**邏輯設計實驗Lab09結報**

**104060012邱怡庭**

**1 Please design an audio-data parallel-to-serial module to generate the speaker control signal with 40MHz system clock, 5MHz bit clock and (5/32) MHz stereo sampling clock.**

**1.1 Design a general frequency divider to generate the required frequencies for speaker clock.**

**1.2 Design a stereo signal parallel-to-serial processor to generate the speaker control signals. Please use verilog simulation waveform to verify your control signal.**

**Design Specification**

**output** : audio\_appsel, // playing mode selection

audio\_sysclk, // control clock for DAC (from crystal)

audio\_bck, // bit clock of audio data (5MHz)

audio\_ws, // left/right parallel to serial control

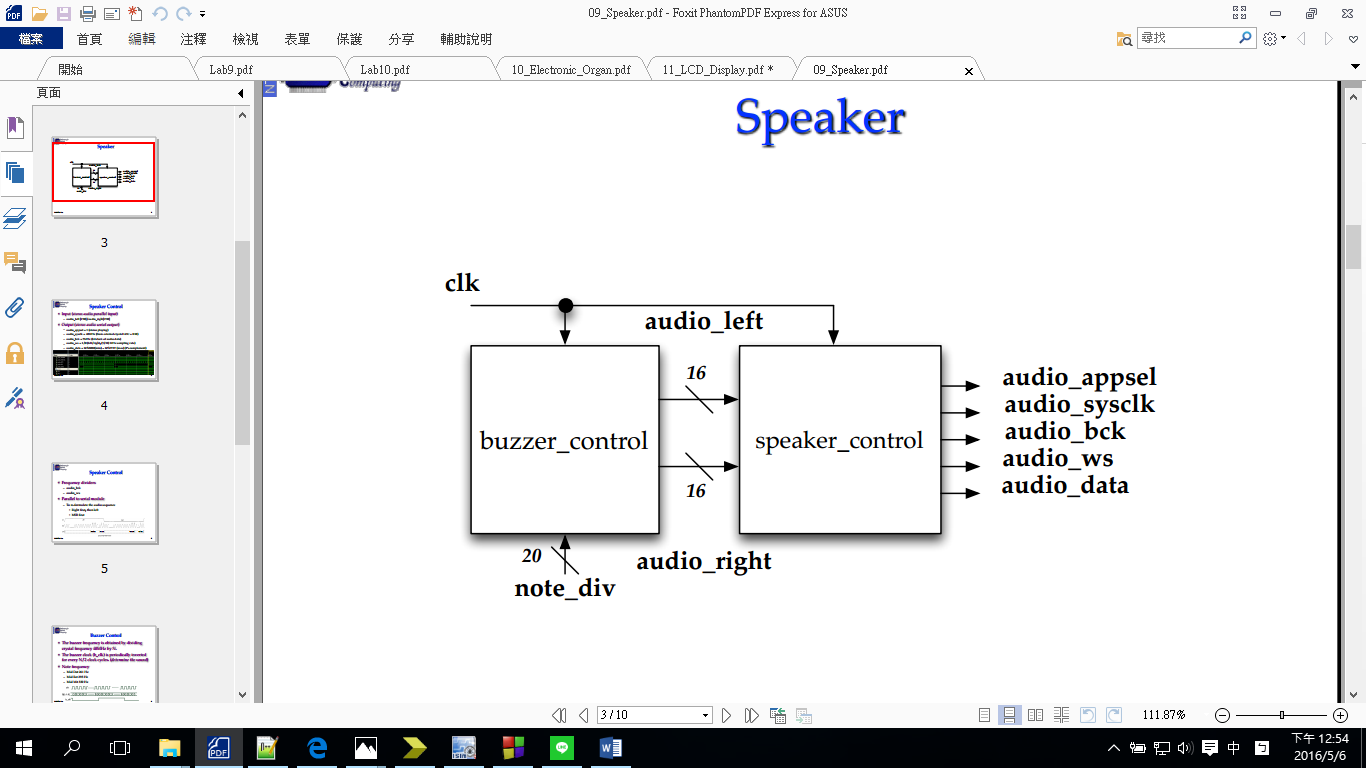
audio\_data

**input** : clk, // clock from crystal

rst\_n, // active low reset

**wire** : [15:0] audio\_in\_left, audio\_in\_right;

**block diagram:**

****

**Design Implementation**

**logic function / logic diagram**:

***speaker\_ctl:***

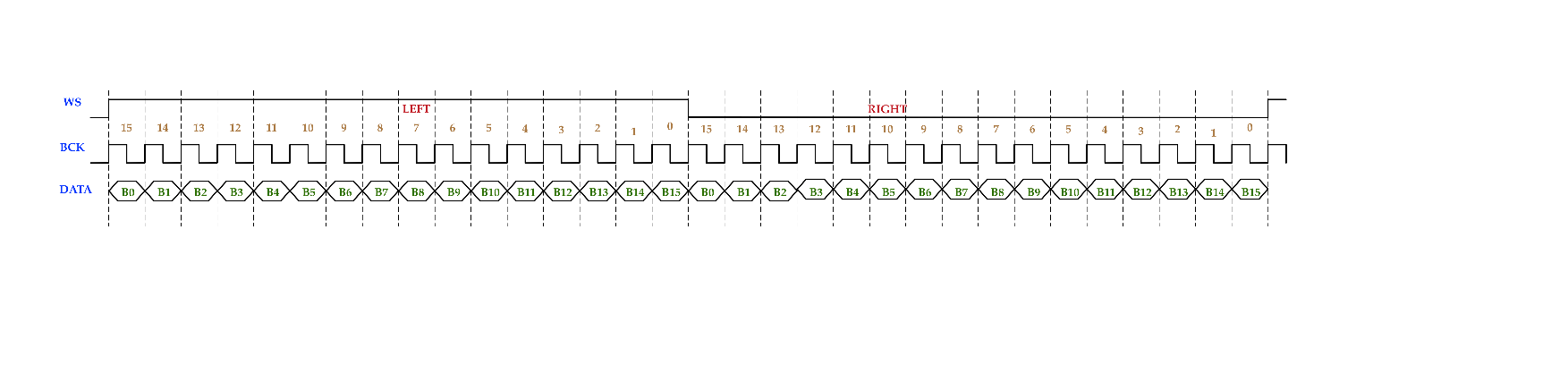
*•Frequency dividers*

*– audio\_bck*

*– audio\_ws*

*•Parallel to serial module*

*– To re-formulate the audio sequence*

*•Right first, then left*

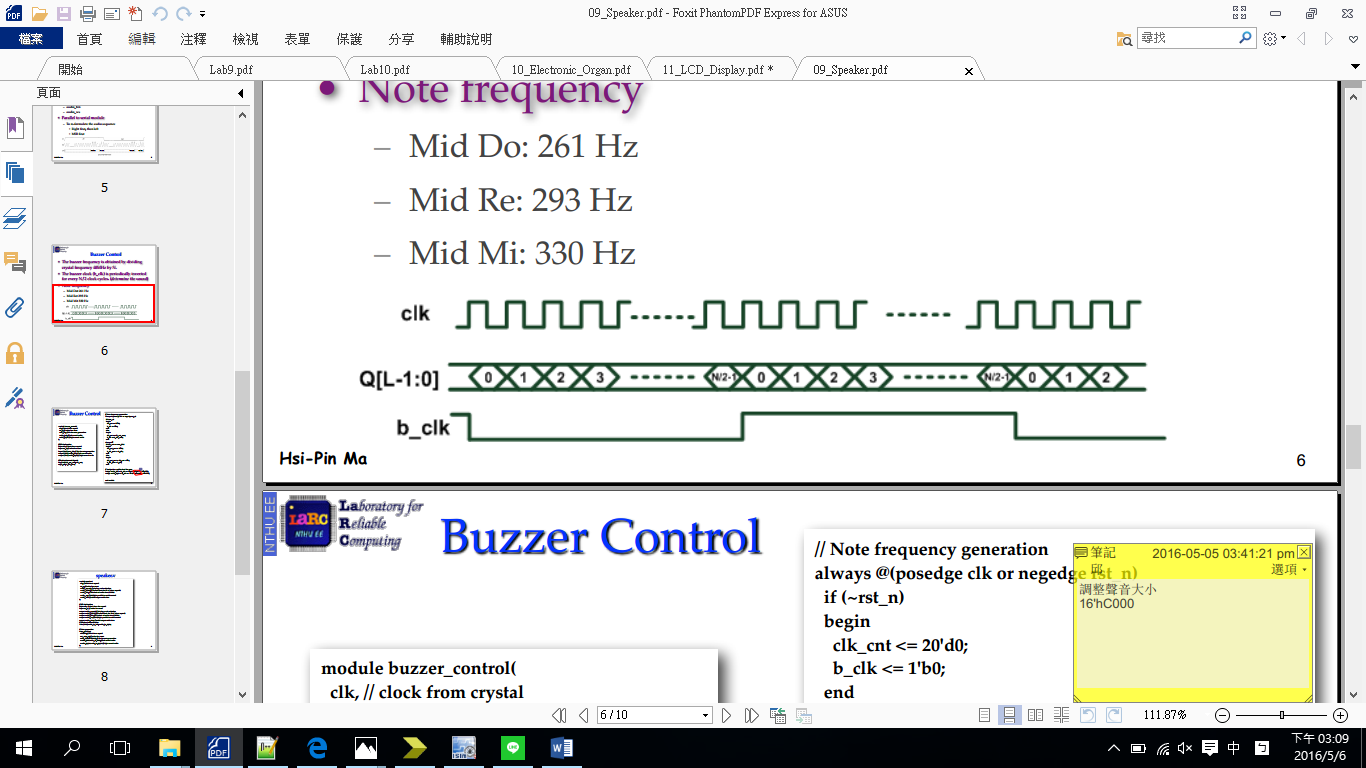
*•MSB first*

***buzzer\_ctl:***

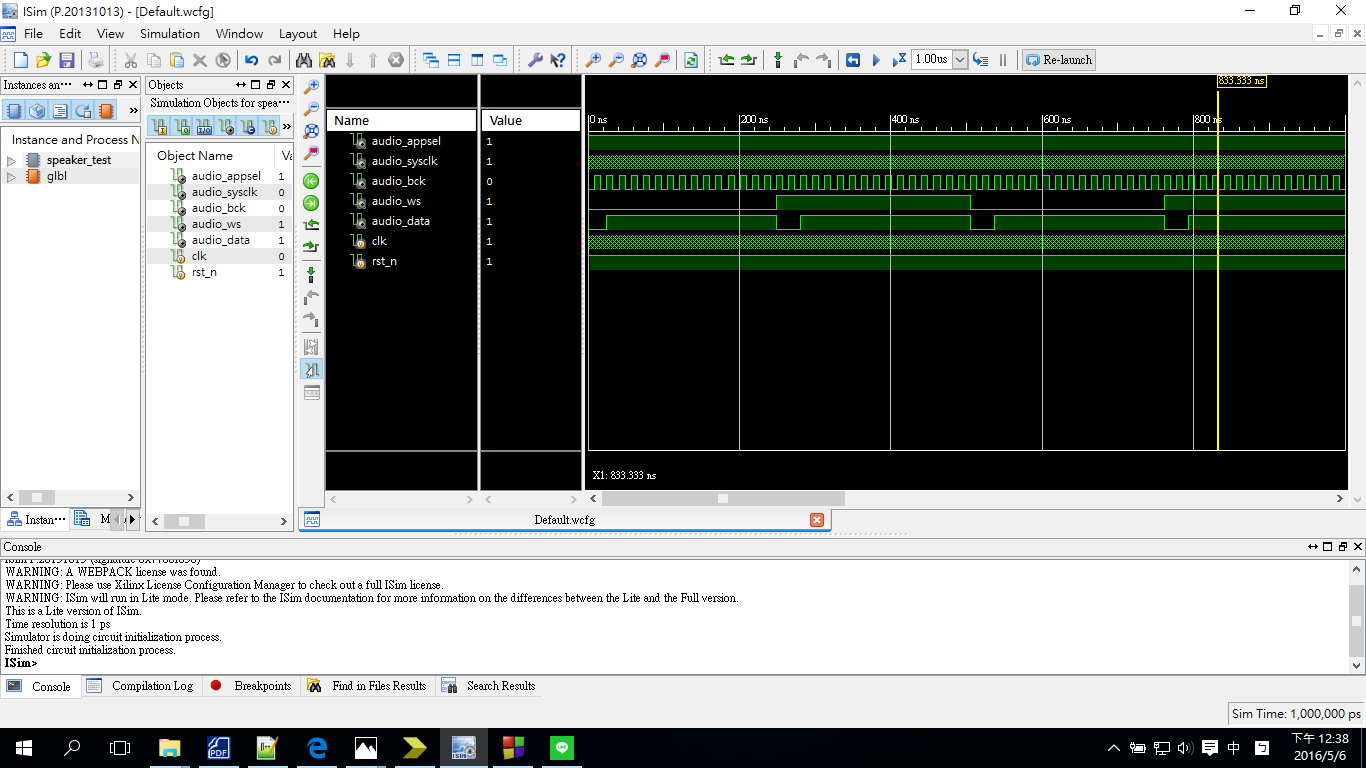
*•The buzzer frequency is obtained by dividing crystal frequency 40MHz by*

*N.*

*•The buzzer clock (b\_clk) is periodically inverted for every N/2 clock cycles. (determine the sound)*

**

**The Final Result:**

****

**Discussion:**

在speaker\_ctl中設置一個3bits 的counter，取第3個bit為*audio\_bck，使其輸出頻率為5MHz。當audio\_ws為0時存入right data，當audio\_ws為1時存入left data；並設置另一個4bits的counter，當數到第15時，使audio\_ws變為~audio\_ws。*

**2 Speaker control**

**2.1 Please produce the buzzer sounds of Do, Re, and Mi by pressing buttons (S1,S2,S3)respectively. When you press down the button, the speaker produces corresponding frequency sound. When you release the switch, the speaker stops the sound.**

**2.2 Please control the volumn of the sound by pressing button (S4) as increase and (S5) and decrease the volumn. Please also quantize the audio dynamic range as 16 levels and show the current sound level in the 14-segment display.**

**Design Specification**

**output** : audio\_appsel // playing mode selection

audio\_sysclk // control clock for DAC (from crystal)

audio\_bck // bit clock of audio data (5MHz)

audio\_ws // left/right parallel to serial control

audio\_data

[3:0] ftsd\_ctl

[14:0] display

**input** : clk // clock from crystal

rst\_n // active low reset

Do

Re

Mi

increase //for volumn

decrease //for volumn

**wire** : clk\_d;

[1:0] ftsd\_ctl\_en;

[15:0] volumn;

[4:0] level;

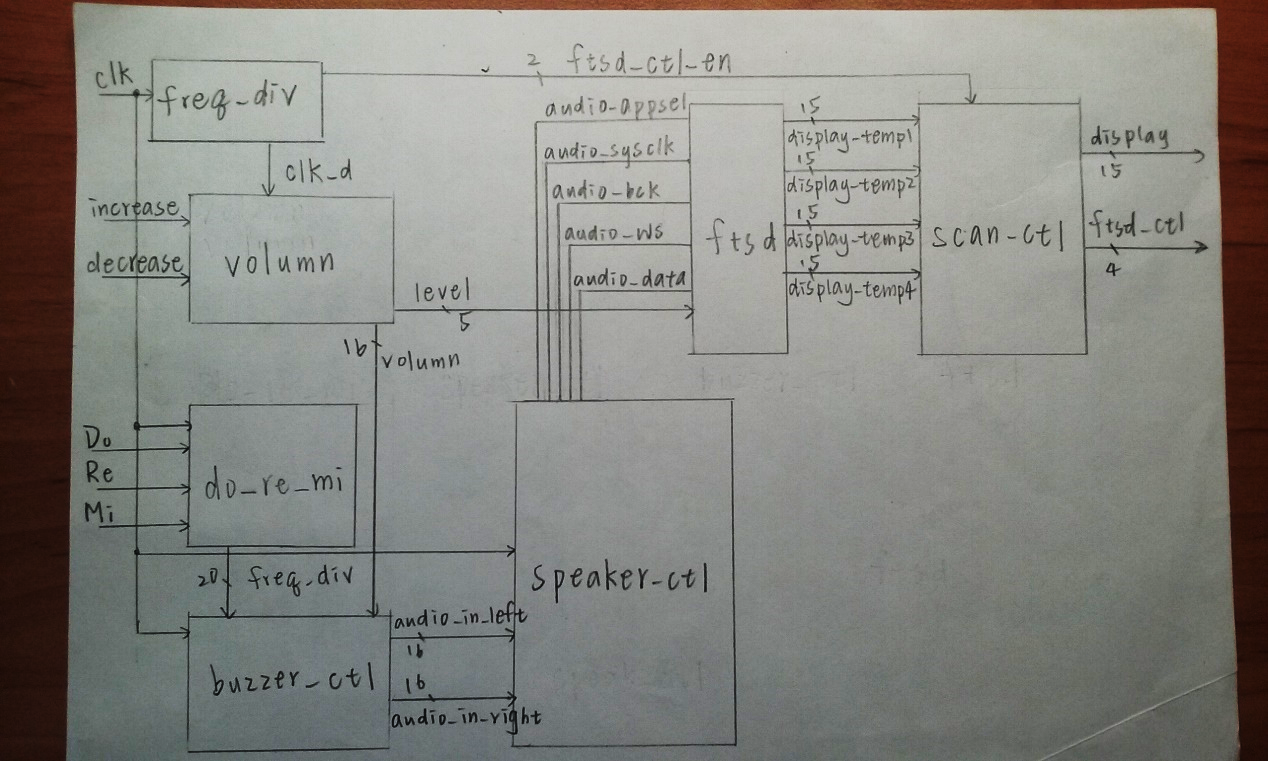
[19:0] freq\_div;

[15:0] audio\_in\_left;

[15:0] audio\_in\_right;

[14:0] display\_temp1,display\_temp2,display\_temp3,display\_temp4;

**block diagram:**



**Design Implementation**

**logic function / logic diagram**:

***speaker\_ctl:***

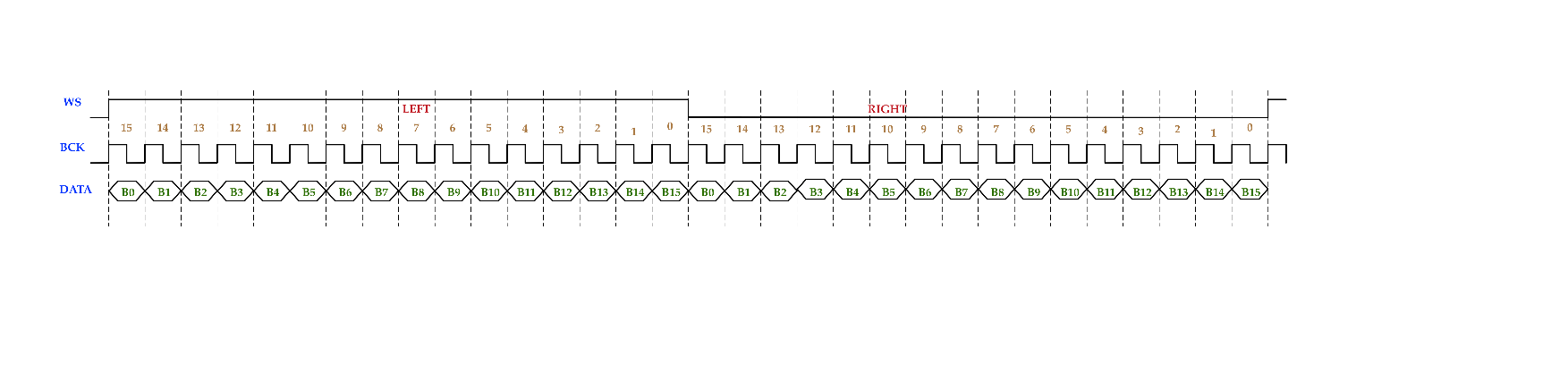
*•Frequency dividers*

*– audio\_bck*

*– audio\_ws*

*•Parallel to serial module*

*– To re-formulate the audio sequence*

*•Right first, then left*

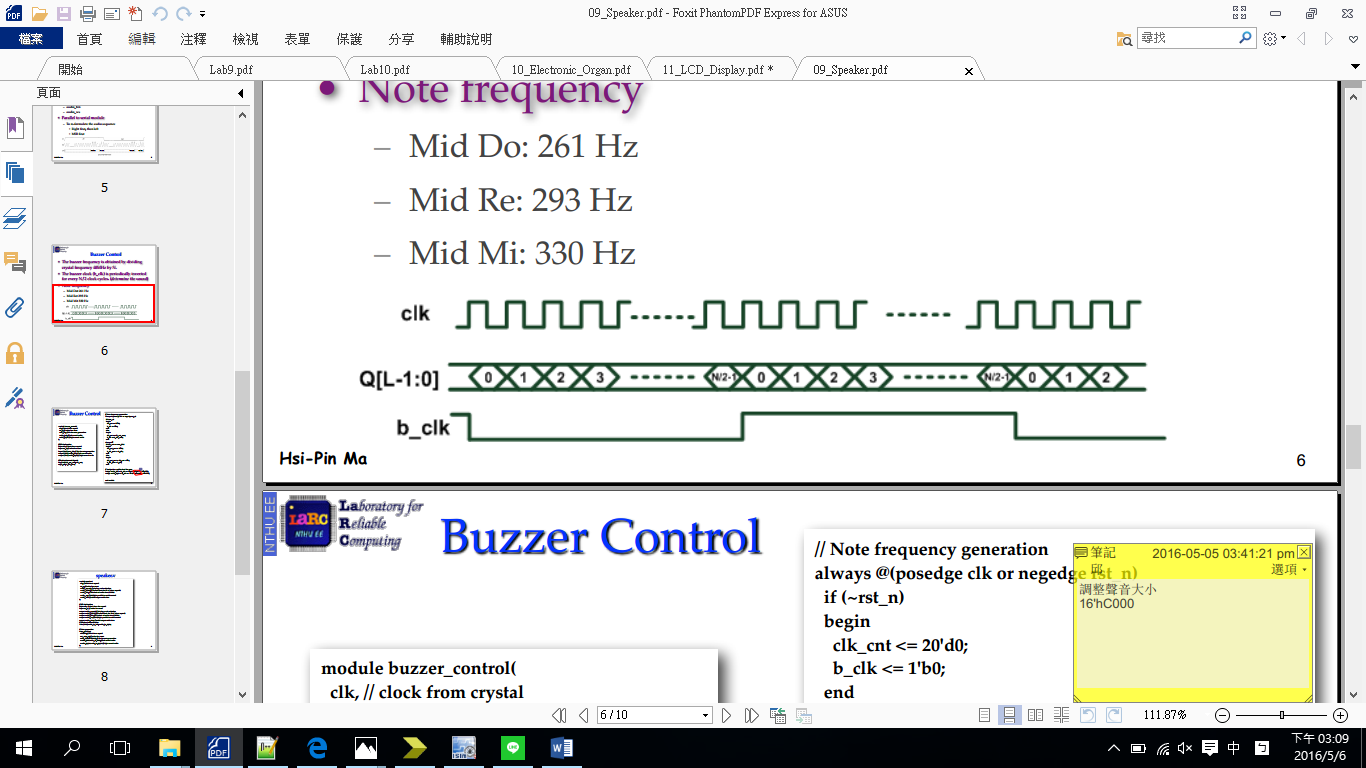
*•MSB first*

***buzzer\_ctl:***

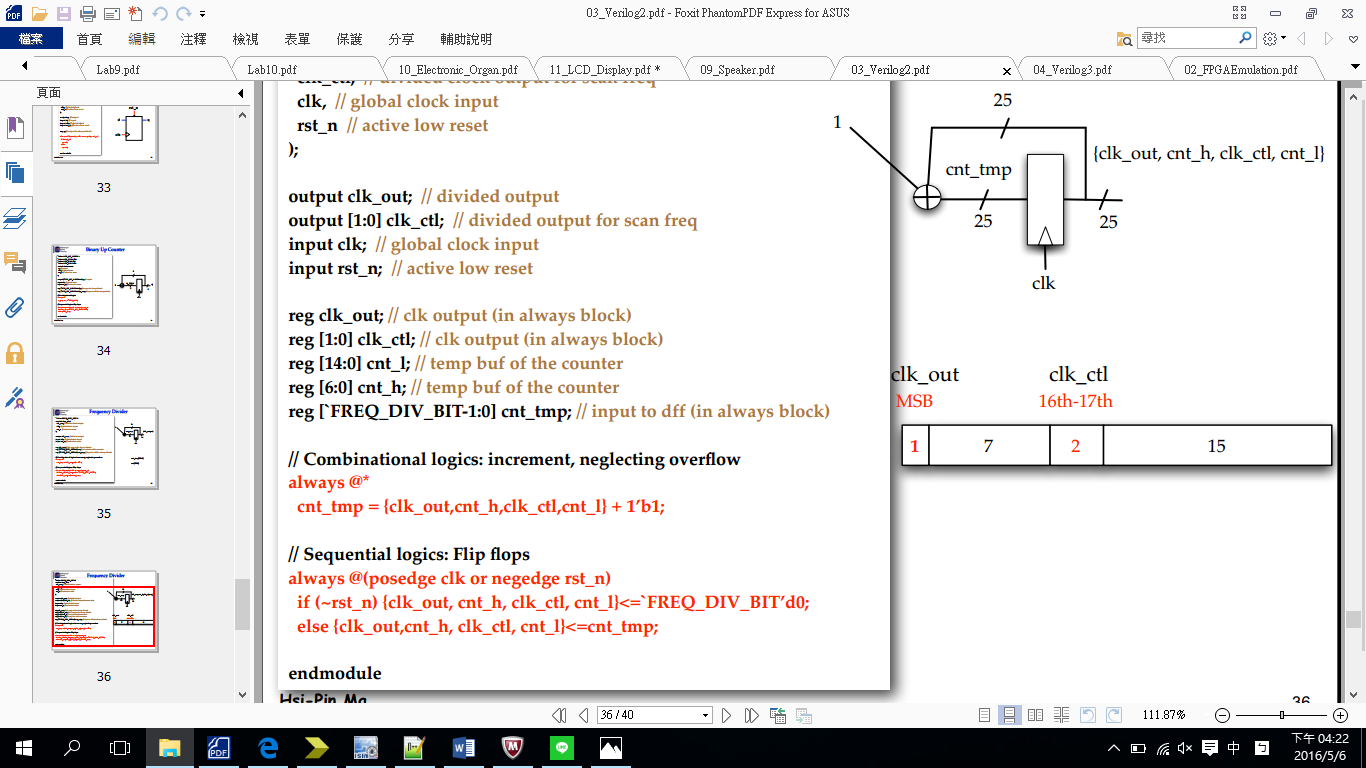
*•The buzzer frequency is obtained by dividing crystal frequency 40MHz by*

*N.*

*•The buzzer clock (b\_clk) is periodically inverted for every N/2 clock cycles. (determine the sound)*

**

***freq\_div:***

**

***scan\_ctl***

if ftsd\_ctl\_en=00

->控制第一個14段顯示器，顯示第一種狀況

if ftsd\_ctl\_en=01

->控制第二個14段顯示器，顯示第二種狀況

if ftsd\_ctl\_en=10

->控制第三個14段顯示器，顯示第三種狀況

if ftsd\_ctl\_en=11

->控制第四個14段顯示器，顯示第四種狀況

***ftsd***

*當level大於9，個位數in1與十位數in2分別控制ftsd*

*當bcd=4'd0: display = 15'b0000\_0011\_1111\_111; //0*

*當bcd=4'd1: display = 15'b1111\_1111\_1011\_011; //1*

*當bcd=4'd2: display = 15'b0010\_0100\_1111\_111; //2*

*當bcd=4'd3: display = 15'b0000\_1100\_1111\_111; //3*

*當bcd=4'd4: display = 15'b1001\_1000\_1111\_111; //4*

*當bcd=4'd5: display = 15'b0100\_1000\_1111\_111; //5*

*當bcd=4'd6: display = 15'b0100\_0000\_1111\_111; //6*

*當bcd=4'd7: display = 15'b0001\_1111\_1111\_111; //7*

*當bcd=4'd8: display = 15'b0000\_0000\_1111\_111; //8*

*當bcd=4'd9: display = 15'b0000\_1000\_1111\_111; //9*

*default: display = 15'b1111\_1111\_1111\_111; //DEF*

***do\_re\_mi***

*Mid Do: 261 Hz*

*Mid Re: 293 Hz*

*Mid Mi: 330 Hz*

***volumn\_ctl***

*用increase和decrease控制level大小，並將16種level對應到16種不同的聲音大小。*

**I/O pin assignment:**

NET "clk" LOC=R10;

NET "rst\_n" LOC=N3;

NET "Do" LOC=P4;

NET "Re" LOC=P3;

NET "Mi" LOC=L6;

NET "increase" LOC=U1;

NET "decrease" LOC=T2;

NET "audio\_appsel" LOC=H18;

NET "audio\_bck" LOC=K16;

NET "audio\_sysclk" LOC=H17;

NET "audio\_ws" LOC=L15;

NET "audio\_data" LOC=L16;

NET "display[14]" LOC = P6;

NET "display[13]" LOC = N4;

NET "display[12]" LOC = V5;

NET "display[11]" LOC = T5;

NET "display[10]" LOC = U7;

NET "display[9]" LOC = R3;

NET "display[8]" LOC = N5;

NET "display[7]" LOC = R5;

NET "display[6]" LOC = T3;

NET "display[5]" LOC = T4;

NET "display[4]" LOC = V4;

NET "display[3]" LOC = V7;

NET "display[2]" LOC = R7;

NET "display[1]" LOC = T7;

NET "display[0]" LOC = U5;

NET "ftsd\_ctl<0>" LOC = V8;

NET "ftsd\_ctl<1>" LOC = U8;

NET "ftsd\_ctl<2>" LOC = V6;

NET "ftsd\_ctl<3>" LOC = T6;

**Discussion:**

改變頻率挑整音高，改變震幅調整音量。

**Conclusion:**

一開始對於頻率轉換有些摸不著頭緒，只要了解buzzer的運作，再配合不同state的控制，即可產生有趣音階變化。