CHECKING THE BELOW EQUATIONS IF THEY ARE TRUE USING ARM(VAMAN)

M Sai Sarath Chandra
chandu.4567890@gmail.com
FWC22117 IITH-Future Wireless Communications Assignment-ARM

Contents

1 Problem

(GATE2019-QP-CS)
Q.6. Which one of the following is NOT a valid identity?

1. $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
2. $(x + y) \oplus z = x \oplus (y + z)$
3. $x \oplus y = x + y$,if $xy = 0$
4. $x \oplus y = (xy + x'y')'$

2 Introduction

The above question can be answered by evaluating the expressions using the digital logic identities and properties,the following cases are evaluated using digital logics and properties:

1. $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
   Explanation:This option follows associative property of XOR,so this is TRUE for every value of x,y and z.
2. $(x + y) \oplus z = x \oplus (y + z)$
   Explanation:This is not a TRUE identity because this identity is not applicable for every value of x,y and z.
3. $x \oplus y = x + y$,if $xy = 0$
   Explanation:As $x \oplus y = (x'y + xy')$,so if x or y is considered to be 0 and after substituting in the equation it results in $x + y$.
   Example: Let us consider $x = 0$ and $y = 1$,so for this case
   $x \oplus y = (0)'1 + 0(1)' = (1)1 + 0(0) = 1 + 0 = 1$.(since (0)'=1,(1)'=0)
   $x + y = 1 + 0 = 1$
   Hence proved that this a TRUE identity.The same can be proved for the rest three cases.
4. $x \oplus y = (xy + x'y')'$
   Explanation:As $(xy + x'y') = (x \odot y)$ and the complement of $(x \odot y)' = (x \oplus y)$.
   So,this is true for all cases of x and y.

3 Components

Components	**Value**	**Quantity**
Breadboard		1
Resistor	$\geq 220\Omega$	1
Vaman	pygmy	1
Seven Segment Display	Common Anode	1
Decoder	7447	1
Flip Flop	7474	2
Jumper Wires		20

Table 1: Components

3.1 Vaman

The Vaman (pygmy) has some ground pins, digital pins that can be used for both input as well as output.It also has two power pins that can generate 3.3V. In the following exercises, we use digital pins,GND and 5V .

3.2 Seven Segment Display

The seven segment display has eight pins, *a,b,c,d,e,f,g* and *dot* that take an active LOW input,i.e. the LED will glow only if the input is connected to ground.Each of these pins is connected to an LED segment.The *dot* pin is reserved for the LED.

4 Implementation

The above problem can be implemented using vaman and seven-segment display by connecting both of them as mentioned in the table below:

Component	PIN.NOs						
VAMAN	4	5	6	7	8	9	10
seven-segment display	a	b	c	d	e	f	g

Table 2: Connections

5 Software

To implement the above code using arm the following code can be used to implement:

```
#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);


void sevenseg_setup(void);
void sevenseg(int a, int b, int c, int d, int e, int f, int g);
void disp(int num);

int main(void)
{
    uint32_t i=0,j=0,k=0;
    uint32_t Z=0,Y=1,X=0;
    uint32_t E,D,C,B,A,F,G,H,I,option=2;


    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();

E = ((X&Y)|(!X)&(!Y));//(X XNOR Y)
D = (Y|Z);
C = (X|Y);
B = (((!Y)&Z)|(Y&(!Z)));//(Y XOR Z)
A = (((!X)&Y)|(X&(!Y)));//(X XOR Y)
F = (((!A)&Z)|(A&(!Z)));//OPTION A
G = (((!X)&B)|(X&(!B)));//OPTION A
H = (((!C)&Z)|(C&(!Z)));//OPTION B
I = (((!X)&D)|(X&(!D)));//OPTION B


     sevenseg_setup();    //Sevenseg ready for display

switch(option)
{
        case 1:
          if(F==G)
          {
                  disp(1);
          }
          else
          {
                  disp(0);
          }
        break;
        case 2:
          if(H==I)
          {
                  disp(1);
          }
          else
          {
                  disp(0);
          }
        break;
        case 3:
          if(A==C)
          {
                  disp(1);
          }
          else
          {
                  disp(0);
          }
        break;
        case 4:
          if(A==(!E))
          {
                  disp(1);
          }
          else
          {
                  disp(0);
          }
        break;
        default:
                sevenseg(1,1,1,1,1,1,1);
                break;
}

    /* 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
        enabling each irq.
    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);
 }

void sevenseg(int a, int b, int c, int d, int e, int f, int g)

{
    //Seven Segment GPIO
    PyHal_GPIO_Set(4,a);//a
    PyHal_GPIO_Set(5,b);//b
    PyHal_GPIO_Set(6,c);//c
    PyHal_GPIO_Set(7,d);//d
    PyHal_GPIO_Set(8,e);//e
    PyHal_GPIO_Set(10,f);//f
    PyHal_GPIO_Set(11,g);//g
}

//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);
}



void sevenseg_setup(void)

{

    //Set GPIO direction
    PyHal_GPIO_SetDir(4,1);
    PyHal_GPIO_SetDir(5,1);
    PyHal_GPIO_SetDir(6,1);
    PyHal_GPIO_SetDir(7,1);
    PyHal_GPIO_SetDir(8,1);
    PyHal_GPIO_SetDir(10,1);
    PyHal_GPIO_SetDir(11,1);

}

void disp(int num)
{
switch(num)
{
        case 0:
          sevenseg(0,0,0,0,0,0,1);
        break;
        case 1:
          sevenseg(1,0,0,1,1,1,1);
        break;
        default:
          sevenseg(0,1,1,0,0,0,0);
        break;
}
}
```