ARM ASSIGNMENT

Tanyala Srihitha
srihithatanyala@gmail.com

May 10, 2023

Contents

1 Problem

(GATE EC-2020)
Q.No 10 The figure(Fig.1) below shows a multiplexer where S_1 and S_0 are select lines, I_0 to I_3 are the input data lines, EN is the enable line, and $F(P, Q, R)$ is the output, F is

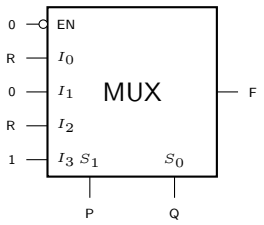

Figure 1:

1. $PQ + Q'R$
2. $P + QR'$
3. $PQ'R + P'Q$
4. $Q' + PR$

2 Components

The components required are given in Table.1

Component	Value	Quantity
Vaman Board	-	1
Breadboard	-	1
Led	-	1
Jumper wires	M-M	20

Table 1:

3 Implementation

We know that the output of a multiplexer is given as:

$$F = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3 \quad (1)$$
$$F = P'Q'R + P'Q(0) + PQ'R + PQ(1) \quad (2)$$
$$F = P'Q'R + PQ'R + PQ \quad (3)$$

3.1 Truth Table

From the above equation, truth table is given in Table.2

P	Q	R	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 2:

3.2 K-map

From the above truth table, Fig.2 represents the K-map:

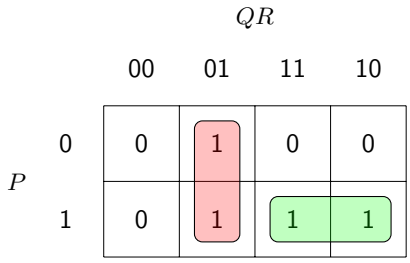

Figure 2:

3.3 Boolean Expression

By Solving the above K-map, we get a boolean equation as: $F = PQ + Q'R$

4 Hardware

1. Set the GPIO pins: 2,3,4 of Vaman as inputs.
2. Set the GPIO pin 10 of Vaman as output.
3. Read the input pins after connecting the Vcc and GND pins.
4. Verify the outputs using the truth table.

5 Software

```
#include "Fw_global_config.h"

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

#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 P,Q,R,F;
    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");
CLI_start_task( my_main_menu );
        HAL_Delay_Init();

PyHal_GPIO_SetDir(4,0);
PyHal_GPIO_SetDir(5,0);
PyHal_GPIO_SetDir(6,0);
PyHal_GPIO_SetDir(10,1);

while(1)
{
        P= PyHal_GPIO_Get(4);
        Q= PyHal_GPIO_Get(5);
        R= PyHal_GPIO_Get(6);

        F=(P&Q)|((!Q)&R);
        PyHal_GPIO_Set(10,F);
}


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

    while(1);
}

static void nvic_init(void)
 {
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)

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));

}

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;
}


int PyHal_GPIO_Set(uint8_t gpionum,
    uint8_t gpioval)
{
    uint32_t tempscratch32;

    if (gpionum > 31)
        return -1;

    tempscratch32 = *(uint32_t*)(
        FGPIO_DIRECTION_REG);

    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;
}
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);
}
```