# Proyecto Optativo 2: Diseño de un controlador de bus

Fernando Aliaga Ramón - 610610

## Diseño: diagrama de estados

El controlador de Bus debe gestionar dos tipos de operaciones: lectura y escritura. Los ciclos son los siguientes:

**Rango = 0**

Estado 0  
Lectura instrucción

**Rango = 1**

**L/E = 1**

Estado 1  
Primer ciclo RAM

**L/E = 0**

Estado 2  
Segundo ciclo lectura

Estado 3  
Soltar BUS

**Master\_Ready = 0**

**Master\_Ready = 1**

Debemos trasladar este diagrama de estados a una función lógica:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| E1 | E0 | MR | LE | E+ | E1+ | E0+ |
| 0 | **0** | **0** | **0** | **1** | **0** | **1** |
| 0 | **0** | **0** | **1** | **1** | **0** | **1** |
| 0 | **0** | **1** | **0** | **1** | **0** | **1** |
| 0 | **0** | **1** | **1** | **1** | **0** | **1** |
| 0 | 1 | 0 | 0 | 2 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 2 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | **0** | **0** | **0** | **3** | **1** | **1** |
| 1 | **0** | **0** | **1** | **3** | **1** | **1** |
| 1 | **0** | **1** | **0** | **2** | **1** | **0** |
| 1 | **0** | **1** | **1** | **2** | **1** | **0** |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Usando un mapa de Karnaugh podemos definir las funciones lógicas de E1+ y E0+:

## E1+

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E1 E0/MR LE** | **00** | **01** | **11** | **10** |
| **00** | 0 | 0 | 0 | 0 |
| **01** | 1 | 0 | 0 | 1 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 1 | 1 | 1 | 1 |

## E0+

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E1 E0/MR LE** | **00** | **01** | **11** | **10** |
| **00** | 1 | 1 | 1 | 1 |
| **01** | 0 | 0 | 0 | 0 |
| **11** | 0 | 0 | 0 | 0 |
| **10** | 1 | 1 | 0 | 0 |

Hemos definido todos los casos posibles respecto al estado y las entradas Master\_Ready y Reg\_LE. Sin embargo, debemos definir una lógica para la señal Rango:

## Slave\_Ready

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| E1 | E0 | Rango | E1+ | E0+ |
| 0 | **0** | 0 | 0 | 0 |
| 0 | **0** | 1 | 0 | 1 |
| 0 | **1** | 0 | L/E | L/E |
| 0 | **1** | 1 | L/E | L/E |
| 1 | **0** | 0 | M\_R | M\_R |
| 1 | **0** | 1 | M\_R | M\_R |
| 1 | **1** | 0 | 0 | 0 |
| 1 | **1** | 1 | 0 | 0 |

La señal de Rango la usamos cuando estamos en el estado 0 para comenzar o no una operación. En los demás estados se ignora y calcularemos el estado siguiente con el resto de señales independientemente del valor de Rango.

Podemos ahora crear las funciones lógicas de E1+ y E0+:

## Diseño: señales de control

Para cada estado tendremos unas señales de control. Estudiaremos cada señal por separado:

## Load\_Input

|  |  |  |
| --- | --- | --- |
| E1 | E0 | Load\_Input |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Load\_Input = (nor)

## Load\_Output

|  |  |  |
| --- | --- | --- |
| E1 | E0 | Load\_Output |
| 0 | **0** | 0 |
| 0 | **1** | 0 |
| 1 | **0** | 1 |
| 1 | **1** | 0 |

Load\_Output =

## Send\_Data

|  |  |  |
| --- | --- | --- |
| E1 | E0 | Send\_Data |
| 0 | **0** | 1 |
| 0 | **1** | 1 |
| 1 | **0** | 1 |
| 1 | **1** | 0 |

Send\_Data = (nand)

## Chip\_Select

|  |  |  |
| --- | --- | --- |
| E1 | E0 | Chip\_Select |
| 0 | **0** | 1 |
| 0 | **1** | 0 |
| 1 | **0** | 0 |
| 1 | **1** | 1 |

Chip\_Select = (xnor)

## Write\_Mem

|  |  |  |  |
| --- | --- | --- | --- |
| E1 | E0 | Reg\_LE | Write\_Mem |
| 0 | **0** | 0 | 1 |
| 0 | **0** | 1 | 1 |
| 0 | **1** | 0 | 1 |
| 0 | **1** | 1 | 0 |
| 1 | **0** | 0 | 1 |
| 1 | **0** | 1 | 1 |
| 1 | **1** | 0 | 1 |
| 1 | **1** | 1 | 1 |

Write\_Mem =

## Slave\_Ready

|  |  |  |  |
| --- | --- | --- | --- |
| E1 | E0 | Reg\_LE | Slave\_Ready |
| 0 | **0** | 0 | 0 |
| 0 | **0** | 1 | 0 |
| 0 | **1** | 0 | 0 |
| 0 | **1** | 1 | 1 |
| 1 | **0** | 0 | 1 |
| 1 | **0** | 1 | 1 |
| 1 | **1** | 0 | 1 |
| 1 | **1** | 1 | 1 |

Slave\_Ready =

## Slave\_Sel

|  |  |  |  |
| --- | --- | --- | --- |
| E1 | E0 | Rango | Slave\_Ready |
| 0 | **0** | 0 | 0 |
| 0 | **0** | 1 | 1 |
| 0 | **1** | 0 | 1 |
| 0 | **1** | 1 | 1 |
| 1 | **0** | 0 | 1 |
| 1 | **0** | 1 | 1 |
| 1 | **1** | 0 | 1 |
| 1 | **1** | 1 | 1 |

Slave\_Sel =

## Resumen Señales de Control

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E1 | E0 | L\_I | L\_O | S\_D | C\_S | W\_M | S\_R | S\_S |
| 0 | **0** | 1 | 0 | 1 | 1 | 1 | 0 | Rango |
| 0 | **1** | 0 | 0 | 1 | 0 | notReg\_LE | Reg\_LE | 1 |
| 1 | **0** | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | **1** | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

## Evaluación temporal

#### ¿Cuántos ciclos tarda como mínimo en gestionar una petición de lectura?

4 ciclos de reloj como mínimo. Si el dato está en el bus y la señal de master\_ready está a 0 el esclavo mantiene el dato tantos ciclos como tarde en valer dicha señal 1.

#### ¿Y de escritura?

2 ciclos de reloj, uno que decodifique la instrucción y otro que lo escriba en la memoria RAM.

## Cronogramas

Vamos a estudiar el cronograma del circuito en 3 casos:

1. Escritura.
2. Lectura sin espera de master\_ready.
3. Lectura con espera de master\_ready.

## Escritura

## 4

## 5

## 3

## 2

## 1

Eventos del cronograma:

1. Comienzo primer ciclo de la escritura
2. El esclavo genera las señales de control. Decodifica la instrucción para distinguir entre lectura y escritura.
3. Comienzo segundo ciclo de escritura. Se escribirá el dato en la memoria RAM.
4. El esclavo genera las señales que permiten escribir en la memoria RAM (chip\_select=0 & write\_mem=0).
5. Fin operación escritura.

## Lectura sin espera

## 9

## 8

## 7

## 6

## 5

## 4

## 3

## 2

## 1

Eventos del cronograma:

1. Comienzo primer ciclo de la lectura.
2. El esclavo genera las señales de control. Decodifica la instrucción para distinguir entre lectura y escritura.
3. Comienzo segundo ciclo de lectura y primer ciclo de uso de la RAM. La lectura de la RAM se realizará en 2 ciclos (retardoRAM = 22ns).
4. El esclavo genera las señales que permiten leer en la memoria RAM (chip\_select=0 & write\_mem=1).
5. Comienzo tercer ciclo de lectura y segundo ciclo de uso de la RAM.
6. Activa la entrada LOAD del registro de la salida de la RAM. Mantendrá el dato leído hasta que la señal Master\_ready valga 0.
7. Comienzo cuarto ciclo de la operación de lectura.
8. Como Master\_ready=0 el esclavo libera el Bus cambiando la señal de control del triestado (send\_data=0).
9. Fin operación lectura.

## Lectura con espera

## 4

## 3

## 2

## 1

Eventos del cronograma:

1. El Master activa la señal de Master\_ready para indicar que no podrá leer hasta que no la desactive.
2. Comienzo cuarto ciclo de lectura. Como la señal Master\_ready sigue valiendo 0 el esclavo mantendrá el dato en el bus.
3. El Master desactiva la señal Master\_Ready, por lo que podrá leer el dato. Como la señal ha sido cambiada a mitad de ciclo el esclavo liberará el Bus en el ciclo siguiente.
4. El esclavo libera el Bus y el Master lee el dato.