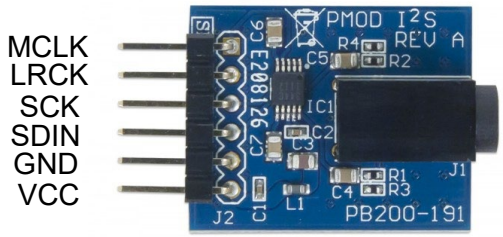
Speaker

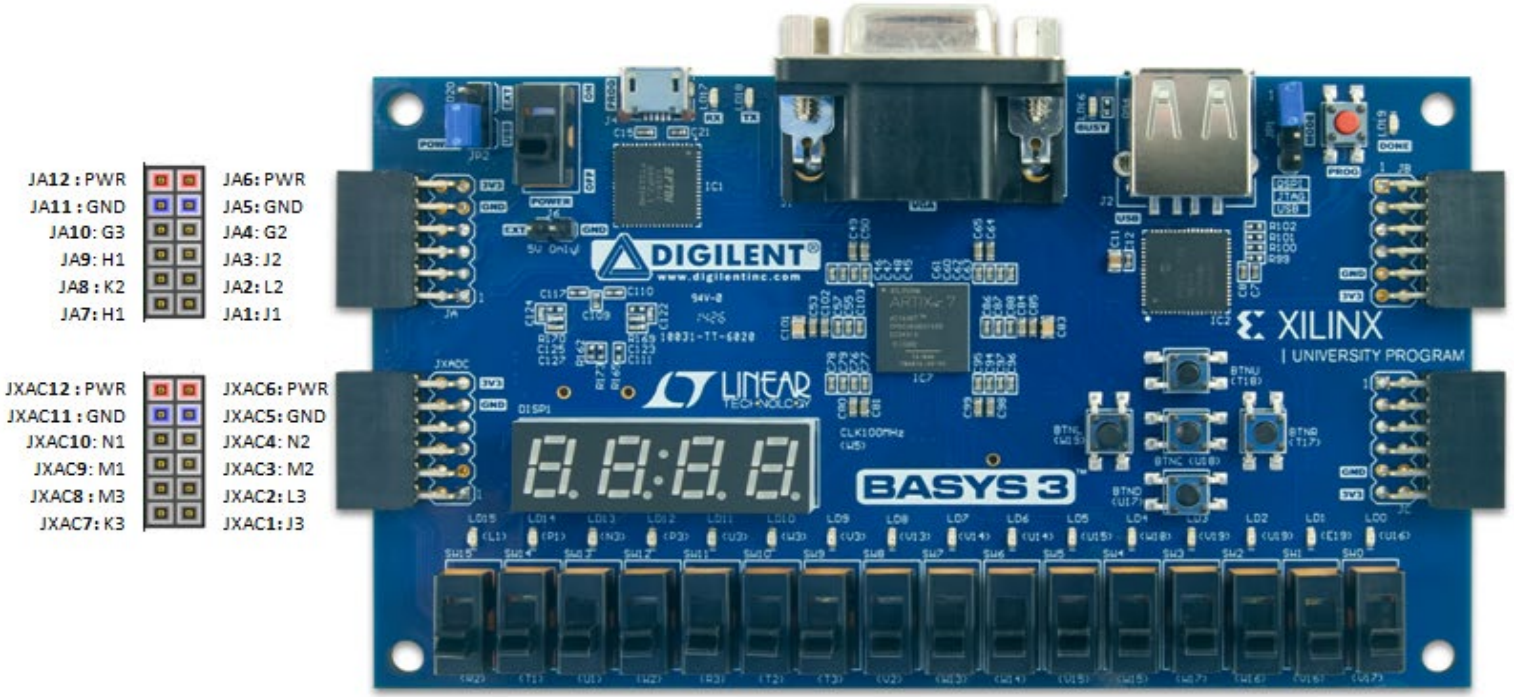
黃元豪
Yuan-Hao Huang

國立清華大學電機工程學系
Department of Electrical Engineering
National Tsing-Hua University

Pmod I2S: Stereo Audio Output



Basys3: Pmod Pin-Out Diagram



upper row

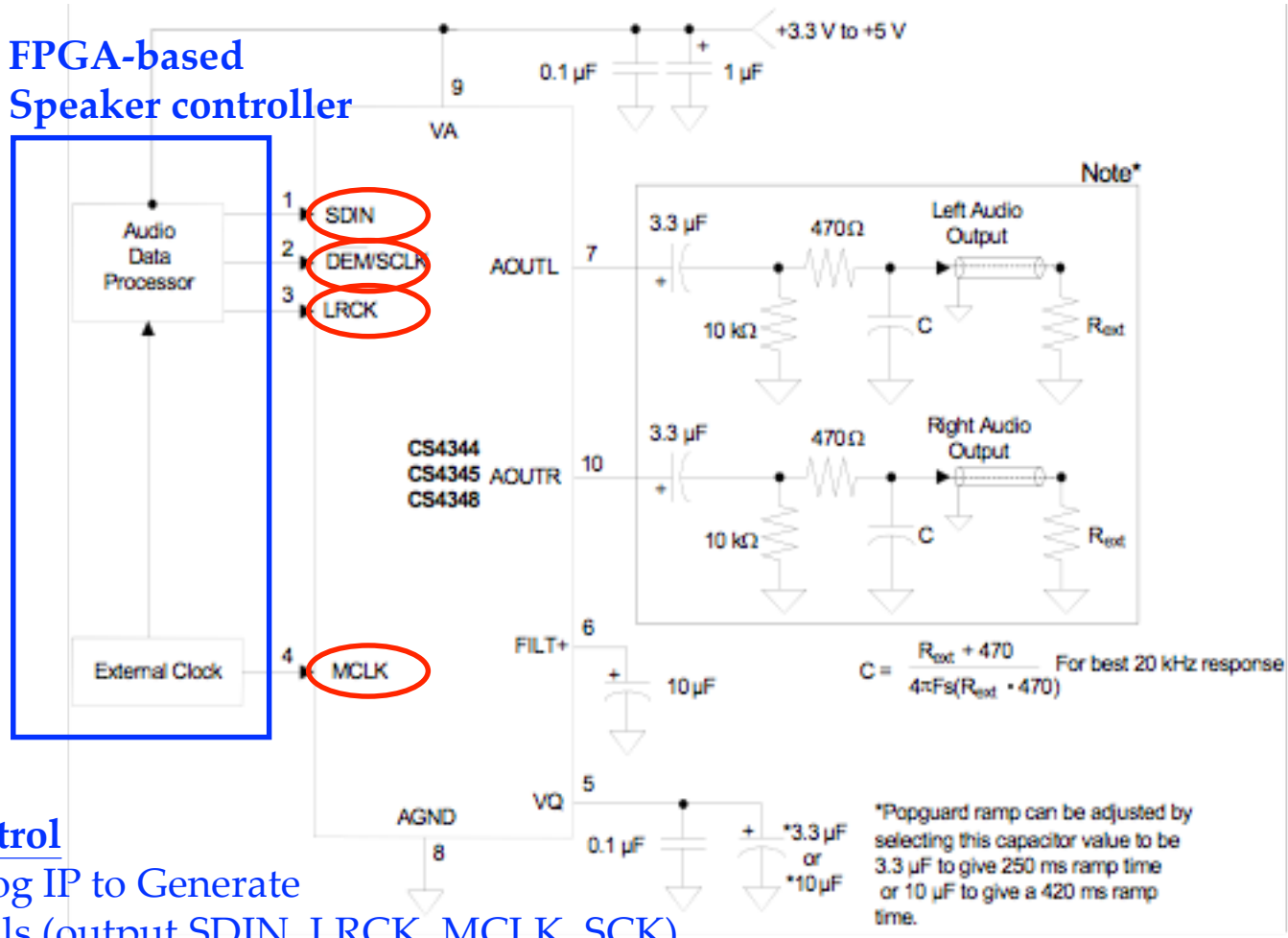
JB1: A14	JB7: A15
JB2: A16	JB8: A17
JB3: B15	JB9: C15
JB4: B16	JB10: C16
JB5: GND	JB11: GND
JB6: PWR	JB12: PWR

lower row

JC1: K17	JC7: L17
JC2: M18	JC8: M19
JC3: N17	JC9: P17
JC4: P18	JC10: R18
JC5: GND	JC11: GND
JC6: PWR	JC12: PWR

Speaker Controller

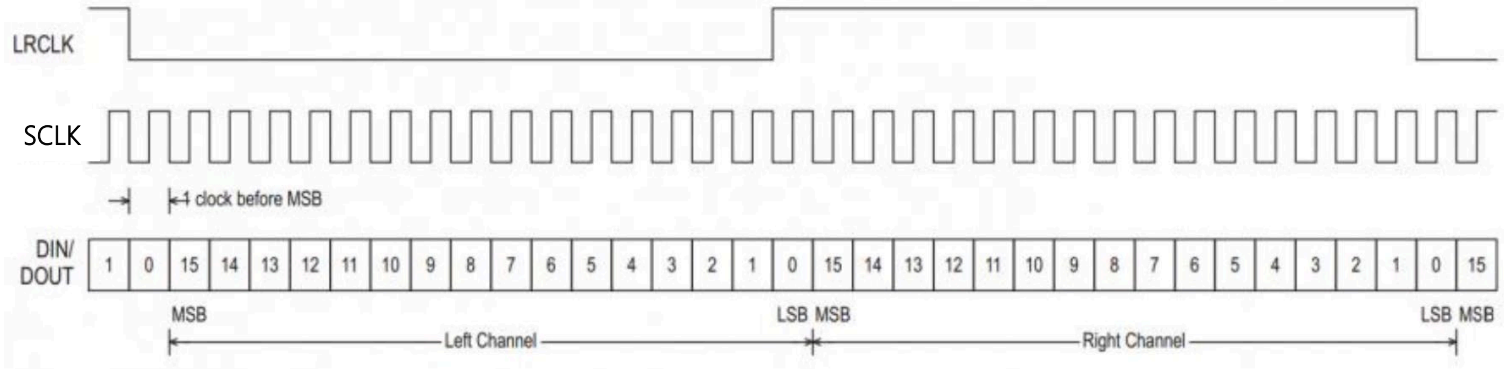
- Control DAC chip (digital to analog converter) CS4344



Speaker Control

Coding Verilog IP to Generate Control signals (output SDIN, LRCK, MCLK, SCK)

Speaker Control




```
set_property PACKAGE_PIN A14 [get_ports {audio_mclk}]
set_property PACKAGE_PIN A16 [get_ports {audio_lrck}]
set_property PACKAGE_PIN B15 [get_ports {audio_sck}]
set_property PACKAGE_PIN B16 [get_ports {audio_sdin}]
```

Pin	Signal	Description
1	MCLK	Master Clock
2	LRCK	Left-right Clock
3	SCK	Serial Clock
4	SDIN	Serial Data input
5	GND	Power Supply Ground
6	VCC	Positive Power Supply

Speaker

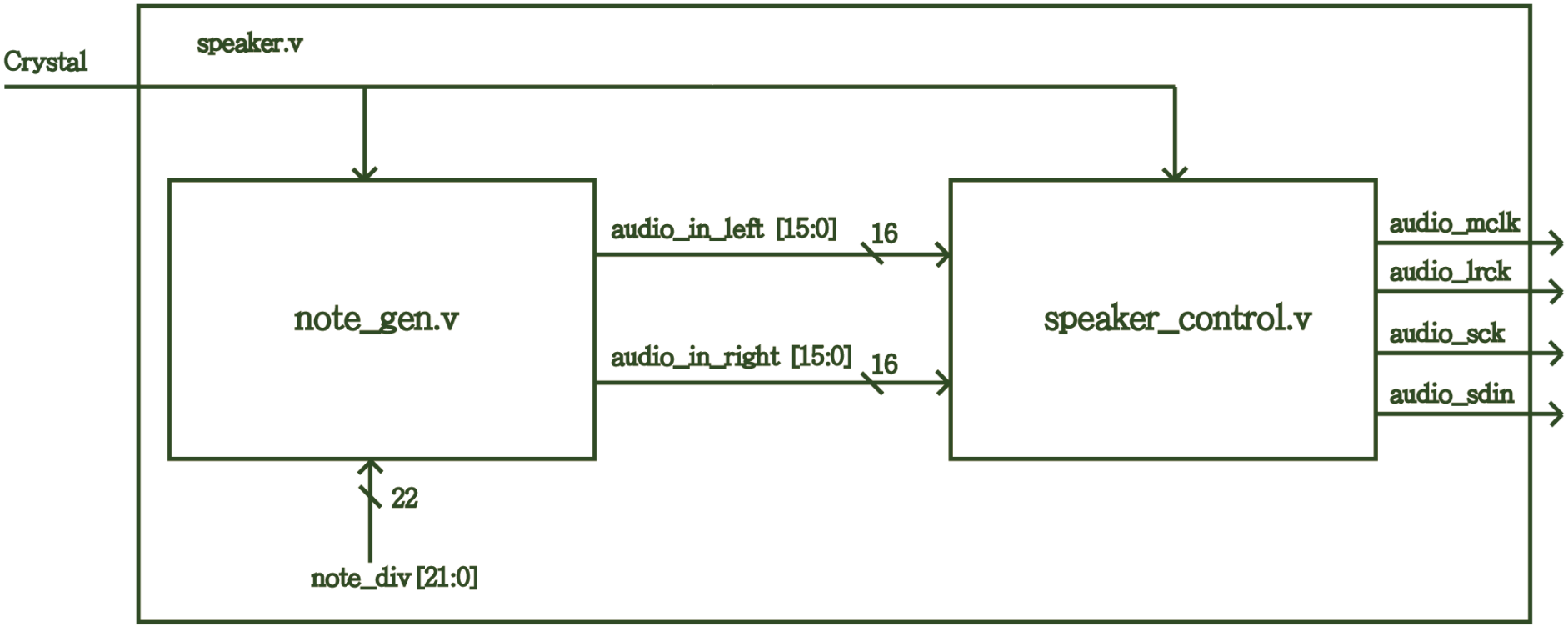
- Control of DAC (digital to analog converter) CS4344
  - LRCK (Left-Right Clock, or Word Select (WS) Clock, or Sample Rate (Fs) Clock) controls the sequence (left or right) of the serial stereo output
  - MCLK (Master Clock) synchronizes the audio data transmission
  - MCLK/LRCK must be an integer ratio 128
  - Serial Clock (SCK) controls the shifting of data into the input data buffers (32*Fs) 25MHz/128*32 = 25MHz/4

25MHz/128 (~192kHz)

25MHz (~24.5760MHz)

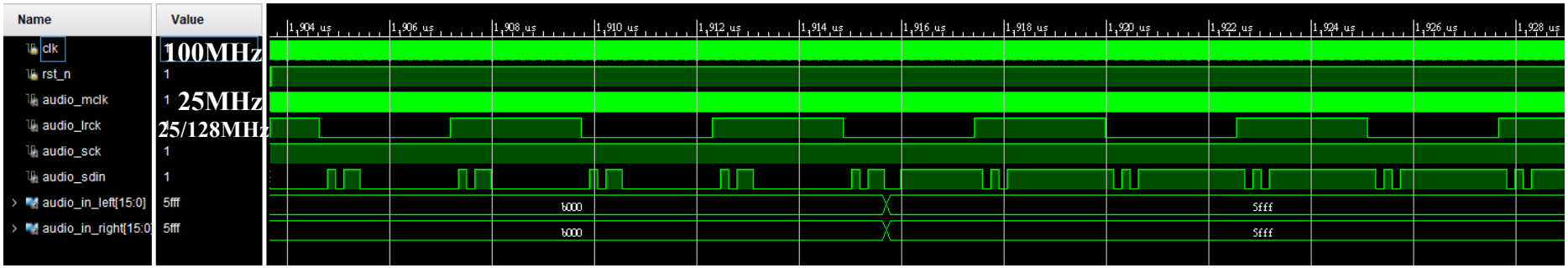
LRCK (kHz)	MCLK (MHz)									
	64x	96x	128x	192x	256x	384x	512x	768x	1024x	1152x
32	-	-	-	-	8.1920	12.2880	-	-	32.7680	36.8640
44.1	-	-	-	-	11.2896	16.9344	22.5792	33.8680	45.1580	-
48	-	-	-	-	12.2880	18.4320	24.5760	36.8640	49.1520	-
64	-	-	8.1920	12.2880	-	-	32.7680	49.1520	-	-
88.2	-	-	11.2896	16.9344	22.5792	33.8680	-	-	-	-
96	-	-	12.2880	18.4320	24.5760	36.8640	-	-	-	-
128	8.1920	12.2880	-	-	32.7680	49.1520	-	-	-	-
176.4	11.2896	16.9344	22.5792	33.8680	-	-	-	-	-	-
192	12.2880	18.4320	24.5760	36.8640	-	-	-	-	-	-
Mode	QSM				DSM		SSM			

Speaker Control Module



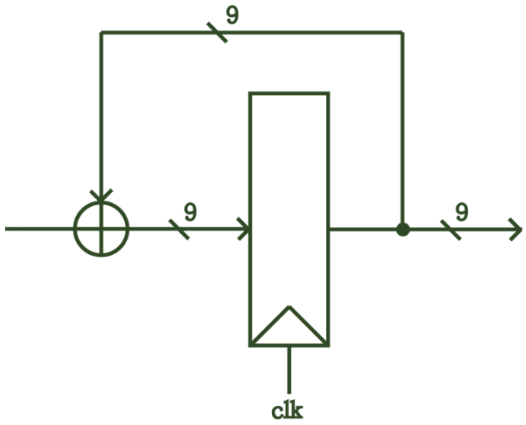
Speaker Control

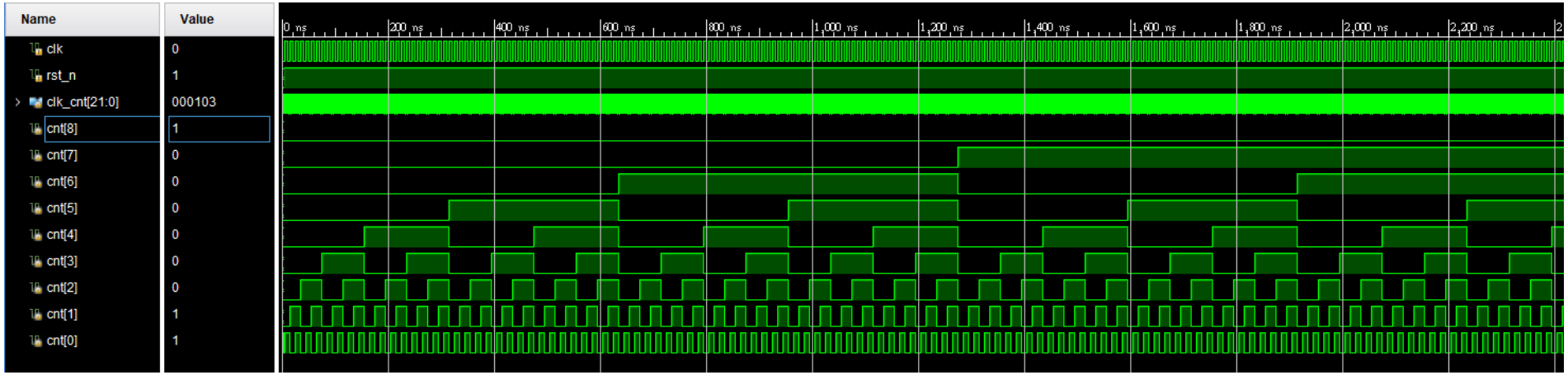
- Stereo audio parallel input
  - audio_in_left [15:0] / audio_in_right [15:0]
- Stereo audio serial output audio_mclk = 25MHz (divided from external crystal PORT=W5)
  - audio_lrck = (25/128)MHZ (left/right Sampling rate)
  - audio_sdin = 16'hB000 (min)~16'h5FFF (max) (two's complement)



Clock Frequency divider (Freq/(2^n))

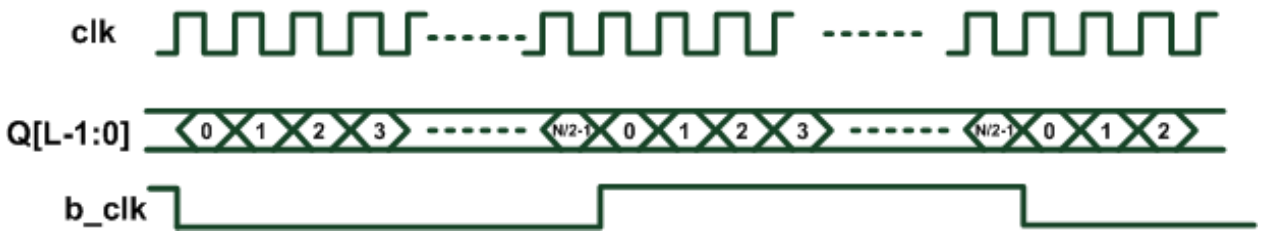
- Free-run n-bit binary counter
  - Speaker control signal can be easily generated by free-run binary counter



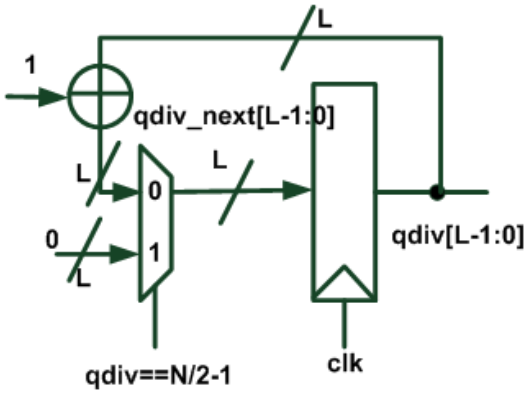


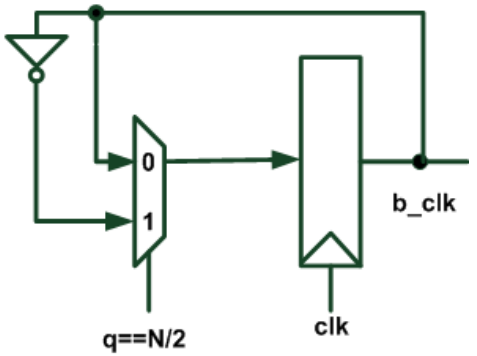
Clock Frequency divider (Freq/N)

- The buzzer frequency is obtained by dividing crystal 100 MHz by N.
- The buzzer clock (note_div) is periodically inverted for every N/2 clock cycles.
  - Master clk frequency = 100 (MHz) ➔ clk period = 10(ns)
  - Buzzer period = 10xN (ns) ➔ Buzzer frequency = 100/N (MHz)
    - "mid Do" sound frequency = 261 Hz
    - "mid Re" sound frequency = 293 Hz
    - "mid Mi" sound frequency = 330 Hz



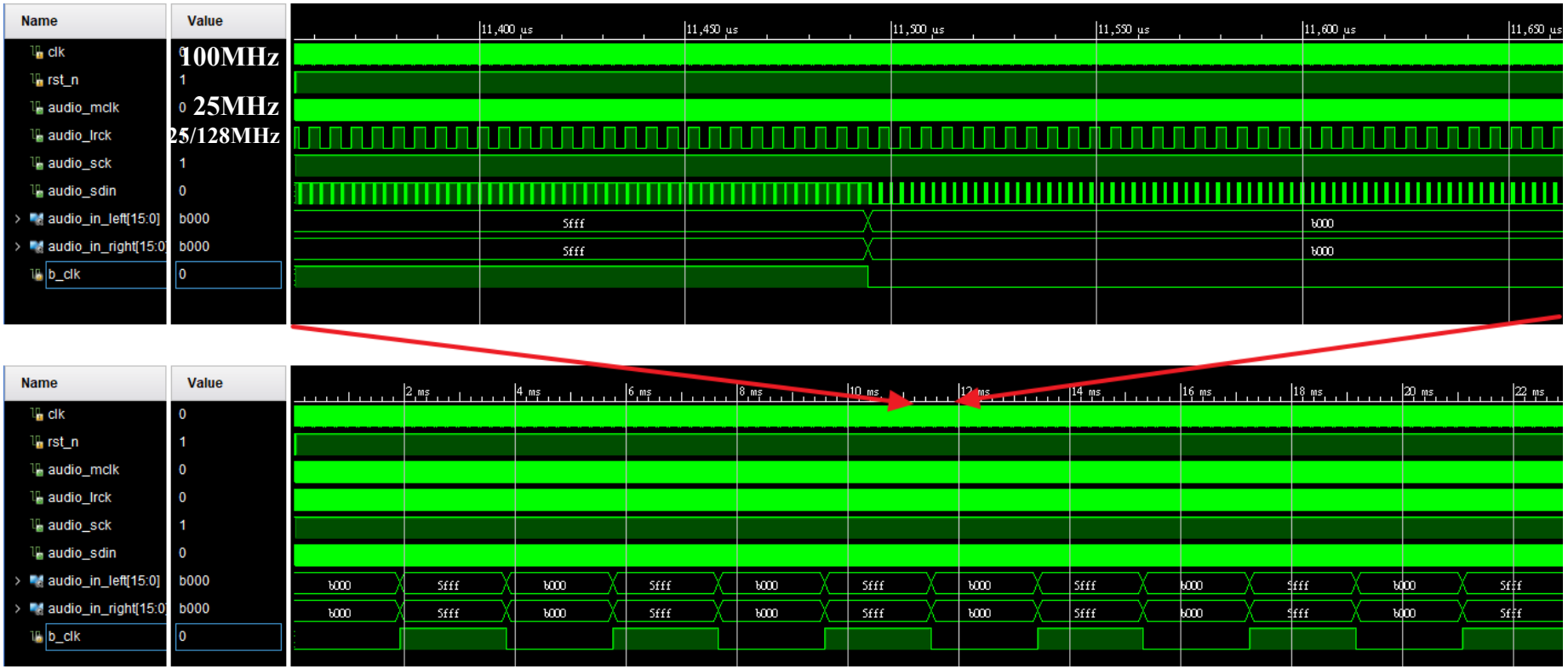
$$2^L > N$$





Buzzer Frequency

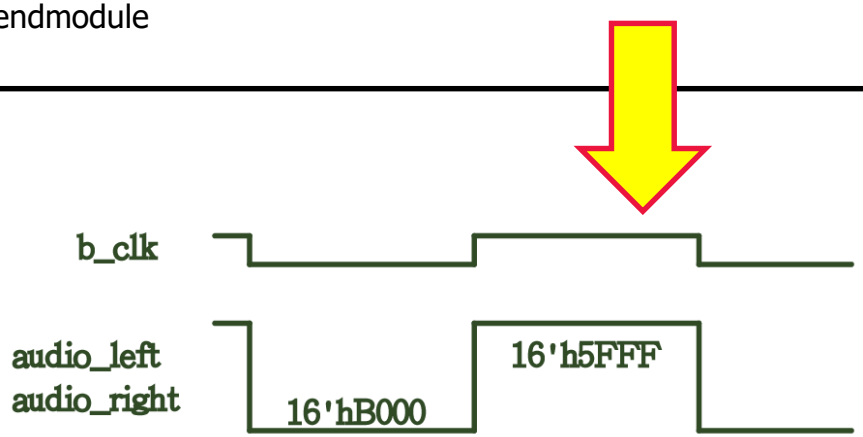
- The buzzer frequency (b_clk) determines the tone of the sound.



note_gen.v

```
module note_gen(  clk, // clock from crystal
                  rst_n, // active low reset
                  note_div, // div for note generation
                  audio_left, // left sound audio
                  audio_right // right sound audio
                );

// I/O declaration
input clk; // clock from crystal
input rst_n; // active low reset
input [21:0] note_div; // div for note generation
output [15:0] audio_left; // left sound audio
output [15:0] audio_right; // right sound audio

// Declare internal signals
reg [21:0] clk_cnt_next, clk_cnt;
reg b_clk, b_clk_next; // Note frequency generation

always @(posedge clk or negedge rst_n)
   if (~rst_n)  begin
      clk_cnt <= 22'd0;
      b_clk <= 1'b0;
   end  else  begin
      clk_cnt <= clk_cnt_next;
      b_clk <= b_clk_next;
   end
```

```
always @*
   if (clk_cnt == note_div)  begin
      clk_cnt_next = 22'd0;
      b_clk_next = ~b_clk;
   end  else  begin
      clk_cnt_next = clk_cnt + 1'b1;
      b_clk_next = b_clk;
   end

// Assign the amplitude of the note
assign audio_left = (b_clk == 1'b0) ? 16'hB000 : 16'h5FFF;
assign audio_right = (b_clk == 1'b0) ? 16'hB000 : 16'h5FFF;

endmodule
```



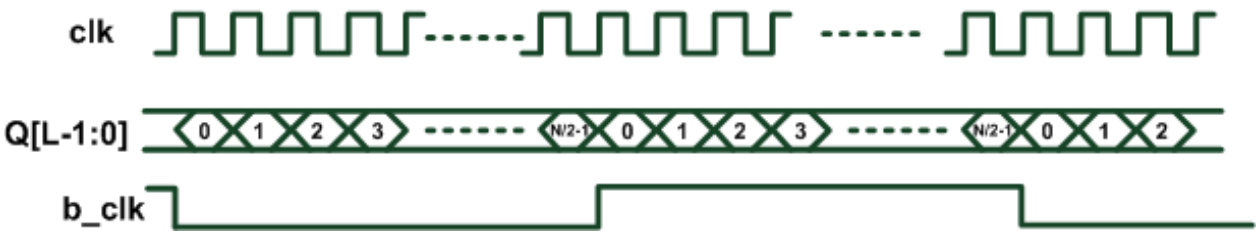
Dynamic Range 16'h5FFF~16'hB000

speaker.v

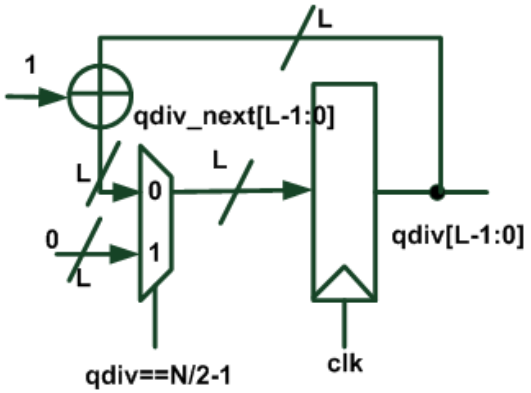
```
module speaker(  clk, rst_n, audio_mclk, audio_lrck, audio_sck, audio_sdin);
// I/O declaration
input clk;  // clock from the crystal
input rst_n;  // active low reset
output audio_mclk; // master clock
output audio_lrck; // left-right clock
output audio_sck; // serial clock
output audio_sdin; // serial audio data input

// Declare internal nodes
wire [15:0] audio_in_left, audio_in_right; // Note generation
note_gen Ung ( .clk(clk), // clock from crystal
                    .rst_n(rst_n), // active low reset
                    .note_div(22'd191571), // div for note generation
                    .audio_left(audio_in_left), // left sound audio
                    .audio_right(audio_in_right) // right sound audio
                  );
// Speaker controllor
speaker_control Usc ( .clk(clk),  // clock from the crystal
                           .rst_n(rst_n),  // active low reset
                           .audio_in_left(audio_in_left), // left channel audio data input
                           .audio_in_right(audio_in_right), // right channel audio data input
                           .audio_mclk(audio_mclk), // master clock
                           .audio_lrck(audio_lrck), // left-right clock
                           .audio_sck(audio_sck), // serial clock
                           .audio_sdin(audio_sdin) // serial audio data input
                         );
endmodule
```

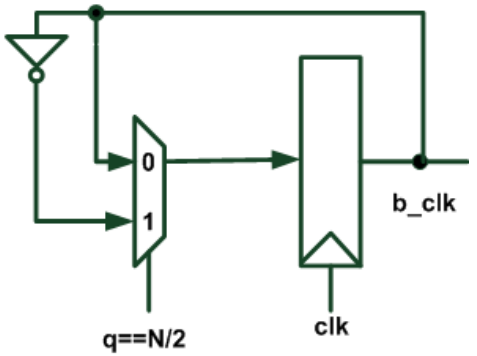
Tone Generation using Clock Frequency divider

- The buzzer frequency is obtained by dividing crystal 40MHz by N.
- The buzzer clock (b_clk) is periodically inverted for every N/2 clock cycles.
  - clk frequency = 40 (MHz) ➔ clk period = 25 (ns)
  - Buzzer period = 25xN (ns) ➔ Buzzer period = 40/N (MHz)
    - "mid Do" sound frequency = 261 Hz
    - "mid Re" sound frequency = 293 Hz
    - "mid Mi" sound frequency = 330 Hz



$$2^L > N$$



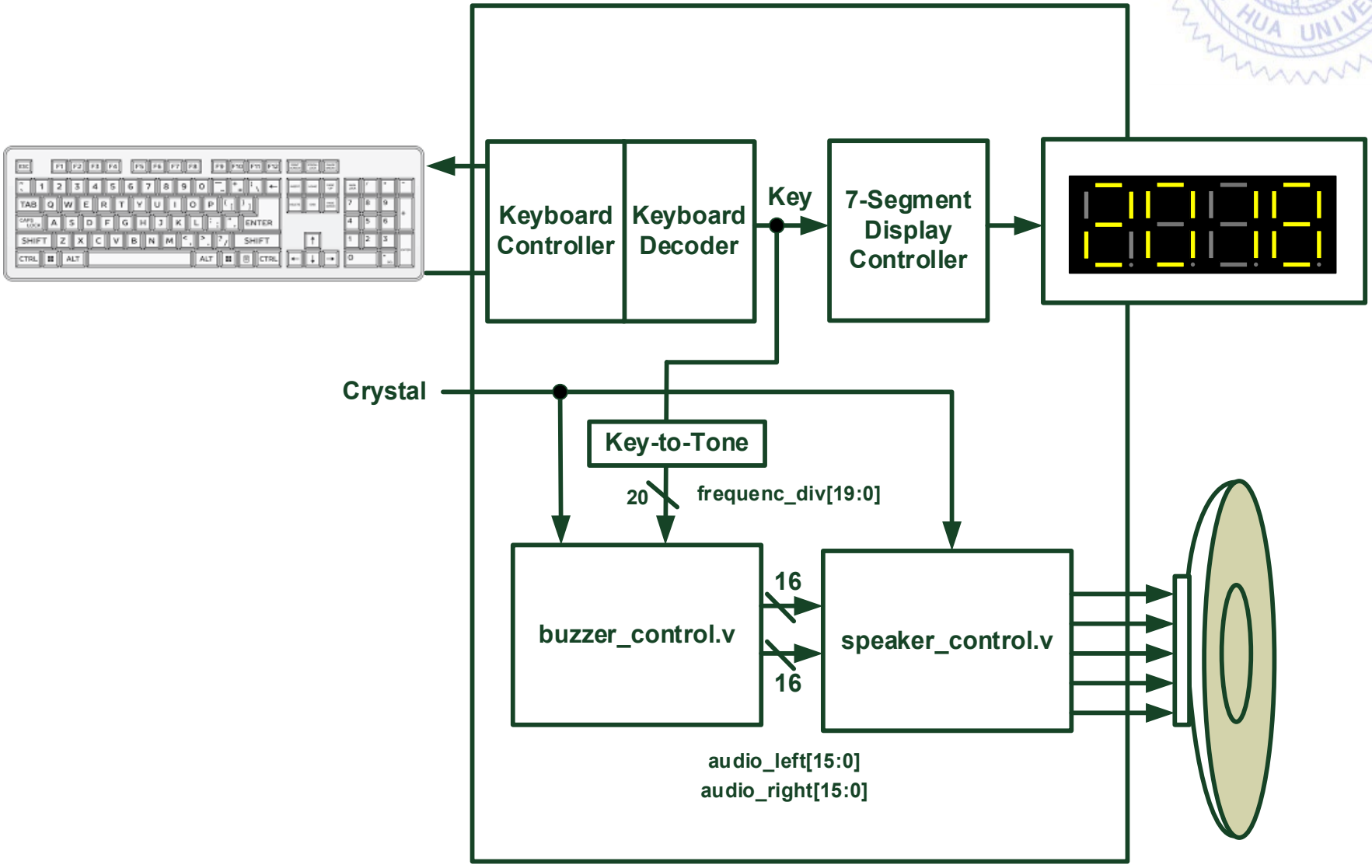


Sound Tone Table

- Table of 16 tone frequencies can be used to compute the periodic of the sound counter

Tone	Do	Re	Me	Fa	So	La	Si
Low (Hz)						220	245
Mid (Hz)	261	293	330	349	392	440	494
High (Hz)	524	588	660	698	784	880	988

Electronic Organ



Lab 7 Speaker

1 Please design an audio-data parallel-to-serial module to generate the speaker control signal with 100MHz system clock, 25 MHz master clock, (25/128) MHz Left-Right clock (Fs), and 6.25 MHz (32Fs) 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.

2 Speaker control

2.1 Please produce the buzzer sounds of **Do**, **Re**, and **Mi** by pressing buttons (Left, Center, Right) 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 (Up) as increase and (Down) and decrease the volumn. Please also quantize the audio dynamic range as 16 levels and show the current sound level in the 7-segment display.

Lab 7 Speaker

3	(Bonus) Playback double tones by separate left and right channels. If you turn one DIP switch off, the electronic organ playback single tone when you press push button. If you turn DIP switch on, left (right) channels play Do(Mi), Re(Fa), Mi(So), Fa(La), So(Si) when you press the five push buttons, respectively.