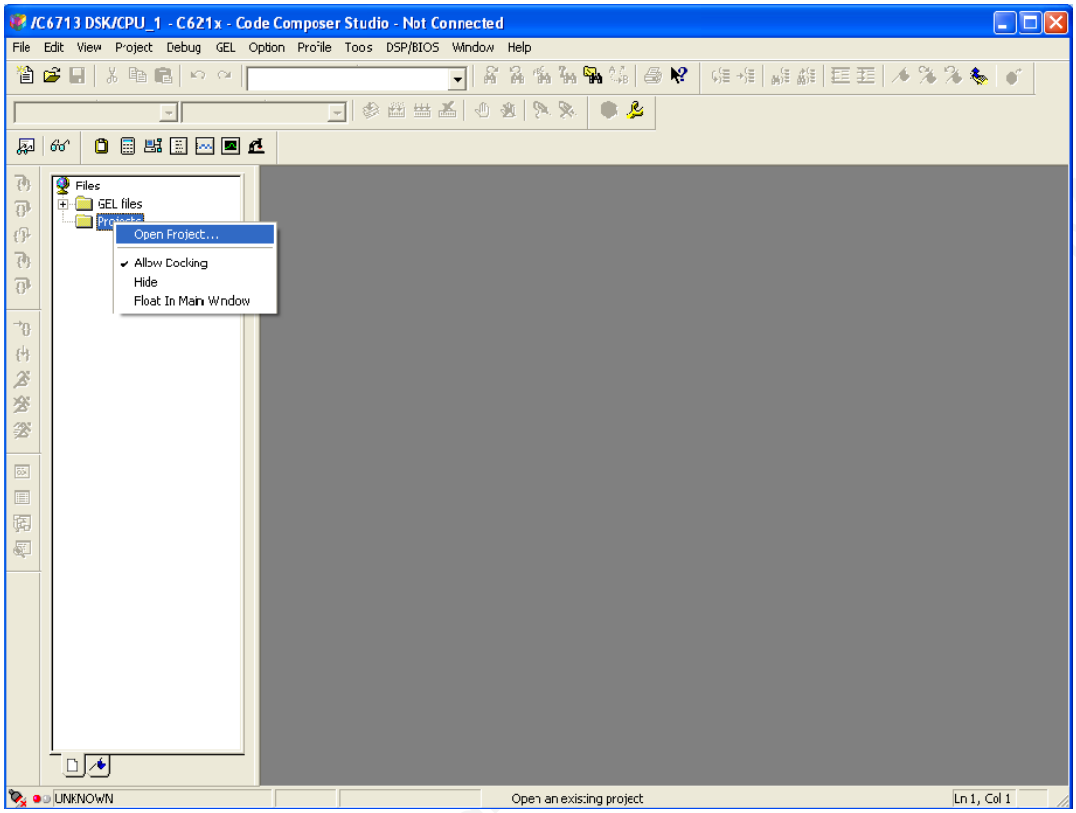

Trong ví dụ này tín hiệu từ máy phát sóng x(t) sau khi đi qua bộ AIC32 trong Kit C6713DSK sẽ là tín hiệu được lấy mẫu x(nT). x(nT) sẽ đi qua hệ thống là vi xử lí DSP6713. Trong ví dụ này, tín hiệu ngõ ra của hệ thống sẽ giống tín hiệu ngõ vào y(n) = x(nT). Sau đó y(n) sẽ được đưa ngược lại bộ DAC của AIC32 và khôi phục lại thành tín hiệu y(t). Tín hiệu y(t) sẽ được đưa vào và hiển thị trên máy tính.

***Chương trình hệ thống cho DSP6713 để lấy tín hiệu ngõ ra là tín hiệu ngõ vào***

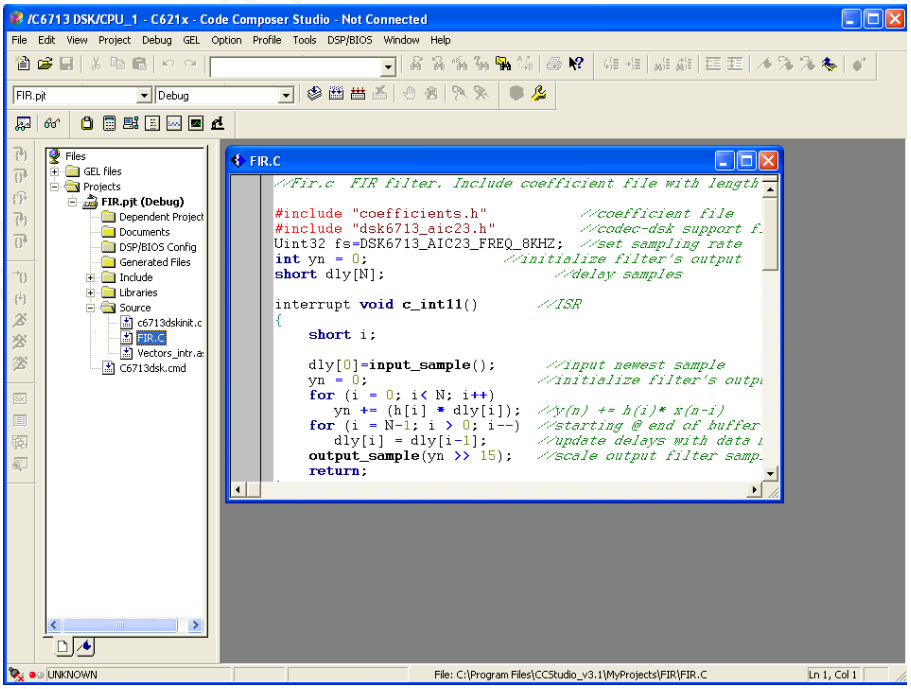
A. Hệ thống này được thực hiện trên kit bằng chương trình sau (viết bằng ngôn ngữ C)

B. Mở project bộ lọc FIR:

1. Trong chương trình




2. Chọn project FIR theo đường dẫn: C:\Program Files\CCStudio_v3.1\myprojects\FIR\FIR.pjt
3. Mở chương trình chính FIR.c



**Viết lại chương trình chính của bộ lọc FIR theo chương trình sau:**

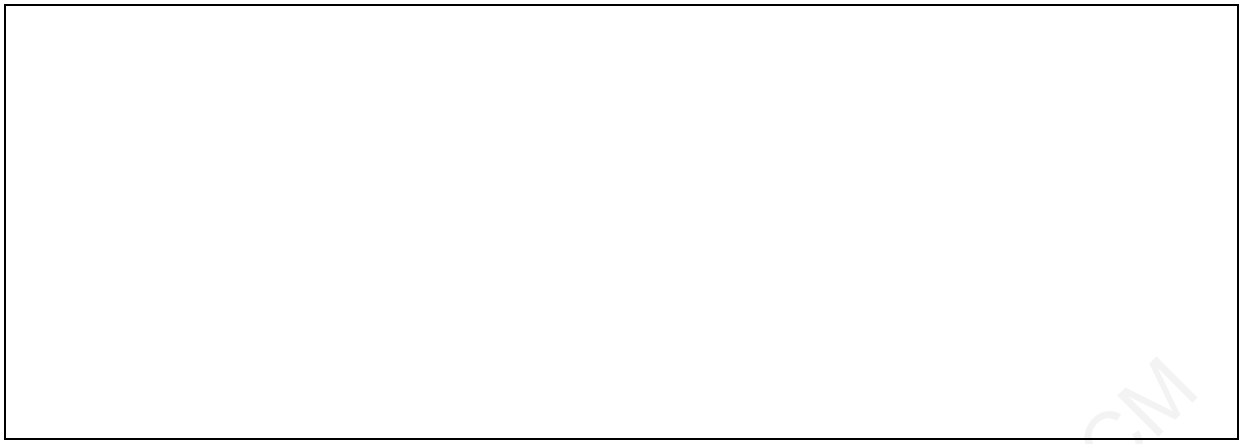
```
//Fir.c  FIR filter. Include coefficient file with length N
#include "coefficients.h"                    //coefficient file
#include "dsk6713_aic23.h"                          //codec-dsk support file

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;        //set sampling rate
int yn = 0;                                          //initialize filter's output
int pulse;
interrupt void c_int11()       //ISR
{
    yn = input_sample();
 output_sample(yn >> 15);   //scale output filter sample
 return;
}

void main()
{
 comm_intr();             //init DSK, codec, McBSP
 while(1);               //infinite loop
}
```

C. Biên dịch và chạy chương trình:

1. Chọn Project → Rebuild All hoặc nhấn nút có hình 3 mũi tên xuống trên toolbar. CCS sẽ dịch tất cả các tập tin C và Assembly. Các tập tin đối tượng tạo ra được liên kết với các tập tin thư viện. Cuối cùng, CCS tạo ra một tập tin thực thi *FIR.out* có thể nạp lên kit để chạy. (Nếu chương trình biên dịch bị lỗi thì kiểm tra lại và sửa lỗi, sau đó biên dịch lại).
2. Chọn Debug → Connect hoặc bấm tổ hợp phím Alt+C để kết nối với kit.
3. Chọn File → Load Program, mở thư mục *Debug* trong thư mục *FIR*, chọn tập tin FIR.*out* để nạp nó lên trên kit. Sau đó, chọn Debug → Run để chạy chương trình trên kit.
4. Khi cần thay đổi hay chỉnh sửa chương trình cho một ví dụ khác, ta chọn Debug → Halt để ngắt kết nối với kit, rồi thực hiện lại các bước như ban đầu.

***Đánh giá kết quả thực hiện***

*Mở nguồn của máy phát sóng. Tạo một tín hiệu hình sine 3KHz từ máy phát sóng và quan sát dạng sóng ngõ ra. Vẽ lại dạng sóng và phổ của tín hiệu ngõ ra. Nhận xét ngắn gọn.*



Sau khi đã thực hiện ví dụ trên, sinh viên tiếp tục thực hiện việc thay đổi tần số lấy mẫu để thấy rõ hiện tượng aliasing khi không thỏa mãn điều kiện lấy mẫu tín hiệu.

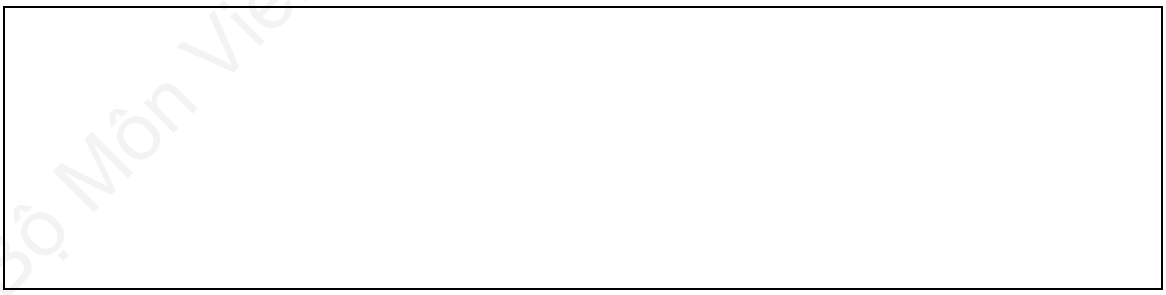
AIC23 được cố định tần số lấy mẫu và ta chỉ có thể thay đổi chương trình trong vi xử lý DSP6713. AIC có tần số lấy mẫu là 8KHz, vậy nên trong 1 giây sẽ có 8000 mẫu được đưa tới vi xử lý DSP6713. Chúng ta có thể giảm tốc độ lấy mẫu xuống còn 4KHz bằng cách thay vì vi xử lý lấy toàn bộ mẫu, ta sẽ lấy một mẫu và bỏ 1 mẫu. Như vậy trong 1 giây, chúng ta chỉ nhận 4000 mẫu, hay nói cách khác, tốc độ lấy mẫu được thay đổi xuống còn 4KHz.

Để thực hiện điều này, ta nhân tín hiệu ngõ vào với một chuỗi tuần hoàn [1, 0, 1, 0, 1, 0,…]. Việc này có thể thực hiện trên chương trình bằng dòng lênh sau:

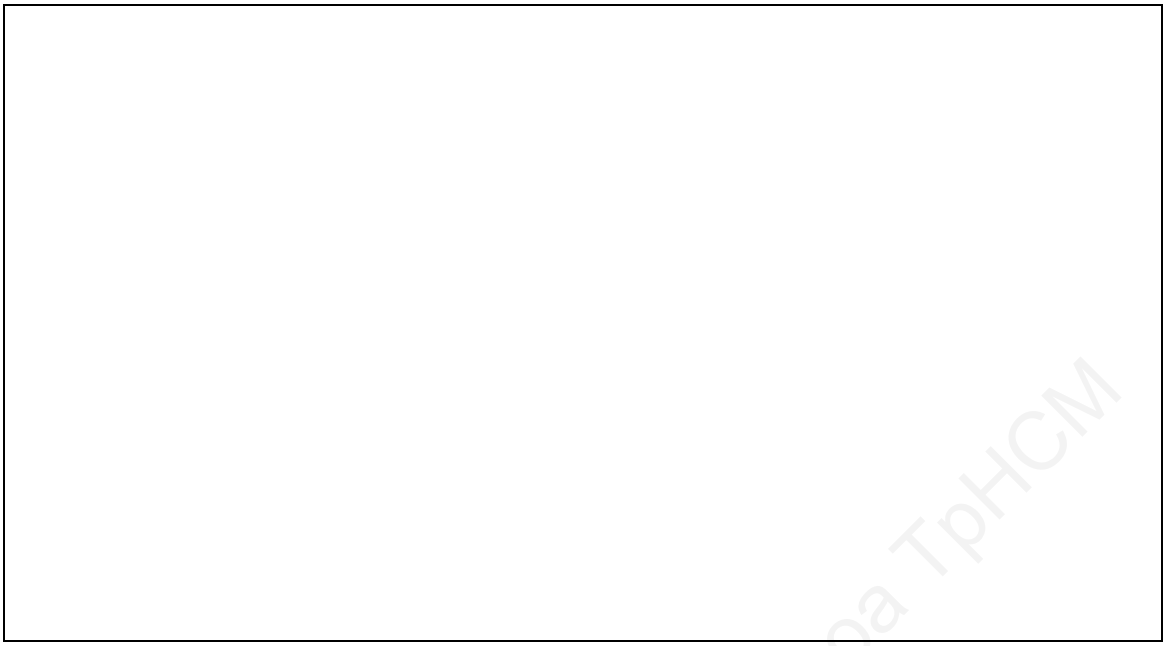
```
yn = pulse * input_sample();
pulse = (pulse==0);
```

**Sinh viên thực hiện**

*Cho tín hiệu hình sin có tần số 3KHz đi qua hệ thống. Tín hiệu được lấy mẫu với tần số 4KHz. Tín hiệu sau đó được đi qua bộ lọc thông thấp tần số 4KHz. Tín hiệu ngõ ra có tần số bao nhiêu?*



*Quan sát và vẽ lại dạng dóng và phổ của tín hiệu ngõ ra. So sánh với trường hợp ví dụ khi ta lấy mẫu với tần số 8KHz. Nhận xét.*



*Cho tín hiệu xung vuông có tần số 0.5KHz đi qua hệ thống. Tín hiệu được lấy mẫu với tần số 8KHz. Tín hiệu sau đó được phục hồi lý tưởng. Quan sát và vẽ lại dạng dóng và phổ của tín hiệu ngõ ra. Nhận xét và giải thích ngắn gọn.*

*Thay đổi tần số lấy mẫu còn 4KHz. Quan sát và vẽ dạng sóng và phổ tín hiệu ngõ ra. So sánh 02 trường hợp khi lấy mẫu với tần số 4KHz và 8KHz của xung vuông tần số 0.5KHz. Nhận xét và giải thích ngắn gọn.*



**Lượng tử hóa tín hiệu**

Mỗi mẫu tín hiệu được AIC32 mã hóa và đưa tới vi xử lý sẽ có dạng chuỗi bit:

$$[b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, 0, \ldots 0] \quad (24 \text{ số không})$$

Trong đó bit $b_1$ là MSB và $b_8$ là LSB

Chúng ta có thể thay đổi số mức lượng tử xuống thấp hơn bằng cách dịch phải rồi dịch trái chuỗi bit. Ví dụ ta có thể thay đổi từ 256 mức lượng tử (tương ứng 8bit) xuống thành 128 mức (tương ứng 7bit) bằng cách bỏ đi bit cuối cùng $b_8$. Điều này được thực hiện bằng cách dịch phải rồi dịch trái 25bit như sau:

**$[b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, 0, \ldots 0] \Rightarrow [0, 0, \ldots 0, b_1, b_2, b_3, b_4, b_5, b_6, b_7,] \Rightarrow [b_1, b_2, b_3, b_4, b_5, b_6, b_7, 0, 0, \ldots 0]$**

Việc này có thể thực hiện trên chương trình bằng dòng lệnh sau:

```
yn = (input_sample() >> 25) << 25;
```

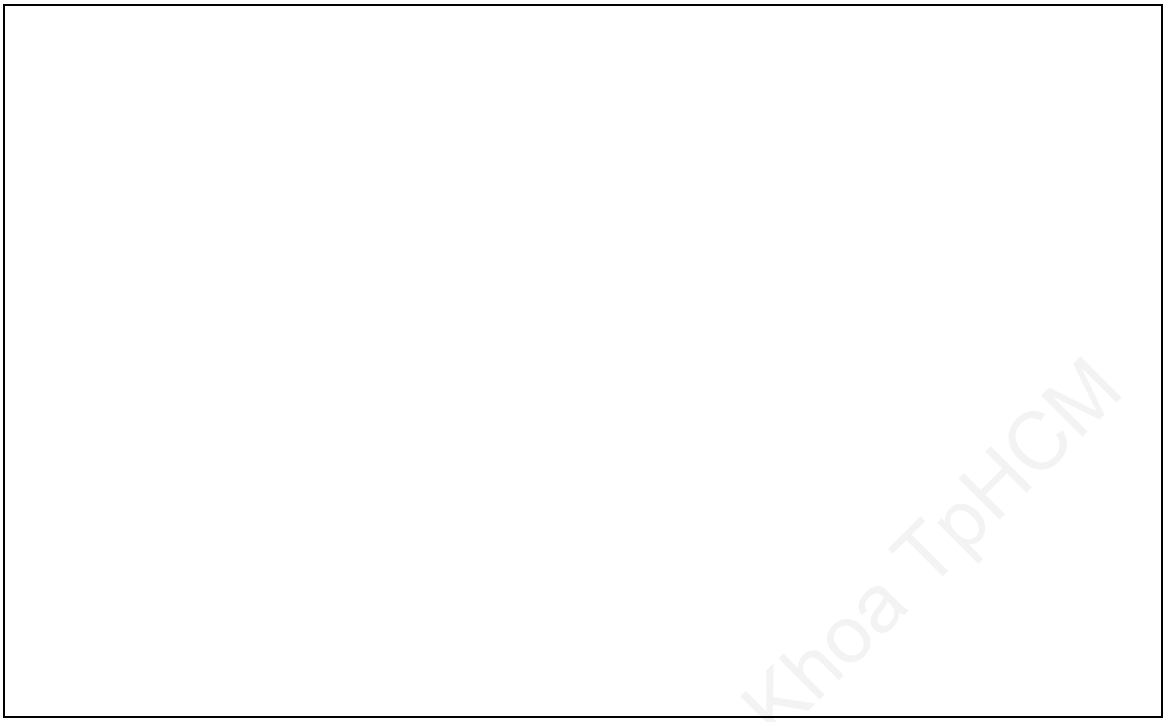
**Sinh viên thực hiện**

*1. Cho tín hiệu hình sin có tần số 3KHz đi qua hệ thống. Tín hiệu được lấy mẫu với tần số 8KHz. Mỗi mẫu tín hiệu được mã hóa thành chuỗi 8bit. Tín hiệu sau đó được phục hồi lý tưởng. Quan sát và vẽ dạng sóng và phổ tín hiệu ngõ ra.*

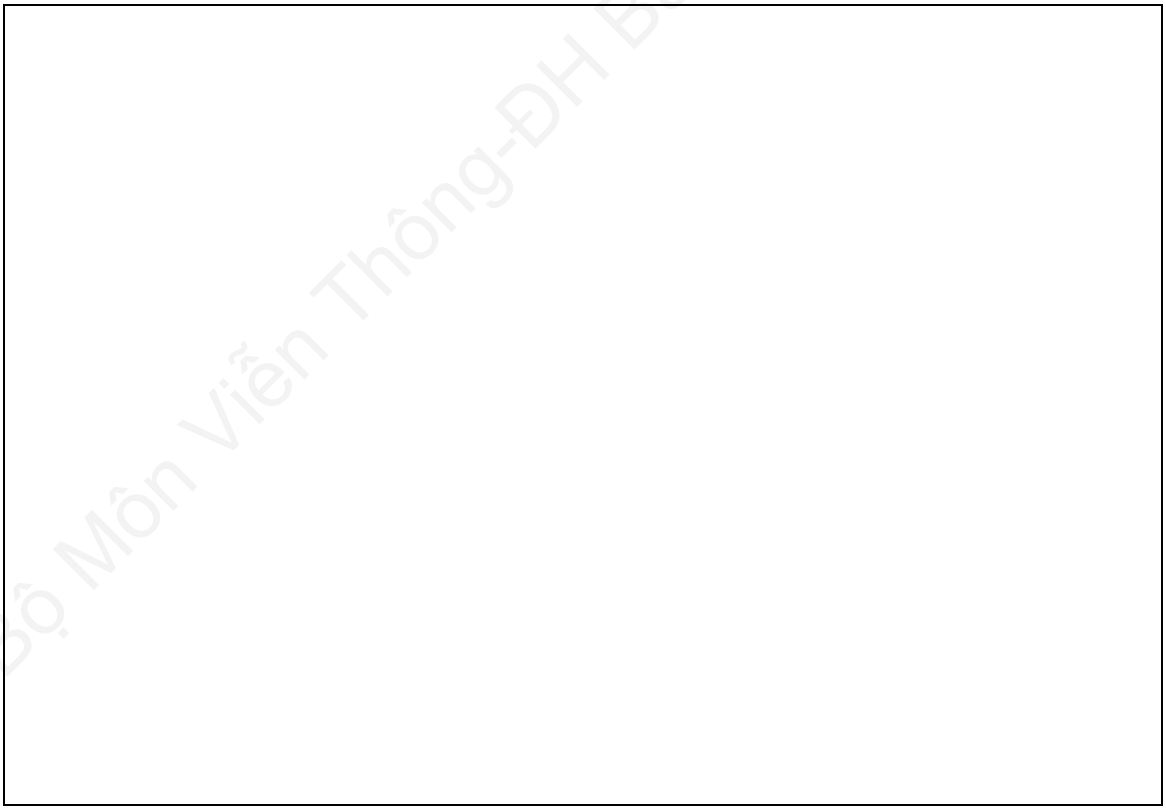
*Thực hiện việc giảm dần số bit mã hóa xuống còn 6, 4, 2, 0 bit. Quan sát và vẽ dạng sóng và phổ tín hiệu ngõ ra. Nhận xét.*

*Trường hợp mỗi mẫu được mã hóa bằng 6bit.*

*Trường hợp mỗi mẫu được mã hóa bằng 4bit.*

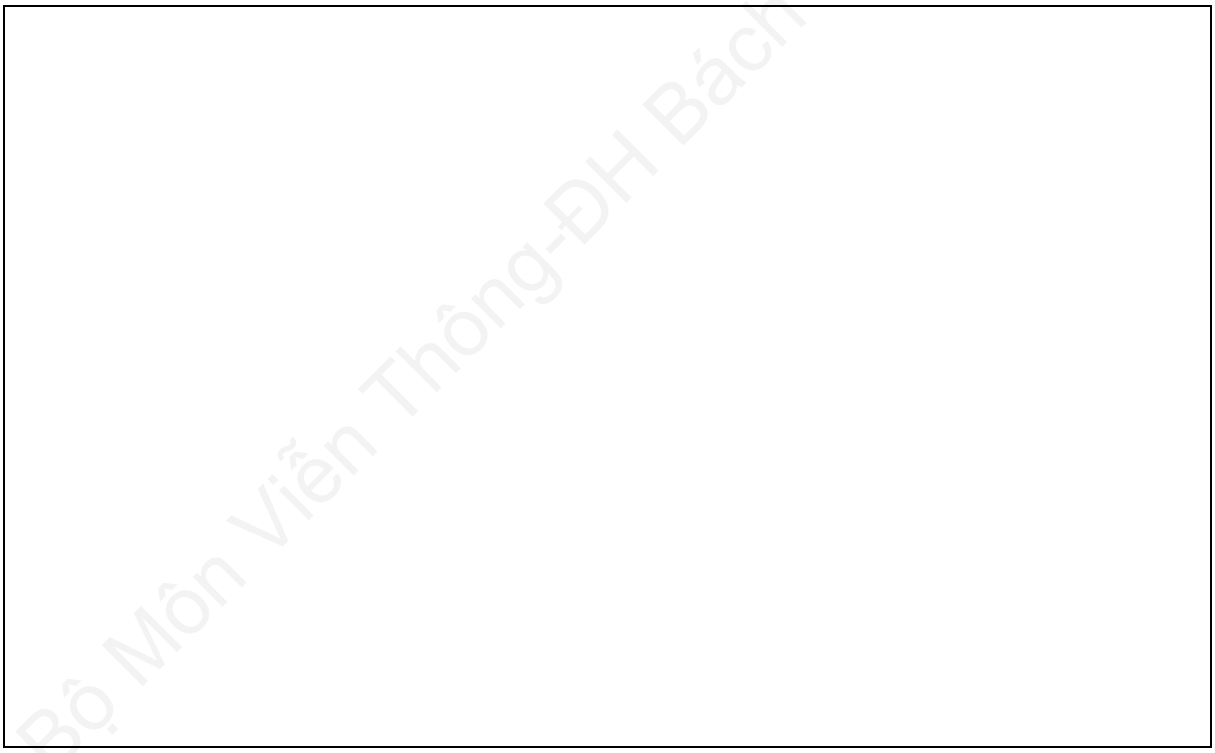


*Trường hợp mỗi mẫu được mã hóa bằng 2bit.*



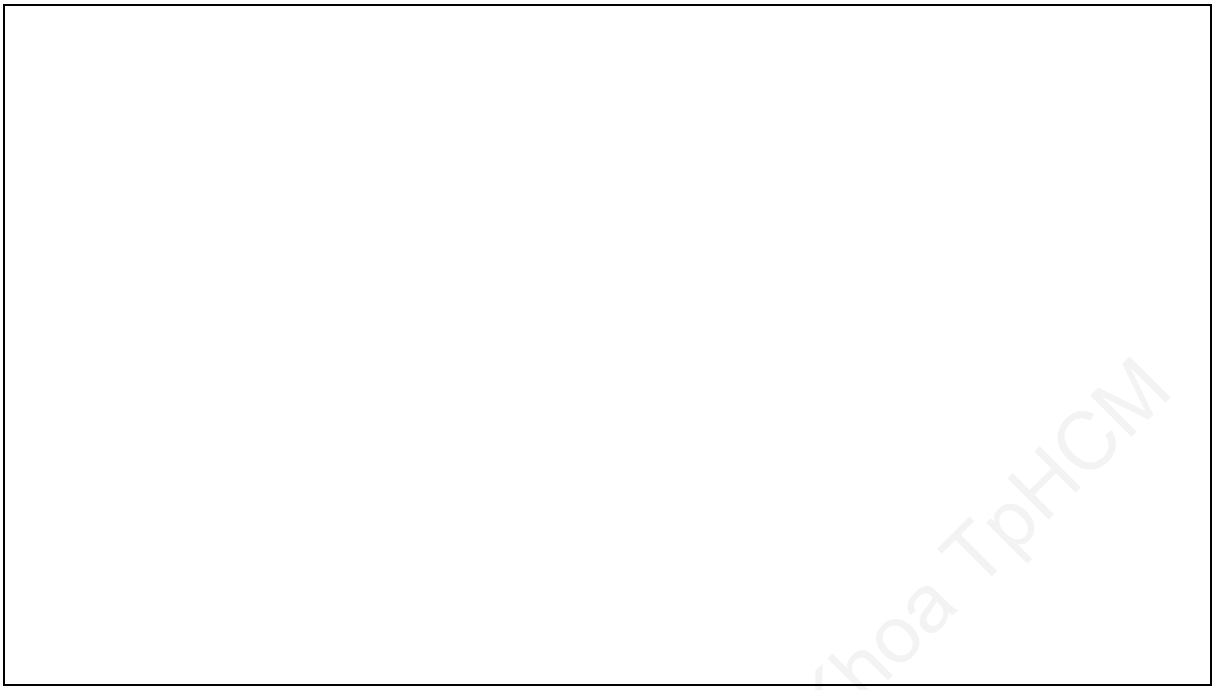
*Trường hợp mỗi mẫu được mã hóa bằng 0bit.*

- *Cho tín hiệu xung vuông có tần số 3KHz đi qua hệ thống. Tín hiệu được lấy mẫu với tần số 8KHz. Mỗi mẫu tín hiệu được mã hóa thành chuỗi 8bit. Tín hiệu sau đó được phục hồi lý tưởng. Quan sát và vẽ dạng sóng và phổ tín hiệu ngõ ra.*

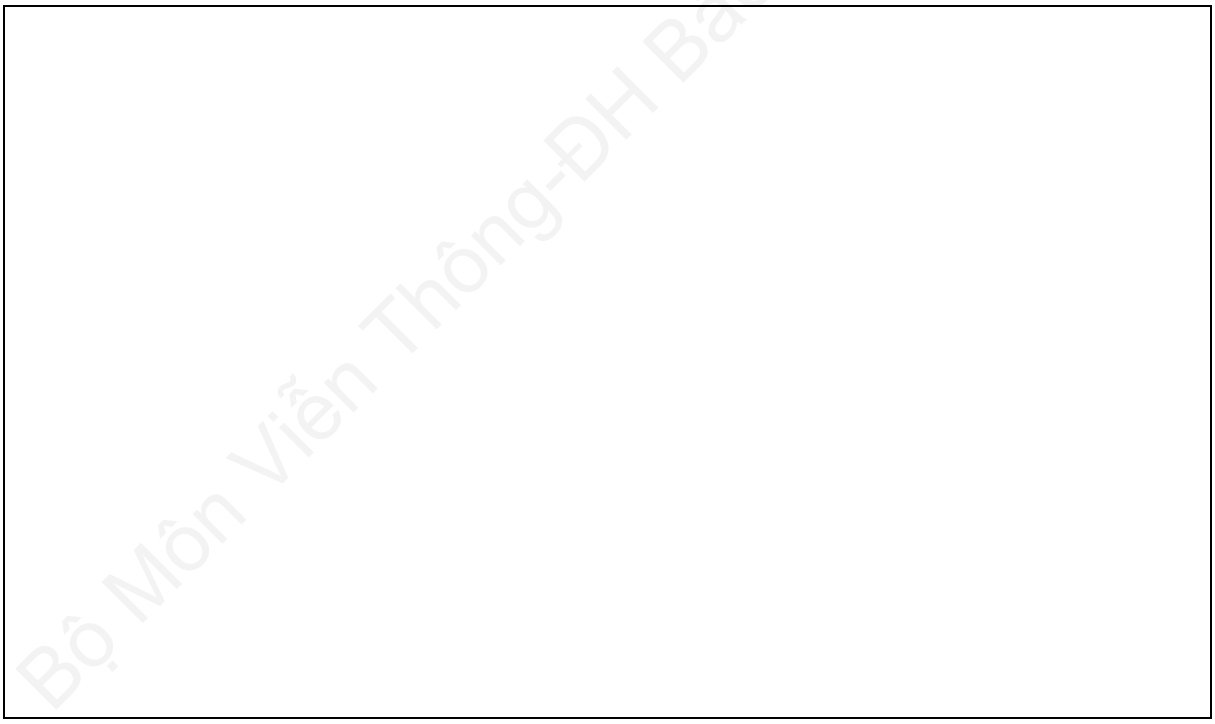


*Thực hiện việc giảm dần số bit mã hóa xuống còn 6, 4, 2, 0 bit. Quan sát và vẽ dạng sóng và phổ tín hiệu ngõ ra. So sánh và nhận xét với trường hợp sóng sin.*

*Trường hợp mỗi mẫu được mã hóa bằng 6bit.*



*Trường hợp mỗi mẫu được mã hóa bằng 4bit.*



*Trường hợp mỗi mẫu được mã hóa bằng 2bit.*

*Trường hợp mỗi mẫu được mã hóa bằng 0bit.*

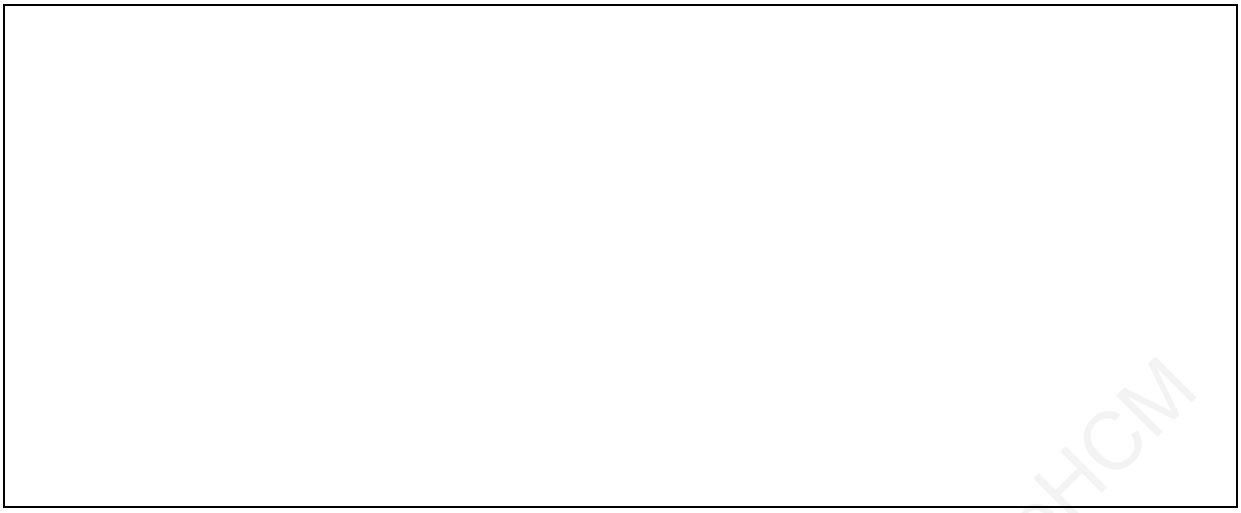
5. Một hệ thống có tần số lấy mẫu fs = 8KHz. Xác định tần số cắt của bộ tiền lọc lý tưởng để không xảy ra hiện tượng aliasing. Giải thích.

6. Một tín hiệu $x(t) = 5\sin(6\pi t)$ (t: ms). Xác định tần số lấy mẫu thấp nhất để có thể phục hồi lại tín hiệu. Tín hiệu được lấy mẫu với tần số 4KHz. Sau đó tín hiệu được phục hồi lý tưởng. Xác định tín hiệu sau khi được phục hồi lý tưởng. Giải thích ngắn gọn.

7. Cho một tín hiệu có tầm toàn thang R = 10V. Xác định số bit B để mã hóa tín hiệu được sai số lượng tử hiệu dụng (rms) không quá 50microV.

8. Cho một tín hiệu lưỡng cực có tầm toàn thang là 16V, được mã hóa thành 4bit bằng phương pháp rounding. Các mẫu tín hiệu có giá trị: -7.9, -7.1,- 6.8,- 5.5,-3.1, 0, 1.3, 2.6, 5.8, 6.9.
    a. Xác định chuỗi bit cho các mẫu trên nếu mã hóa bằng bộ mã offset binary.

    b. Lặp lại câu trên với bộ mã bù bậc 2.



3

# BỘ LỌC FIR/IIR TRÊN KIT C6713 DSK

**Họ và tên SV báo cáo 1: ……. MSSV: …….....**

**Họ và tên SV báo cáo 2: ……. MSSV: …….....**

**Họ và tên SV báo cáo 3: ……. MSSV: …….....**

**Họ và tên SV báo cáo 4: ……. MSSV: …….....**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: ……..……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |
| | | | | |

## 1. MỤC ĐÍCH THÍ NGHIỆM

- Hiểu rõ các bước từ thiết kế đến hiện thực bộ lọc FIR/IIR lên trên một kit DSP.
- Quan sát đáp ứng xung và đáp ứng tần số của bộ lọc.
- Kiểm tra đặc tính (thông thấp, thông cao, thông dải, chắn dải) của bộ lọc.
- Khảo sát ngõ ra của bộ lọc khi ngõ vào là tín hiệu xung vuông.
- Hệ thống lại các lý thuyết đã học.

# 2. THIẾT BỊ THÍ NGHIỆM

| STT | Tên thiết bị | Số lượng |
|---|---|---|
| 01 | Máy vi tính | 01 |
| 02 | Kit C6713 DSK | 01 |
| 03 | Máy phát sóng | 01 |
| 04 | Bộ dây nối tín hiệu | 01 |

# 3. CƠ SỞ LÝ THUYẾT

Lọc là một trong những hoạt động xử lý tín hiệu quan trọng. Một bộ lọc tương tự hoạt động trên các tín hiệu liên tục và thường được thực hiện với các linh kiện như khuếch đại thuật toán, các điện trở và các tụ điện. Một bộ lọc số hoạt động trên tín hiệu thời gian rời rạc và có thể thực hiện với một bộ xử lý số tín hiệu như họ TMS320C6x. Quá trình lọc bao gồm sử dụng một bộ biến đổi A/D để nhận tín hiệu vào, xử lý các mẫu vào rồi gửi kết quả ra thông qua một bộ biến đổi D/A.

Các bộ lọc số có rất nhiều ưu điểm so với các bộ lọc tương tự. Các ưu điểm này bao gồm độ tin cậy cao hơn, độ chính xác cao hơn và ít nhạy với nhiệt độ và tuổi đời. Các đặc tính lọc như tần số trung tâm, băng thông và loại bộ lọc có thể thay đổi dễ dàng. Một số công cụ có sẵn cho việc thiết kế và thực hiện các bộ lọc số một cách nhanh chóng trên kit TMS320C6x.

### 3.1. Bộ lọc FIR

Bộ lọc FIR nhân quả bậc M có đáp ứng xung h = [$h_0$, $h_1$, …, $h_M$] (chiều dài bằng M + 1). Ngõ ra của bộ lọc được xác định theo công thức tích chập:

$$y(n) = \sum_m h(m)x(n-m) = \sum_m x(m)h(n-m)$$

trong đó x(n) là ngõ vào của bộ lọc.

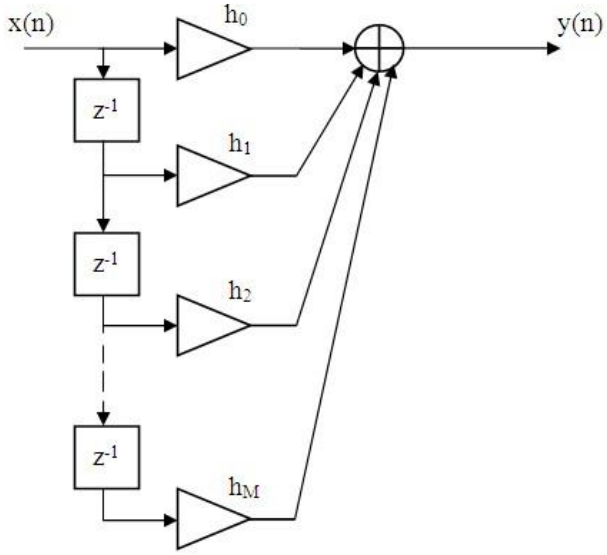
Hàm truyền của bộ lọc được xác định từ biến đổi Z của h(n):

$$H(z) = \sum_{n=0}^{M} h(n)z^{-n} = h_0 + h_1 z^{-1} + ... + h_M z^{-M}$$

trong đó các hệ số của hàm truyền chính là đáp ứng xung h của bộ lọc.

Bộ lọc FIR có thể thiết kế bằng nhiều phương pháp, trong đó phương pháp đơn giản nhất là phương pháp cửa sổ.

Bộ lọc có thể được thực hiện bằng hai phương pháp: Phương pháp xử lý khối và phương pháp xử lý mẫu. Với phương pháp xử lý mẫu, bộ lọc có thể được thực hiện dạng trực tiếp như sau:




**Hình 25. Thực hiện bộ lọc FIR dạng trực tiếp.**

Nếu đặt các biến trạng thái:

$v_0(n) = x(n)$

$v_1(n) = x(n - 1)$

…

$v_M(n) = x(n - M)$

Ta sẽ có giải thuật xử lý mẫu ứng với sơ đồ khối trên như sau:

Với mỗi mẫu vào x:

$v_0 = x$

$$y = \sum_{k=0}^{M} h_k v_k$$

$v_M = v_{M-1}$

…

$v_1 = v_0$

### 3.2. Bộ lọc IIR

Hãy xem xét một phương trình I/O tổng quát có dạng:

$$\begin{aligned} y(n) &= \sum_{k=0}^{N} a_k x(n-k) - \sum_{j=1}^{M} b_j y(n-j) \\ &= a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) + \cdots + a_N x(n-N) \\ &\quad - b_1 y(n-1) - b_2 y(n-2) - \cdots - b_M y(n-M) \end{aligned}$$

Dạng phương trình đệ quy này biểu diễn một bộ lọc IIR. Ngõ ra y(n) ở thời điểm y(n) không chỉ phụ thuộc vào ngõ vào hiện tại x(n) ở thời điểm n và các ngõ vào trong quá khứ x(n – 1),

x(n – 2), …, x(n – N), mà còn phụ thuộc vào các ngõ ra trước đó y(n – 1), y(n – 2), …, y(n – M).

Nếu chúng ta giả sử các điều kiện ban đầu đều bằng 0, biến đổi Z phương trình trên sẽ cho:

$$Y(z) = a_0 X(z) + a_1 z^{-1} X(z) + a_2 z^{-2} X(z) + \cdots + a_N z^{-N} X(z)$$
$$- b_1 z^{-1} Y(z) - b_2 z^{-2} Y(z) - \cdots - b_M z^{-M} Y(z)$$

Khi N = M, hàm truyền H(z) là

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots a_N z^{-N}}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_N z^{-N}} = \frac{N(z)}{D(z)}$$

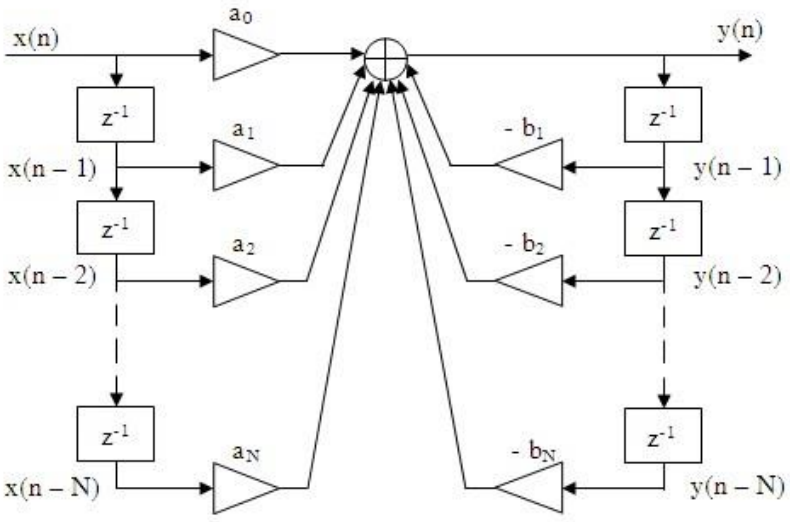
trong đó N(z) và D(z) biểu diễn đa thức tử số và đa thức tử số của hàm truyền. Nhân và chia cho $z^N$, H(z) trở thành:

$$H(z) = \frac{a_0 z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots a_N}{z^N + b_1 z^{N-1} + b_2 z^{N-2} + \cdots b_N} = C \prod_{i=1}^{N} \frac{z - z_i}{z - p_i}$$

Đây là một hàm truyền với N zero và N cực. Nếu tất cả các hệ số $b_j$ bằng 0, hàm truyền này trở thành hàm truyền của một bộ lọc FIR. Để hệ thống ổn định, tất cả các cực phải nằm trong vòng tròn đơn vị.

Các bộ lọc IIR có thể được thực hiện theo các cấu trúc sau:

### 1. Dạng trực tiếp 1




**Hình 26. Thực hiện bộ lọc IIR dạng trực tiếp 1.**

Khi thực hiện ở dạng này, một bộ lọc bậc N cần dùng 2N khối làm trễ.

### 2. Dạng trực tiếp 2 (Dạng chính tắc)

Đây là một trong những cấu trúc thường được sử dụng. Nó chỉ cần một nửa số khối trễ so với dạng trực tiếp 1.

Đặt $U(z) = \frac{X(z)}{D(z)}$ trong đó D(z) là mẫu số của hàm truyền bộ lọc IIR.

Khi đó:

$$Y(z) = \frac{N(z)}{D(z)} X(z) = N(z)U(z)$$
$$= U(z)(a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots a_N z^{-N}$$

với N(z) là tử số của hàm truyền.

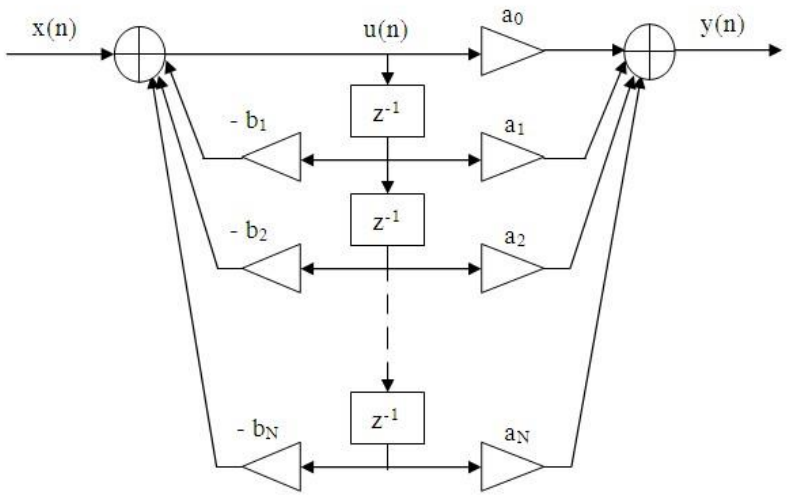
Và:

$$X(z) = U(z)D(z) = U(z)(1 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_N z^{-N})$$

Biến đổi Z ngược ta sẽ có:

$$u(n) = x(n) - b_1 u(n-1) - b_2 u(n-2) - \cdots - b_N u(n-N)$$
$$y(n) = a_0 u(n) + a_1 u(n-1) + a_2 u(n-2) + \cdots + a_N u(n-N)$$

Thực hiện dưới dạng sơ đồ khối:




**Hình 27. Thực hiện bộ lọc IIR dạng trực tiếp 2.**

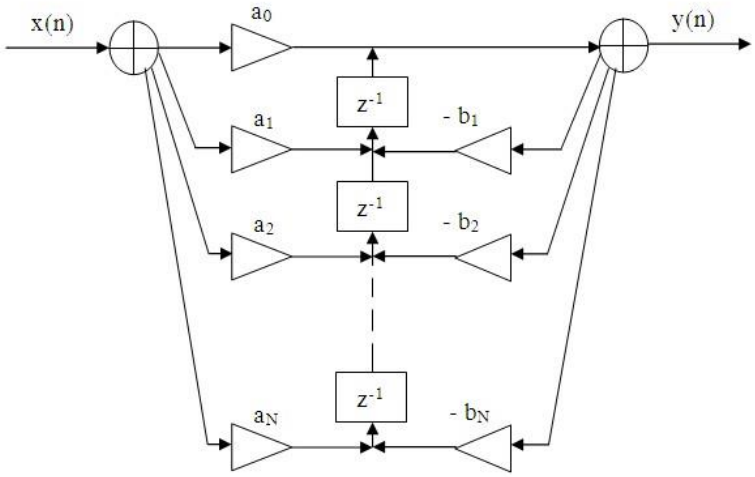
### 3. Dạng trực tiếp 2 chuyển vị

Dạng trực tiếp 2 chuyển vị là một biến thể của dạng trực tiếp 2 và cần cùng số khối trễ. Các bước sau chuyển một bộ lọc từ dạng trực tiếp 2 sang dạng chuyển vị:

i. Đảo hướng tất cả các nhánh

ii. Đổi đầu vào với đầu ra

iii. Vẽ lại sơ đồ sao cho đầu vào ở bên trái và đầu ra ở bên phải



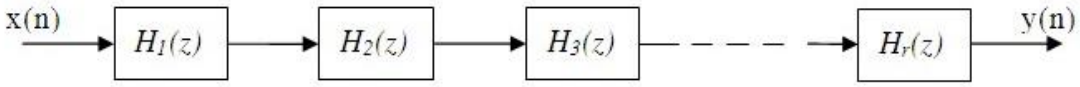

**Hình 28. Thực hiện bộ lọc IIR dạng trực tiếp 2 chuyển vị.**

## 4. Dạng cascade các tầng bậc 2

Hàm truyền trên có thể được phân tích thành tích các hàm truyền bậc 1 hoặc bậc 2 như sau:

$$H(z) = CH_1(z)H_2(z)\cdots H_r(z)$$

Cấu trúc nối tiếp (cascade) này được vẽ như sau:

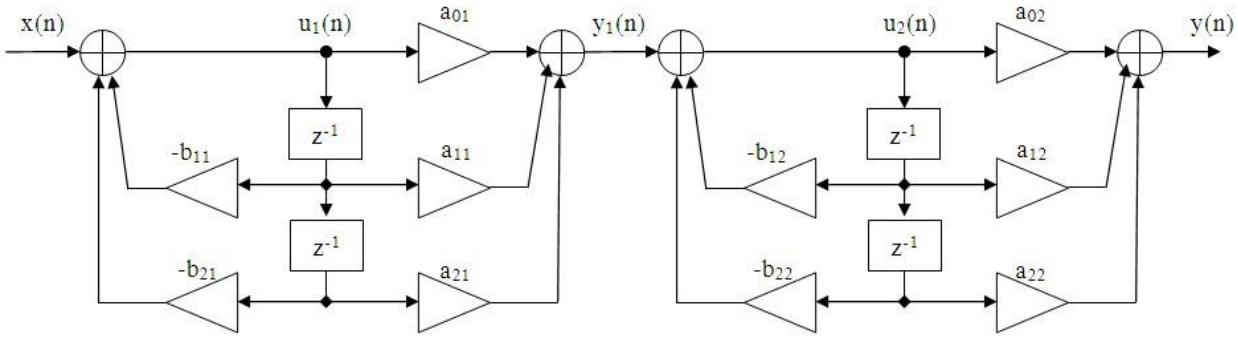



**Hình 5. Cấu trúc cascade của bộ lọc IIR.**

Hàm truyền toàn bộ có thể được biểu diễn bằng sự ghép cascade các hàm truyền. Đối với mỗi phần, dạng trực tiếp 2 hoặc chuyển vị của nó có thể được sử dụng. Hàm truyền H(z) dưới dạng cascade các hàm truyền bậc hai có thể viết như sau:

$$H(z) = \prod_{i=1}^{N/2} \frac{a_{0i} + a_{1i}z^{-1} + a_{2i}z^{-2}}{1 + b_{1i}z^{-1} + b_{2i}z^{-2}}$$

Hình sau vẽ một bộ lọc IIR bậc 4 dưới dạng cascade của hai phần bậc 2.



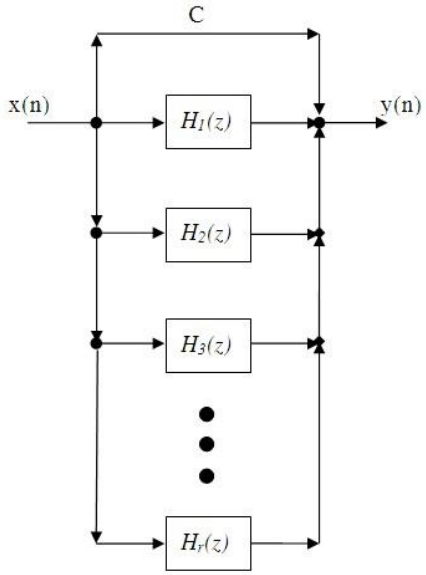

**Hình 29. Bộ lọc IIR bậc 4 với 2 phần bậc 2 dạng trực tiếp 2.**

### 5. Dạng song song

Hàm truyền bộ lọc IIR cũng có thể được biểu diễn như sau (bằng phương pháp khai triển phân số từng phần):

$$H(z) = C + H_1(z) + H_2(z) + \cdots H_r(z)$$

Cấu trúc song song này có thể vẽ như sau:




**Hình 30. Cấu trúc song song của bộ lọc IIR.**

## 4. CHUẨN BỊ LÝ THUYẾT THÍ NGHIỆM

### 4.1. Bộ lọc FIR

Cho một bộ lọc FIR có đáp ứng xung h = [1; -2; -3; -4].

❖ **Câu hỏi chuẩn bị (lý thuyết)**

1. Xác định bậc bộ lọc?

2. Viết phương trình sai phân I/O của bộ lọc?

3. Vẽ sơ đồ khối thực hiện dạng trực tiếp và giải thuật xử lý mẫu.

4. Viết biểu thức hàm truyền của bộ lọc?

5. Vẽ đáp ứng biên độ-tần số và pha-tần số của bộ lọc?

6. Xác định đặc tính (thông thấp, thông cao, thông dải, chắn dải) của bộ lọc?

7. Xác định tần số cắt -3dB và độ dốc của bộ lọc?

### 4.2. Bộ lọc IIR

Cho một bộ lọc có hàm truyền như sau:

$$H(z) = \frac{5}{1+0.25z^{-2}} - \frac{4}{1-0.25z^{-2}}$$

❖ **Câu hỏi chuẩn bị (lý thuyết)**

1. Xác định bậc bộ lọc?

2. Viết phương trình sai phân I/O của bộ lọc?

3. Hãy vẽ cách thực hiện dạng trực tiếp (direct form) của bộ lọc.

4. Hãy vẽ cách thực hiện dạng chính tắc (canonical form) của bộ lọc.

5. Hãy vẽ cách thực hiện dạng ghép nối tiếp các tầng bậc hai (cascade form) của bộ lọc.

6. Vẽ đáp ứng biên độ-tần số và pha-tần số của bộ lọc?

7. Xác định đặc tính (thông thấp, thông cao, thông dải, chắn dải) của bộ lọc?

8. Xác định tần số cắt -3dB và độ dốc của bộ lọc?

❖ **Câu hỏi chuẩn bị (thí nghiệm):** sinh viên đọc kỹ tiến trình thí nghiệm để trả lời.

1. Số lượng bộ lọc cần phải thực hiện trong bài thí nghiệm?
2. Các phương pháp thiết kế bộ lọc FIR/IIR dùng MATLAB?
3. Tóm tắt quy trình thiết kế bộ lọc FIR/IIR dùng MATLAB?
4. Các hệ số đáp ứng xung của bộ lọc FIR/IIR dùng trong chương trình DSP được lưu trong tập tin nào?
5. So sánh sự khác biệt về cách thức biểu diễn của đáp ứng xung bộ lọc FIR/IIR trong phần mềm MATLAB và C (kit DSP)?
6. Tóm tắt quy trình thực hiện bộ lọc FIR/IIR trên kit DSP?
7. Những thông số cần điều chỉnh khi quan sát đáp ứng xung và đáp ứng tần số của bộ lọc dùng chương trình CCS của kit DSP?
8. Những chức năng nào của nguồn máy phát sóng cần điều chỉnh trong bài thí nghiệm? Những lưu ý cần quan tâm khi sử dụng nguồn máy phát sóng?
9. Tìm hiểu một số lệnh trong MATLAB liên quan đến xử lý chuỗi để hỗ trợ chuyển đổi kết quả từ MATLAB sang C dùng cho kit DSP? Viết đoạn chương trình MATLAB hỗ trợ việc chuyển đổi này?

# 5. TIẾN TRÌNH THÍ NGHIỆM

Trong phần thí nghiệm này có 3 yêu cầu chính cần phải thực hiện:

1. Thiết kế bộ lọc: Kết quả của phần này là có được đáp ứng xung $h(n)$ của bộ lọc.
2. Thực hiện bộ lọc lên trên kit C6713 DSK: Sử dụng đáp ứng xung thu được từ phần thiết kế, viết chương trình thực hiện mạch lọc lên kit. Chương trình sẽ đọc từng mẫu dữ liệu vào và tiến hành giải thuật xử lý mẫu để tính ngõ ra.
3. Kiểm tra bộ lọc đã thực hiện: Trong phần này, bộ lọc đã thực hiện trên kit sẽ được kiểm tra xem có đáp ứng yêu cầu đặt ra hay không. Chúng ta sẽ sử dụng một máy phát sóng để tạo tín hiệu ngõ vào và quan sát tín hiệu ngõ ra của bộ lọc khi thay đổi tín hiệu ngõ vào.

## 5.1. Các bộ lọc FIR

### 5.1.1. Bộ lọc FIR chắn dải

Thiết kế, thực hiện và khảo sát bộ lọc FIR chắn dải bằng phương pháp cửa sổ Kaiser với các thông số sau:

- *Chiều dài của đáp ứng xung: N = 63 (MATLAB hiển thị bậc bộ lọc bằng 62)*
- *Tần số trung tâm: 2700 Hz*
- *Tần số cắt: 2500 Hz và 2900 Hz*
- *Giá trị của $\beta = 4$*
- *Tần số lấy mẫu 8000 Hz*

❖ **Thiết kế bộ lọc dùng MATLAB**

1. Khởi động SPTool. Dưới cột Filters, nhấn nút New để mở cửa sổ Filter Designer.
2. Trong giao diện của Filter Designer:
   a. Trong text box Filter: Tên bộ lọc được tự đặt (ở đây là **filt2**). Tên này có thể thay đổi sau này.
   b. Nhập các thông số thiết kế vào:
      - Response Type = Bandstop
      - Design Method = FIR Window
      - Specify Order: 62
      - Window: Kaiser, Beta: 4
      - Frequency Specifications: Fs = 8000, Fc1 = 2500, Fc2 = 2900.
   c. Nhấn Design Filter. Khi đó đáp ứng tần số của bộ lọc thiết kế sẽ được hiển thị.
3. Trở về cửa sổ SPTool, trong cột Filters sẽ xuất hiện thêm một dòng **filt2 [design]**. Đây chính là bộ lọc vừa thiết kế. Thay đổi tên bộ lọc trên thành **bs2700** bằng cách chọn Edit → Name…→filt2 [design]. Trong cửa sổ mới xuất hiện, nhập tên mới.

Ghi lại kết quả và kiểm tra xem đây có phải bộ lọc chắn dải như mong muốn không?

Khi thiết kế một bộ lọc FIR như trên, kết quả mà ta cần nhận được sau khi thiết kế là các giá trị của vector đáp ứng xung **h** của bộ lọc thiết kế. Để lấy các giá trị của vector đáp ứng xung, ta thực hiện như sau:

1. Từ cửa sổ SPTool, chọn File → Export… Trong Export list xuất hiện, chọn *Filter: bs2700 [design]* rồi nhấn nút **Export to workspace**
2. Đóng cửa sổ SPTool lại. Một thông báo xuất hiện hỏi có muốn lưu lại phiên làm việc hiện tại hay không. Nếu muốn lưu lại, chọn Save.
3. Mở cửa sổ Workspace của MATLAB, ta sẽ thấy trong workspace sẽ xuất hiện biến mới là **bs2700**. Đây chính là bộ lọc mà ta đã thiết kế trong SPTool và xuất ra workspace của MATLAB. Biến này được lưu dưới dạng một cấu trúc mô tả bộ lọc đã thiết kế. Nhấn đúp chuột vào tên biến bs2700 trong workspace, ta sẽ thấy được các field của cấu trúc này.
4. Trong các field này, field **tf** thể hiện hàm truyền của bộ lọc. Field này cũng là một cấu trúc gồm 2 field: **tf.num** và **tf.den** thể hiện tương ứng các hệ số của đa thức tử số và đa thức mẫu số. Đối với bộ lọc FIR, hàm truyền chỉ có tử số và các hệ số của tử số chính là đáp ứng xung của bộ lọc. Do đó, với bộ lọc trên, các giá trị của vector đáp ứng xung được lưu trong **bs2700.tf.num**. Trong cửa sổ Array Editor trên, lần lượt nhấn đúp vào field **tf** rồi nhấn đúp vào **num**, ta sẽ thấy các hệ số đáp ứng xung của bộ lọc. Để gán các hệ số này vào một vector h, trong MATLAB có thể dùng lệnh sau:

   ```
   >> h = bs2700.tf.num
   ```

5. Các giá trị thu được của vector đáp ứng xung sẽ được sử dụng để thực hiện bộ lọc số lên trên kit DSP. Do chương trình khảo sát bộ lọc số lên trên kit DSP sử dụng chế độ 16 bit có dấu trong khi chương trình thiết kế bộ lọc bằng MATLAB chuẩn hóa các hệ số đáp ứng xung trong khoảng [-1 1] nên các hệ số đáp ứng xung này cần nhân với $2^{15}$ và làm tròn về số nguyên trước khi đưa vào thực hiện bộ lọc số lên trên kit DSP như sau:

   ```
   >> cof = round(h*2^15)
   ```

Ghi nhận giá trị các hệ số của đáp ứng xung này.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

❖ **Thực hiện bộ lọc trên kit DSP**

Bộ lọc này được thực hiện trên kit bằng chương trình sau (viết bằng ngôn ngữ C)

```
//Fir.c  FIR filter. Include coefficient file with length N
#include "coefficients.h"                    //coefficient file
#include "dsk6713_aic23.h"                   //codec-dsk support file

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
int yn = 0;                                  //initialize filter's output
short dly[N];              //delay samples

interrupt void c_int11()    //ISR
{
    short i;
    dly[0]=input_sample();          //input newest sample
    yn = 0;                  //initialize filter's output
    for (i = 0; i< N; i++)
        yn += (h[i] * dly[i]);  //y(n) += h(i)* x(n-i)
    for (i = N-1; i > 0; i--)  //starting @ end of buffer
        dly[i] = dly[i-1];      //update delays with data move
 output_sample(yn >> 15);   //scale output filter sample
 return;
```

```
}

void main()
{
 comm_intr();            //init DSK, codec, McBSP
 while(1);              //infinite loop
}
```

Trong chương trình này, N là chiều dài của đáp ứng xung của bộ lọc (bằng M + 1 với M là bậc của bộ lọc) và đáp ứng xung của bộ lọc là mảng h có kích thước N. Giá trị của N và vector h được khai báo trong tập tin coefficients.h. Tập tin này được gộp vào nhờ chỉ dẫn #include. Như vậy, khi muốn thay đổi bộ lọc, chỉ cần thay đổi nội dung của tập tin coefficients.h.

Chương trình trên có sử dụng ngắt. Khi có có xung lấy mẫu (tần số chọn ở đây là 8KHz), trình phục vụ ngắt c_int11 được gọi, đọc mẫu vào và thực hiện giải thuật xử lý mẫu để tính ngõ ra.

*(Hướng dẫn: nên sao chép thư mục FIR đã có thành một thư mục với tên khác và thực hiện trên thư mục mới này ứng với từng bộ lọc).*

- **Tóm lại, các bước để thực hiện bộ lọc FIR lên kit như sau:**

1. Lấy các hệ số đáp ứng xung cof của bộ lọc thiết kế ở định dạng 16 bit có dấu.
2. Mở CCS (nhớ mở nguồn của DSK trước khi mở CCS). Kiểm tra kết nối.
3. Mở tập tin project (đã được tạo sẵn) FIR.pjt trong C:\CCStudio_v3.1\myprojects\FIR.
4. Trong cửa sổ Project View, tab File View, mở rộng phần Include, mở tập tin coeficients.h.
5. Đặt các hệ số đáp ứng xung của bộ lọc vừa thiết kế vào trong tập tin này. (Có thể copy và paste từ cửa sổ Array Editor trên). Điều chỉnh giá trị N cho đúng với chiều dài đáp ứng xung. Lưu ý rằng các giá trị của đáp ứng xung cách nhau bằng một dấu phẩy (,). Lưu tập tin sau khi sửa đổi.
6. Xác lập các tùy chọn phù hợp (xem phần hướng dẫn sử dụng trong tài liệu này) rồi tiến hành biên dịch chương trình. Sau khi dịch thành công, hãy nạp chương trình lên trên kit và chạy chương trình.

❖ **Đánh giá kết quả thực hiện**

1. Chọn View → Graph → Time/Frequency. Thay đổi các tùy chọn trong cửa sổ Graph Property Dialog để vẽ trong miền thời gian. Địa chỉ bắt đầu của bộ đệm chính là tên mảng *h* được nhập vào Start Address. Các tùy chọn khác có thể để như mặc định.

   Ghi nhận dạng sóng đáp ứng xung của bộ lọc:

2. Chọn View → Graph → Time/Frequency, sau đó chọn **Display type** là ***FFT Magnitude*** và địa chỉ bắt đầu (***Start Address***) là h. Chọn bậc của FFT (FFT Order) sao cho FFT Framesize = $2^{order}$.

   Ghi nhận đáp ứng biên độ-tần số và pha-tần số của bộ lọc:

❖ **Kiểm tra bộ lọc:**

1. Mở nguồn của máy phát sóng. Tạo một tín hiệu vào hình sine từ máy phát sóng, lần lượt thay đổi tần số của tín hiệu vào từ 100Hz đến 4KHz (mỗi lần 100Hz), ghi nhận biên độ dạng sóng và biên độ phổ của tín hiệu ngõ ra từ đó xác định đặc tính của bộ lọc.



***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am mức vừa phải trên màn hình hiển thị. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

2. Tạo một sóng vuông từ máy phát sóng, lần lượt thay đổi tần số của tín hiệu vào ghi nhận các thành phần tần số của ngõ ra. Giải thích tại sao có dạng phổ này?

| $f_i$ (Hz) | 100 | 200 | 550 | 900 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.1.2. Bộ lọc FIR thông dải

Tương tự như phần trên, hãy thiết kế, thực hiện và kiểm tra bộ lọc FIR thông dải bằng phương pháp Kaiser Window với các thông số như sau:

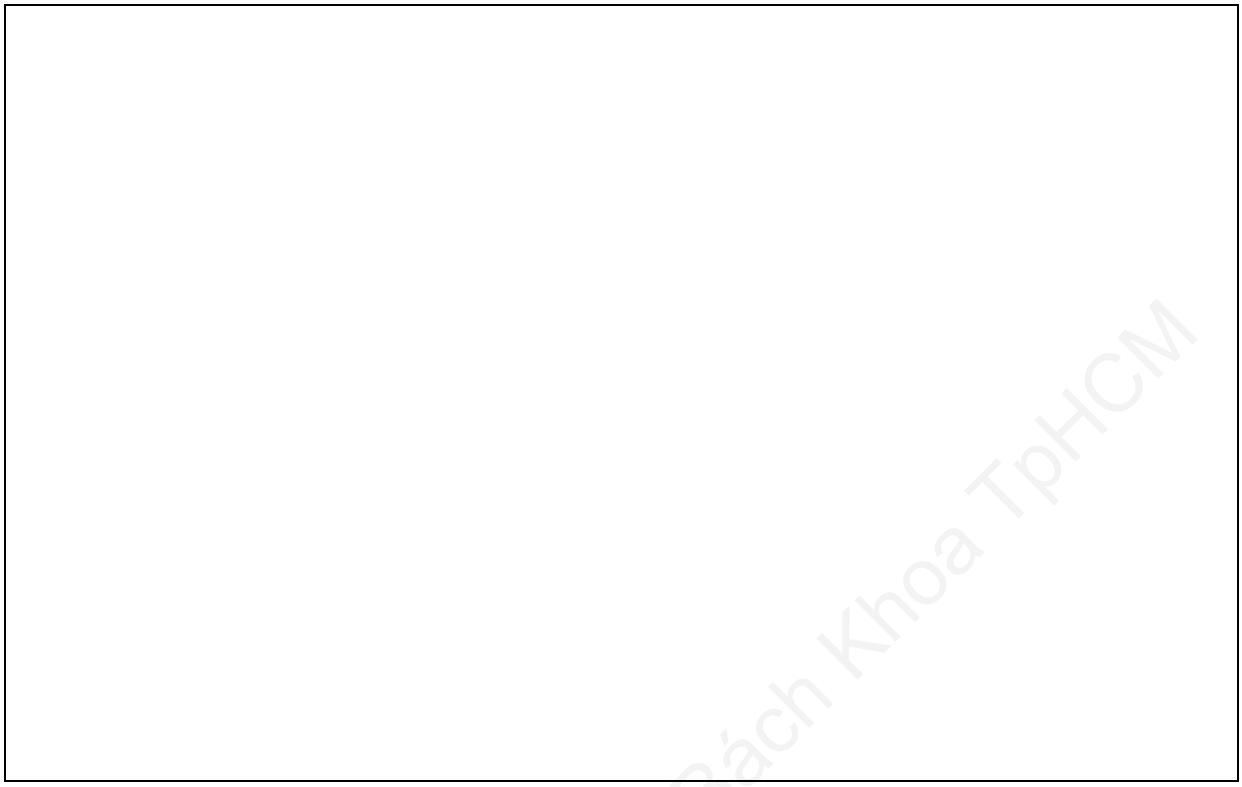
- Chiều dài đáp ứng xung: 63
- Tần số cắt: 1500 và 2000 Hz.
- Giá trị của $\beta = 5$
- Tần số lấy mẫu: 8 kHz.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:

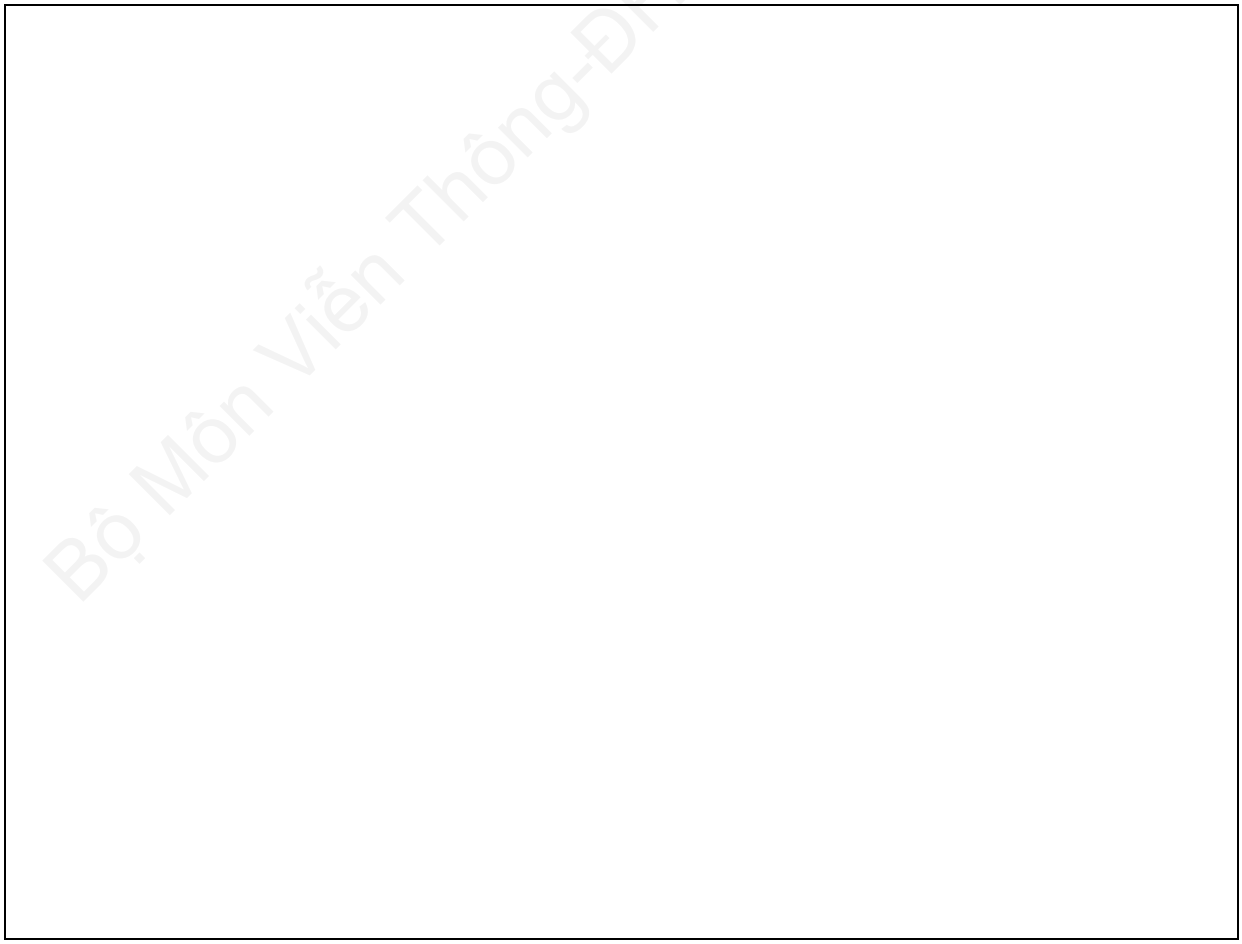
2. Gía trị các hệ số đáp ứng xung của bộ lọc thực hiện trên kit DSP:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

3. Dạng sóng đáp ứng xung của bộ lọc thực hiện trên kit DSP:



4. Đáp ứng biên độ-tần số và pha-tần số của bộ lọc thực hiện trên kit DSP:



5. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:



***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am mức vừa phải trên màn hình hiển thị. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

6. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 100 | 200 | 350 | 600 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.1.3. Bộ lọc FIR thông cao

Tương tự như trên, hãy thiết kế, thực hiện và kiểm tra bộ lọc FIR thông cao bằng phương pháp Kaiser Window với các thông số như sau::

- Chiều dài đáp ứng xung: 63
- Tần số cắt: 2200 Hz.
- Giá trị của $\beta = 4$
- Tần số lấy mẫu: 8 kHz.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:

| |
|---|
| |

2. Gía trị các hệ số đáp ứng xung của bộ lọc thực hiện trên kit DSP:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

3. Dạng sóng đáp ứng xung của bộ lọc thực hiện trên kit DSP:

4. Đáp ứng biên độ-tần số và pha-tần số của bộ lọc thực hiện trên kit DSP:

5. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:



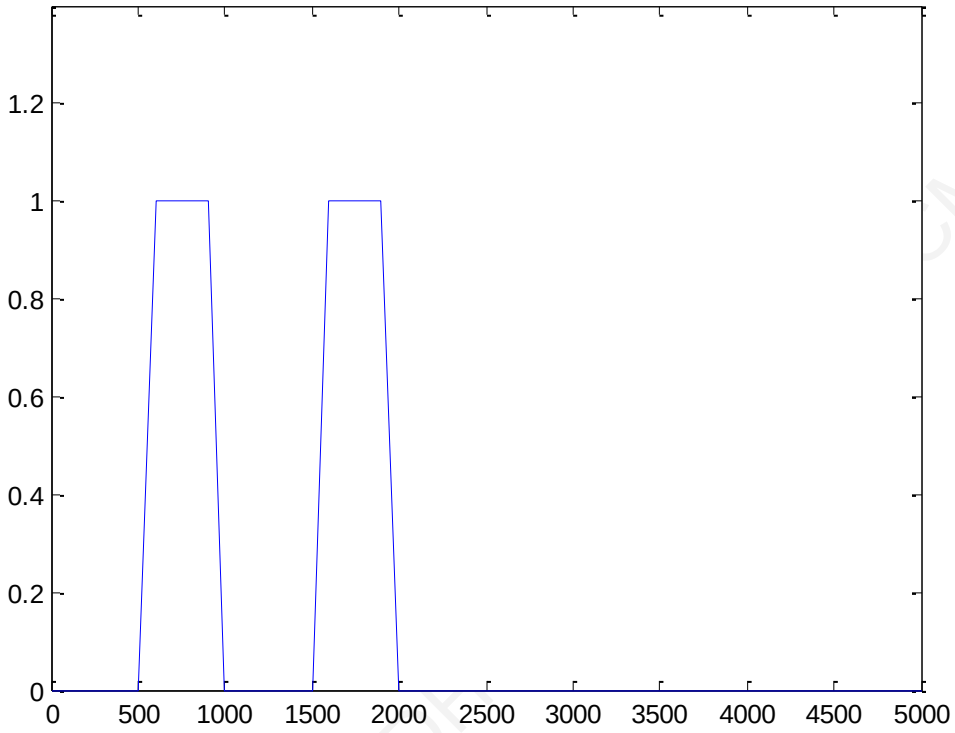
***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am mức vừa phải trên màn hình hiển thị. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

6. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 500 | 600 | 800 | 1000 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.1.4. Bộ lọc FIR multiband

Thiết kế, thực hiện và kiểm tra một bộ lọc FIR multiband gồm 63 hệ số, tần số lấy mẫu là 10 kHz, dải thông [500 1000] Hz và [1500 2000] Hz, độ rộng dải chuyển tiếp 100 Hz. Bộ lọc cần thiết kế có đáp ứng tần số như sau:




**Hình 31. Đáp ứng tần số của bộ lọc multiband FIR cần thiết kế.**

❖ **Thiết kế bộ lọc dùng MATLAB**

Bộ lọc mong muốn có 2 dải thông, được biểu diễn bởi 5 dải như sau:

| Dải | Tần số (Hz) | Tần số chuẩn hóa $f/F_N$ | Biên độ |
|---|---|---|---|
| **1** | 0 – 500 | 0 – 0.1 | 0 |
| **2** | 600 – 900 | 0.12 – 0.18 | 1 |
| **3** | 1000 – 1500 | 0.2 – 0.3 | 0 |
| **4** | 1600 – 1900 | 0.32 – 0.38 | 1 |
| **5** | 2000 - 5000 | 0.4 - 1 | 0 |

trong đó $F_N$ là tần số Nyquist, bằng ½ tần số lấy mẫu.

Chúng ta viết một file .m để thiết kế bộ lọc này, lưu lại với tên multibandfir63.m. Nội dung của file này như sau:

```
%multibandfir63.m: Multiband FIR filter with 63 coefficients
f = [0 0.1 0.12 0.18 0.2 0.3 0.32 0.38 0.4 1];
m = [0 0 1 1 0 0 1 1 0 0];
n = 63;
cof = remez(n-1,f,m);

% frequency response with 256 points
[h w] = freqz(cof,1,256);
% plot magnitude of the filter
plot(5000*f,m);
figure;
plot(w/pi,abs(h));
```

Trong đoạn chương trình trên, có một số lưu ý:

- Dòng lệnh **cof = remez(n-1,f,m)** trả về vector hệ số của bộ lọc FIR bậc n – 1, với f và m xác định các dải tần số theo bảng ở trên.
- Lệnh freqz để tính đáp ứng tần số của bộ lọc
- Lệnh plot thứ nhất vẽ đáp ứng tần số *mong muốn* dựa trên f và m.
- Lệnh figure tạo ra một cửa sổ mới và lệnh plot thứ hai vẽ đáp ứng tần số của bộ lọc *đã thiết kế được* lên cửa sổ mới này.

Ở đây, kết quả của quá trình thiết kế mà ta cần nhận được chính là các hệ số chứa trong biến **cof**. Chúng được sử dụng khi thực hiện bộ lọc lên trên kit DSP (lưu ý đưa về định dạng 16 bit có dấu khi thực hiện trên kit DSP). Tiếp theo thực hiện tương tự như phần II.1.1.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:

2. Gía trị các hệ số đáp ứng xung của bộ lọc thực hiện trên kit DSP:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

3. Dạng sóng đáp ứng xung của bộ lọc thực hiện trên kit DSP:

4. Đáp ứng biên độ-tần số và pha-tần số của bộ lọc thực hiện trên kit DSP:

5. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:



***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am mức vừa phải trên màn hình hiển thị. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

6. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 200 | 350 | 600 | 800 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

## 5.2. Các bộ lọc IIR

### 5.2.1. Bộ lọc IIR chắn dải

Thiết kế, thực hiện và kiểm tra một bộ lọc IIR chắn dải bậc 10, tần số trung tâm 1750Hz, sử dụng phương pháp Elliptic với các thông số như sau:

- *Tần số cắt: 1700 Hz và 1800 Hz*
- *Độ gợn dải thông và dải chắn tương ứng là 1 dB và 60 dB*
- *Tần số lấy mẫu: 8000 Hz*

❖ **Thiết kế bộ lọc dùng MATLAB:**

1. Khởi động SPTool. Dưới cột Filters, nhấn nút New để mở cửa sổ Filter Designer.
2. Trong giao diện của Filter Designer:
   a. Trong text box Filter: Tên bộ lọc được tự đặt (ở đây là **filt2**). Tên này có thể thay đổi sau này.
   b. Nhập các thông số thiết kế vào:
      - Response Type = Bandstop
      - Design Method = IIR Elliptic
      - Specify Order: 10
      - Frequency Specifications: Fs = 8000, Fc1 = 1700, Fc2 = 1800.
      - Apass = 1, Astop = 60.
   c. Nhấn Design Filter. Khi đó đáp ứng tần số của bộ lọc thiết kế sẽ được hiển thị. Lưu lại kết quả và kiểm tra xem đây có phải bộ lọc chắn dải như mong muốn không?

3. Trở về cửa sổ SPTool, trong cột Filters sẽ xuất hiện thêm một dòng **filt2 [design]**. Đây chính là bộ lọc vừa thiết kế. Thay đổi tên bộ lọc trên thành **bs1750** bằng cách chọn Edit → Name…→filt2 [design]. Trong cửa sổ mới xuất hiện, nhập tên mới.
4. Từ cửa sổ SPTool, chọn File → Export… Trong Export list xuất hiện, chọn *Filter: bs1750 [design]* rồi nhấn nút **Export to workspace**
5. Đóng cửa sổ SPTool lại. Một thông báo xuất hiện hỏi có muốn lưu lại phiên làm việc hiện tại hay không. Nếu muốn lưu lại, chọn Save.
6. Mở cửa sổ Workspace của MATLAB, ta sẽ thấy trong workspace sẽ xuất hiện biến mới là **bs1750**. Đây chính là bộ lọc mà ta đã thiết kế trong SPTool và xuất ra workspace của MATLAB. Biến này được lưu dưới dạng một cấu trúc mô tả bộ lọc đã thiết kế. Nhấn đúp chuột vào tên biến bs1750 trong workspace, ta sẽ thấy được các field của cấu trúc này.
7. Trong các field này, field **tf** thể hiện hàm truyền của bộ lọc. Field này cũng là một cấu trúc gồm 2 field: **tf.num** và **tf.den** thể hiện tương ứng các hệ số của đa thức tử số và đa thức mẫu số. Để chuyển các hệ số này sang dạng mỗi tầng bậc hai (second-order section), trong MATLAB có thể dùng các lệnh sau:

```
>> [z,p,k] = tf2zp(bs1750.tf.num, bs1750.tf.den) ;
>> sos = zp2sos(z,p,k);
```

Ma trận *sos* trong MATLAB như sau:

$$sos = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

trong đó $b_{0i}$, $b_{1i}$, $b_{2i}$ là các hệ số tử số hàm truyền của phần bậc 2 thứ i và 1, $a_{1i}$, $a_{2i}$ là các hệ số mẫu số hàm truyền của phần bậc 2 thứ i.

```
>> sos = round(sos*2^15)
```

Các phần tử của ma trận sos này sẽ được sử dụng để thực hiện bộ lọc IIR lên kit DSP. Ghi nhận các giá trị này.

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

### ❖ Thực hiện bộ lọc IIR trên kit DSP

Bộ lọc này được thực hiện trên kit bằng chương trình sau (viết bằng ngôn ngữ C)

```
//IIR.c IIR filter using cascaded Direct Form II
//Coefficients a's and b's correspond to b's and a's from MATLAB
#include "DSK6713_AIC23.h"              //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#include "coefficients.h"                    //BS @ 1750 Hz coefficient file
short dly[stages][2] = {0};                  //delay samples per stage
interrupt void c_int11()             //ISR
{
    short i, input;
    int un, yn;
    input = input_sample(); //input to 1st stage
    for (i = 0; i < stages; i++) //repeat for each stage
    {
        un=input-((b[i][0]*dly[i][0])>>15)- ((b[i][1]*dly[i][1])>>15);
        yn=((a[i][0]*un)>>15)+((a[i][1]*dly[i][0])>>15)+
       ((a[i][2]*dly[i][1])>>15);
        dly[i][1] = dly[i][0];       //update delays
        dly[i][0] = un;              //update delays
        input = yn;                  //intermed out->in to next stage
    }
    output_sample((short)yn); //output final result for time n
    return;                          //return from ISR
}
void main()
{
    comm_intr(); //init DSK, codec, McBSP
    while(1); //infinite loop
}
```

Trong chương trình này, stages là số tầng của bộ lọc (số phần bậc 2). Ma trận a và b tương ứng chứa các hệ số của tử số và mẫu số của các phần bậc 2. **Lưu ý: a và b trong chương trình này ứng với b và a trong ma trận sos của MATLAB.** Giá trị của stages, ma trận a, ma trận b được khai báo trong tập tin coefficients.h. Tập tin này được include vào nhờ chỉ dẫn #include. Như vậy, khi muốn thay đổi bộ lọc, chỉ cần thay đổi nội dung của tập tin coefficients.h.

Nội dung của tập tin coefficients.h có dạng như sau:

```
//coefficients.cof coefficient file

#define stages 5                //number of 2nd-order stages

int a[stages][3]= {             //numerator coefficients
{27940, -10910, 27940},             //a10, a11, a12 for 1st stage
{32768, -11841, 32768},             //a20, a21, a22 for 2nd stage
{32768, -13744, 32768},             //a30, a31, a32 for 3rd stage
{32768, -11338, 32768},             //a40, a41, a42 for 4th stage
{32768, -14239, 32768}  };

int b[stages][2]= {             //*denominator coefficients
{-11417, 25710},                    //b11, b12 for 1st stage
{-9204, 31581},                     //b21, b22 for 2nd stage
{-15860, 31605},                    //b31, b32 for 3rd stage
{-10221, 32581},                    //b41, b42 for 4th stage
{-15258, 32584}  };                 //b51, b52 for 5th stage
```

Tập tin này khai báo một bộ lọc có 5 tầng bậc 2. Giá trị của các phần tử của a và b thu được từ việc thiết kế bộ lọc ở trên. **Lưu ý: a và b trong chương trình này ứng với b và a trong ma trận sos của MATLAB.**

Chương trình trên được thực hiện có sử dụng ngắt. Khi có xung lấy mẫu (tần số ở đây là 8 KHz), trình phục vụ ngắt *c_int11( )* được gọi, đọc mẫu vào và thực hiện giải thuật xử lý mẫu để tính ngõ ra. Phần lặp của đoạn mã trong chương trình được thực hiện *stages* lần với mỗi giá trị của n. Đối với tầng đầu tiên, *x(n)* là mẫu mới nhận vào. Đối với các tầng tiếp theo, *x(n)* là ngõ ra của tầng trước đó. Các giá trị dly[i][0] và dly[i][1] tương ứng với các delay *u(n – 1)* và *u(n – 2)* ở tầng thứ i.

- **Tóm lại, các bước để thực hiện bộ lọc IIR lên kit như sau:**

1. Lấy các hệ số đáp ứng xung mỗi tầng bậc 2 sos của bộ lọc thiết kế ở định dạng 16 bit có dấu.
2. Mở CCS (nhớ mở nguồn của DSK trước khi mở CCS). Kiểm tra kết nối.
3. Mở tập tin project (đã được tạo sẵn) FIR.pjt trong C:\CCStudio_v3.1\myprojects\IIR.
4. Trong cửa sổ Project View, tab File View, mở rộng phần Include, mở tập tin coeficients.h.
5. Đặt các hệ số đáp ứng xung tương ứng với mỗi tầng bậc 2 của bộ lọc vừa thiết kế vào trong tập tin này. (Có thể copy và paste từ cửa sổ Array Editor trên). Điều chỉnh giá trị N cho đúng với chiều dài đáp ứng xung. Lưu ý rằng các giá trị của đáp ứng xung cách nhau bằng một dấu phẩy (,). Lưu tập tin sau khi sửa đổi.
6. Xác lập các tùy chọn phù hợp (xem phần hướng dẫn sử dụng trong tài liệu này) rồi tiến hành biên dịch chương trình. Sau khi dịch thành công, hãy nạp chương trình lên trên kit và chạy chương trình.

*(Hướng dẫn: nên sao chép thư mục IIR đã có thành một thư mục với tên khác và thực hiện trên thư mục mới này ứng với từng bộ lọc).*

❖ **Kiểm tra bộ lọc**

1. Mở nguồn của máy phát sóng. Tạo một tín hiệu vào hình sine từ máy phát sóng, lần lượt thay đổi tần số của tín hiệu vào từ 100Hz đến 4KHz (mỗi lần 100Hz), ghi nhận biên độ dạng sóng và biên độ phổ của tín hiệu ngõ ra từ đó xác định đặc tính của bộ lọc.



***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

2. Tạo một sóng vuông từ máy phát sóng, lần lượt thay đổi tần số của tín hiệu vào ghi nhận các thành phần tần số của ngõ ra. Giải thích tại sao có dạng phổ này?

| $f_i$ (Hz) | 100 | 200 | 600 | 1000 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.2.2. Bộ lọc IIR thông thấp

Tương tự như phần II.2.1, hãy thiết kế, thực hiện và kiểm tra một bộ lọc IIR thông thấp thuộc loại Chebyshev 2 với các thông số như sau:

- Bậc bộ lọc: 10
- Cạnh dải dải chắn: 1.6 KHz.
- Độ gợn dải chắn: 60 dB
- Tần số lấy mẫu: 8 kHz.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:



2. Gía trị các hệ số đáp ứng xung mỗi tầng bậc 2 của bộ lọc thực hiện trên kit DSP:

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

3. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:



***Chú ý:*** Đầu tiên nên phát tín hiệu sine ở tần số nằm giữa dải thông của bộ lọc để xác định biên độ tín hiệu ngõ ra Am mức vừa phải trên màn hình hiển thị. Sau đó, hiệu chỉnh tần số máy phát theo yêu cầu. Để bảo đảm ngõ ra không bị méo dạng khi ngõ vào quá lớn, nên kéo nút AMPLITUDE ra ngoài để giảm mức tối đa của biên độ ngõ vào.

4. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 200 | 350 | 600 | 800 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.2.3. Thực hiện bộ lọc IIR thông dải

Tương tự như phần II.2.1, hãy thiết kế, thực hiện và kiểm tra một bộ lọc IIR thông dải thuộc loại Chebyshev 2 với các thông số như sau:

- Bậc bộ lọc: 36
- Tần số cắt dải chắn: 1600 và 2400 Hz
- Độ gợn dải chắn 100 dB.
- Tần số lấy mẫu: 8 kHz.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:

| |
|---|
| |

2. Gía trị các hệ số đáp ứng xung mỗi tầng bậc 2 của bộ lọc thực hiện trên kit DSP:

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

3. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:

4. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 200 | 350 | 600 | 800 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

### 5.2.4. Thiết kế bộ lọc IIR multiband

Trong thí nghiệm này, chúng ta thiết kế, thực hiện và kiểm tra một bộ lọc IIR multiband có đáp ứng mong muốn như bộ lọc ở phần II.1.4.

Các bước thiết kế giống như ở phần II.1.4 với nội dung file multibandiir63.m dùng để thiết kế như sau:

```
%multibandiir63.m: Multiband IIR filter with 63 coefficients
f = [0 0.1 0.12 0.18 0.2 0.3 0.32 0.38 0.4 1];
m = [0 0 1 1 0 0 1 1 0 0];
n = 63;
[num, den] = yulewalk(n-1,f,m);

% frequency response with 256 points
[h w] = freqz(num,den,256);
% plot magnitude of the filter
plot(5000*f,m);
figure;
plot(w/pi,abs(h));
```

Trong đoạn chương trình trên, lưu ý:

- Lệnh **remez** được thay bằng lệnh **yulewalk** dùng để thiết kế bộ lọc IIR. Lệnh này trả về các hệ số của tử số và mẫu số của hàm truyền bộ lọc, được chứa tương ứng trong biến **num** và **den**.
- Lệnh freqz được sửa đổi để vẽ đáp ứng tần số của bộ lọc IIR đã thiết kế.

Các giá trị của **num** và **den** được sử dụng khi cần thực hiện bộ lọc này lên kit DSP.

1. Đáp ứng tần số của bộ lọc thiết kế dùng MATLAB:

2. Gía trị các hệ số đáp ứng xung mỗi tầng bậc 2 của bộ lọc thực hiện trên kit DSP:

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

3. Kiểm tra bộ lọc với ngõ vào tín hiệu sin:

4. Kiểm tra bộ lọc với ngõ vào tín hiệu xung vuông:

| $f_i$ (Hz) | 200 | 350 | 600 | 800 |
|---|---|---|---|---|
| $f_o$ (Hz) | | | | |
| Giải thích | | | | |

5. So sánh các kết quả với phần trước:

### ❖ Câu hỏi ôn tập

1. Cho biết cách kiểm tra và một số đặc tính cơ bản của bộ lọc tuyến tính bất biến.
2. Phân loại bộ lọc tuyến tính bất biến dựa vào đáp ứng xung và đáp ứng tần số.
3. Cho biết tên đầy đủ của các thuật ngữ viết tắt sau: FIR, IIR, LPF, HPF, BPF, BSF.
4. Cho biết cách kiểm tra và ý nghĩa các đặc tính nhân quả và ổn định của bộ lọc.
5. Cho biết cách xác định và ý nghĩa bậc của bộ lọc.
6. Trình bày tóm tắt các bước thiết kế một bộ lọc FIR/IIR bằng SPTool của MATLAB. Có mấy phương pháp thiết kế bộ lọc FIR/IIR trong SPTool? Hãy liệt kê.
7. Hãy phân biệt đặc điểm của đáp ứng tần số của các loại bộ lọc Butterworth, Chebyshev 1, Chebyshev 2 và Elliptic.
8. Thế nào là một bộ lọc multiband? Thử phát họa đáp ứng tần số của một bộ lọc multiband? Có thể dùng MATLAB để thiết kế một bộ lọc FIR/IIR multiband hay không?
9. Trình bày tóm tắt các bước thực hiện một bộ lọc FIR/IIR trên kit DSP.

4

# THỰC HIỆN FFT TRÊN KIT C6713 DSK

**Họ và tên SV báo cáo 1: …… MSSV: ……..…..**

**Họ và tên SV báo cáo 2: …… MSSV: ……..…….....**

**Họ và tên SV báo cáo 3: …… MSSV: ……..…….....**

**Họ và tên SV báo cáo 4: …… MSSV: ……..…….....**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: …….……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |
| | | | | |

## 1. MỤC ĐÍCH THÍ NGHIỆM

- Hiểu rõ giải thuật thực hiện FFT lên trên một kit DSP.
- Hệ thống lại các lý thuyết đã học.

## 2. THIẾT BỊ THÍ NGHIỆM

| **STT** | **Tên thiết bị** | **Số lượng** |
|---|---|---|
| 01 | Máy vi tính | 01 |
| 02 | Kit C6713 DSK | 01 |

| | | |
|---|---|---|
| 03 | Máy phát sóng | 01 |
| 04 | Bộ dây nối tín hiệu | 01 |

# 3. CƠ SỞ LÝ THUYẾT

Biến đổi Fourier nhanh (FFT) là một thuật toán cực kì hiệu quả để chuyển đổi một tín hiệu rời rạc miền thời gian sang miền tần số dựa trên biến đổi Fourier rời rạc (DFT). Phép biến đổi DFT phân tích một dãy các số thành các thành phần ở các tần số khác nhau. Nó được ứng dụng trong nhiều lĩnh vực khác nhau nhưng tính toán trực tiếp từ định nghĩa thường quá chậm trong thực tế. FFT là một cách để đạt được cùng kết quả như DFT nhưng nhanh hơn nhiều: tính DFT của $N$ điểm trực tiếp theo định nghĩa đòi hỏi $O(N^2)$ phép tính, trong khi FFT tính ra cùng kết quả đó trong $O(N \log N)$ phép tính.

Thuật toán FFT phổ biến nhất là thuật toán FFT Cooley-Tukey. Đây là một thuật toán chia để trị dùng đệ quy để chia bài toán tính DFT có kích thước $N=N_1N_2$, thành nhiều bài toán tính DFT nhỏ hơn có kích thước $N_1$ và $N_2$. Dạng phổ biến nhất của thuật toán Cooley-Tukey là chia biến đổi thành hai nửa kích thước $N/2$ ở mỗi bước (vì vậy chỉ dùng được cho kích thước là lũy thừa của 2, còn gọi là thuật toán cơ số 2), nhưng bất kì cách phân tích ra thừa số nào cũng đều có thể dùng được. Mặc dù ý tưởng cơ bản là đệ quy, khi lập trình, người ta thường sắp xếp lại thuật toán để tránh đệ quy.

FFT có thể thực hiện theo hai hướng: Giải thuật phân chia miền thời gian và giải thuật phân chia miền tần số.

### 3.1. DFT

DFT của một tín hiệu rời rạc x(nT) có chiều dài N được định nghĩa như sau:

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk} \qquad k = 0, 1, \ldots, N-1$$

trong đó

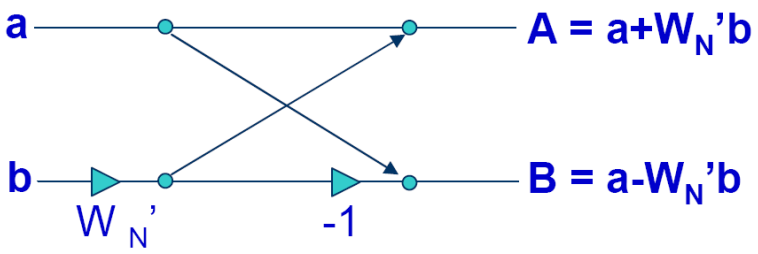
$$W = e^{-j2\pi/N}$$

IDFT được xác định bởi biểu thức sau:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W^{-nk} \qquad n = 0, 1, \ldots, N-1$$

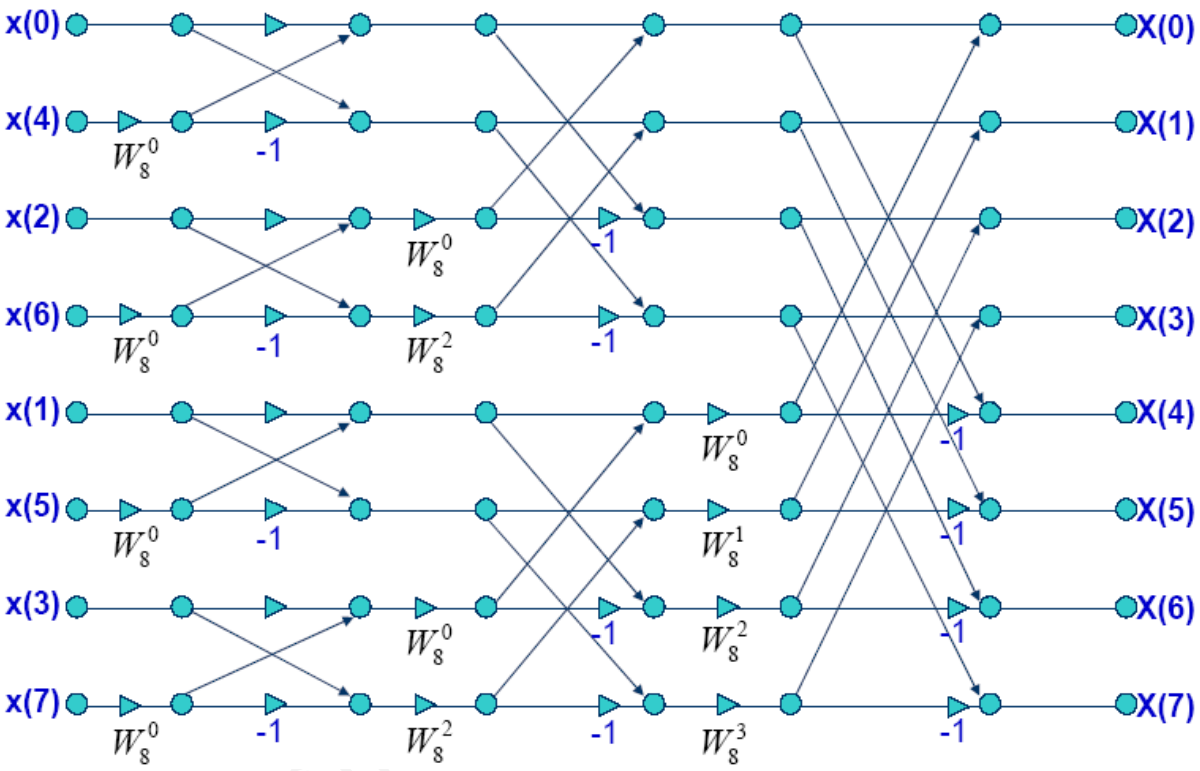
### 3.2. FFT

Giải thuật FFT cơ số 2 có đơn vị xử lý nhỏ nhất là FFT-2 điểm, thường được gọi là sơ đồ cánh bướm có dạng như sau:




**Hình 32. Sơ đồ cánh bướm FFT-2 điểm.**

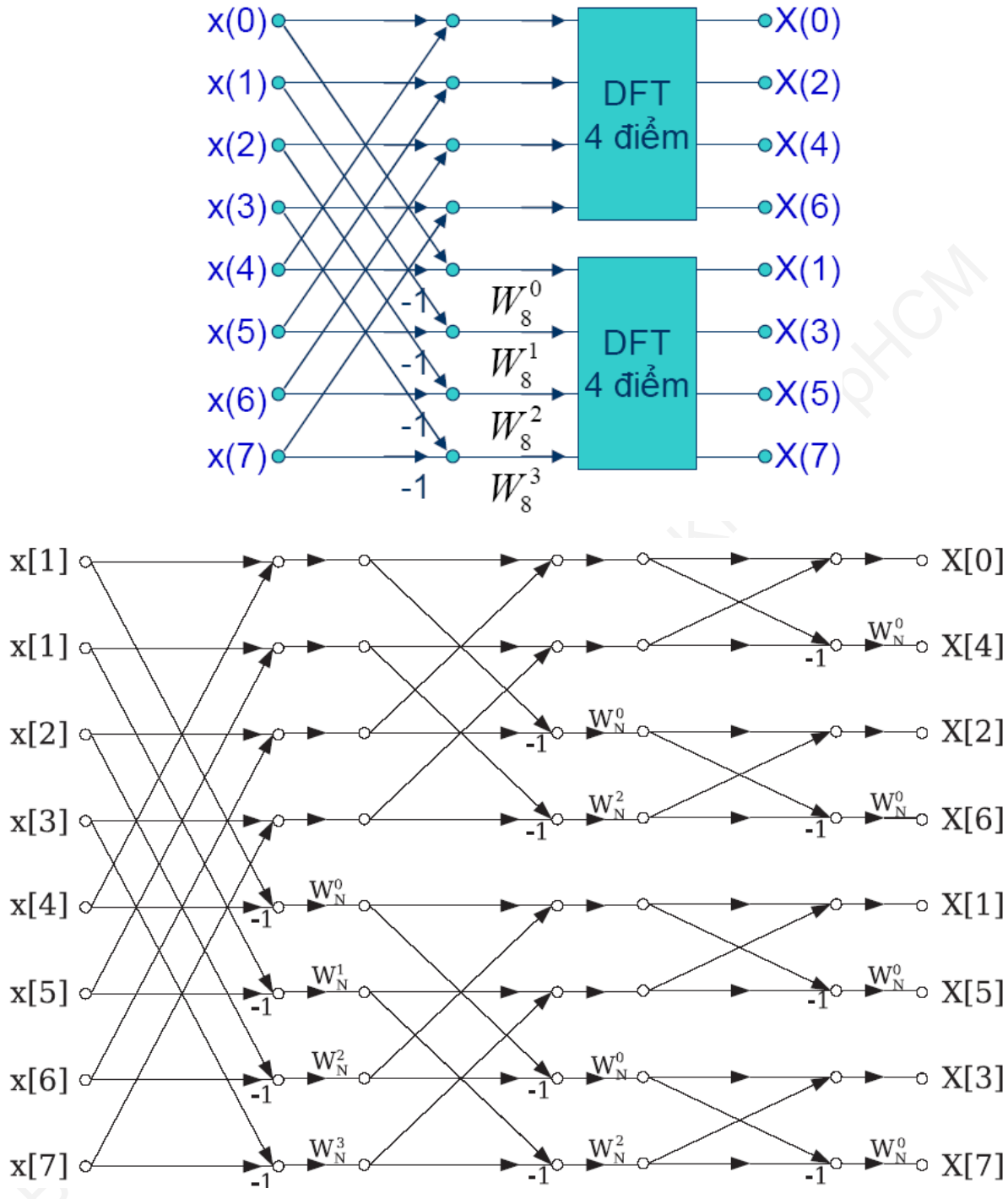
## 1. Giải thuật phân chia miền thời gian




| Bộ nhớ | Địa chỉ | Phân chia | Bộ nhớ | Địa chỉ | Phân chia | Bộ nhớ | Địa chỉ |
|---|---|---|---|---|---|---|---|
| x(0) | 000 | | x(0) | 000 | | x(0) | 000 |
| x(1) | 001 | | x(2) | 010 | | x(4) | 100 |
| x(2) | 010 | | x(4) | 100 | | x(2) | 010 |
| x(3) | 011 | | x(6) | 110 | | x(6) | 110 |
| x(4) | 100 | | x(1) | 001 | | x(1) | 001 |
| x(5) | 101 | | x(3) | 011 | | x(5) | 101 |
| x(6) | 110 | | x(5) | 101 | | x(3) | 011 |
| x(7) | 111 | | x(7) | 111 | | x(7) | 111 |

**Hình 33. Giải thuật FFT-8 điểm phân chia miền thời gian.**

**2. Giải thuật phân chia miền tần số**




**Hình 34. Giải thuật FFT-8 điểm phân chia miền tần số.**

## 4. CHUẨN BỊ LÝ THUYẾT THÍ NGHIỆM

Cho một tín hiệu rời rạc x(n) = [1; -2; -3; -4].

❖ **Câu hỏi chuẩn bị**

1. Viết biểu thức hàm truyền (biến đổi Z) và DTFT của tín hiệu x(n)? Vẽ phổ biên độ-tần số và pha-tần số của tín hiệu trên?

2. Tính DTF-4 điểm của tín hiệu x(n)? Vẽ biên độ và pha của DTF-4 điểm của tín hiệu trên? Nhận xét?

3. Tính DTF-8 điểm của tín hiệu x(n)? Vẽ biên độ và pha của DTF-8 điểm của tín hiệu trên? Nhận xét?

4. Vẽ sơ đồ và thực hiện tính FFT-8 điểm dùng giải thuật phân chia miền thời gian?

5. Vẽ sơ đồ và thực hiện tính FFT-8 điểm dùng giải thuật phân chia miền tần số?

6. Vẽ sơ đồ và thực hiện tính IFFT-8 điểm dùng giải thuật phân chia miền thời gian?

7. Vẽ sơ đồ và thực hiện tính IFFT-8 điểm dùng giải thuật phân chia miền tần số?

# 5. TIẾN TRÌNH THÍ NGHIỆM

Trong phần thí nghiệm này có 2 yêu cầu chính cần phải thực hiện:

1. Thực hiện FFT-128 điểm lên trên kit C6713 DSK: Sử dụng chương trình có sẵn để có thể dùng dao động ký quan sát phổ tần số của tín hiệu sin tạo ra từ máy phát sóng.
2. Thực hiện FFT-256 điểm lên trên kit C6713 DSK: Chỉnh sửa chương trình có sẵn để có thể dùng dao động ký quan sát phổ tần số của tín hiệu sin tạo ra từ máy phát sóng.

## 5.1. Thực hiện FFT-128 điểm

Thực hiện FFT-128 điểm của một tín hiệu sin thực lên kit DSP và quan sát phổ dùng dao động ký.

❖ **Thực hiện FFT-128 điểm trên kit DSP**

1. Mở CCS (nhớ mở nguồn của DSK trước khi mở CCS). Kiểm tra kết nối.
2. Mở tập tin project (đã được tạo sẵn) FIR.pjt trong C:\CCStudio_v3.1\myprojects\FFT.
3. Xác lập các tùy chọn phù hợp (xem phần hướng dẫn sử dụng trong tài liệu này) rồi tiến hành biên dịch chương trình. Sau khi dịch thành công, hãy nạp chương trình lên trên kit và chạy chương trình.

*(Hướng dẫn: nên sao chép thư mục FFT đã có thành một thư mục với tên khác và thực hiện trên thư mục mới này ứng với từng bộ lọc).*

FFT được thực hiện trên kit bằng chương trình sau (viết bằng ngôn ngữ C)

```
//fft128c.c

#include "DSK6713_AIC23.h"    //codec-DSK interface support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_LINE;

#include <math.h>
#include "fft.h"
#define PI 3.14159265358979
#define TRIGGER 32000
#define N 128
#include "hamm128.h"

short buffercount = 0;            //number of new input samples in iobuffer
short bufferfull = 0;                   //set by ISR to indicate iobuffer
full
COMPLEX A[N], B[N], C[N];
COMPLEX *input_ptr, *output_ptr, *process_ptr, *temp_ptr;
COMPLEX twiddle[N];
short outbuffer[N];


interrupt void c_int11(void)      //ISR
{
  output_left_sample((short)((output_ptr + buffercount)->real));
  outbuffer[buffercount] = -(short)((output_ptr + buffercount)->real);
  (input_ptr + buffercount)->real = (float)(input_left_sample());
  (input_ptr + buffercount++)->imag = 0.0;
```

```
  if (buffercount >= N)       //for overlap-add method iobuffer
  {                             // is half size of FFT used
    buffercount = 0;
    bufferfull = 1;
  }
}


main()
{
  int n;

  for (n=0 ; n<N ; n++)         //set up DFT twiddle factors
  {
    twiddle[n].real = cos(PI*n/N);
    twiddle[n].imag = -sin(PI*n/N);
  }
  input_ptr = A;
  output_ptr = B;
  process_ptr = C;
  comm_intr();                    //initialise DSK, codec, McBSP
  while(1)                       //frame processing loop
  {
    while(bufferfull==0);    //wait for new frame of input samples
    bufferfull = 0;

    temp_ptr = process_ptr; //rotate buffer/frame pointers
    process_ptr = input_ptr;
    input_ptr = output_ptr;
    output_ptr = temp_ptr;

    fft(process_ptr,N,twiddle); //process contents of buffer

    for (n=0 ; n<N ; n++)       // compute magnitude of frequency domain
representation
    {                           // and place in real part
      (process_ptr+n)->real = -sqrt((process_ptr+n)->real*(process_ptr+n)-
>real
                              + (process_ptr+n)->imag*(process_ptr+n)-
>imag)/16.0;
    }
    (process_ptr)->real = TRIGGER; // add oscilloscope trigger pluse
  }                                //end of while(1)
}                                  //end of main()
```

Trong chương trình này, các đỉnh xung âm (giá trị 32000) dùng để đánh dấu. Khoảng cách giữa 2 đỉnh xung âm liên tiếp tương ứng với tần số lấy mẫu. Khi đó, vị trí của đỉnh xung dương đầu tiên tương ứng với tần số của tín hiệu sin ngõ vào.

### ❖ Quan sát phổ tần số

1. Mở nguồn của máy phát sóng. Tạo một tín hiệu vào hình sine tần số 2KHz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

***Chú ý:*** *Để thuận tiện hơn trong quá trình quan sát dạng sóng, cần phải chỉnh trigger để tránh trôi tín hiệu hoặc có thể dùng chức năng chụp màn hình để ghi nhận kết quả. Điều chỉnh thang đo phù hợp để xuất hiện ít nhất 2 đỉnh xung âm.*

2. Tạo một tín hiệu vào hình sine tần số 1KHz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

3. Tạo một tín hiệu vào hình sine tần số 500Hz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

4. Tạo một tín hiệu vào hình xung vuông tần số 500Hz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

### 5.2. Thực hiện FFT-256 điểm

Tương tự như phần II.1., chỉnh sửa chương trình phù hợp để thực hiện FFT-256 điểm của một tín hiệu sin thực lên kit DSP và quan sát phổ dùng dao động ký.

1. Giá trị hàm cửa sổ Hamming chiều dài 256.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

2. Chương trình FFT-256 điểm.

3. Tạo một tín hiệu vào hình sine tần số 2KHz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

4. Tạo một tín hiệu vào hình sine tần số 1KHz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

5. Tạo một tín hiệu vào hình sine tần số 500Hz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

6. Tạo một tín hiệu vào hình xung vuông tần số 500Hz (biên độ vừa phải) từ máy phát sóng, ghi nhận dạng sóng của tín hiệu ngõ ra và giải thích.

### ❖ Câu hỏi ôn tập

1. Cho biết tên đầy đủ và mối liên hệ của các thuật ngữ viết tắt sau: DTFT, DFT, FFT và IFFT?
2. Vẽ sơ đồ thực hiện FFT(IFFT)-4/8/16 điểm dùng giải thuật phân chia miền thời gian?
3. Vẽ sơ đồ thực hiện FFT(IFFT)-4/8/16 điểm dùng giải thuật phân chia miền tần số?
4. Trình bày đặc tính tần số của tín hiệu thoại?
5. Trình bày tóm tắt các bước thực hiện giải thuật FFT trên kit DSP?

5

# ĐIỀU CHẾ PAM VÀ PWM

**Họ và tên SV báo cáo 1: …… MSSV: ……......**

**Họ và tên SV báo cáo 2: …… MSSV: ……..…….....**

**Họ và tên SV báo cáo 3: …… MSSV: ……..…….....**

**Họ và tên SV báo cáo 4: …… MSSV: ……..…….....**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: …….……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |

## 1. MỤC ĐÍCH THÍ NGHIỆM

Sử dụng kit DSK7613 thiết kế bộ điều chế PAM và PWM.

- Thực hiện điều PAM trên kit C6713 cho ứng dụng truyền dữ liệu.
- Thực hiện điều PWM trên kit C6713 cho ứng dụng điều khiển.

## 2. THIẾT BỊ SỬ DỤNG

| **STT** | **Tên thiết bị** | **Số lượng** |
|---|---|---|
| 01 | Máy vi tính | 01 |
| 02 | Kit C6713 DSK | 01 |
| 03 | Bộ dây nối tín hiệu | 01 |

# 3. CƠ SỞ LÝ THUYẾT

## 3.1. Giới thiệu

Điều chế là quá trình làm biến đổi một tín hiệu sóng bằng các phương pháp khác nhau nhằm sử dụng tín hiệu này vào một mục đích cụ thể. Thông thường, một dạng sóng hình sine cao tần được sử dụng làm sóng mang. Ba thông số của một sóng sine là biên độ, pha và tần số đều có thể làm cho biến đổi theo tín hiệu thông tin để tạo ra tín hiệu điều chế.

Thiết bị để thực hiện điều chế được gọi là bộ điều chế (modulator) và thiết bị thực hiện quá trình ngược lại gọi là bộ giải điều chế (demodulator). Thiết bị có thể thực hiện cả hai quá trình gọi là modem(modulator and demodulator).

Có hai dạng cơ bản là điều chế số và điều chế tương tự dựa trên việc biến đổi tín hiệu bằng hiệu số hoặc tín hiệu tương tự.

Mục đích của *điều chế số* là để truyền một chuỗi bit trên một kênh truyền tương tự bandpass, ví dụ như trên đường dây điện thoại (các bộ lọc giới hạn dải tần số từ 300 → 3400 Hz) hoặc trên một dải tần số radio. Trong khi đó, mục đích của *điều chế tương tự* là truyền một tín hiệu tương tự tần số thấp, ví dụ như tín hiệu âm tần, trên một kênh truyền tương tự bandpass. Ngoài ra còn có các điều chế khác với các chức năng cụ thể ví dụ như:

Mục đích của các phương pháp *điều chế số dải nền (baseband)*, còn gọi là *mã hóa đường truyền*, là truyền một chuỗi bit trên một kênh truyền thông thấp (lowpass).

Mục đích của các phương pháp *điều chế xung* là để truyền một tín hiệu tương tự băng hẹp (narrowband) hoặc trong một số trường hợp như là một cách thay đổi năng lượng trung bình của tín hiệu xung vuông.

## 3.2. Một số kỹ thuật điều chế cơ bản

### 3.2.1. Các kỹ thuật điều chế tương tự

Trong điều chế tương tự, quá trình điều chế được áp dụng liên tục theo tín hiệu thông tin tương tự.

Các kỹ thuật điều chế tương tự thông dụng gồm:

- Điều chế biên độ:
  - Double - sideband modulation (DSB)
    - AM modulation
    - Double – sideband suppressed-carrier (DSB – SC)
  - Single – sideband modulation (SSB)
  - Vestigial sideband modulation (VSB)
  - Quadratude amplitude modulation (QAM)
- Điều chế góc:
  - Frequency modulation (FM)
  - Phase modulation (PM)

### 3.2.2. Các kỹ thuật điều chế số

Trong điều chế số, một sóng mang tương tự được điều chế bởi một chuỗi bit của tín hiệu thông tin. Sự thay đổi của tín hiệu sóng mang được chọn từ một số hữu hạn các ký hiệu (symbol).

- Trong PSK, một tập hợp hữu hạn các pha được sử dụng.
- Trong FSK, một tập hợp hữu hạn các tần số được sử dụng.
- Trong ASK, một tập hợp hữu hạn các biên độ được sử dụng.
- Trong QAM, một tín hiệu cùng pha (tín hiệu I, ví dụ dạng sóng cosine) và một tín hiệu vuông pha (tín hiệu Q, ví dụ dạng sóng sine) được điều biên với một số lượng hữu hạn các mức biên độ (ASK). Tín hiệu thu được là kết hợp của PSK và ASK.

Các kỹ thuật điều chế số cơ bản nhất bao gồm:

- Phase – shift keying (PSK)
- Frequency – shift keying (FSK)
- Amplitude – shift keying (ASK) và dạng thường gặp của nó là On – off keying (OOK)
- Quadratude Amplitude Modulation (QAM): một kết hợp của PSK và ASK.
- Polar modulation: giống QAM, là kết hợp của PSK và ASK.
- Continuous phase modulation (CPM)
  - Minimum shift keying (MSK)
  - Gaussian minimum – shift keying (GMSK)
- Orthogonal frequency division multiplexing (OFDM) modulation
- Wavelet modulation
- Trellis coded modulation (TCM) hay còn gọi là trellis modulation

### 3.2.3. Điều chế số dải nền (Digital baseband modulation)

Thuật ngữ điều chế số dải nền đồng nghĩa với mã hóa đường truyền (line coding), là tập hợp các phương pháp để truyền một chuỗi bit trên một kênh truyền tương tự thông thấp, sử dụng một số rời rạc các mức tín hiệu, bằng cách điều chế một chuỗi xung (một sóng vuông). Các ví dụ thường gặp là unipolar, non-return-to-zero (NRZ), Manchester và AMI (alternate mark inversion).

### 3.2.4. Các phương pháp điều chế xung

Các phương pháp điều chế xung nhằm truyền một tín hiệu tương tự băng hẹp trên một kênh truyền thông thấp như một tín hiệu được lượng tử hai mức, bằng cách điều chế một chuỗi xung.

Một vài dạng điều chế xung cũng cho phép tín hiệu tương tự băng hẹp được truyền như một tín hiệu số với một tốc độ bit cố định,và trong một vài trường hợp được xem như các kỹ thuật biến đổi A/D.

Các phương pháp này bao gồm:

- Pulse code modulation (PCM): điều chế xung mã
- Pulse – width modulation (PWM): điều chế độ rộng xung
- Pulse – amplitude modulation (PAM): điều chế biên độ xung
- Pulse – position modulation (PPM): điều chế vị trí xung
- Pulse – density modulation (PDM): điều chế mật độ xung
- Sigma – delta modulation (DM)
- Adaptive delta modulation (ADM)

Trải phổ trực tiếp (DSSS – direct-sequence spread spectrum) dựa trên điều chế PAM.

*Trong bài thí nghiệm này, hai phương pháp điều chế sẽ được thực hiện là điều chế biên độ xung (PAM) và điều chế độ rộng xung (PWM).*

### 3.2.5. Điều chế biên độ xung (PAM)

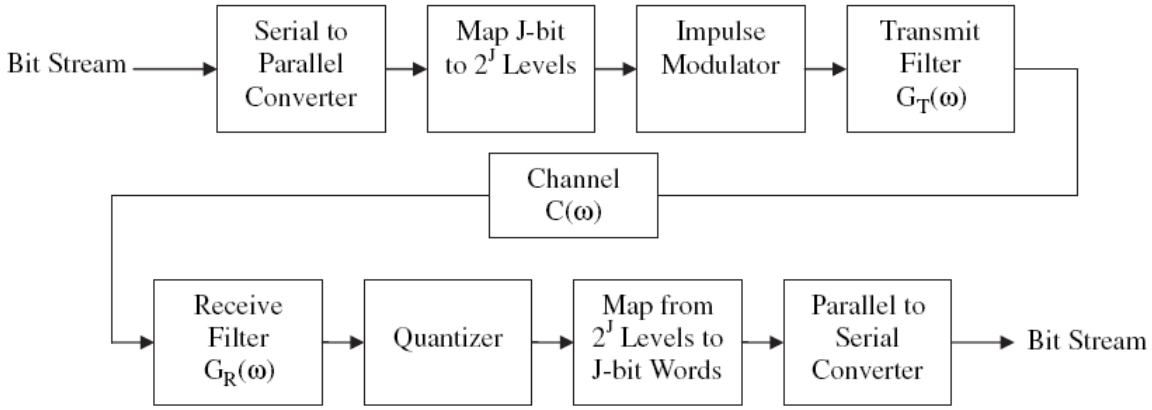
Điều chế biên độ xung (PAM) là một dạng điều chế tín hiệu trong đó thông tin được mã hóa trong biên độ của một chuỗi xung. Ví dụ: một bộ điều chế 2 bit (PAM – 4) sẽ lấy 2 bit một và ánh xạ biên độ tín hiệu thành một trong bốn mức, ví dụ như -3V, -1V, 1V và 3V.

Giải điều chế được thực hiện bằng cách đọc mức biên độ của sóng mang tại mỗi chu kỳ ký hiệu.

Hiện nay, PAM ít được sử dụng và hầu như được thay thế bởi PCM và PPM. Tất cả các modem điện thoại nhanh hơn 300 bps sử dụng kỹ thuật QAM.

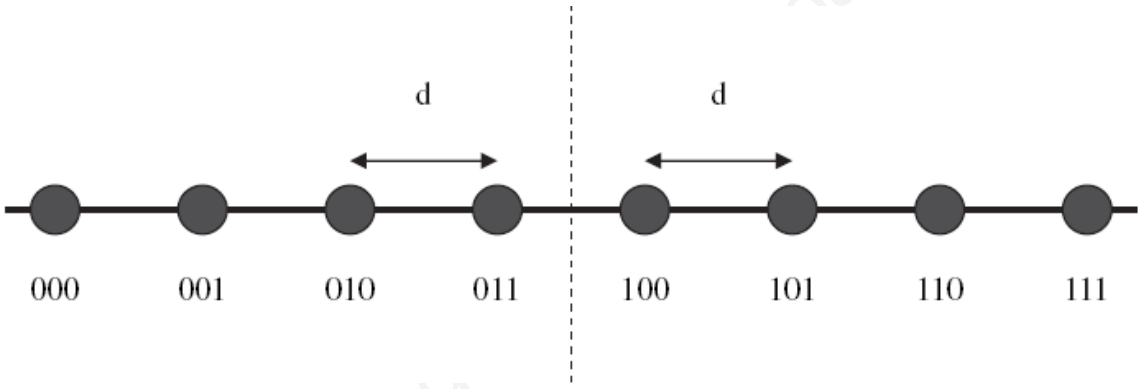
Tuy nhiên, chuẩn giao tiếp Ethernet vẫn sử dụng PAM. Ví dụ, 100BASE-T2 Ethernet (ở tốc độ 100Mb/s) sử dụng điều chế PAM 5 mức chạy ở tốc độ 25 megapulses/s trên 2 cặp dây dẫn. Một kỹ thuật đặc biệt được sử dụng để giảm nhiễu liên ký tự giữa các cặp không dây. Sau đó, 1000BASE-T nâng lên sử dụng 4 cặp dây dẫn ở tốc độ 125 megapulses/s để đạt tốc độ dữ liệu 1000 Mb/s, vẫn sử dụng PAM 5 mức cho mỗi cặp dây.

Dưới đây là sơ đồ khối của một hệ thống PAM đơn giản (bỏ qua bộ cân bằng thích nghi và bộ phục hồi xung clock).




**Hình 35. Hệ thống PAM**

Chuỗi bit vào được xử lý thành các từ dài J bits. J bits này được ánh xạ thành một trong $2^J$ mức. Ví dụ, khi J = 3 thì sẽ có 8 mức. Các mức này các đều nhau trên giản đồ constellation và đối xứng quanh mức zero như trên hình sau:

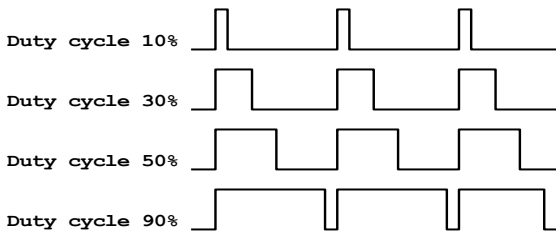



**Hình 36. Giản đồ constellation của PAM 8 mức**

Tám điểm trên giản đồ này biểu diễn cho 8 mức với mỗi mức được biểu diễn bằng một chuỗi 3 bits.

### 3.3. Điều chế độ rộng xung (PWM)

Điều chế độ rộng xung (PWM), là một kỹ thuật điều chế kiểm soát chiều rộng của của một xung điện (trong thời gian), trong chu kỳ của xung, dựa trên thông tin tín hiệu điều biến. Mặc dù kỹ thuật điều chế này có thể được sử dụng để mã hóa thông tin cho truyền dữ liệu,nhưng sử dụng chính của nó là kiểm soát năng lượng điện cung cấp cho các thiết bị điện, đặc biệt là động cơ. Ngoài ra, PWM là một trong hai cách chính được sử dụng trong bộ sạc pin năng lượng mặt trời quang điện, cách kia là MPPT(Maximum power point tracking) không thuộc phạm vi tài liệu này.

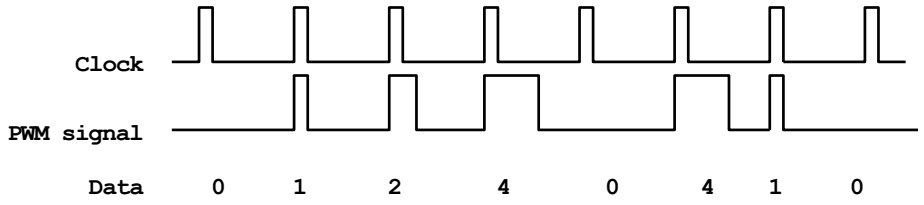



**Hình 37. Dạng sóng trong điều chế PWM**

**Trong lĩnh vực truyền thông**

PWM là một hình thức điều chế tín hiệu mà độ rộng của các xung tương ứng với giá trị dữ liệu cụ thể được mã hóa ở một đầu và giải mã ở đầu kia.

Xung có độ rộng khác nhau ứng với các thông tin riêng của mình sẽ được gửi đi thông qua sóng mang đã được điều chế PWM.




**Hình 38. Điều chế PWM trong truyền thông**

**Trong lĩnh vực điều khiển**

Nếu cấp điện liên tục cho tải là trạng thái ON.

Nếu không cấp điệm cho tải là trạng thái OFF.

Nếu xen kẻ on/off như một sóng vuông với tần số không đổi và độ rộng xung thay đổi ta sẽ thay đổi được năng lương cấp cho tải.

Giá trị trung bình của năng lượng cấp cho tải tương ứng với điện áp trung bình trên tải được kiểm soát bằng PWM.

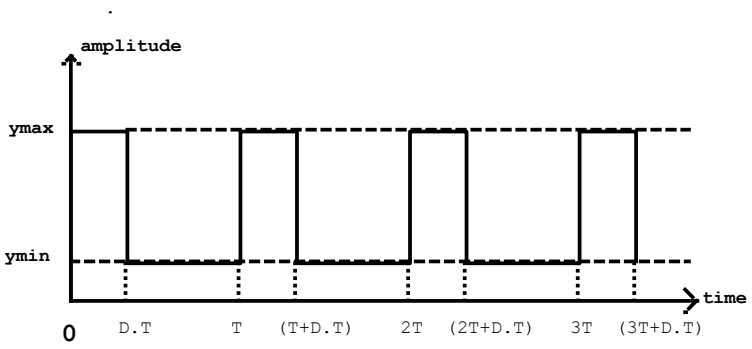
Các tần số chuyển mạch PWM có thể cao hơn nhiều so với những gì sẽ ảnh hưởng đến tải (các thiết bị có sử dụng điện), để để cho dang sóng trên tải càng mịn càng tốt(độ gợn nhỏ nhất). Ví dụ chọn tần số PWM trong dimmer(bộ thay đổi độ sáng)của bóng đèn đốt tim khoảng 100hz là đủ, còn trong điều khiển động cơ DC 3000vòng/phút tần số có thể lên đến vài chục Kz.

Ưu điểm chính của PWM là tổn thất điện năng trong các thiết bị chuyển mạch là rất thấp. Khi bộ chuyển mạch tắt thực tế là không có dòng qua bộ chuyển mạch và khi nó được bật năng lượng được chuyển giao cho tải, hầu như không có điện áp rơi trên các switch(bộ chuyển mạch). Tổn thất điện năngtrong cả hai trường hợp gần bằng không.

### 3.4. Lý thuyết cơ bản

Điều chế độ rộng xung sử dụng một sóng xung hình chữ nhật có chiều rộng xung được điều chế dẫn đến sự thay đổi của giá trị trung bình của các dạng sóng. Nếu chúng ta xem xét một dạng sóng xung f (t), với thời gian T, có giá trị thấp $y_{min}$, một giá trị cao $y_{max}$ và một chu kỳ nhiệm vụ D(duty cycle) (xem hình 39), giá trị trung bình của các dạng sóng được cho bởi:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)dt \qquad (1)$$




**Hình 39. Cơ sở lý thuyết PWM**

Khi f (t) là sóng xung(pulse wave), giá trị của nó là: $y_{max}$ cho 0 <t <D.T và $y_{min}$ cho D.T <t <T. Khi đó $\bar{y}$ trở thành

$$\bar{y} = \frac{1}{T}\left( \int_0^{DT} y\max dt + \int_{DT}^{T} y\min dt \right) = \frac{D.T.y\max + T(1-D)y\min}{T}.$$

Cuối cùng ta được: $\bar{y}$ = D. $y_{max}$+(1-D)$y_{min}$ . (2)

Từ (2 ) ta thấy với ymin=0 thì y trung bình phụ thuộc vào giá trị của D.

# 4. CHUẨN BỊ LÝ THUYẾT THÍ NGHIỆM

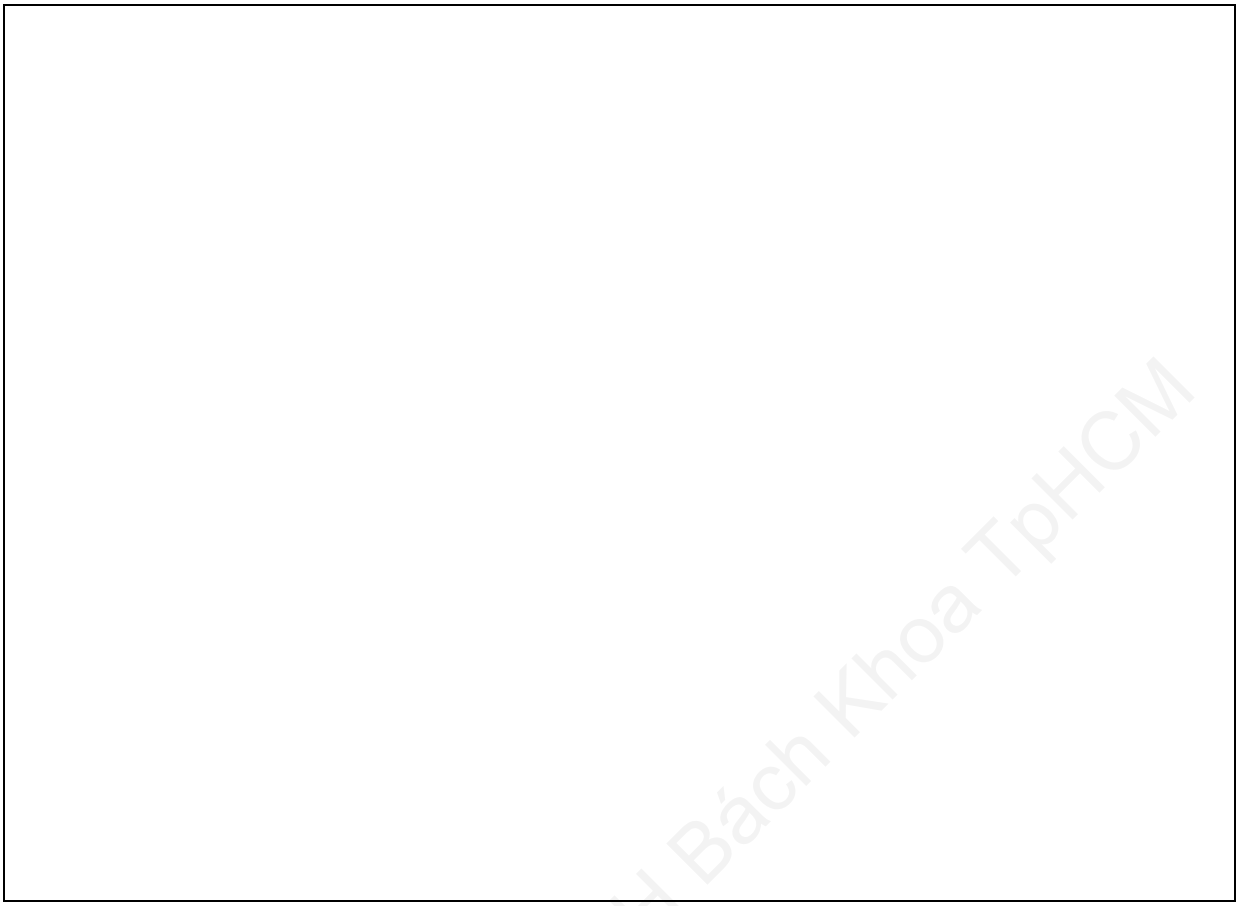
## 4.1. Điều chế PAM

1. Phân biệt ngắn gọn các phương pháp điều chế xung: PAM, PCM, PPM, PWM.

2. Hãy vẽ hình minh họa quá trình điều chế PAM một tín hiệu hình sine và giải thích?

3. Hãy tìm một số ứng dụng của điều chế PAM.

4. Hãy vẽ giản đồ constellation ứng với điều chế PAM 16 mức.



5. Đọc qua phần tiến hành thí nghiệm,tóm tắt các bước  sẽ làm để thực hiện các phần thí nghiệm PAM.

### 4.2. Điều chế PWM

1. Hãy vẽ hình minh họa sơ đồ khối quá trình điều chế PWM và giải thích?

2. Hãy tìm một số ứng dụng của điều chế PWM.

3. Đọc qua phần tiến hành thí nghiệm,tóm tắt các bước  sẽ làm để thực hiện các phần thí nghiệm PWM.



# 5. TIẾN TRÌNH THÍ NGHIỆM

## 5.1. Thực hiện mạch điều chế PAM trên kit C6713 DSK

Trong thí nghiệm này, mạch điều chế PAM 4 mức, 8 mức và 16 mức sẽ được lần lượt thực hiện trên kit DSP. Trong đó, mạch điều chế 4 mức đã được thực hiện sẵn trong một project để giúp hiểu rõ về giải thuật thực hiện. Sau đó, sinh viên sẽ tự thực hiện mạch điều chế PAM 8 mức và 16 mức.

### *Giải thuật điều chế*

Mỗi mẫu vào để điều chế gồm 16 bits. Tùy vào loại điều chế PAM, một mặt nạ (mask) thích hợp được sử dụng.

Hãy xem xét trường hợp PAM 16 mức (16-PAM). Trong trường hợp này, để biểu diễn 16 mức cần 4 bits, do đó mỗi ký hiệu sẽ có chiều dài 4 bits. Để đạt được tốc độ ký hiệu mong muốn, mẫu vào được chia làm các đoạn dài 4 bits. Như vậy, mỗi mẫu vào sẽ gồm 4 đoạn. Việc xử lý mẫu vào bao gồm việc áp dụng mặt nạ và dịch. Khối ký hiệu đầu tiên nhận được bằng cách dùng mẫu vào AND với mặt nạ 0x000F để lấy 4 bit LSB (0x000F là biểu diễn trong hệ thập lục phân của 0000 0000 0000 1111, khi AND một mẫu với mặt nạ này, chỉ 4 bit cuối của mẫu được giữ lại, tạo thành 1 ký hiệu). Khối ký hiệu thứ hai nhận được bằng cách dịch mẫu ban đầu sang phải 4 bit và áp dụng lại mặt nạ. Các bước này được lặp lại cho đến khi hết chiều dài của mẫu vào và tạo ra 4 ký hiệu.

Do mỗi ký hiệu có chiều dài 4 bit nên sẽ có tất cả 16 ký hiệu. 16 ký hiệu này được ánh xạ thành 16 mức điện áp cách đều nhau dựa theo một bảng tra như sau:

**Bảng 1. Bảng tra PAM 16 mức**

| Khối ký hiệu | Mức điện áp (biểu diễn dạng số Hex) |
|---|---|
| **0000** | 0x7FFF |
| **0001** | 0x6EEE |
| **0010** | 0x5DDD |
| **0011** | 0x4CCC |
| **0100** | 0x3BBB |
| **0101** | 0x2AAA |
| **0110** | 0x1999 |
| **0111** | 0x0888 |
| **1000** | - 0x0889 |
| **1001** | - 0x199A |
| **1010** | - 0x2AAB |
| **1011** | - 0x3BBC |
| **1100** | - 0x4CCD |
| **1101** | - 0x5DDE |
| **1110** | - 0x6EEF |
| **1111** | - 0x8000 |

Ví dụ, mẫu vào là 0xA52E (10100101 00101110). Khi đó, 1110 (sau khi lọc mặt nạ lấy 4 bit LSB) được ánh xạ thành mức -0x6EEF. Mỗi ký hiệu gồm 4 bit được ánh xạ lên 16 mức cách đều nhau từ -0x8000 đến 0x7FFF. Khoảng cách giữa các mức được chọn là 0x111 để có khoảng cách đều nhau.

Mức điện áp được chọn sau đó sẽ được truyền đi như một sóng vuông. Chu kỳ của sóng vuông đạt được bằng cách xuất cùng một mức điện áp đó nhiều lần (ví dụ 12 lần) để có thể đạt được 1 dạng sóng vuông tốt ở ngõ phát (sau khi qua bộ D/A).

Cùng một cách thực hiện của bộ phát được áp dụng cho PAM 4 mức (4-PAM) và 8 mức (8-PAM) với sự khác nhau về mặt nạ, dịch và bảng tra. Đối với 8-PAM, bit có trọng số thấp nhất (LSB) của mẫu vào được loại bỏ để số bit còn lại (15) là một bội số của 3. Điều này gây tác động không đáng kể lên dạng sóng điều chế và dạng sóng khôi phục lại.

Các bảng tra đối với bộ điều chế 4 – PAM và 8 – PAM lần lượt như sau:

**Bảng 2. Bảng tra PAM 4 mức**

| Khối ký hiệu | Mức điện áp (dạng Hex) |
|---|---|
| **00** | 0x7FFF |
| **01** | 0x2AAA |
| **10** | - 0x2AAB |
| **11** | - 0x8000 |

**Bảng 3. Bảng tra PAM 8 mức**

| Khối ký hiệu | Mức điện áp (dạng số Hex) |
|---|---|
| **000** | 0x7FFF |
| **001** | 0x5B6D |
| **010** | 0x36DB |
| **011** | 0x1249 |
| **100** | - 0x1249 |
| **101** | - 0x36DB |
| **110** | - 0x5B6D |
| **111** | - 0x7FFF |

***Chương trình thực hiện***

Hãy thực hiện theo các bước sau:

1. Cấp nguồn điện cho kit và chạy chương trình Code Compose Studio trên máy tính.
2. Mở project PAM4 ở folder C:\CCStudio_v3.1\MyProjects\PAM4\.
3. Trong cửa sổ File View, double click lên tập tin PAM4.c để mở nó ra. Đây là tập tin chứa mã nguồn chính của chương trình. Nội dung của tập tin này như sau:

```
// PAM
#include "DSK6713_aic23.h"
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;

#include <math.h>
//Initialization:
int i_PAM;
int j_PAM;
int k;
int masked_value, output;
int data_4PAM[4] = {0x7FFF, 0x2AAA, -0x2AAB, -0x8000};

int out_buffer[256];
int i=0;

interrupt void c_int11()   //interrupt service routine
{
     int sample_data;

     if (i_PAM==96)
     {
               sample_data = input_sample();        //inputs data
     i_PAM=0;
               j_PAM=0;
  }
  masked_value = sample_data & 0x0003;
  output = data_4PAM[masked_value];
     output_sample(output);
  out_buffer[i++] = output;
  if (i==256)
     i = 0;
  j_PAM++;                          //repeated output counter
```

```
    if (j_PAM==12)
    {
     j_PAM=0;
     sample_data = sample_data >> 2;
    }
  i_PAM++;

    return;
}

void main()
{
    i_PAM=0;
    comm_intr();       //init DSK, codec, McBSP
    while(1);                    //infinite loop
}
```

Trong chương trình này, hàm **main( )** đặt giá trị biến i_PAM = 0 và khởi động kit. Sau đó sẽ thực hiện một vòng lặp vô tận với lệnh **while (1)**.

Khi có tín hiệu xung lấy mẫu (tần số 8KHz), ngắt 11 xảy ra và trình phục vụ ngắt **c_int11( )** được gọi. Trong trình phục vụ ngắt này, dữ liệu vào được đọc vào biến **sample_data** (bằng lệnh **sample_data = input_sample( )**). Mẫu dữ liệu này biểu diễn ở dạng số nguyên 16 bit có dấu.

Do ở đây thực hiện điều chế PAM 4 mức, mà để biểu diễn được 4 mức thì cần 2 bit. Do đó, mẫu dữ liệu vào sẽ được chia ra làm 8 ký hiệu (symbol), mỗi ký hiệu 2 bit. Để thực hiện việc chia này, mẫu dữ liệu được AND với mặt nạ 00000000 00000011 (tức là 0x0003) để lấy 2 bit LSB của mẫu dữ liệu. 2 bit này được ánh xạ thành 1 trong 4 mức điện áp bằng cách sử dụng bảng data_4PAM. Bảng này có 4 giá trị ứng với 4 mức điện áp (0x7FFF, 0x2AAA, -0x2AAB, -0x8000) được khai báo ở đầu chương trình. Mỗi symbol sẽ cho ra 1 mức điện áp. Để cho dạng sóng xuất ra sau khi qua bộ A/D có dạng sóng vuông, mỗi mức sẽ được xuất ra 12 lần trước khi chuyển sang mức ứng với symbol kế tiếp. Biến đếm j_PAM để đếm số lần xuất ra cho đủ 12 lần. Như vậy, ứng với mỗi mẫu vào 16 bit, ta có 8 symbol. Mỗi symbol sẽ xuất ra 12 lần, như vậy số lần xuất ra ứng với mỗi mẫu là 8 x 12 = 96 được thể hiện bởi biến đếm i_PAM.

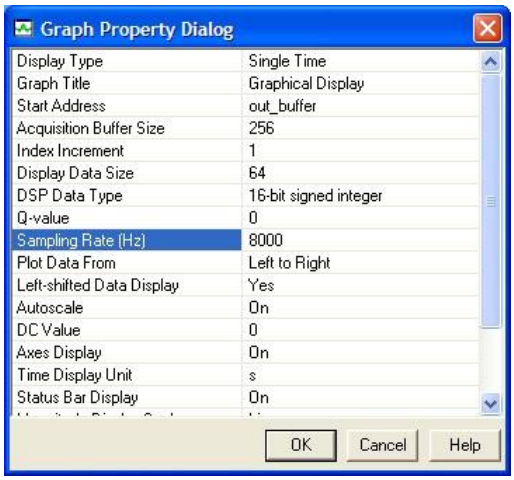
**Ví dụ:** Mẫu vào có giá trị 0x001B (tức là 00000000 00011011). Mẫu này đầu tiên được chứa vào biến sample_data.

- Đầu tiên: sample_data = 00000000 00011011
  - + Sample_data AND 0x0003 = 00000000 00000011 (= 3).
  - + data_4PAM(3) = - 0x8000 → mức điện áp – 0x8000 được xuất ra 12 lần

- Tiếp theo: sample_data dịch phải 2 bit → 00000000 00000110
  + Sample_data AND 0x0003 = 00000000 00000010 ( = 2)
  + data_4PAM(2) = - 0x2AAB → mức điện áp -0x2AAB được xuất ra 12 lần
- Quá trình tiếp tục cho đến khi hết 16 bit của mẫu vào, tức là dịch 8 lần, mỗi lần xuất 12 lần → số lần xuất tổng cộng i_PAM = 8 x 12 = 96 lần. Sau đó sẽ đọc vào mẫu kế tiếp và thực hiện điều chế.

Ở đây, một bộ đệm nội **out_buffer** có kích thước 256 mẫu được dùng để lưu lại giá trị xuất ra nhằm phục vụ cho việc vẽ tín hiệu xuất ra bằng công cụ Plot trong CCS (xem phần hướng dẫn vẽ đồ thị trên CCS).

1. Chọn Project → Rebuild All để biên dịch chương trình.
2. Sau khi biên dịch thành công, nạp chương trình lên kit (File → Load Program) và chạy thử chương trình.
3. Mở máy phát sóng lên và quan sát dạng sóng xuất ra. Ở đây hãy quan sát theo 2 cách: (1) sử dụng Code Compose Studio để vẽ các giá trị của bộ đệm nội **out_buffer.** (2) quan sát tín hiệu xuất ra ở ngõ ra Line Out của kit bằng Oscilloscope.
4. Nếu sử dụng máy phát sóng, dạng sóng thay đổi liên tục nên kết quả ra khó kiểm chứng. Hãy sửa câu lệnh **sample_data = input_sample( )** thành **sample_data = 0x????** (một giá trị bất kỳ 16 bit dạng số hex) rồi biên dịch, nạp và chạy lại chương trình.Vẽ lại dạng sóng xuất ra trong CCS. Lúc này, mẫu vào là giá trị cố định do nhập vào và dạng sóng xuất ra sẽ dễ dàng kiểm chứng hơn.
5. Chọn View → Graph → Time/Frequency. Thay đổi các tùy chọn trong cửa sổ Graph Property Dialog như sau để vẽ trong miền thời gian. Địa chỉ bắt đầu của bộ đệm chính là tên mảng *out_buffer* được nhập vào Start Address. Các tùy chọn khác có thể để như mặc định.




**Hình 40. Các tùy chọn để vẽ trong miền thời gian**

6. Hãy thử một vài giá trị nhập vào và vẽ dạng sóng xuất ra.

**Bài tập**

*Hãy chép toàn bộ Folder PAM4 sang một Folder mới và đặt tên tùy ý. Thực hiện các bài tập sau trên folder mới để không làm ảnh hưởng đến phần đã làm.*

1. Chỉnh sửa lại chương trình trên để thực hiện điều chế PAM 8 mức.

   *Gợi ý:*

   - *Mỗi symbol gồm 3bits. Do mẫu vào có 16 bits, cần bỏ 1 bits LSB (bằng cách dịch phải sample_data 1 bit) để còn lại 15 bits là bội số của 3 và mỗi mẫu sẽ tạo ra 5 symbols.*
   - *Mặt nạ để lấy 3 bit LSB là 00000000 00000111 (0x0007)*
   - *Sử dụng bảng tra ứng với 8-PAM để khai báo các mức áp ra tương ứng.*

*Hãy viết lại đoạn chương trình của file.c đã sửa vào đây. Giải thích và vẽ một vài dạng sóng điều chế thu được để kiểm chứng.*

- Chỉnh sửa lại chương trình trên để thực hiện điều chế PAM 16 mức.

*Hãy viết lại đoạn chương trình của file.c đã sửa vào đây. Giải thích và vẽ một vài dạng sóng điều chế thu được để kiểm chứng.*



## 5.2. Thực hiện mạch điều chế PWM trên kit C6713 DSK

***Giải thuật điều chế***

Trong Kit C6713DSK có phần khai báo tần số lấy mẫu.

```
#include "DSK6713_aic23.h"

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;
```

Mỗi lần xảy ra ngắt tại tần số lấy mẫu trình phục vụ ngắt **c_int11( )** được gọi.Toàn bộ chương trình tạo tín hiệu PWM được xử lý trong hàm này.

Giải thuật đơn giản nhất để thực hiện là khai báo một biến có tên là **duty_cycle** chứa giá trị D(duty cycle). Biến này có thể thay đổi tùy theo yêu cầu. Ví dụ như 25,40,70.

Hai biến đếm lần lượt là i và j dùng để đếm số lần vào ngắt thực hiện việc xuất giá trị ở mức cao (**H**igh) hoặc xuất ở mức thấp(**L**ow). Ở đây mức thấp tương ứng ngỏ ra 0V.

Việc xuất tín hiệu điện áp ra ngỏ ra audio jack nhờ vào hàm output_sample().

Mỗi lần vào ngắt ta sẽ xem giá trị biến đếm i > D chưa? Nếu chưa thì vẫn tiếp tục xuất mức cao và tăng biến đếm i,j và thoát. Nếu đã >D thì xuất mức thấp.

Bên cạnh đó biến j cũng được xem xét, nếu đạt được 100 thì i,j được xóa về 0 và quá trình lập lại từ đầu.

***Chương trình thực hiện***

Hãy thực hiện theo các bước sau:

1. Cấp nguồn điện cho kit và chạy chương trình Code Compose Studio trên máy tính.
2. Mở project PWM ở folder C:\CCStudio_v3.1\MyProjects\PWM\.
3. Trong cửa sổ File View, double click lên tập tin PWM.c để mở nó ra. Đây là tập tin chứa mã nguồn chính của chương trình. Nội dung của tập tin này như sau:

```
// PWM

#include "DSK6713_aic23.h"
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;

#include <math.h>
//Initialization:
int i_PWM;                      // counter  in high state.
int j_PWM;                      // counter  max to 100.

int out_buffer[1024];  // buffer for view graph.
int i=0;
int duty_cycle;



interrupt void c_int11()   //interrupt service routine
{
int output;
```

```
    if(j_PWM==100){                    // when reach 100 wil reset to zero.
            i_PWM=0;
    j_PWM=0;
            output=0x0000;   // set output value to zero.
            output_sample(output);// calll funtion output to line out audio jack.
    }
    else
    {
            if(i_PWM<= duty_cycle)// compare with variable //"time_duration" return from
GEL(General Extension Language)
// find in "help" keyword"slider param_definition".
            {
                    output=0x7FFF;  // set output value to max.
                    output_sample(output);     // calll funtion output.
                    i_PWM++;
                    j_PWM++;
            }
            else
            {
                    output=0x0000;                    // set output value to zero.
                    output_sample(output);     // calll funtion output
                    i_PWM++;
                    j_PWM++;
            }

    }

  out_buffer[i++] = output;                // record in buffer for view graph.
  if (i==1024)
    i = 0;

    return;
}

void main()
{
    time_duration=10;
    i_PWM=0;                    //init i_PWM.
    j_PWM=0;                    //init j_PWM.
    comm_intr();       //init DSK, codec, McBSP.
```

```
    while(1);                   //infinite loop.
}
```

4. Chọn Project → Rebuild All để biên dịch chương trình.
5. Sau khi biên dịch thành công, nạp chương trình lên kit (File → Load Program) chạy
6. Chọn File →Load Gel và chọn file PWM_vari.gel.
7. Sau đó chọn GEL → *PWM* → *PWM*. Một cửa sổ mới xuất hiện với thanh trượt ở trên đó. Khi thanh trượt này ở vị trí số 10, giá trị ngõ ra tương ứng độ rộng xung 10%. Khi thanh trượt ở vị trí cao nhất, giá trị ngõ ra tương ứng độ rộng xung 90%.
8. Chọn Debug→run.
9. Dùng công cụ **dsptool** (Oscilloscope)quan sát dạng sóng xuất ra mỗi lần thay đổi vị trí cần gạt. Hoặc quan sát theo cách 2: sử dụng Code Compose Studio để vẽ các giá trị của bộ đệm nội **out_buffer.** (Tương tự phần PAM ở trên.).

**Hãy trả lời các câu hỏi sau:**

1. Có thể sửa chương trình PWM.c chỉ dùng một biến i hoặc j được không, vì sao?

2. Làm thế nào để thay đổi tần số xung PWM?

3. Quan sát bằng công cụ dsptool và view graph có gì khác nhau? Vì sao?

4. Với một chương trình đã hoạt động trên kit DSP như trên làm thế nào để ứng dụng thực tế điều khiển độ sáng của một bóng đèn, hoặc tốc độ của động cơ DC.

6

# PHÂN TÍCH TÍN HIỆU TIẾNG NÓI VÀ ỨNG DỤNG XỬ LÝ TIẾNG NÓI

**Họ và tên SV báo cáo 1: ……. MSSV: …….....**

**Họ và tên SV báo cáo 2: ……. MSSV: …….....**

**Họ và tên SV báo cáo 3: ……. MSSV: …….....**

**Họ và tên SV báo cáo 4: ……. MSSV: …….....**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: …….……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |

## 1. MỤC ĐÍCH THÍ NGHIỆM

- Hiểu được bản chất quá trình phát âm tiếng nói của con người.
- Nắm rõ một số đặc điểm quan trọng của tín hiệu tiếng nói như băng thông, sự phân bố năng lượng, tốc độ lấy mẫu tối thiểu.
- Biết cách phân đoạn tiếng nói và tính đặc trưng trên từng đoạn như Năng lượng, tần số cơ bản.
- Biết cách thực hiện một số ứng dụng đơn giản của hệ thống xử lý tiếng nói như: triệt nhiễu, phát hiện và tách tín hiệu tiếng nói ra khỏi đoạn tín hiệu ghi âm được, nhận dạng giới tính người phát âm dựa vào tiếng nói.

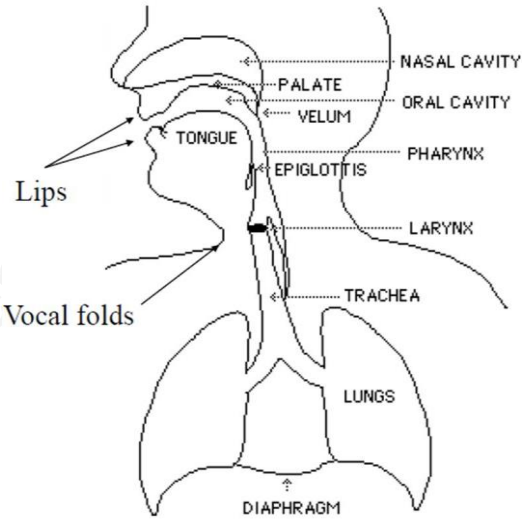
# 2. THIẾT BỊ THÍ NGHIỆM

| STT | Tên thiết bị | Số lượng |
|---|---|---|
| 01 | Máy vi tính có gắn SoundCard và cài chương trình Matlab | 01 |
| 02 | Micro | 01 |
| 03 | Loa | 01 |

# 3. LÝ THUYẾT

## 3.1. Giới thiệu

Tiếng nói là một trong những công cụ giao tiếp thông dụng và hiệu quả nhất của con người. Tín hiệu tiếng nói được dùng trong rất nhiều hệ thống thực tế như điện thoại, phát thanh, truyền hình, các hệ thống thu phát âm thanh…Các kỹ thuật xử lý số tín hiệu tiếng nói (DSP - Digital Signal Processing) thường được dùng trong các ứng dụng như: Mã hóa tiếng nói (speech coding); nhận dạng tiếng nói (speech recognition); nhận dạng người nói (speaker identification); nhận dạng các đặc điểm của người nói như giới tính, độ tuổi, cảm xúc…; triệt nhiễu trong tiếng nói (noise reduction) hoặc tổng hợp tiếng nói (speech synthesis). Để có thể xây dựng thành công các giải thuật DSP trong các ứng dụng này, chúng ta cần hiểu rõ bản chất của quá trình phát âm tiếng nói bên trong cơ thể con người. Từ đó hiểu được một số đặc điểm cơ bản của tín hiệu tiếng nói.




**Hình 41. Bộ máy phát âm**

Mục đích của bài thí nghiệm này là giúp sinh viên hiểu được quá trình phát âm tiếng nói của con người, một số đặc điểm cơ bản của tín hiệu tiếng nói như băng thông, sự phân bố năng lượng trên các băng tần số khác nhau, lấy mẫu. Bài thí nghiệm cũng giúp sinh viên biết được cách xây dựng các giải thuật đơn giản để thực hiện 03 ứng dụng là:

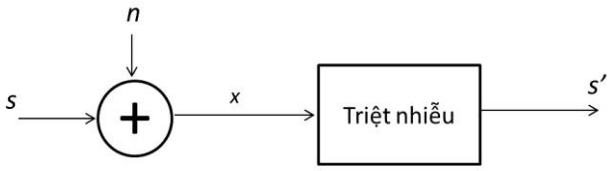
1. Triệt nhiễu nhằm nâng cao chất lượng tiếng nói (noise reduction).
2. Phát hiện và tách tín hiệu tiếng nói ra khỏi đoạn tín hiệu thu âm được (VAD: voice activity detection).

3. Nhận dạng giới tính của người nói là nam hay nữ dựa vào tiếng nói (gender recognition).

### 3.2. Cơ sở lý thuyết

Khi con người phát âm, luồng không khí từ phổi truyền qua hệ thống phát âm sẽ làm rung dây thanh quản (vocal tract). Hệ thống phát âm bao gồm những bộ phận liên quan đến việc phát âm của con người bắt đầu từ khoảng mở của vocal folds và kết thúc tại môi. Đặc trưng cộng hưởng của hệ thống thanh quản sẽ biến đổi theo hình dạng của hệ thống phát âm mà hình dạng này sẽ thay đổi trong quá trình phát âm do sự dịch chuyển của hàm răng, lưỡi, môi và các bộ phận khác bên trong khoang miệng. Con người có thể điều khiển âm thanh được phát ra bằng cách dịch chuyển các bộ phận phát âm trong miệng theo các kiểu khác nhau để phát ra các tiếng nói khác nhau.

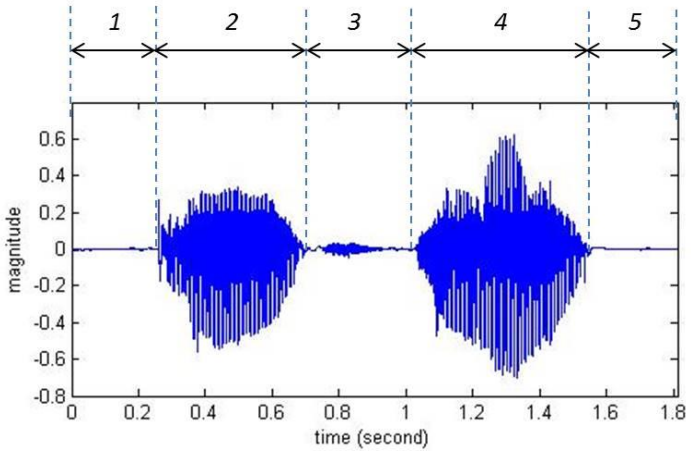
Tín hiệu tiếng nói trong thực tế thường bị ảnh hưởng của nhiễu (noise) là các tín hiệu tạp âm không mong muốn. Điều này sẽ làm ồn và làm giảm chất lượng tiếng nói. Vì vậy việc triệt nhiễu nhằm mục đích nâng cao chất lượng tiếng nói là cần thiết. Ảnh hưởng của nhiễu có thể được mô hình như là một tín hiệu khác cộng vào trong tín hiệu tiếng nói như Hình 2.




**Hình 42. Tác động của nhiễu**

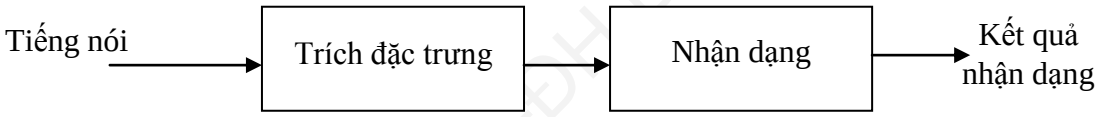
Trong mô hình trên, s là tiếng nói sạch (clean speech) chưa bị ảnh hưởng bởi nhiễu, n là nhiễu (noise), *x* là tiếng nói sau khi ảnh hưởng bởi nhiễu (noisy speech), *s'* là tiếng nói sau khi triệt nhiễu (denoised speech). Lưu ý là $x = s + n$. Chức năng của bộ triệt nhiễu là làm suy hao thành phần nhiễu *n* trong tín hiệu *x* để cải thiện chất lượng tiếng nói. Trong thực tế, tín hiệu có thể bị ảnh hưởng của rất nhiều loại nhiễu khác nhau như nhiễu động cơ xe hơi; nhiễu ồn do các tiếng nói tạp âm trong khác trong lớp học; văn phòng; sân bay….; nhiễu tiếng ồn trong nhà máy. Nếu tín hiệu hiệu nhiễu phân bố năng lượng trên vùng tần số khác với tín hiệu tiếng nói, việc triệt nhiễu có thể được thực hiện đơn giản bằng cách dùng bộ lọc để loại bỏ nhiễu ra khỏi tín hiệu tiếng nói. Do vậy, việc hiểu rõ sự phân bố năng lượng của tiếng nói và của tín hiệu nhiễu là cần thiết.

Đoạn tín hiệu tiếng nói thu được trong quá trình ghi âm sẽ có hai phần: phần có tín hiệu tiếng nói tương ứng với khoảng thời gian con người phát âm và phần không có tín hiệu tiếng nói tương ứng với khoảng thời gian yên lặng. Một trong những công đoạn đầu tiên của các hệ thống xử lý tín hiệu tiếng nói là nhận ra và tách riêng đoạn tín hiệu tiếng nói ra khỏi đoạn tín hiệu thu được (VAD: Voice Activity Detection). Các đặc trưng của tiếng nói dùng trong các hệ thống xử lý tiếng nói chỉ được tách từ đoạn tín hiệu thực sự có tiếng nói.




**Hình 43. Dạng sóng tín hiệu tiếng nói. Đoạn 2 và 4 có tín hiệu tiếng nói, đoạn 1, 3, 5 không có tín hiệu tiếng nói.**

Tín hiệu tiếng nói chứa đựng các đặc trưng có thể được sử dụng để xác định nhiều thông tin hữu ích như: nội dung các từ phát âm; đặc điểm của người phát âm như: giới tính; độ tuổi; cảm xúc, mức độ căng thẳng. Hệ thống nhận dạng các thông tin về người nói thường có hai phần chính như sau:



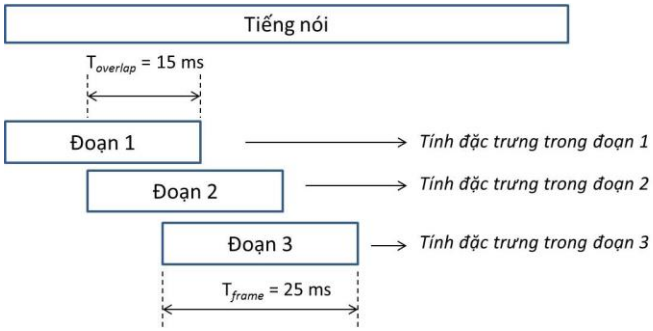

Hình 44. Sơ đồ một bộ nhận dạng dùng tiếng nói

Nhiệm vụ của bộ trích đặc trưng là tính toán các đặc trưng hữu ích để phục vụ cho quá trình nhận dạng. Các đặc trưng cơ bản của tiếng nói là: chu kỳ cao độ (pitch period), năng lượng (energy), các tần số cộng hưởng (formants), tốc độ qua điểm zero, MFCC (Mel frequency cepstrum coefficients)…Tùy vào mục tiêu nhận dạng mà ta có thể sử dụng một hay nhiều đặc trưng khác nhau trong số các đặc trưng này. Đặc trưng tiếng nói thông thường được tính trên mỗi đoạn tín hiệu có chiều dài $T_{frame}$ =25ms. Các đoạn kế tiếp nhau chồng lấn nhau $T_{overlap}$=15ms theo Hình 5. Như vậy, các vị trí bắt đầu của 2 khung kế tiếp nhau cách nhau 10 ms. Giá trị này được chọn vì trong khoảng thời gian này, tín hiệu tiếng nói tương đối ổn định do bộ máy phát âm của con người ít thay đổi trong khoảng thời gian ngắn.

Năng lượng của tín hiệu tiếng nói có thể được xác định trong miền thời gian hoặc tần số:

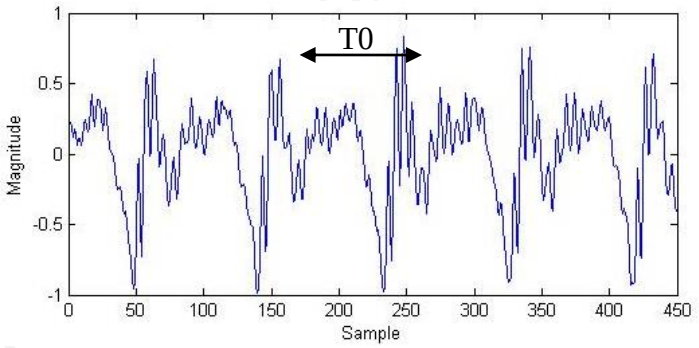
$$SpeechEnergy = \sum_{n=0}^{L-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2$$

Trong đó L là tổng số mẫu tiếng nói; x[n] là mẫu thứ n trong đoạn tiếng nói; N là tổng số điểm tính DFT (Discrete Fourier Transform); X(k) là giá trị vạch phổ thứ k của biến đổi DFT.

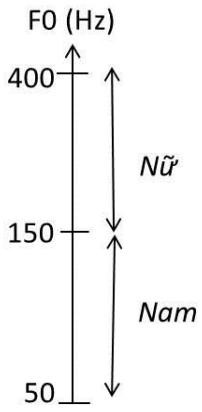



**Hình 45. Phân đoạn tiếng nói**

Khi con người phát ra tiếng nói, dây thanh quản sẽ rung theo một chu kỳ nhất định. Có nhiều yếu tố tác động đến chu kỳ này như: từ được phát âm, độ tuổi, cảm xúc giới tính…Về mặt giới tính, sự khác nhau về bộ máy phát âm của nam và nữ làm cho tốc độ rung dây thanh quản của nữ sẽ nhanh hơn so với nam. Nói cách khác, chu kỳ rung dây thanh quản của nữ sẽ nhỏ hơn của nam. Chu kỳ rung này sẽ thể hiện trong đặc trưng chu kỳ cao độ T0 (hoặc tần số cao độ F0=1/T0) của tín hiệu tiếng nói như trong Hình 6, Lưu ý tín hiệu tiếng nói trong vùng nguyên âm có dạng gần tuần hoàn, tín hiệu tiếng nói trong các chu kỳ kế tiếp nhau là gần giống nhau (nhưng không giống nhau hoàn toàn). Vì vậy giá trị T0 cũng thay đổi (rất nhỏ) trong các chu kỳ khác nhau. Chu kỳ cao độ trong một đoạn tiếng nói được tính là trung bình tất cả các giá trị T0 trong đoạn tín hiệu tiếng nói đó.

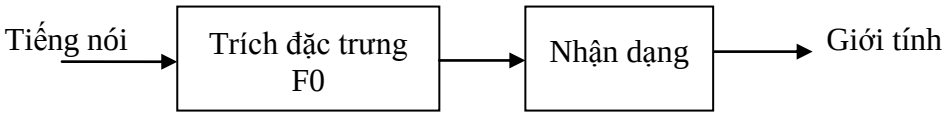



**Hình 46. Dạng sóng của một đoạn tín hiệu tiếng nói**




Tần số cao độ được định nghĩa là nghịch đảo của chu kỳ cao độ F0=1/T0. Tần số cao độ của tiếng nói con người thay đổi trong khoảng từ 50Hz đến 400Hz. Tần số cao độ của nam là thấp hơn so với nữ do bộ máy phát âm của hai giới là khác nhau. ***Trong phạm vi bài thí nghiệm này***, tần số ngưỡng 150Hz được dùng để phân chia tiếng nói là của nam hay nữ. Cụ thể là 50Hz< $F0_{nam}$ < 150Hz và 150Hz≤$F0_{nữ}$ <400Hz. Lưu ý rằng việc phân chia này chỉ mang tính chất tương đối vì trong thực tế có giọng nữ trầm (F0<150Hz) và giọng nam cao (F0>150Hz) nên ranh giới 150Hz là

không hoàn toàn đúng cho tất cả mọi trường hợp. Vì vậy, độ chính xác của việc nhận dạng giới tính bằng phương pháp này có thể không đạt đến mức 100%. Sơ đồ của một bộ nhận dạng giới tính như Hình 7.




**Hình 47. Sơ đồ bộ nhận dạng giới tính**

Chức năng của phần trích đặc trưng là trích F0 và tính F0 trung bình của tiếng nói ngõ vào. Thông số này sẽ được phần Nhận dạng so sánh với ngưỡng 150Hz để xác định giới tính của người phát âm là nam hay nữ.

## 4. CHUẨN BỊ THÍ NGHIỆM

1. Bên cạnh nội dung tiếng nói (từ được phát âm), hãy liệt kê những thông tin khác có thể biết được khi người nghe nghe một đoạn tín hiệu tiếng nói.

2. Khi nghe một đoạn tiếng nói, người nghe có thể biết được tiếng nói này là của nữ hay nam không? Giải thích.

3. Tín hiệu tiếng nói khi được truyền trên kênh truyền điện thoại, băng thông của nó sẽ được giới hạn trong khoảng tần số nào? Cho biết lý do của việc giới hạn băng thông.

4. Việc thu tiếng nói trong Matlab có thể được thực hiện bằng lệnh y = wavrecord(N, Fs, DataType). Hãy cho biết ý nghĩa các thông số y, N, Fs, DataType? Vì sao cần phải xác định 3 thông số này trong quá trình thu tiếng nói.

5. Việc phát tiếng nói trong Matlab có thể được thực hiện bằng lệnh wavplay(y, Fs). Fs là tần số lấy mẫu của y. Hãy cho biết nếu nhập tần số lấy mẫu không đúng thì kết quả tiếng nói được phát âm sẽ như thế nào?

6. Tín hiệu tiếng nói y có tổng số 1000 mẫu. Biết rằng y được thu tại tốc độ lấy mẫu Fs=16000Hz. Cho biết tín hiệu y được thu trong bao nhiêu mili giây?.....

7. Tín hiệu tiếng nói y trong câu 6 được biết đổi Fourier rời rạc (Discrete Fourier Transform) tại N=1024 điểm. Biết rằng độ rộng giữa hai vạch phổ kế tiếp nhau là Δf. Tính Δf.

   Δf = …….Hz

8. Vì sao việc triệt nhiễu trong tín hiệu tiếng nói là cần thiết? Cho biết một số cách có thể được dùng để đánh giá của hiệu quả của quá trình triệt nhiễu trong tiếng nói

9. Dựa vào đặc trưng nào để nhận biết đoạn tín hiệu thực sự có tiếng nói trong toàn bộ đoạn tín hiệu ghi âm được. Giải thích ngắn gọn.

# 5. TIẾN HÀNH THÍ NGHIỆM

## 5.1. Ước lượng băng thông và khảo sát sự phân bố năng lượng

a. Sử dụng hàm wavread trong Matlab để đọc file tiếng nói 'CleanSpeech.wav', gán tín hiệu tiếng nói trong file này là biến sp. Xác định tần số lấy mẫu fs của tín hiệu tiếng nói lưu trong file này.

fs= ……mẫu/giây

b. Biết rằng đoạn tiếng nói trên được thu trong khoảng thời gian *t* giây. Từ giá trị fs tìm được ở trên, hãy xác định *t*.

*t*= ……giây

c. Sử dụng lệnh wavplay, phát và nghe lại đoạn tiếng nói trên tại ba tần số lấy mẫu khác nhau là f1=0.5fs, f2=fs, f3=2fs. Cho nhận xét về đoạn âm thanh nghe được trong mỗi trường hợp.

d. Sử dụng Matlab, vẽ dạng sóng tín hiệu tiếng nói sp trên với trục ngang biểu diễn theo đơn vị giây. Tính và vẽ phổ biên độ của sp trong khoảng tần số từ 0 đến fs/2 (sử dụng công cụ fft). Lưu ý: trục ngang của hình vẽ phổ biên độ phải được biểu diễn theo đơn vị Hz.


e. Dựa trên hình vẽ phổ biên độ này, hãy ước lượng băng thông của tiếng nói (độ rộng của đoạn tần số tính từ 0Hz, chứa khoảng hơn 95% năng lượng của tín hiệu tiếng nói).

Băng thông của tín hiệu tiếng nói là: BW =…… Hz

f. Dựa trên kết quả tính fft trong câu 6.1, hãy tính phần trăm năng lượng của tiếng nói phân bố trên các vùng tần số khác nhau có cùng độ rộng là 1kHz và hoàn thành Bảng 1.

Bảng 1: Phần trăm năng lượng tiếng nói trên các băng tần khác nhau

| | Băng Tần | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Băng tần | 0-1kHz | 1-2kHz | 2-3kHz | 3-4kHz | 4-5kHz | 5-6kHz | 6-7kHz | 7-8kHz |
| % năng lượng | | | | | | | | |

Cho biết vùng tần số nào chứa nhiều năng lượng tiếng nói nhất?.....

g. Dựa vào kết quả trong Bảng 1 hãy giải thích vì sao tín hiệu tiếng nói truyền trên đường truyền điện thoại bị giới hạn tần số dưới 3.5kHz?

**5.2. Xác định tần số lấy mẫu tối thiểu**

a. Từ kết quả trong câu 6.1, hãy ước lượng tần số lấy mẫu nhỏ nhất $fs_{min}$ của tiếng nói để tín hiệu sau khi lấy mẫu không bị aliasing (chồng phổ).

$fs_{min}$=…….mẫu/giây

b. Tín hiệu tiếng nói trong câu 6.1 có thể được giảm tần số lấy mẫu để tạo nên tín hiệu có tần số lấy mẫu nhỏ hơn theo dùng lệnh

```
spDownSampling=sp(1:N:end);
```

Trong đó N = 2, 3, 4…. là tốc độ giảm tần số lấy mẫu. Việc giảm tần số lấy mẫu của tín hiệu được thực hiện bằng cách trong mỗi đoạn N mẫu tín hiệu gốc liên tiếp nhau thì giữ lại một mẫu và bỏ đi N-1 mẫu còn lại để tạo thành tín hiệu mới.

c. Hãy cho biết lợi ích của việc giảm tần số lấy mẫu?

d. Hãy tiến hành giảm tần số lấy mẫu của tiếng nói gốc trong câu 6.1. Ứng với mỗi giá trị N, thực hiện các bước sau và hoàn thành Bảng 2.

   i. Tính và vẽ phổ biên độ của tín hiệu tiếng nói spDownSampling trong khoảng tần số từ [0 fs/(2N)] Hz.

   ii. Vẽ dạng sóng tín hiệu tiếng nói trong 4 mili giây đầu tiên dùng lệnh stem(spDownSampling)

iii. Nghe tín hiệu tiếng nói sau khi giảm tốc độ lấy mẫu theo lệnh sau:

wavplay(spDownSampling,fs/N)

| | |
|---|---|
| Phổ biên độ tiếng nói với N=1 | Dạng sóng tiếng nói với N=1 |
| Phổ biên độ tiếng nói với N=2 | Dạng sóng tiếng nói với N=2 |
| Phổ biên độ tiếng nói với N=3 | Dạng sóng tiếng nói với N=3 |
| Phổ biên độ tiếng nói với N=4 | Dạng sóng tiếng nói với N=4 |
| Phổ biên độ tiếng nói với N=8 | Dạng sóng tiếng nói với N=8 |

Bảng 2: Giảm tần số lấy mẫu

| Tốc độ giảm tần số lấy mẫu | Tần số lấy mẫu mới (kHz) | Tổng số mẫu của spDownSampling | Mức độ aliasing |
|---|---|---|---|
| N=1 | | | |
| N=2 | | | |
| N=3 | | | |
| N=4 | | | |
| N=8 | | | |

e. Trong các giá trị N đã chọn trong Bảng 2, bắt đầu từ giá trị nào thì hiện tượng aliasing xảy ra? Giải thích nguyên nhân.

### 5.3. Triệt nhiễu nâng cao chất lượng tiếng nói

Tín hiệu tiếng nói trong câu 6.1 bị ảnh hưởng của nhiễu cộng là một tín hiệu SIN tần số cao. Tiếng nói bị ảnh hưởng bởi nhiễu được lưu trong file 'NoisySpeech.wav'.

Sử dụng hàm wavread trong Matlab để đọc file tiếng nói bị tác động bởi nhiễu, gán tín hiệu tiếng nói trong file này là biến spNoisy. Sinh viên được yêu cầu ứng dụng bộ lọc để triệt nhiễu nhằm mục đích nâng cao chất lượng tiếng nói.

a. Tính và vẽ phổ biên độ của tín hiệu spNoisy trong khoảng tần số từ 0 đến fs/2. Lưu ý: trục ngang của hình vẽ phổ biên độ phải được biểu diễn theo đơn vị Hz. So sánh với phổ của tín hiệu tiếng nói chưa bị tác động của nhiễu (sp) trong câu 6.1 để xác định tần số của tín hiệu nhiễu $f_{Noise}$.

Phổ biên độ của tín hiệu tiếng nói bị nhiễu

$f_{Noise}$=……..Hz

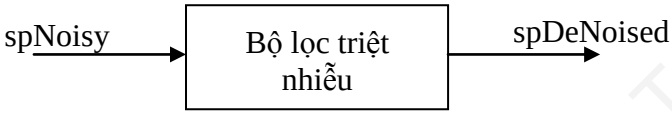
Tín hiệu spNoisy được truyền qua bộ lọc như Hình 8 nhằm triệt nhiễu để nâng cao chất lượng tiếng nói. Hãy cho biết để khử nhiễu hiệu quả, bộ lọc triệt nhiễu cần có đặc tính gì?

Loại bộ lọc (thông cao, thấp, dải, chắn dải)?.....

Tần số cắt của bộ lọc:

Giải thích ….....

……




**Hình 48. Bộ lọc triệt nhiễu**

b. Sử dụng 03 bộ lọc thông thấp Chebyshev loại 1 khác nhau để triệt nhiễu trong tín hiệu spNoisy. Cả ba bộ lọc này đều có tần số cắt fc=4kHz và độ gợn dải thông Rp=0.2dB. Bậc của bộ lọc thay đổi như trong Bảng 3.

Sử dụng lệnh cheby1 trong Matlab để xác định các hệ số B và A của các bộ lọc 1, 2 và 3. Sau đó dùng lệnh filter để thực hiện việc lọc nhiễu trong tín hiệu spNoisy. Hãy thực hiện các cách sau để đánh giá hiệu quả của quá trình triệt nhiễu và hoàn thành Bảng 3.

i. Nghe tín hiệu trước và sau khi triệt nhiễu.

ii. Vẽ và so sánh dạng sóng của tín hiệu trước và sau khi triệt nhiễu.

iii. Vẽ và so sánh phổ biên độ của tín hiệu tiếng nói trước và sau khi triệt nhiễu.

iv. Tính thông số Tỷ số tín hiệu trên nhiễu được ký hiệu là SNR (Signal to Noise Ratio) theo công thức sau:

$$SNR\,(dB) = 10\log_{10}\left(\frac{\text{SpeechPower}}{\text{NoisePower}}\right)$$

Trong đó: *SpeechPower* là công suất tín hiệu tiếng nói chưa bị ảnh hưởng bởi nhiễu (sp). *NoisePower* là công suất của nhiễu. Nhiễu được xác định là hiệu số của tín hiệu sau khi xử lý và tín hiệu trước khi xử lý (sp). Dễ hiểu là SNR càng lớn thì chất lượng tín hiệu tiếng nói càng tốt. Lưu ý: Công suất tín hiệu có thể được tính là tổng bình phương tất cả các mẫu của

tín hiệu đó. $Power = \sum_{k=0}^{L-1}(s[k])^2$ . Trong đó s[k] là biên độ của mẫu tín hiệu thứ k trong tổng số L mẫu của tín hiệu s.

Xác định SNR của tín hiệu tiếng nói trước khi triệt nhiễu spNoisy

$SNR_{spNoisy}$ = …….dB

Bảng 3: So sánh hiệu quả các bộ lọc triệt nhiễu

| | **Bậc bộ lọc** | **SNR sau khi triệt nhiễu (dB)** | **Chất lượng tiếng nói sau khi triệt nhiễu** |
|---|---|---|---|
| Bộ lọc 1 | N=2 | | |
| Bộ lọc 2 | N=4 | | |
| Bộ lọc 3 | N=8 | | |

c. Hãy vẽ dạng sóng và phổ biên độ của sp, spNoisy và spDeNoised hình vẽ sau.

| | |
|---|---|
| Dạng sóng tiếng nói gốc (chưa bị nhiễu) | Phổ biên độ tiếng nói gốc |
| Dạng sóng tiếng nói bị nhiễu | Phổ biên độ tiếng nói bị nhiễu |
| Dạng sóng tiếng nói sau triệt nhiễu dùng Bộ lọc 1 | Phổ biên độ tiếng nói sau triệt nhiễu dùng Bộ lọc 1 |

| | |
|---|---|
| Dạng sóng tiếng nói sau triệt nhiễu dùng Bộ lọc 2 | Phổ biên độ tiếng nói sau triệt nhiễu dùng Bộ lọc 2 |
| Dạng sóng tiếng nói sau triệt nhiễu dùng Bộ lọc 3 | Phổ biên độ tiếng nói sau triệt nhiễu dùng Bộ lọc 3 |

d. Cho nhận xét và giải thích về hiệu quả triệt nhiễu của các bộ lọc 1, 2, 3.

| |
|---|
| |

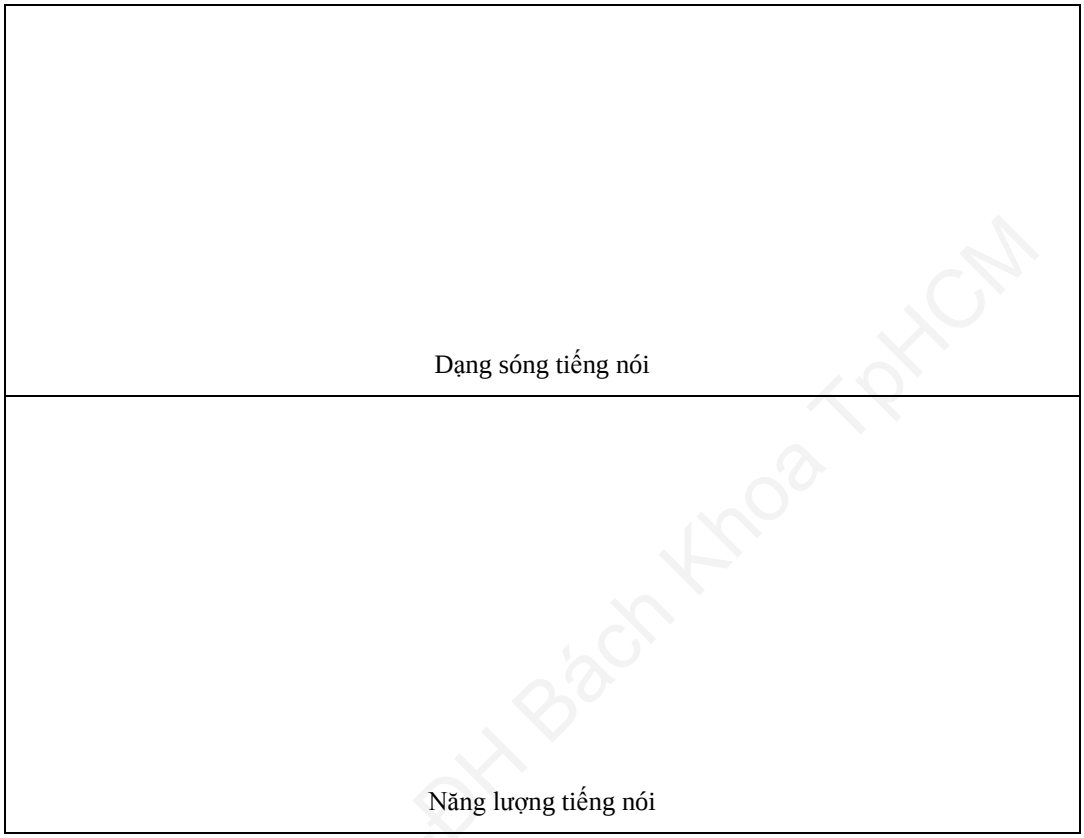


### 5.4. Tách tiếng nói ra khỏi đoạn tín hiệu thu được

a. Hãy phân đoạn đoạn tín hiệu tiếng nói sp trong mục 6.1 theo hướng dẫn trong Mục 4 và hãy cho biết tổng số đoạn tiếng nói nhận được là ……..đoạn.

b. Viết một đoạn chương trình tính Năng lượng từng đoạn tiếng nói và vẽ thông số này trên cùng màn hình với dạng sóng tiếng nói. Lưu ý: Hình vẽ dạng sóng tiếng nói và Năng lượng phải được sắp xếp thẳng hàng. Trục ngang hình vẽ dạng sóng

tiếng nói biểu diễn theo chỉ số mẫu, trục ngang của hình vẽ năng lượng biểu diễn theo chỉ số đoạn.




Từ kết quả nhận được hãy cho biết sự thay đổi của Năng lượng ứng với các vùng tiếng nói khác nhau

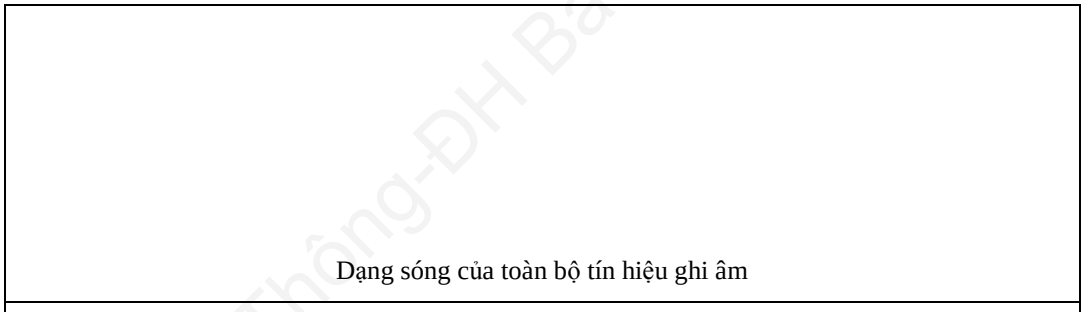
c. Tiếng nói sau khi thu được sẽ bao gồm hai phần, phần có tín hiệu tiếng nói và phần không có tín hiệu tiếng nói (tương ứng với khoảng thời gian yên lặng trong quá trình phát âm). Việc trích đặc trưng trong các ứng dụng xử lý tiếng nói được thực hiện trong đoạn có tiếng nói. Vì vậy, việc nhận biết và tách đoạn tiếng nói ra khỏi đoạn tín hiệu thu được là cần thiết. Quá trình này được gọi là phát hiện tiếng nói (Voice Activity Detection VAD).

Từ sự quan sát về sự thay đổi của Năng lượng trong phần 6.4b, hãy đề nghị một giải thuật tách đoạn tín hiệu tiếng nói ra khỏi đoạn tín hiệu thu được sử dụng Năng lượng. Viết đoạn chương trình thực hiện giải thuật này bằng Matlab.

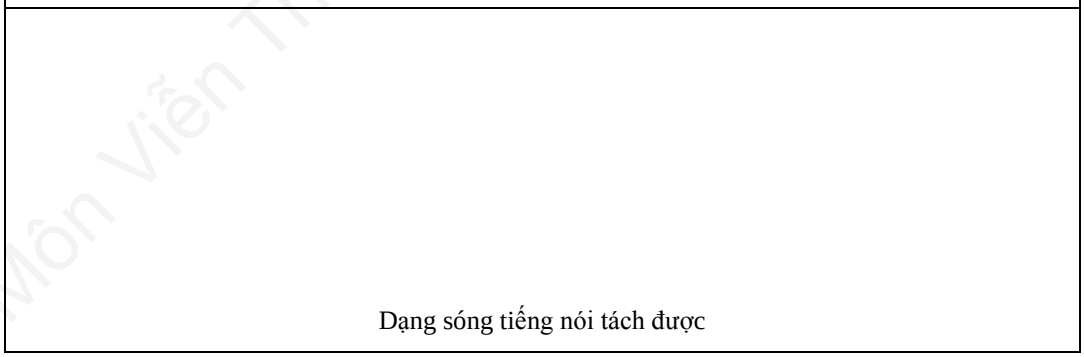
Giải thuật tách đoạn tín hiệu tiếng nói ra khỏi đoạn tín hiệu ghi âm được:

Đoạn chương trình thực hiện VAD bằng Matlab

Kết quả thực hiện.



Dạng sóng của toàn bộ tín hiệu ghi âm



Dạng sóng tiếng nói tách được



### 5.5. Nhận dạng giới tính.

Lệnh pitch_rapt được dùng để trích F0 trên tất cả các đoạn tiếng nói (chiều dài 25ms và chồng lấn 15ms giữa hai đoạn kế tiếp). Lệnh này được dùng như sau:

[mf0]=pitch_rapt(s,fs) trong đó mf0 là vector chứa giá trị F0 trên các đoạn tiếng nói của trong tín hiệu s, fs là tần số lấy mẫu của s.

a. Dùng lệnh pitch_rapt hãy trích và vẽ F0 của 02 file tiếng nói, 1 file tiếng nói của nữ (‘*Female.wav*’), 1 file tiếng nói của nam (‘*Male.wav*’). Tính F0 trung bình của tiếng nói người nữ Mean_F0$_{nữ}$ và người nam Mean_F0$_{nam}$. Lưu ý rằng F0 của những đoạn không có tiếng nói không xác định được và pitch_rapt sẽ gán giá trị F0=0 cho những đoạn này. F0 trung bình chỉ tính cho những đoạn có F0>0.

Hình vẽ F0 của nữ, file ‘Female.wav’.

Hình vẽ F0 của nam, file ‘Male.wav’.

Mean_F0$_{nữ}$ =……Hz

Mean_F0$_{nam}$ =……Hz

Nhận xét về khoảng dao động và giá trị trung bình của F0 người nữ và người nam.

b. Sinh viên được cung cấp 10 file tiếng nói của 10 người nói (speaker) lưu dưới dạng file .wav, thực hiện các bước sau và hoàn thành Bảng 4.

i. Hãy nghe 10 file tiếng nói này và sau khi nghe xác định giới tính người nói là nam hay nữ. Điền kết quả vào hàng (1).

ii. Sử dụng lệnh pitch_rapt hãy xác định viết một đoạn chương trình để tính F0 trung bình của từng file tiếng nói. Từ đó so sánh với ngưỡng 150Hz để xác định giới tính người phát âm. Điền kết quả vào hàng (3) và (4).

Bảng 4: Kết quả nhận dạng giới tính

| Speaker | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| (1) Nghe | | | | | | | | | | |
| (2) Kết luận | | | | | | | | | | |
| (3) F0 (Hz) | | | | | | | | | | |
| (4)Chương trình | | | | | | | | | | |
| (5) Kết luận | | | | | | | | | | |

Đối chiếu kết quả nhận được với Giáo viên hướng dẫn để biết kết quả nhận dạng theo mỗi phương pháp là *đúng* hay *sai* và ghi vào ô tương ứng trong hàng (2)&(5) Bảng 4. Độ chính xác của quá trình nhận dạng theo phần trăm được tính như sau:

*Độ chính xác (%) = (Số lần nhận dạng đúng/Tổng số lần nhận dạng)*100.*

c. Tính độ chính xác việc nhận dạng giới tính 10 file cho sẵn bằng cách lắng nghe …….%

Tính độ chính xác việc nhận dạng giới tính bằng chương trình Matlab…….%

d. Sử dụng wavread, đọc và thu vào máy tính dòng chữ: "*Bộ môn Viễn thông, Khoa Điện, Trường Đại học Bách khoa Thành phố Hồ Chí Minh*". Tốc độ lấy mẫu fs=16kHz, datatype='double', gán tín hiệu thu được vào biến s trong Matlab. Dùng pitch_rapt hãy tính F0 trung bình của chuỗi tiếng nói này. Hãy lặp lại việc này 5 lần và ghi nhận kết quả vào phần sau:

Giới tính của bạn……

Lần 1: F01=……Hz

Lần 2: F02=……Hz

Lần 3: F03=……Hz

Lần 4: F04=……Hz

Lần 5: F05=……Hz

F0TrungBình=……Hz

### 5.6. Phần mở rộng

Lưu ý: *Phần này không bắt buộc sinh viên thực hiện. Nếu hoàn thành đúng phần này, sinh viên được cộng 2 điểm vào bài báo cáo*.

Sử dụng Matlab, hãy viết một đoạn chương trình để khi người đọc đọc vào máy tính một đoạn tiếng nói (chậm và rời rạc), chương trình sẽ thực hiện một số yêu cầu sau:

a. Xác định tổng số từ trong đoạn tiếng nói thu vào.

b. Tách tiếng nói của mỗi từ riêng biệt và lưu vào các file '.wav'. Tên file sẽ là số thứ tự của từ tương ứng trong đoạn tiếng nói. Ví dụ '1.wav', '2.wav'…

c. Tính chiều dài của tiếng nói mỗi từ theo đơn vị mili giây và chiều dài trung bình của tất cả các từ đọc vào

7

# XỬ LÝ ẢNH SỐ

**Họ và tên SV báo cáo 1: …… MSSV: ……..…..**

**Họ và tên SV báo cáo 2: …… MSSV: ……..…..**

**Họ và tên SV báo cáo 3: …… MSSV: ……..…..**

**Họ và tên SV báo cáo 4: …… MSSV: ……..…..**

**Nhóm lớp: …..…… Tiểu nhóm: …..…..….. Ngày thí nghiệm: ……..……..**

| **Điểm đánh giá** | | | | **CBGD nhận xét và ký tên** |
|---|---|---|---|---|
| **Chuẩn bị lý thuyết** | **Báo cáo và kết quả TN** | **Kiểm tra** | **Kết quả** | |

## 1. MỤC ĐÍCH THÍ NGHIỆM

Bài thí nghiệm này nhằm giúp sinh viên tìm hiểu và kiểm chứng lại lý thuyết đồng thời thực hiện mô phỏng các quá trình xử lý ảnh sau:

- Triệt nhiễu muỗi tiêu và nhiễu tuần hoàn.
- Làm nhòe và làm sắc nét ảnh.

## 2. THIẾT BỊ SỬ DỤNG

- Máy tính cá nhân.
- Phần mềm MATLAB.

## 3. LÝ THUYẾT VÀ CHUẨN BỊ THÍ NGHIỆM

## 3.1. Giới thiệu ảnh số

### 3.1.1. Ảnh số:

- Ảnh f(x,y) được miêu tả bằng những mẫu cách đều nhau ở dạng ma trận (N-M):

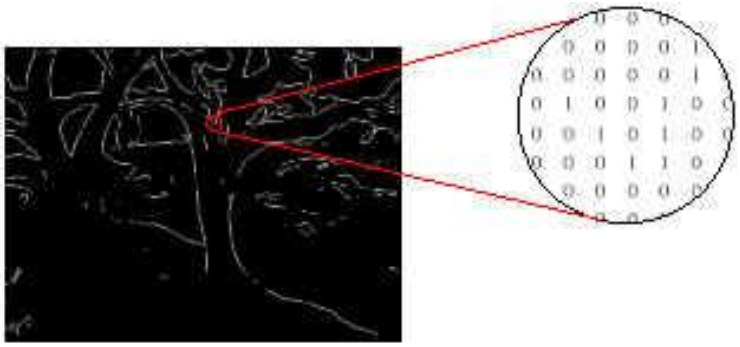
$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} = A$$

- Ma trận A được gọi là ảnh số, mỗi thành phần của A được gọi là một thành phần ảnh, hay pixel, hoặc pel.
- Lấy mẫu: chia mặt phẳng xy thành mắt lưới, tọa độ của mỗi mắt lưới là (x, y), trong đó x,y là số nguyên.
- Lượng tử: f được gán bằng một giá trị mức xám G (thực hoặc nguyên).
- Trong thực tế, $N = 2^n$, $M = 2^k$, $G = 2^m$. Tổng số bit cần chứa ảnh là: NxMxm
- Độ phân giải: mức độ chi tiết điểm ảnh, phục thuộc và số mẫu và số mức xám.

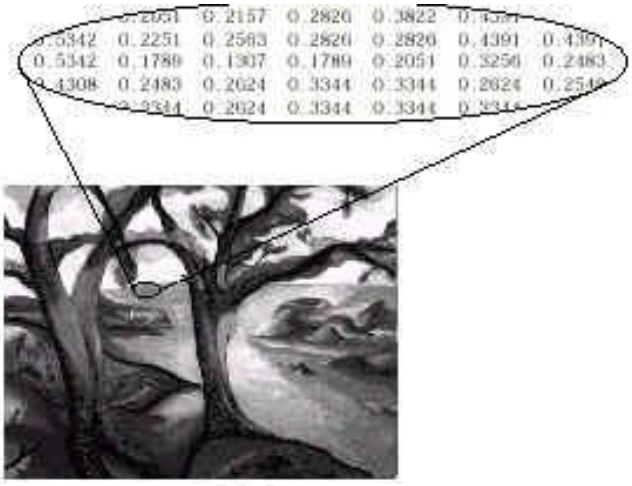

Màu sắc: kết hợp của 3 màu RGB


**Hình 49. Ảnh 2 chiều**

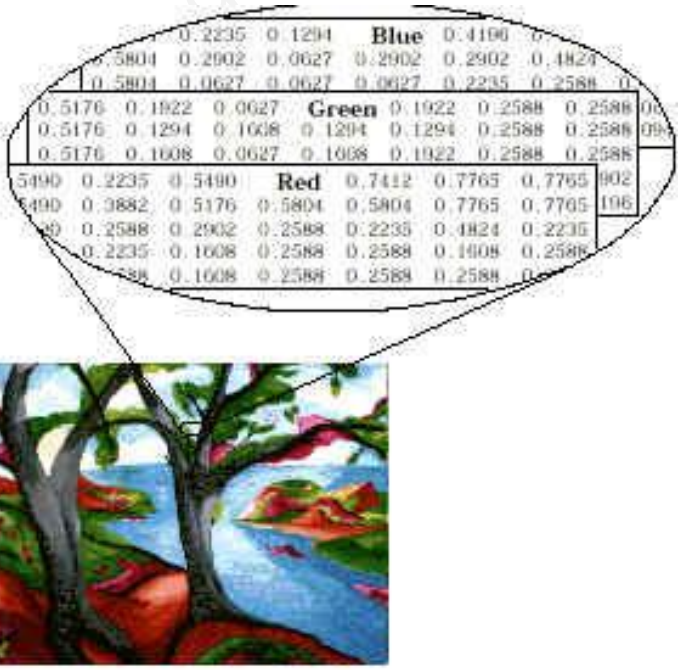

**Hình 50. Ảnh động**



**Hình 51. Ảnh nhị phân**



**Hình 52. Ảnh mức xám**




**Hình 53. Ảnh màu**

**Câu hỏi chuẩn bị:**

1. Tính số MB cần để lưu trữ một ảnh màu RGB kích thước 1080x1920.

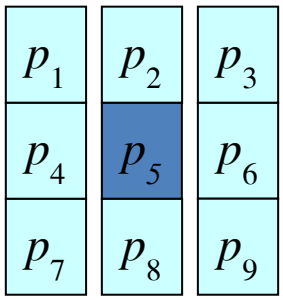
| G= |
|---|

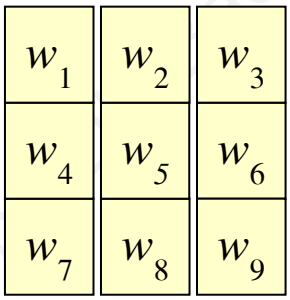
2. Tính băng thông Mbps cần để truyền một phim HD 1080x1920, tốc độ 30 khung hình/s, chiều dài 2h.

| B= |
|---|

### 3.1.2. Tác vụ đại số:

- ❑ Tác vụ đại số giữa hai pixel p và q: bao gồm các tác vụ cộng, trừ, nhân, chia (thực hiện trên từng pixel).
- ❑ Tác vụ mặt nạ (cửa sổ):




$$p = \sum_{i=1}^{9} w_i p_i \rightarrow p_5$$

Với sự chọn lựa thông số w thích hợp, tác vụ có thể được dùng để triệt nhiễu, làm mỏng hay phát hiện cạnh.

**Câu hỏi chuẩn bị:**

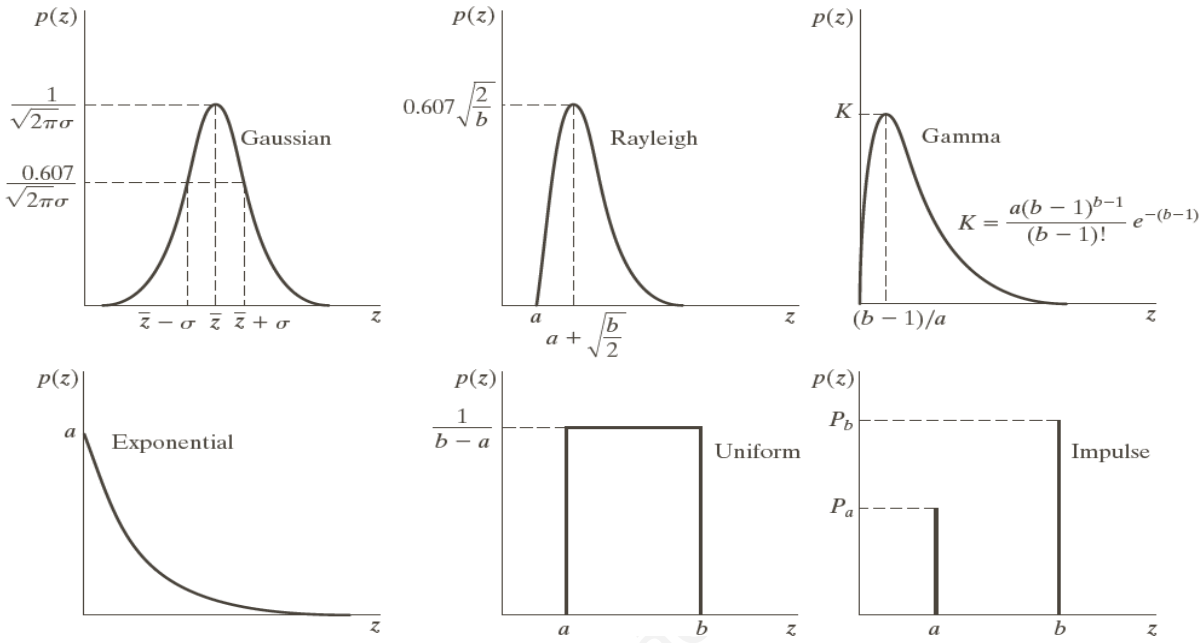
1. Nếu $w_i$=1/9 thì tác vụ là lọc loại gì (thông cao, thông dãy hay thông thấp).

| Loại bộ lọc = |
|---|

2. Nếu $w_i$=1( i≠5) và $w_5$=8 thì tác vụ là lọc loại gì (thông cao, thông dãy hay thông thấp):

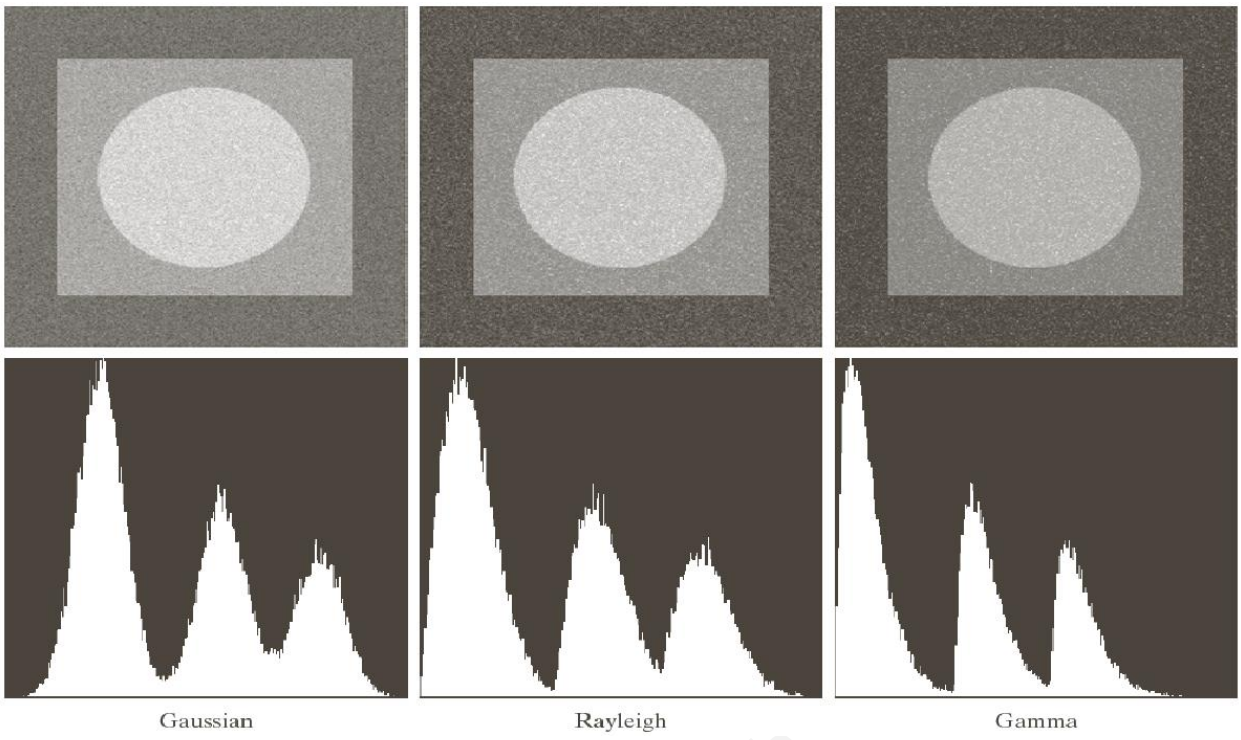
| Loại bộ lọc = |
|---|

### 3.1.3. Các loại nhiễu ảnh:

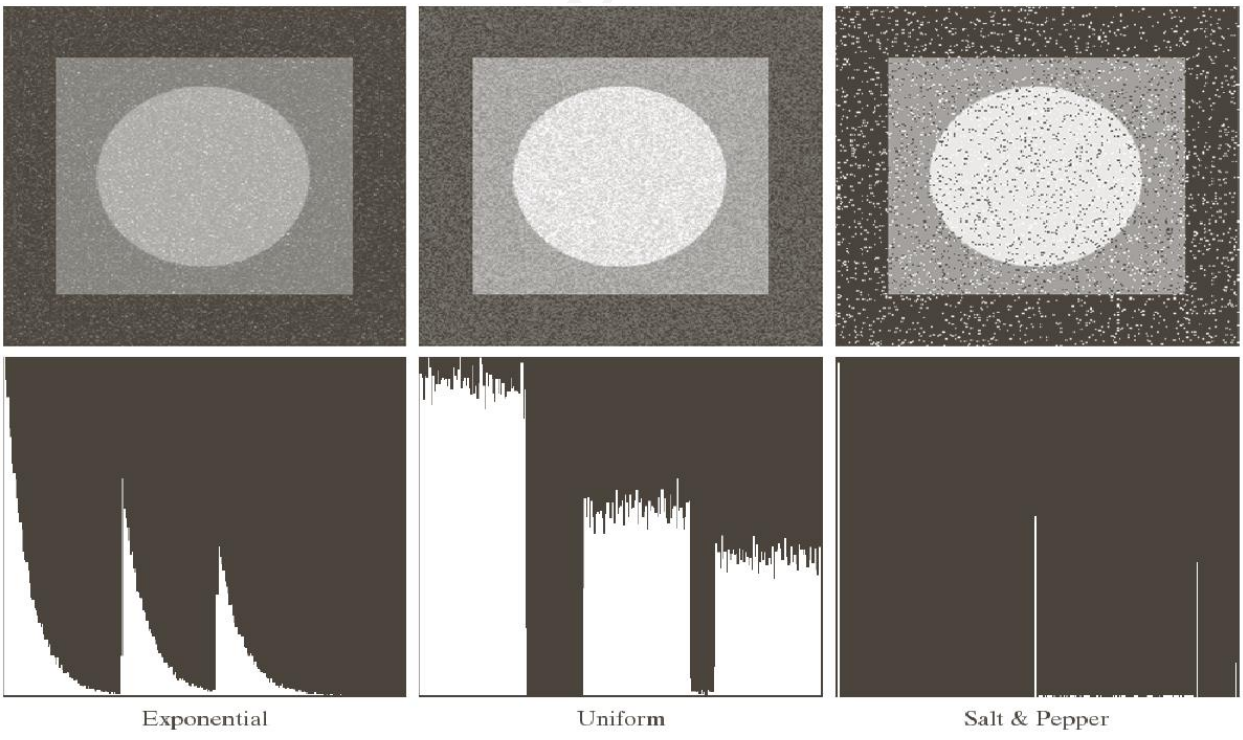



**Hình 54. Một số hàm mật độ xác suất nhiễu**

**Hình 55. Ảnh gốc**




**Hình 56. Ảnh và phân bố xác suất với các loại nhiễu khác nhau 1.**




**Hình 57. Ảnh và phân bố xác suất với các loại nhiễu khác nhau 2.**

Hàm phân bố xác suất (PDF) của biến ngẫu nhiên Gauss được cho bởi:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\bar{z})^2/2\sigma^2}$$

Trong đó σ là variance và $\bar{z}$ là giá trị trung bình.

Hàm phân bố xác suất (PDF) của nhiễu muối tiêu (salt and pepper) được cho bởi:

$$p(z) = \begin{cases} P_a & z = a \\ P_b & z = b \\ khác \end{cases}$$

Trong đó a=0 và b=255 cho ảnh mức xám 8 bit.

**Câu hỏi chuẩn bị:**

1. Nhiễu Gauss là nhiễu có giá trị liên tục hay rời rạc.

| Loại nhiễu= |
|---|

2. Nhiễu muối tiêu là nhiễu có giá trị liên tục hay rời rạc.

| Loại nhiễu= |
|---|

**3.1.4. Các loại bộ lọc theo sắp xếp thứ tự:**

Bộ lọc median

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$

Bộ lọc max

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}$$

Bộ lọc min

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

Bộ lọc midpoint

$$\hat{f}(x,y) = \frac{1}{2}\left(\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\}\right)$$

**Câu hỏi chuẩn bị:**

1. Để loại bỏ nhiễu muối (giá trị mức xám 255), thì có thể sử dụng loại bộ lọc nào ở trên.

| Loại bộ lọc = |
|---|

2. Để loại bỏ nhiễu tiêu (giá trị mức xám 0), thì có thể sử dụng loại bộ lọc nào ở trên:

Loại bộ lọc =

3. Để loại bỏ nhiễu muối tiêu, thì có thể sử dụng loại bộ lọc nào ở trên:

Loại bộ lọc =

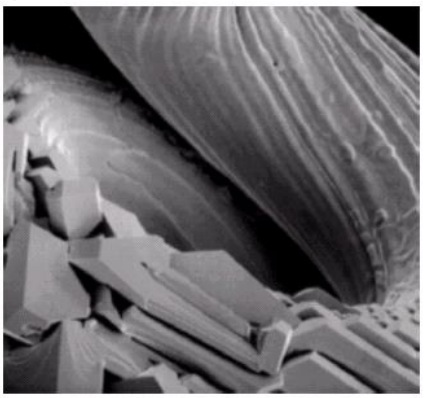
### 3.2. Làm sắc nét hình:

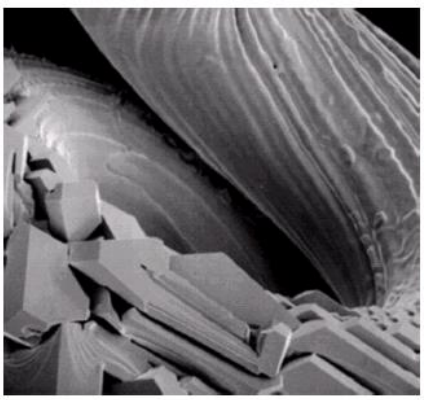
Ứng dụng này sử dụng tác vụ đại số để lọc thành phần tần số cao của ảnh bằng cách sử dụng đạo hàm vi phân. Nếu việc lấy trung bình (tích phân) tương đương với bộ lọc thong thấp (tác dụng là nhòe hình) thì việc lấy đạo hàm (vi phân) tương ứng với bộ lọc thông cao. Thành phần tần số cao này được cộng lại với ảnh gốc để được ảnh có tần số cao được khuyếch đại, làm cho ảnh sắc nét hơn.

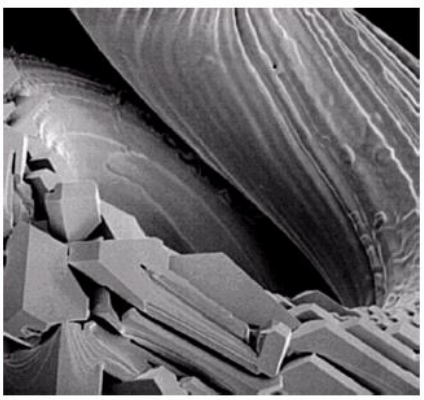
Ảnh bên dưới là một ví dụ làm sắc nét hình. Hình hàng trên bên phải là hình gốc, trong khi hình hàng dưới bên trái là hình nâng sắc nét dùng bộ lọc với mặt nạ bên trái, và hình hàng dưới bên phải là hình nâng sắc nét dùng bộ lọc với mặt nạ bên phải. Kết quả cho thấy cả hai hình thu được có độ sắc nét tốt hơn hình gốc.

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 5 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 9 | −1 |
| −1 | −1 | −1 |







# 4. TIẾN TRÌNH THÍ NGHIỆM

Trong quá trình thí nghiệm SV sẽ thực hiện các nhiệm vụ sau:

- Viết mã MATLAB denoisesaltpepper.m để triệt nhiễu muối tiêu dùng bộ lọc trung vị.
- Viết mã MATLAB denoiseperiodic.mđể triệt nhiễu tuần hoàn dùng bộ lọc trung vị.
- Viết mã MATLAB makeJPEG.m nhằm:
  - Lọc ảnh dùng chuẩn JPEG.
  - Tính toán giá trị PSNR.
  - Kiểm tra chất lượng của ảnh nén.
  - Nén video chuẩn Motion JPEG.

## 4.1. Tạo và triệt nhiễu muối tiêu

### 4.1.1. Đọc ảnh gốc vào

Đọc ảnh gốc tên pep.tif và ảnh nhiễu tên n2.tif.Ảnh nhiễu có thể được tạo bởi các lệnh sau:

```
noi = rand(256);
n2 = zeros(size(pep));
n2 = n2 + pep .* (noi >0.2) + 255 .* (noi<0.1);
```

a. Ảnh gốc b. Ảnh bị nhiễu muối tiêu

**Hình 58. Ví dụ về nhiễu muối tiêu.**

Dùng hàm figure và subplot, imshow để hiện cả hai hình này trên cùng Figure 1.

Phần mã báo cáo:

Phần Figure 1. báo cáo:



### 4.1.2. Lọc ảnh dùng bộ lọc trung vị

1. Viết hàm thực hiện lọc trung vị hai chiều bằng bộ lọc trung vị dạng 5 điểm đường chéo (5-point cross-shaped).

| | | |
|---|---|---|
| **X** | **0** | **X** |
| **0** | **X** | **0** |
| **X** | **0** | **X** |

Áp dụng bộ lọc này để triệt nhiễu ở Hình 59.b.

Phần mã báo cáo:

2. Tìm sai số trung bình bình phương giữa ảnh đã triệt nhiễu và ảnh gốc.

Kết quả báo cáo:

3. Hiện thêm hình đã triệt nhiễu trên hình Figure 1.

Phần Figure 1. báo cáo:

### 4.1.3. Lọc ảnh dùng bộ lọc trung vị có sẵn medfil2

1. Dùng hàm medfilt2 để lọc trung vị với các bộ lọc kích thước khác nhau: bên cạnh dạng chữ thập, sử dụng thêm các kích thước sau: 1x2 1x3 2x2 3x3 3x4 4x4 4x5 5x5 7x7.

Phần mã báo cáo:

2. Hiện các hình này trên cùng một Figure 2. để dễ so sánh chất lượng. Có thể thêm cả hình gốc và hình nhiễu để tiện so sánh. Gán tên cho các hình con này.

Phần Figure 2. báo cáo:

3. Nếu một lọc trung vị có thể lọc nhiễu hoàn hảo, sai số trung bình bình phương giữa ảnh gốc và ảnh lọc nhiễu sẽ bằng 0. Trong thực tế thì điều này không xảy ra. Giá trị sai số trung bình bình phương có thể được dùng như một phép đo đơn giản định tính chất lượng của bộ lọc trung vị. Vẽ giá trị sai số trung bình bình phương theo số điểm của bộ lọc cho các bộ lọc xét ở trên. Lưu ý hình vẽ Figure 3. nên bao gồm cả giá trị sai số trung bình bình phương giữa pep và n2, như giá trị chưa được lọc nhiễu.

Phần Figure 3. báo cáo:

4. Nhận xét kết quả.

Báo cáo nhận xét:

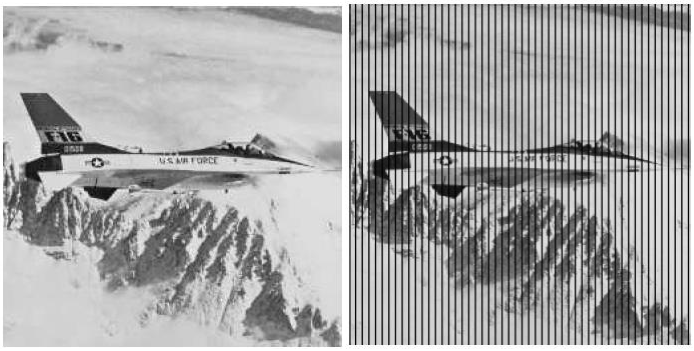


### 4.2. Giảm nhiễu tuần hoàn

#### 4.2.1. Tạo nhiễu tuần hoàn

1. Đọc ảnh gốc airplane.tif
2. Chuyển từ ảnh màu sang ảnh trắng đen dùng công thức Y= (R+G+B)/3
3. Hiện cả hình màu và hình trắng đen trên Figure 4.

4. Giảm chất lượng hình bằng nhiễu tuần hoàn: mỗi cột bội 5 được gán giá trị 0 như Hình 57.b.



a. Ảnh gốc b. Ảnh bị nhiễu

**Hình 59. Ví dụ về nhiễu tuần hoàn.**

Phần mã báo cáo:

Phần Figure 4. báo cáo:

### 4.2.2. Lọc nhiễu tuần hoàn

5 phương pháp lọc được đề xuất để thực hiện lọc nhiễu tuần hoàn này.

(a) Lọc trung vị theo chiều ngang 3 điểm (horizontal 3-point median filter)

(b) Lọc trung bình theo chiều ngang 3 điểm (horizontal 3-point mean filter)

(c) Lọc trung điểm theo chiều ngang 3 điểm (horizontal 3-point midpoint filter)

(d) Lọc trung vị 6 điểm như hình bên dưới. Ngõ ta tại trung tâm cửa sổ, là giá trị trung vị của 6 điểm của bộ lọc đánh dấu bởi điểm X. Lưu ý rằng pixel đang xét không được dùng cho tính toán trung vị trong phương pháp này.

| | | |
|---|---|---|
| X | | X |
| X | | X |
| X | | X |

(e) Lọc trung vị có trọng số 3x3 điểm với trọng số cho bởi ma trận sau

| | | |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 3 | 1 |
| 1 | 3 | 1 |

Ở mỗi phương pháp, chỉ những pixel ở cột bị nhiễu (mỗi 5 cột) sẽ được xử lý. Pixel ở những cột không bị nhiễu có giá trị giống như hình gốc và sẽ không bị tác động bởi bộ lọc.

1. Thực hiện lọc nhiễu sử dụng 5 phương pháp trên. Hiện kết quả hình sau lọc nhiễu trên Figure 5.

Phần mã báo cáo:

Figure 5. báo cáo:

2. Tính sai số trung bình bình phương (MSE) của từng phương pháp, bỏ qua sai số ở gần 4 biên do chèn thêm điểm 0 (zero-padding) tại 4 biên. Sắp xếp 5 phương pháp này theo chỉ số MSE từ cao đến thấp. Giải thích lý do.

Phần báo cáo và giải thích:

## 4.3. Làm nhòe và làm sắc nét hình

### 4.3.1. Làm nhòe hình

1. Đọc ảnh gốc airplane.tif
2. Áp dụng làm nhòe dùng bộ lọc thông thấp. Phép toán này được thực hiện bằng toán tử đại số bằng cách nhân mặt nạ trung bình 3x3 lên hình gốc

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

3. Hiện cả hình gốc và hình nhòe trên Figure 6.

Phần mã báo cáo:

Phần Figure 6. báo cáo:



### 4.3.2. Làm sắc nét hình

1. Đọc ảnh gốc airplane.tiff

2. Áp dụng tách thành phần tần số cao dùng bộ lọc thông cao. Phép toán này được thực hiện bằng toán tử đại số bằng cách nhân mặt nạ trung bình 3x3 lên hình gốc

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

3. Cộng hình gốc và hình tần số cao để tạo thành hình sắc nét.
4. Hiện cả hình gốc, hình tần số cao và hình sắc nét trên Figure 7.

Phần mã báo cáo:

Phần Figure 7. báo cáo:

5. Thực hiện lại bước 2 đến 4 với mặt nạ sau

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |

6. Cộng hình gốc và hình tần số cao để tạo thành hình sắc nét. Hiện cả hình gốc, hình tần số cao và hình sắc nét trên Figure 8.

Phần mã báo cáo:

Phần Figure 8. báo cáo:



7. Đánh giá so sánh độ sắc nét của hình sử dụng hai mặt nạ trên bằng cách hiện cả hình gốc và hình sắc nét dùng hai mặt nạ trên

Phần Figure 9. báo cáo:

Nhận xét đánh giá: