Implementation of 4x1 mux using Arm(VAMAN)

Marikundam Harshitha
marikundamdec@gmail.com
FWC22120
IIT Hyderabad-Future Wireless Communication Assignment - 4

March 2023

Contents

1 Problem

(GATE EC-2022)

Q.19. Consider the 2-bit multiplexer(MUX) shown in the figure.For output to be the XOR of R and S,the values for W, X, Y and Z are ?

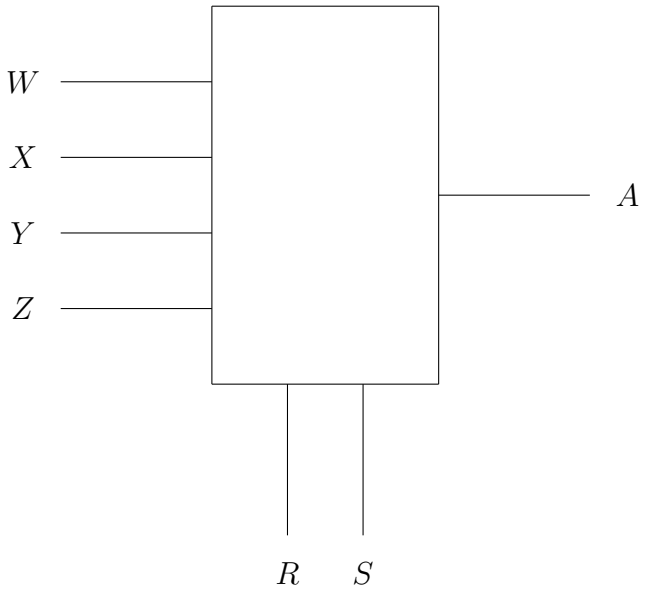

Figure 1: mux

1. $W = 0, X = 0, Y = 1, Z = 1$
2. $W = 1, X = 0, Y = 1, Z = 0$
3. $W = 0, X = 1, Y = 1, Z = 0$
4. $W = 1, X = 1, Y = 0, Z = 0$

2 Introduction

The above diagram is a 4:1 multiplexer where W, X, Y, Z are the inputs of the multiplexer and A is the output of the multiplexer.R, S are the select

lines of the multiplexer,which means:

1. For $R = 0, S = 0$,the first input line W is selected.
2. For $R = 0, S = 1$,the second input line X is selected.
3. For $R = 1, S = 0$,the third input line Y is selected.
4. For $R = 1, S = 1$,the fourth input line Z is selected.

Therefore,the resultant output expression of the multiplexer is $R'S'W + R'SX + RS'Y + RSZ$.

3 Components

COMPONENTS		
Component	Value	Quantity
Resistor	220 ohm	1
Arduino	UNO	1
Seven Segment Display		1
Jumper Wires	M-M	20
Breadboard		1

Table 1: contents

4 Hardware

1. Set the GPIO pins: 4,5,6,7,8,9 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.

Truth table		
R	S	A
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: truth table

The K-map for this truth table will be a two variable K-map and it will be as follows:

		R	
		0	1
S	0	0	1
	1	1	0

Figure 2: k-map

So,the resultant expression of A is $A = R'S + RS'$.

5 Software

The embedded code for the given circuit is

```
#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 W,X,Y,Z,R,S,A,B;
    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(7,0);
```

```
PyHal_GPIO_SetDir(8,0);
PyHal_GPIO_SetDir(9,0);
PyHal_GPIO_SetDir(10,1);

while(1){
        W= PyHal_GPIO_Get(4);
        X= PyHal_GPIO_Get(5);
        Y= PyHal_GPIO_Get(6);
        Z= PyHal_GPIO_Get(7);
        R= PyHal_GPIO_Get(8);
        S= PyHal_GPIO_Get(9);
        A=(!R&&!S&&W) || (!R&&S&&X)
                || (R&&!S&&Y) || (R&&S&&Z);
        B=(!R&&S) || (R&&!S);

        if(A==B){
        PyHal_GPIO_Set(10,1);
        }
        else{
        PyHal_GPIO_Set(10,0);
        }
}


    /* 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);
}
```