Implementation of the below circuit using Vaman Arm

Sai Harshith Kalithkar
harshith.work@gmail.com
FWC22118
IIT Hyderabad-Future Wireless Communication Assignment-4.1

April 2023

Contents

1 Problem

GATE EC-2019
Q.25. In the circuit shown,the clock frequency, i.e.,the frequency of the clock signal ,is 12 KHz.The frequency of the signal at Q2 is ..... KHz.

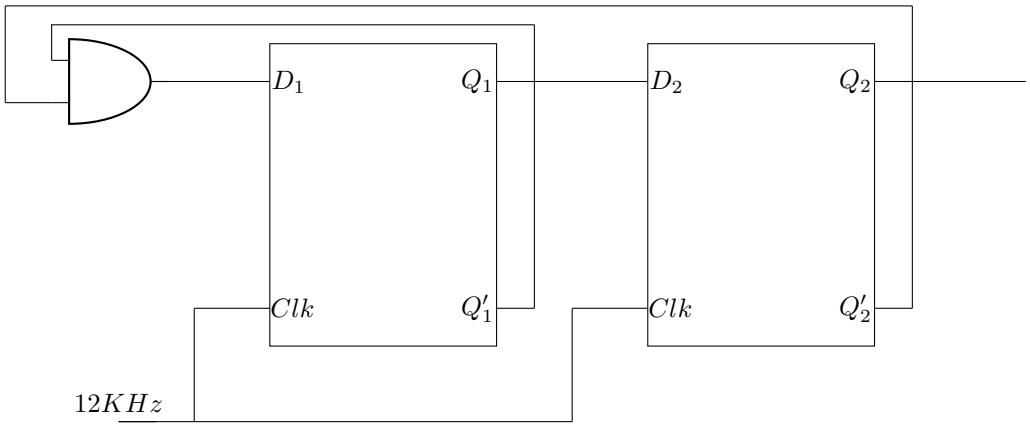

Figure 1: circuit

2 Introduction

The aim is to implement the above sequential circuit using D flip-flops (IC 7474) and to find out the frequency of the signal at Q2(it is given that the frequency of the clock signal is 12KHz).IC 7474 is a dual positive edge triggered D type flip flop,which means it has two separate flip-flop that are triggered by the rising edge of a clock signal.

In the above circuit Q_1,Q_2 are inputs and D_1,D_2 are outputs.So,from the circuit the expressions of D_1 and D_2 are:

$D_1 = Q_1'Q_2'$.

$D_2 = Q_1$.

Below is the transition table of the above circuit which is as follows:

INPUT		OUTPUT	
Q_1	Q_2	D_1	D_2
0	0	1	0
1	0	0	1
0	1	0	0

Table 1: Transition table

3 Components

COMPONENTS		
Component	Value	Quantity
Resistor	=220 Ohm	1
Arduino	UNO	1
Seven Segent Display	Common Anode	1
Decoder	7447	1
Flip Flop	7474	1
Jumper Wires		20
Breadboard		1

Table 2: Components

4 Hardware

IC 7474 is a D flip-flop integrated circuit that is commonly used in digital electronics applications.It is a dual positive edge-triggered by the rising edge of a clock signal.Below is the pin diagram of IC 7474:

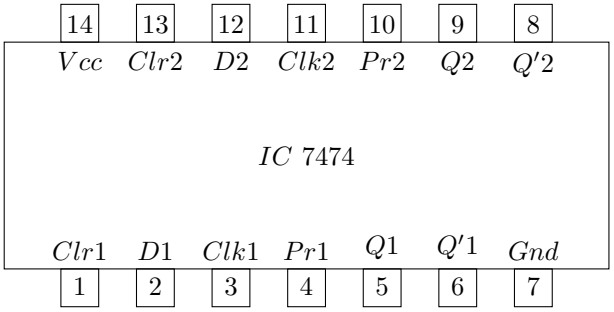

Figure 2: 7474

The connections between the arduino and IC 7474 is as follows:

	INPUT		OUTPUT		CLOCK		VCC			
ARDUINO	D2	D3	D5	D6	D13		5V			
7474	5	9	2	12	3	11	1	4	10	13
7447			1	7				16		

Table 3: connections

5 Software

The code to implement the above circuit is :

```
#include "Fw_global_config.h"   // This defines application specific charactersitics

#include <stdio.h>
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "timers.h"
#include "RtosTask.h"

/*Include the generic headers required for QORC */
#include "eoss3_hal_gpio.h"
#include "eoss3_hal_rtc.h"
#include "eoss3_hal_timer.h"
#include "eoss3_hal_fpga_usbserial.h"
#include "ql_time.h"
#include "s3x_clock_hal.h"
#include "s3x_clock.h"
#include "s3x_pi.h"
#include "dbg_uart.h"

#include "cli.h"


extern const struct cli_cmd_entry my_main_menu[];


const char *SOFTWARE_VERSION_STR;


/*
 * Global variable definition
 */


extern void qf_hardwareSetup();
static void nvic_init(void);

#define GPIO_OUTPUT_MODE (1)
```

```
#define GPIO_INPUT_MODE (0)
void PyHal_GPIO_SetDir(uint8_t gpionum,uint8_t iomode);
int PyHal_GPIO_GetDir(uint8_t gpionum);
int PyHal_GPIO_Set(uint8_t gpionum, uint8_t gpioval);
int PyHal_GPIO_Get(uint8_t gpionum);

int main(void)
{
    uint32_t Q1, Q2, D1, D2;
    SOFTWARE_VERSION_STR = "qorc-onion-apps/qf_hello-fpga-gpio-ctlr";

    qf_hardwareSetup();
    nvic_init();

    dbg_str("\n\n");
    dbg_str( "############################\n");
    dbg_str( "Quicklogic␣QuickFeather␣FPGA␣GPIO␣CONTROLLER␣EXAMPLE\n");
    dbg_str( "SW␣Version:␣");
    dbg_str( SOFTWARE_VERSION_STR );
    dbg_str( "\n" );
    dbg_str( __DATE__ "␣" __TIME__ "\n" );
    dbg_str( "############################\n\n");

    dbg_str( "\n\nHello␣GPIO!!\n\n");   // <<<<<<<<<<<<<<<<<<<<  Change me!

    CLI_start_task( my_main_menu );
        HAL_Delay_Init();

    PyHal_GPIO_SetDir(2,0); //Q1
    PyHal_GPIO_SetDir(3,0); //Q2
    PyHal_GPIO_SetDir(4,1); //D1
    PyHal_GPIO_SetDir(5,1); //D2
    PyHal_GPIO_SetDir(6,1); //CLK

    PyHal_GPIO_Set(2,0);
    PyHal_GPIO_Set(3,0);

    while(1){
        PyHal_GPIO_Set(6,0);
        HAL_DelayUSec(500000);


        Q1 = PyHal_GPIO_Get(2);
        Q2 = PyHal_GPIO_Get(3);
        D1 = (!Q1)&&(!Q2);
        D2 = Q1;
        PyHal_GPIO_Set(4,D1);
        PyHal_GPIO_Set(5,D2);

        PyHal_GPIO_Set(6,1);
        HAL_DelayUSec(500000);
    }


    /* Start the tasks and timer running. */
    vTaskStartScheduler();
    dbg_str("\n");
```

```
    while(1);
}

static void nvic_init(void)
 {
    // To initialize system, this interrupt should be triggered at main.
    // So, we will set its priority just before calling vTaskStartScheduler(), not the time of
    NVIC_SetPriority(Ffe0_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
    NVIC_SetPriority(SpiMs_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
    NVIC_SetPriority(CfgDma_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
    NVIC_SetPriority(Uart_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
    NVIC_SetPriority(FbMsg_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY);
 }

//needed for startup_EOSS3b.s asm file
void SystemInit(void)
{

}

//gpionum --> 0 --> 31 corresponding to the IO PADs
//gpioval --> 0 or 1
#define FGPIO_DIRECTION_REG (0x40024008)
#define FGPIO_OUTPUT_REG (0x40024004)
#define FGPIO_INPUT_REG (0x40024000)
//Set GPIO(=gpionum) Mode: Input(iomode = 0) or Output(iomode = 1)
//Before Set/Get GPIO value, the direction must be correctly set
void PyHal_GPIO_SetDir(uint8_t gpionum,uint8_t iomode)
{
    uint32_t tempscratch32;

    if (gpionum > 31)
        return;

    tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);
    if (iomode)
        *(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 | (0x1 << gpionum);
    else
        *(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 & (~(0x1 << gpionum));

}


//Get current GPIO(=gpionum) Mode: Input(iomode = 0) or Output(iomode = 1)
int PyHal_GPIO_GetDir(uint8_t gpionum)
{
    uint32_t tempscratch32;
    int result = 0;

    if (gpionum > 31)
        return -1;

    tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);

    result = ((tempscratch32 & (0x1 << gpionum)) ? GPIO_OUTPUT_MODE : GPIO_INPUT_MODE);
```

```
    return result;
}

//Set GPIO(=gpionum) to 0 or 1 (= gpioval)
//The direction must be set as Output for this GPIO already
//Return value = 0, success OR -1 if error.
int PyHal_GPIO_Set(uint8_t gpionum, uint8_t gpioval)
{
    uint32_t tempscratch32;

    if (gpionum > 31)
        return -1;

    tempscratch32 = *(uint32_t*)(FGPIO_DIRECTION_REG);

    //Setting Direction moved out as separate API, we will only check
    //*(uint32_t*)(FGPIO_DIRECTION_REG) = tempscratch32 | (0x1 << gpionum);
    if (!(tempscratch32 & (0x1 << gpionum)))
    {
        //Direction not Set to Output
        return -1;
    }

    tempscratch32 = *(uint32_t*)(FGPIO_OUTPUT_REG);

    if(gpioval > 0)
    {
        *(uint32_t*)(FGPIO_OUTPUT_REG) = tempscratch32 | (0x1 << gpionum);
    }
    else
    {
        *(uint32_t*)(FGPIO_OUTPUT_REG) = tempscratch32 & ~(0x1 << gpionum);
    }

    return 0;
}
//Get GPIO(=gpionum): 0 or 1 returned (or in erros -1)
//The direction must be set as Input for this GPIO already
int PyHal_GPIO_Get(uint8_t gpionum)
{
    uint32_t tempscratch32;
    uint32_t gpioval_input;

    if (gpionum > 31)
        return -1;

    tempscratch32 = *(uint32_t*)(FGPIO_INPUT_REG);
    gpioval_input = (tempscratch32 >> gpionum) & 0x1;

    return ((int)gpioval_input);
}
```