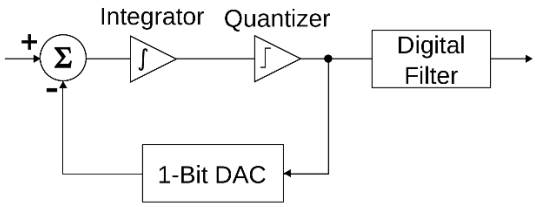
一、發聲原理

在本次 project 中我們利用ΔΣ調變來將 PCM 轉換為 PWM，因此先介紹一下ΔΣ調變：



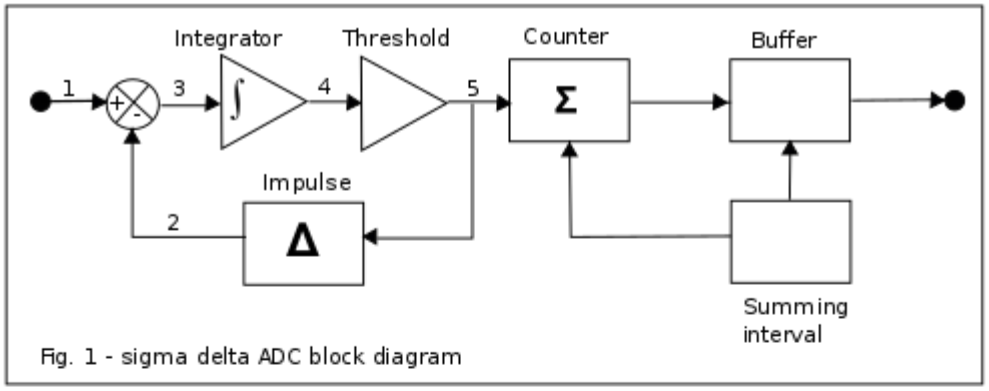
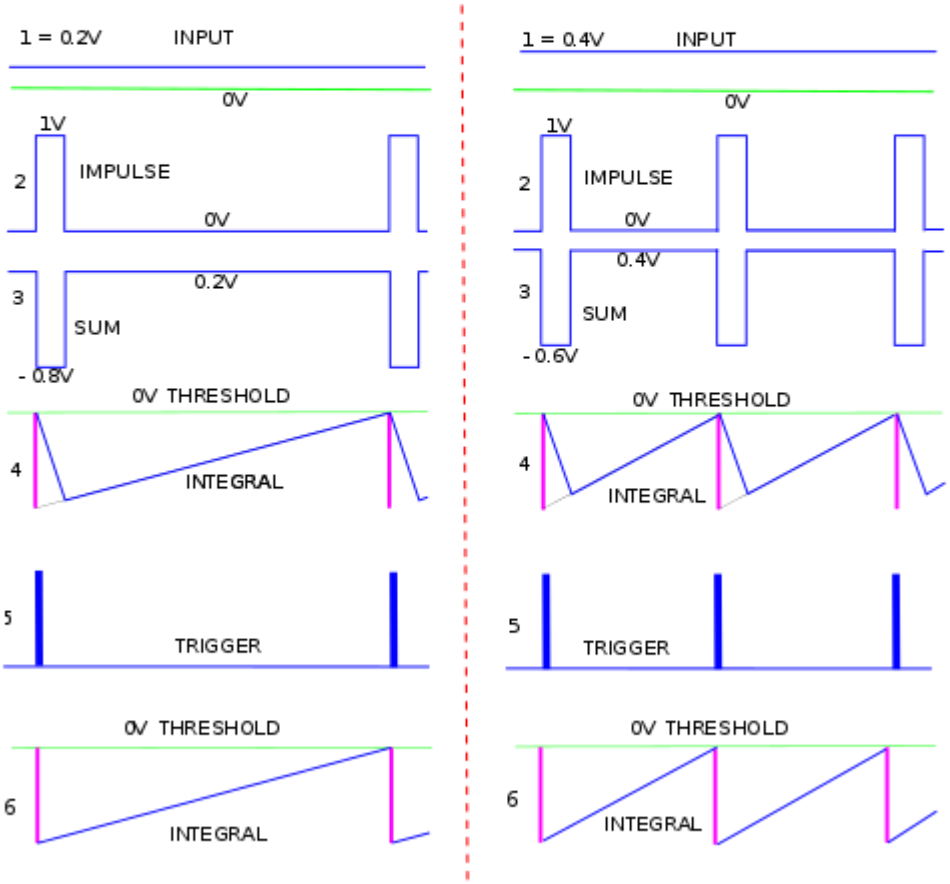


Fig. 1 - sigma delta ADC block diagram



Typical waveforms

ΔΣ調變是一種常用的 ADC，圖一為簡單的一階ΔΣ調變器，圖二則為其運作的大致流程：

圖二中的(1)為輸入的類比訊號，與脈衝(2)的差異值則會變成(3)，而(3)再經過積分器的累積成為(4)。

當(4)越過閾值時會產生脈衝(2)直到(4)下降到閾值以下。

我們實作的部分就為圖二的前半段，圖二的下半部分則顯示出 0.2V 以及 0.4V 的訊號如何被轉換為數位訊號。

在解釋如何實作ΔΣ調變之前，先解釋如何產生輸入的 PCM：

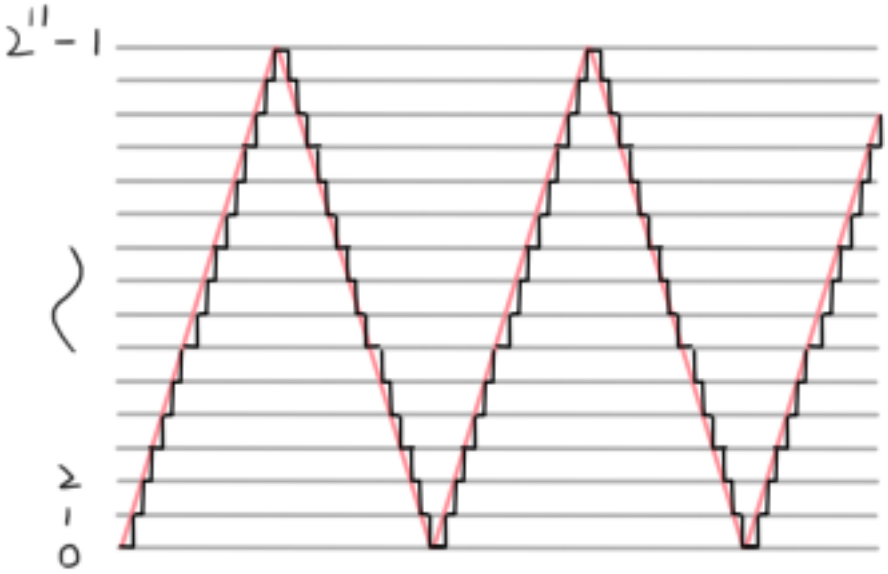
```
module gen_triangle (
    input clk,
    input reset,
    output reg [9:0] num,
    input on_off, // if this note is to be turned on or off
    input [7:0] note // 0x00~0x7F, 3C is Middle-C, 262 Hz.
    //"Each value is a 'half-step' above or below the adjacent values"
);
```

這是一個三角波的產生器(此外我們還有製作鋸齒波產生器)，我們輸出一個 10bits 的訊號，主要是利用 counter 來產生波形。

```
wire [31:0] freq;
wire [31:0] count_max = 32'd100_000_000/freq;
wire [31:0] count_duty = count_max / 32'd2048;
reg [31:0] count;
```

```
if (count < count_max) begin
    count <= count + 1;
    if (count%count_duty==0)begin
        if (count < count_max/2) begin
            num <= (num<10'b11_1111_1111) ? num + 1: num;
        end else begin
            num <= (num>10'b0) ? num - 1: num;
        end
    end
end else begin
    count <= 0;
    num <= 10'b0;
end
```

輸出的波型如下：



其週期為輸入的 freq。

由於我們的一個 track 最多會有 4 個音，因此最後輸入 PWM 產生器的數值為一個 12bits 的數字。

```
assign num = (on) ? (num_1 + num_2 + num_3 + num_4): 0;
```

至於 Δ Σ 調變的實作，主要是藉由此 module 來進行實作：

```
module PWM_gen_x (
    input clk,
    input reset,
    input [11:0] num_1,
    input [11:0] num_2,
    output reg PWM
);
```

他的實作相當簡單，我們將 2^11-1 作為閾值，並把 sav_num 作為積分容器：

```
reg [12:0] sav_num;

always @(posedge clk, posedge reset) begin
    if (reset) begin
        sav_num <= 0;
        PWM <= 0;
    end else begin
        if (sav_num>12'b1111_1111_1111) begin
            PWM <= 1;
            sav_num <= sav_num + num_1/8 + num_2/8 - 12'b1111_1111_1111;
        end else begin
            PWM <= 0;
            sav_num <= sav_num + num_1/8 + num_2/8;
        end
    end
end
```

每一個 num 之所以除以 8，是因為我們總共有兩個 track，每一個 track 都有 4 個音。

二、track 讀取

再往下另一個同學的部分會講解如何讀取 midi 檔，這邊的 track 讀取則是講解如何讀取 midi 輸出的訊號並產生該有的波型。

```
module gen_track_triangle(
    input clk,
    input reset,
    input on,
    input out_valid, // ignore the following if it's invalid
    input on_off, // if this note is to be turned on or off
    input [7:0] note,
    output wire [11:0] num
);
```

這是一個用來產生三角波的 track 讀取器，input on 是用來實作 stop 的部分。

當 out_valid==1 時，track 讀取器便會開始讀取，而 on_off 則是告知一個音符的開與關，note 為音高。

```
wire [9:0] num_1, num_2, num_3, num_4;
reg [3:0] select;
reg [7:0] note_1, note_2, note_3, note_4;
```

因為我們的一個 track 最多會有 4 個音，因此設置了 select(select 同時也是每個軌道的 on_off)來選擇現在的 on_off 是針對哪一個音。

```
if (out_valid) begin
    if (on_off) begin
        if (select[0]==0) begin
            select[0] <= 1'b1;
            note_1 <= note;
        end else if (select[1]==0) begin
```

如果 on_off==1，就利用 select 從 0~3 開始偵測哪一個軌道現在是空的。

```
 if (note_1==note) begin
     select[0] <= 1'b0;
     note_1 <= 8'b0;
 end
```

如果 on_off==0，則偵測哪一個軌道是目標 note。