Práctica 2: Simulación del BUS PCI

Periféricos e Interfaces

David Morales Sáez

Introducción

En esta práctica se nos ha solicitado hacer un programa que simule un BUS PCI, en el cual podremos manejar las distintas señales y ver el correcto funcionamiento de este BUS.

Desarrollo

La práctica se ha desarrollado íntegramente con el entorno de desarrollo Qt, basado en el lenguaje de programación C++. El código de la práctica es el siguiente:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

bool frame=false, irdy=false, trdy=false, devset=true, LECTURA, inicializado=false, activado =
false;
int dir, escrito;

MainWindow::MainWindow(QWidget *parent) :
        QMainWindow(parent),
        ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->tabla->setRowCount(10);
    ui->tabla->setColumnCount(1);
    QStringList encabezado;
    encabezado << "Valor" ;
    ui->tabla->setHorizontalHeaderLabels(encabezado);
    ui->tabla->setColumnWidth(0, 105);
    ui->tabla->setRowHeight(0, 20);
    ui->tabla->setRowHeight(1, 20);
    ui->tabla->setRowHeight(2, 20);
    ui->tabla->setRowHeight(3, 20);
    ui->tabla->setRowHeight(4, 20);
    ui->tabla->setRowHeight(5, 20);
    ui->tabla->setRowHeight(6, 20);
    ui->tabla->setRowHeight(7, 20);
    ui->tabla->setRowHeight(8, 20);
    ui->tabla->setRowHeight(9, 20);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_FRAME_clicked()
{
    if(!frame)
    {
        frame = true;
    }
    else
    {
        frame = false;
    }
}
```

```
void MainWindow::on_IRDY_clicked()
{
    if(frame)
    {
        if(irdy)
            irdy = false;
        else
            irdy = true;
    }
}

void MainWindow::on_NEXT_clicked()
{
    if(frame)
    {
        if(inicializado)
        {
            if(activado)
            {
                if(dir<10)
                {
                    if((irdy)&&(trdy))
                    {
                        if(LECTURA)
                        {       // Si estamos leyendo
                            // Ponemos al principio de lo escrito lo nuevo
                            // Y lo guardamos en su posición del vector
                            QString mascara = ui->MASK->toPlainText();
                            QString palabro = ui->DATA->toPlainText();
                            QTableWidgetItem *p1 = ui->tabla->item(dir, 0);
                            if(p1==0)
                            {
                                irdy = false;
                                trdy = false;
                                frame = false;
                                devset = false;
                                inicializado = false;
                                activado = false;
                                ui->IRDY->setChecked(false);
                                ui->TRDY->setChecked(false);
                                ui->FRAME->setChecked(false);
                                ui->DEVSEL->setChecked(false);
                                ui->DIR->clear();
                                ui->MASK->clear();
                                return;
                            }
                            int i;
                            for(i=0; i<4; i++)
                            {
                                if(mascara[i]=='1')
                                {
                                    palabro[i] = p1->text()[i];
                                }
                            }
                            ui->DATA->clear();
                            ui->DATA->insertPlainText(palabro);
                            dir++;
                        }
                        else
                        {       // Si estamos escribiendo
```

```
                    // Ponemos el dato en su hueco debido
                    QString mascara = ui->MASK->toPlainText();
                    int i;
                    QTableWidgetItem *p1 = ui->tabla->item(dir, 0);
                    QString palabro;
                    for(i=0; i<4; i++)
                    {
                        if(mascara[i]=='1')
                        {
                            palabro[i] = ui->DATA->toPlainText()[i];
                        }
                    }
                    QTableWidgetItem *palabra = new QTableWidgetItem(palabro);
                    palabra->setFlags(palabra->flags() & (~Qt::ItemIsEditable));
                    palabra->setTextColor(Qt::blue); // color de los items
                    ui->tabla->setItem(dir,0,palabra);
                    dir++;
                }
            }
        }
        else
        {
            irdy = false;
            trdy = false;
            frame = false;
            devset = false;
            inicializado = false;
            activado = false;
            ui->IRDY->setChecked(false);
            ui->TRDY->setChecked(false);
            ui->FRAME->setChecked(false);
            ui->DEVSEL->setChecked(false);
            ui->DIR->clear();
            ui->MASK->clear();
            return;
        }
    }
    else
    {
        ui->IRDY->setChecked(true);
        ui->TRDY->setChecked(true);
        ui->DEVSEL->setChecked(true);
        irdy = true;
        trdy = true;
        devset = true;
        activado = true;
    }
}
else
{       // Iniciamos el proceso
    dir = ui->DIR->toPlainText().toInt()-1;
    // Si intentamos insertar en una memoria no válida
    // no lo permitimos
    if((dir<0)||(dir>9))
        return;
    inicializado = true;
}
}
}
```

```
void MainWindow::on_READ_clicked()
{
   LECTURA = true;
}

void MainWindow::on_WRITE_clicked()
{
   LECTURA = false;
}
```

Descripción de los resultados

La interfaz gráfica es la siguiente:

Donde podemos ver que hay 3 cuadros de texto. El primero es el lugar donde indicamos la dirección de memoria con la que nos vamos a comunicar; el segundo es el lugar donde insertamos el dato a escribir o donde obtenemos el dato leído; y el tercer campo es la máscara de 4 bytes que aplicaremos en la operación.

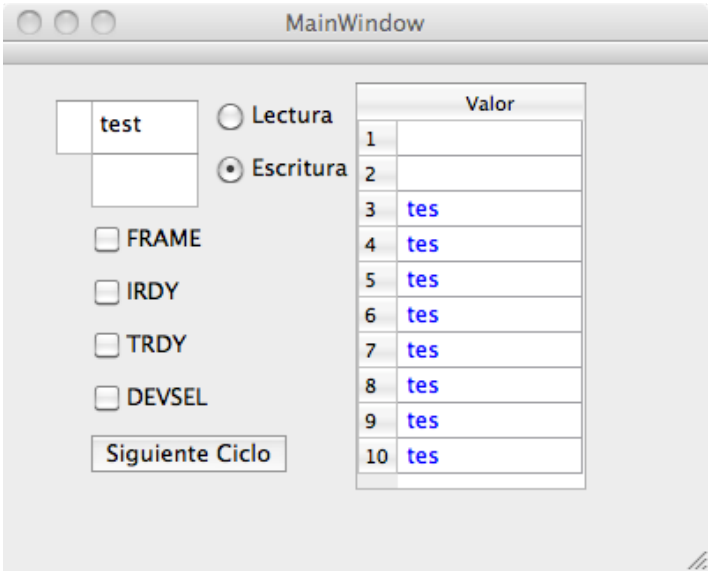
Supongamos que deseamos escribir un dato en la posición 3 de la memoria, por ejemplo, deseamos insertar los 3 primeros bytes de la palabra “test”:

Tras insertar los datos y seleccionar la operación que vamos a realizar, activamos la señal FRAME:

A partir de ahora, lo único que debemos es pulsar el botón de “Siguiente Ciclo”, mientras vemos como se van activando las señales

y se va escribiendo en la memoria:

esto se llevará a cabo tantas veces como queramos, y cada “iteración” incrementará el puntero a la dirección de memoria donde vamos a escribir, hasta llegar al final de la misma, donde finalizará la operación:



En el caso que queramos leer, sólo debemos seleccionar la operación de lectura, indicar la dirección y elegir la máscara. En este caso, leeremos el 2º byte de la dirección 3:

Al igual que con la escritura, activamos la señal de FRAME y el resto de señales se activarán a medida que avancemos y comenzará la lectura.

Conclusiones

El control del BUS PCI es bastante sencillo, por lo que su implementación no ha requerido un gran esfuerzo. Aún así, no podemos simular todos los posibles casos, como un acceso a un registro dañado. De todas formas, esta práctica ha facilitado las labores de comprensión del BUS PCI, pasando de ser una mera cuestión teórica a verlo de manera práctica.