Project Design

The design includes the following parts.

1. Square wave output

2. Key input

3. OLED display

FPGA drives the square wave generation, and the MCU drives the key and OLED, and the two communicate directly through the AHB bus.

Flow: OLED displays the frequency and performs key operation when a key operation is monitored, and sends the set frequency to the FPGA through AHB bus.

MCU Program Design

Here the address 0xA0000000 is used to store the frequency to be sent to the FPGA (the data is the accumulated value converted according to the 32-bit overflow and the 27MHz clock).

1. Judge the state of the three keys. When the key is pressed, it will first de-jitter and then wait for the key to be stable. The functions includes: move the cursor, decrease the value of the frequency where the cursor is located, and increase the value of the frequency where the cursor is located.

2. Write the converted frequency to the 0xA0000000 address (frequency\*2^32/27000000).

3. Modify the display on the screen

1. uint32\_t freq = 0; //frequency
2. uint8\_t place = 0; //select
3. SystemInit();
4. GPIOInit();
5. OLED\_Init();                        //initialize OLED
6. OLED\_Clear();
7. while(1)
8. {
9. if((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_2) == 0)
10. {
11. delay\_ms(20);
12. while((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_2) == 0){}
14. freq += reduce\_freq[place];
16. }
17. if((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_3) == 0)
18. {
19. delay\_ms(20);
20. while((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_3) == 0){}
21. if(freq > reduce\_freq[place])
22. freq -= reduce\_freq[place];
23. else
24. freq = 0;
26. }
27. if((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_4) == 0)
28. {
29. delay\_ms(20);
30. while((GPIO\_ReadBits(GPIO0)&GPIO\_Pin\_4) == 0);
32. place++;
33. if(place > 7)
34. place = 0;
35. }
37. \*((uint32\_t \*)0xA0000000) = freq\*159.0728628148148148;  // set the frequency to fpga
38. {
39. OLED\_ShowChar(16+8\*0, 3, '0'+freq/10000000%10, place==7?1:0);
40. OLED\_ShowChar(16+8\*1, 3, '0'+freq/1000000%10, place==6?1:0);
41. OLED\_ShowChar(16+8\*2, 3, ',',0);
42. OLED\_ShowChar(16+8\*3, 3, '0'+freq/100000%10, place==5?1:0);
43. OLED\_ShowChar(16+8\*4, 3, '0'+freq/10000%10, place==4?1:0);
44. OLED\_ShowChar(16+8\*5, 3, '0'+freq/1000%10, place==3?1:0);
45. OLED\_ShowChar(16+8\*6, 3, ',',0);
46. OLED\_ShowChar(16+8\*7, 3, '0'+freq/100%10, place==2?1:0);
47. OLED\_ShowChar(16+8\*8, 3, '0'+freq/10%10, place==1?1:0);
48. OLED\_ShowChar(16+8\*9, 3, '0'+freq/1%10, place==0?1:0);
49. OLED\_ShowChar(16+8\*10, 3 ,'H',0);
50. OLED\_ShowChar(16+8\*11, 3 ,'z',0);
51. }
52. }

FPGA Program Design

It includes top, hardcore, AHB parsing, and square wave generation.

Top: instantiate each module.

Hardcore: use gpio and AHB2 Master, and the ip cores are very easy to configure via software.

AHB parsing: At position 0 of AHB, a 32 bit reg\_freq register used to receive the frequency of the square wave sent from MCU.

1. **`time**s**cal**e 100 ps/100 ps
2. module Gowin\_AHB\_Multiple
3. (
4. output [31:0] freq,
5. output        wire        [31:0]        AHB\_HRDATA,
6. output        wire                        AHB\_HREADY,
7. output        wire        [ 1:0]        AHB\_HRESP,
8. input        wire        [ 1:0]  AHB\_HTRANS,
9. input        wire        [ 2:0]  AHB\_Hburst,
10. input        wire        [ 3:0]  AHB\_HPROT,
11. input        wire        [ 2:0]        AHB\_HSIZE,
12. input        wire                        AHB\_HWRITE,
13. input        wire                        AHB\_HMASTLOCK,
14. input        wire        [ 3:0]        AHB\_HMASTER,
15. input        wire        [31:0]        AHB\_HADDR,
16. input        wire        [31:0]  AHB\_HWDATA,
17. input        wire                        AHB\_HSEL,
18. input        wire                        AHB\_HCLK,
19. input        wire                        AHB\_HRESETn
20. );
21. //The AHB BUS is always ready
22. assign AHB\_HREADY = 1'b1; //ready signal, slave to MCU master
23. //Response OKAY
24. assign AHB\_HRESP  = 2'b0;//response signal, slave to MCU master
25. //Define Reg for AHB BUS
26. reg [31:0]  ahb\_address;
27. reg                 ahb\_control;
28. reg         ahb\_sel;
29. reg         ahb\_htrans;
30. always @(posedge AHB\_HCLK or negedge AHB\_HRESETn)
31. begin
32. if(~AHB\_HRESETn)
33. begin
34. ahb\_address  <= 32'b0;
35. ahb\_control  <= 1'b0;
36. ahb\_sel      <= 1'b0;
37. ahb\_htrans   <= 1'b0;
38. end
39. else              //Select The AHB Device
40. begin                          //Get the Address of reg
41. ahb\_address  <= AHB\_HADDR;
42. ahb\_control  <= AHB\_HWRITE;
43. ahb\_sel      <= AHB\_HSEL;
44. ahb\_htrans   <= AHB\_HTRANS[1];
45. end
46. end
47. wire write\_enable = ahb\_htrans & ahb\_control    & ahb\_sel;
48. wire read\_enable  = ahb\_htrans & (!ahb\_control) & ahb\_sel;
49. //The register of Multiple AHB bus
50. reg [32:0] reg\_freq;
51. //write data to AHB bus
52. always @(posedge AHB\_HCLK or negedge AHB\_HRESETn)
53. begin
54. if(~AHB\_HRESETn)
55. begin
56. reg\_freq   <= 32'b0;
57. end
58. else if(write\_enable)
59. begin
60. case (ahb\_address[15:0])
61. 16'h0000: reg\_freq         <= AHB\_HWDATA[31:0];
62. endcase
63. end
64. end
65. //register address
66. reg [31:0] ahb\_rdata;
67. always @(\*)
68. begin
69. if(read\_enable)  //read cmd
70. begin
71. case (ahb\_address[15:0])
72. 32'h0000:  ahb\_rdata = reg\_freq;
73. default:ahb\_rdata = 32'hFFFFFFFF;
74. endcase
75. end
76. else
77. begin
78. ahb\_rdata = 32'hFFFFFFFF;
79. end
80. end
81. assign AHB\_HRDATA = ahb\_rdata;
82. assign freq = reg\_freq;
84. Endmodule

Square wave generation: 32-bit accumulator, by modifying the size of the single accumulation to control the period of the waveform.

1. module wave (
2. input sys\_clk,
3. input reset\_n,
4. input [31:0] freq,
5. output wave\_out
6. );
7. reg [31:0] phase\_acc;
8. always @(posedge sys\_clk) phase\_acc <= phase\_acc + freq;
9. wire [31:0] phase = phase\_acc;
10. assign wave\_out = phase[31];
11. endmodule

**Resource Utilization**  
MCU:

1. ==============================================================================
2. Code (inc. data)   RO Data    RW Data    ZI Data      debug
3. 2880        142       2324         16       1632      38415   Grand Totals
4. 2880        142       2324         16       1632      38415   ELF Image Totals
5. 2880        142       2324         16          0          0   ROM Totals
6. ==============================================================================
7. Total RO  Size (Code + RO Data)                 5204 (   5.08kB)
8. Total RW  Size (RW Data + ZI Data)              1648 (   1.61kB)
9. Total ROM Size (Code + RO Data + RW Data)       5220 (   5.10kB)
10. ==============================================================================