Experiment 3. Simple Music Box Design

【Purpose】 Creating a control circuit for playing music.

i. Applying encoding and decoding principles.
ii. Utilizing memory scanning principles.
iii. Understanding the principles of music generation.
iv. Understanding the control methods for music output.
v. Understanding the process of creating sheet music.

【Experimental background】

Verilog design experience

【Principle and explanation】

1. The principles of music generation

The principle of sound generation in a speaker is based on different frequencies producing varying pitches. Following this principle, by using the frequencies on the board(50MHz) and designing a frequency divider, the required range of musical notes can be generated.

In the realm of musical notes, there is a principle that the frequency between the original note and its octave higher is **doubled**, and there are a total of 12 semitones between two notes. This means that the frequency difference between each half-tone is a factor of $\sqrt[12]{2}$, making it easy to design the pitch for each note.

Regarding note duration, it's important to note that not every note in music has the same duration. To adjust note duration, it's possible to design the output time for each frequency differently, allowing for the desired note durations to be achieved.

Hz	Scale	Hz	Scale	Hz	Scale
195.99	So(bass)	207.65	#So(bass)	220	La(bass)
233.08	#La(bass)	246.94	Si(bass)	261.62	Do
277.18	#Do	293.66	Re	322.12	#Re
329.62	Mi	349.22	Fa	369.99	#Fa
391.99	So	415.30	#So	440	La
466.16	#La	493.88	Si	523.25	Do(treble)
554.36	#Do(treble)	587.32	Re(treble)	622.25	#Re(treble)
659.25	Mi(treble)	698.45	Fa(treble)	739.98	#Fa(treble)

783.99	So(treble)	830.66	#So(treble)	880	La(treble)
932.32	#La(treble)	987.76	Si(treble)	1046.50	Do(Super high)

Table 1. Frequency and Pitch Reference Table

2. The methods of controlling music output

In the previous section, we learned that pitch is determined by the frequency's highness or lowness, and note duration is determined by how long a frequency is sustained. Therefore, in music production, creating a single note requires incorporating both pitch and note duration as essential elements.

Pitch:

From a digital perspective, there are 12 notes in an octave. For most music, covering 2 octaves is usually sufficient. Therefore, in the design, 5 bits are used to represent pitch, allowing for the representation of more than 2 octaves. In practice, the divisor for each note is stored in memory. When you want to output a specific note, you utilize a frequency divider by dividing the clock signal by the value stored in memory.

For example, if you are using an experimental board with a built-in clock signal of 50 MHz and you want to generate La. , as we learned from the previous section, the output frequency is 440 Hz. Therefore, you would store the value 113636 in memory. The method of outputting the notes involves first storing the corresponding pitches at each memory location. By selecting the address lines appropriately, you determine which note to output. For instance, in the current configuration, memory location 0 stores La, and memory location 1 stores Si. If you want to output La, you would input 0 as the location.

Note Length:

A musical note can be divided into durations like 1/32, 1/16, and 1/8 notes. To represent these durations while meeting the requirements of most songs, a 3-bit design is used. It can display 7 different note lengths. For example, 3'b000 represents a 1/32 note, and 3'b001 represents a 1/16 note.

In the circuit, a decoder must be designed to convert the 3-bit data into their respective durations. Then, a comparator at the back end determines when to switch to the next note. For instance, you can set the shortest note (1/32) as the reference duration, equal to 1. Then, a 1/16 note becomes 2, and a 1/8 note becomes 4, effectively converting

the notes into more familiar numerical values.

At this point, the circuit includes an up-counter that counts at the rate of the shortest note (1/32). The value of the up-counter is compared with the value generated by the decoder using a comparator. When the values match, it indicates that the note duration has been reached, and a control signal is sent to request the output of the next note. This way, note duration control is achieved. The table below shows the values corresponding to notes after decoding。

note	Beat	Decoder Value
1/32 note	1/8 beat	1
1/16 note	1/4 beat	2
1/8 note	Half beat	4
1/4 note	1 beat	8
1/2 note	2 beat	16
3/4 note	3 beat	24
1 note	4 beat	32

Table 2. Correspondence Between Notes and Decoded Values

Rests, Key Changes, and Endings:

When music is playing, it is common to encounter consecutive identical notes and rests. To prevent consecutive identical notes from sounding connected and forming a longer note, it is necessary to break them apart in the design. The principle of this separation is to issue a reset signal to the circuit simultaneously when switching to the next note. This momentarily reduces the frequency to zero, creating a slight pause in the sound. The principle for rests is similar; during a rest, the total output frequency is set to **zero.** The control for the ending involves keeping the counter for switching to the next note inactive. This way, it does not switch to a new note, effectively stopping at the end.

3. Music Sheet Creation

The creation of sheet music primarily involves storing information about pitch and duration in memory. In our design, we use 3 bits for duration and 5 bits for pitch. Therefore, the sheet music is stored in an

8-bit memory, with duration bits at the beginning and pitch bits at the end. Additionally, we designate the case where all 8 bits are set to 1 as an end symbol. In other words, when the system encounters 8'hFF, it marks the end of a song and triggers the stop playback action. Below is the table mapping pitch and duration:

Bit[7..5]	Duration	Bit[4..0]	Pitch	Bit[4..0]	Pitch	Bit[4..0]	Pitch	Bit[4..0]	Pitch
000	1/32note	11000	#Fa (H)	10000	#La	01000	Re	00000	Rest
001	1/16note	11001	So (H)	10001	Si	01001	#Re	00001	So (L)
010	1/8note	11010	#So (H)	10010	Do (H)	01010	Mi	00010	#So (L)
011	1/4note	11011	La (H)	10011	#Do (H)	01011	Fa	00011	La (L)
100	1/2note	11100	#La (H)	10100	Re (H)	01100	#Fa	00100	#La (L)
101	3/4note	11101	Si (H)	10101	#Re (H)	01101	So	00101	Si (L)
110	1note	11110	Do (HH)	10110	Mi (H)	01110	#So	00110	Do
111		11111	Music end	10111	Fa (H)	01111	La	00111	#Do

Table 3. Mapping of Pitch and Duration

To create a song, simply input the required pitch and duration based on the table above. To mark the end of the song, add 8'hFF at the appropriate point, and your song will be complete.

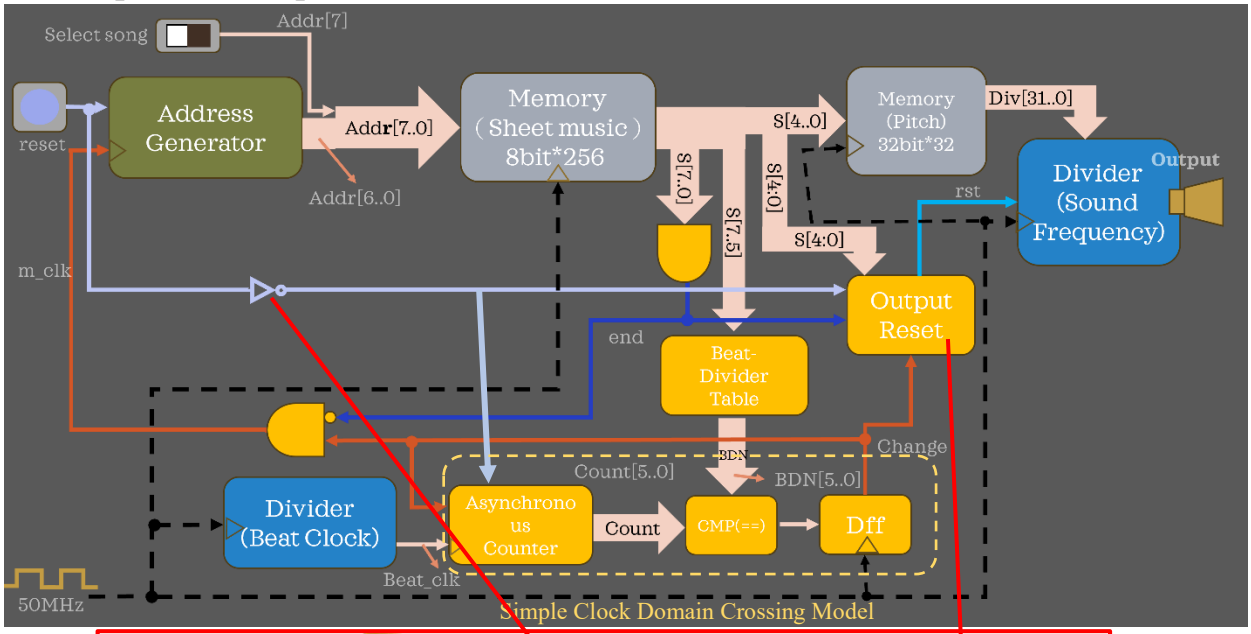
Song:butterfly

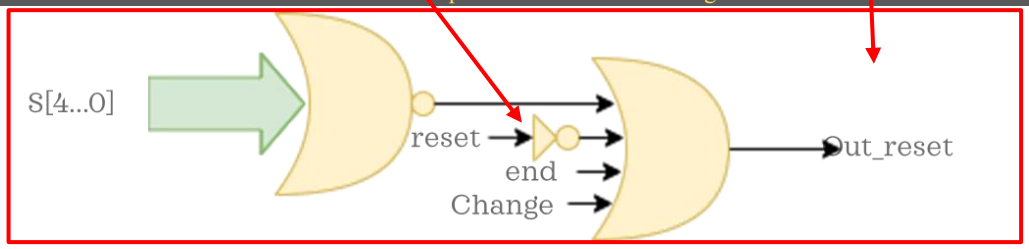
	+0	+1	+2	+3	+4	+5	+6	+7
0	60	4A	4A	4F	2F	4F	51	72
8	60	52	32	52	54	76	60	74
16	54	51	6D	71	52	54	52	51
24	6F	40	4F	4A	2F	4F	51	72
32	60	52	32	52	54	76	60	74
40	54	51	6D	71	52	54	52	51
48	6F	6A	76	56	59	7B	56	59
56	5B	5E	5B	59	76	60	74	54
64	56	79	56	52	54	56	54	52

72	6F	60	76	56	59	7B	56	59
80	5B	5E	5B	59	76	60	74	54
88	56	79	56	52	54	56	54	52
96	6f	60	FF					

【Implementation】

Create a control circuit for a simple music box that allows song selection, decodes the corresponding sheet music from memory, and outputs it to a speaker.





This architecture, as shown in the diagram above, includes:

2 dividers:

Beat frequency divider : Convert the 50MHz clock provided by the board into the shortest note duration (32nd note).

Audio frequency generation divider : Convert the 50MHz clock provided by the board into the desired output frequency for the notes.

"If a 1/4 note is equal to 1 beat, then a 1/32 note is equal to 1/8 of a beat. Given that the tempo of the music is set to 100 beats per minute, you can calculate that each beat is equal to 6/10 seconds or 10/6 Hz. Therefore, the reference for the divider for a 1/32 note is 1/8 of a beat, which is equivalent to (1/8 * 6/10) seconds = 80/6 Hz."

Address Generator:
Send the starting address of the selected song to the Memory (sheet music), and increment the address by 1 when the m_clk signal triggers on the positive edge, causing the Memory (sheet music) to output the data for the next note. Reset sets the address back to zero.
2 Memory:
Sheet music memory: Store the sheet music data for all songs in memory with a size of 8 bits * 256, and choose song2f.mif.
Pitch music memory: Store all the divisors corresponding to each note so that the divider (output sound) can generate the frequency of the desired note. The memory size is 32 bits * 32, and choose sound1.mif.
Decoder(Duration):
Decode the first 3 bits of the music score into 6 bits of note duration for comparison by the comparator.(Table 2)
Comparator:
Check if the counter has counted to the correct number
Asynchronous counter:
Increments on the rising edge of CLK, Resets on the rising edge of the change signal., This counter is an asynchronous counter (reset is not synchronized with CLK).
Output Reset:
When reset(music reset),end(music end),change(change the sound) or rest(S[4..0] is 0),the output(rst) is High.

Classroom Implementation

1. Complete all of the above modules and implement them using Verilog syntax.
2. Connect to the speaker to implement the music box.
   Additional Information:
   - This experiment is for a simple music box control circuit, so the memory module's write enable and data input should both be connected to **GND**.
   - The signal source for m_clk is ~end & change. This is done to ensure that when the music reaches its end, it won't continue outputting the next note and will stop at the end symbol.
   - The output of the divider (beat) is 13.33Hz because the song's tempo is 100 beats per minute.

Addition bonus:

Creating an extra creative feature. For example: using a 7-segment display to show the track, composing your own songs.

Experiment Report:

An overall architectural diagram, waveform simulations for each module (with explanations on how functionality was verified), Verilog code for each module (excluding the memory module) with comments, an introduction to creative additions (if any), and an experiment summary.