Blinking LED Core

Andrew Clinkenbeard

November 1, 2021

Video
GitHub

1 Introduction

In this assignment, core for the FPro system was created that blinks four LEDs with a period of XXXX milliseconds. Each LED blinks at a predetermined by software interval. With the core, the processor only needs to write to the registers that hold the interval time.

2 Method

2.1 Overview of Method

In order to produce a device that did this, first a counter was created that gives out a millisecond tick, then a blinker that blinks the LEDs was created, and finally a blinker top level module was created that blinked four LEDs. This describes the hardware creation. This was then put inside a wrapped which then plugged into the Chu mmio bus which interacts with the Microblaze processor. For the software, a header file was created along with a .cpp file which created the functions to write to the registers.

2.2 Counter

Below is the counter which was used to count the milliseconds passed since the start button was pressed.

```
module counter(
    input logic clk,
    input logic rst,
    output logic ms_tick
    );

    logic [20:0] count, ncount;
    logic [20:0] target = 20'b00011000011010011111;
```

```
    always_ff @(posedge clk, posedge rst)
        if(rst)
            count <= 0;
        else
            count <= ncount;



     always_comb
        if (count == target)
             begin
                ms_tick = 1;
                ncount = 0;
            end
        else
            begin
                ms_tick = 0;
                ncount = count + 1;
            end

endmodule
```

2.3 Blinker

Below is the blinker which switches the LED state when the target time has passed.

```
module blinker (
    input logic clk,
    input logic rst,
    input logic ms_tick,
    input logic [15:0] target,
    output logic led
);

    logic [15:0] count = 0;
    logic nled;
    always_ff @(posedge clk, posedge rst)
        if (rst)
            led <= 0;
        else
            led <= nled;

    always @(posedge ms_tick) begin
        count = count + 1;

        if (count > target)
```

```
            count = 0;
        else if (count < target[15:1])
            nled = 0;
        else if (count >= target[15:1])
            nled = 1;
        end


endmodule
```

2.4 Blinker Top Level

Below is the top level of the blinker which inserts the counter as well as four instances of the blinker module in order to blink four LEDs.

```
module blinker_top(
    input logic clk,
    input logic reset_n,
    input [15:0] target0, target1, target2, target3,
    output logic [3:0] led_out
    );



    logic ms_tick;
    counter my_counter(
    .clk(clk),
    .rst(reset_n),
    .ms_tick(ms_tick)
    );


    blinker blinker0 (
    .clk(clk),
    .rst(reset_n),
    .ms_tick(ms_tick),
    .target(target0),
    .led(led_out[3])
    );

    blinker blinker1 (
    .clk(clk),
    .rst(reset_n),
    .ms_tick(ms_tick),
    .target(target1),
    .led(led_out[2])
    );
```

```
    blinker blinker2 (
    .clk(clk),
    .rst(reset_n),
    .ms_tick(ms_tick),
    .target(target2),
    .led(led_out[1])
    );

    blinker blinker3 (
    .clk(clk),
    .rst(reset_n),
    .ms_tick(ms_tick),
    .target(target3),
    .led(led_out[0])
    );
endmodule
```

2.5 Wrapper

Below is the wrapper which wraps the blinker's top level in order to allow an easy plug-in into the mmio bus.

```
module hw3(
    input  logic clk,
    input  logic reset,
    // slot interface
    input  logic cs,
    input  logic read,
    input  logic write,
    input  logic [4:0] addr,
    input  logic [31:0] wr_data,
    output logic [31:0] rd_data,
    // external port
    output logic [3:0] dout
   );

   blinker_top my_blinker(
   .clk(clk),
   .reset_n(!reset),
   .target0(wait0),
   .target1(wait1),
   .target2(wait2),
   .target3(wait3),
   .led_out(dout)
   );
```

```
   // declaration
   logic [15:0] buf_reg0, buf_reg1, buf_reg2, buf_reg3;
   logic [15:0] wait0, wait1, wait2, wait3;
   logic wr_en;

   // body
   // output buffer register
   always_ff @(posedge clk, posedge reset)
      if (reset)begin
         buf_reg0 <= 0;
         buf_reg1 <= 0;
         buf_reg2 <= 0;
         buf_reg3 <= 0;
         end
      else begin
         if (wr_en && addr == 4'b0000)
            buf_reg0 <= wr_data[15:0];
         else if (wr_en && addr == 4'b0001)
            buf_reg1  <= wr_data[15:0];
         else if (wr_en && addr == 4'b0010)
            buf_reg2  <= wr_data[15:0];
         else if (wr_en && addr == 4'b0011)
            buf_reg3  <= wr_data[15:0];
      end



   // decoding logic
   assign wr_en = cs && write;
   // slot read interface
   assign rd_data =  0;
   // external output
   assign buf_reg0 = wait0;
   assign buf_reg1 = wait1;
   assign buf_reg2 = wait2;
   assign buf_reg3 = wait3;
endmodule
```

2.6 Software

Finally, the software was created. Fist, below is the header file for the blinking LED core.

```
#ifndef _hw3_H_INCLUDED
#define _hw3_H_INCLUDED

#include "chu_init.h"

class HW3Core {
```

```
public:
/**
* register map
*
*/
enum {
BLINK0_REG = 0, /**< Blink rate for LED 12 in ms */
BLINK1_REG = 1, /**< Blink rate for LED 13 */
BLINK2_REG = 2, /**< Blink rate for LED 14 */
BLINK3_REG = 3 /**< Blink rate for LED 15 */
};
/**
* constructor.
*
*/
HW3Core(uint32_t core_base_addr);
~HW3Core();                    // not used


/*
* function to write a timer to all of the timer reg
*
*/
void write_reg(uint32_t time0, uint32_t time1,
uint32_t time2, uint32_t time3);

private:
uint32_t base_addr;
uint32_t wr_data0;
uint32_t wr_data1;
uint32_t wr_data2;
uint32_t wr_data3;
};

#endif  // _HW3_H_INCLUDED
```

Next the drivers for the header file was created.

```
#include "hw3_core.h"

HW3Core::HW3Core(uint32_t core_base_addr) {
base_addr = core_base_addr;
wr_data0 = 0;
wr_data1 = 0;
wr_data2 = 0;
wr_data3 = 0;
}

HW3Core::~HW3Core() {
}


void HW3Core::write_reg(uint32_t time0, uint32_t time1,
uint32_t time2, uint32_t time3) {
wr_data0 = time0;
```

```
wr_data1 = time1;
wr_data2 = time2;
wr_data3 = time3;

io_write(base_addr, BLINK0_REG, wr_data0);
io_write(base_addr, BLINK1_REG, wr_data1);
io_write(base_addr, BLINK2_REG, wr_data2);
io_write(base_addr, BLINK3_REG, wr_data3);
}
```

Finally, the following code was added at the appropriate place in order to test the blinking LED core.

```
// instantiate  blink rate class
HW3Core time(get_slot_addr(BRIDGE_BASE, S4_USER));

// function to call in main to set blink rate
 void set_blink_rate(int t1, int t2, int t3, int t4) {
time.write_reg(t1, t2, t3, t4);
}

// function called in main to set blink rate
set_blink_rate(10000, 5000, 2000, 1000);
```

3 Testing

In order to test, several different values were inserted into the registers to test that they would all blink at different times.

4 Results

The LEDs blinked at the appropriate rate.

5 Conclusion

In conclusion, the blinking LED core was able to be created and working. The software allows for the desired period of blinking to quickly be inserted with out having to change the hardware. The video of the working device can be seen at this link. The code for the gitHub can be found at this link.