Clase 10

Práctico de Diseño Lógico

Clase 10 – RTL

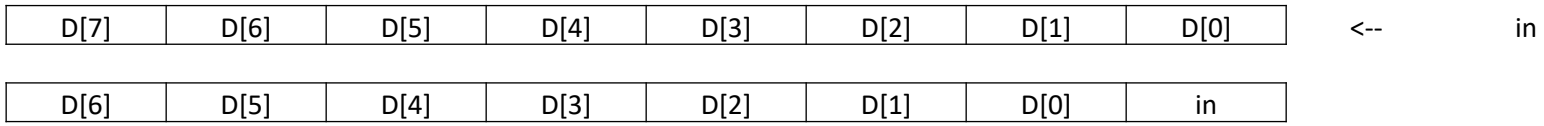
- Repaso rápido de RTL
- Shift-register en RTL
- Ejercicio de diseño RTL
  - Secuencia
  - Circuito

Repaso rápido RTL

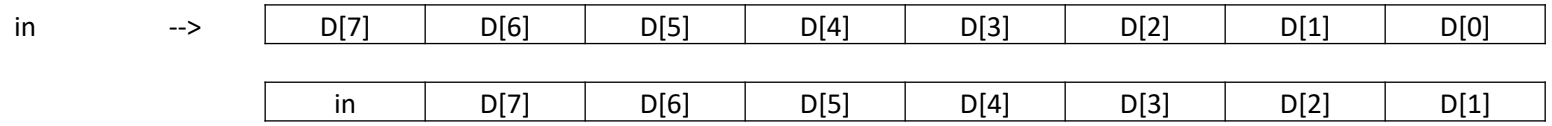
- Estructura de circuito RTL:
  - Bloque de Control.
  - Bloque de Datos.
  - Señales de control.
  - Señales de reloj. Usamos FF con Enable para las transferencias de datos y el flanco activo es el de subida.

- Descripción RTL
  - Encabezado (inputs, outputs, nodes, memory).
  - Lo que está fuera del ENDSEQUENCE está fijo, es válido para todos los pasos.
  - Control Reset indica el paso de inicio.

Shift register en RTL



1. D[7..0] ← D[6..0], in



1. D[7..0] ← in, D[7..1]

Ej 8 práctico 9

Ejercicio 8. (H&P 84. Prob. 10.12)
Un sistema digital tiene dos líneas de entrada, **control** y **data**, 8 líneas de salida **Z**[8], y una línea de salida **ready**. Los datos representando palabras de 8 bits llegarán serialmente (un bit por vez) sobre la línea **data**. Será convertido a formato paralelo para sacarlo por las líneas **Z**. El primer bit en llegar de los datos serie puede ser el más significativo o el menos significativo. En cualquier caso la salida paralelo debe ser de la forma normal, con el más significativo a la izquierda.
Se provee sincronización por la interacción entre la entrada **control** y la salida **ready**. Cuando el sistema está pronto para recibir una nueva palabra, subirá la línea **ready** y empezará a monitorear **control**. Dos unos consecutivos (igual a uno durante dos períodos de reloj) en **control** indicará que el siguiente bit en control indicará la dirección de los datos serie, 0 para el MSB primero, 1 para el LSB primero. Los 8 bits de datos aparecerán a continuación en **data** durante los siguientes 8 períodos de reloj. **Ready** deberá bajar cuando los dos unos consecutivos en **control** se detecten y deberá volver a 1 cuando los 8 bits hayan sido recibidos. La palabra paralelo deberá presentarse en las líneas **Z** siempre que **ready** esté alta.
Realizar la descripción RTL del sistema descrito y dibujar los bloques de datos y control.

Entradas y salidas del sistema

Un sistema digital tiene dos líneas de entrada, **control** y **data**, 8 líneas de salida **Z**[8], y una línea de salida **ready**. Los datos representando palabras de 8 bits llegarán

- Empezamos esqueleto de la secuencia

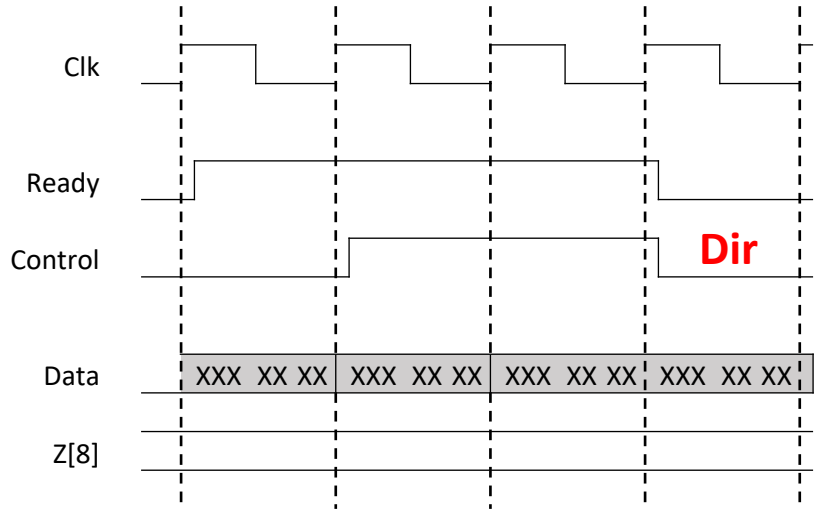

```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY:
OUTPUTS: Z[8], ready
```

Un sistema digital tiene dos líneas de entrada, **control** y **data**, 8 líneas de salida **Z**[8], y una línea de salida **ready**. Los datos representando palabras de 8 bits llegarán serialmente (un bit por vez) sobre la línea **data**. Será convertido a formato paralelo para sacarlo por las líneas **Z**. El primer bit en llegar de los datos serie puede ser el más significativo o el menos significativo. En cualquier caso la salida paralelo debe ser de la forma normal, con el más significativo a la izquierda.

Dirección de los datos (MSB o LSB primero)

Se provee sincronización por la interacción entre la entrada **control** y la salida **ready**. Cuando el sistema está pronto para recibir una nueva palabra, subirá la línea **ready** y empezará a monitorear **control**. Dos unos consecutivos (igual a uno durante dos períodos de reloj) en **control** indicará que el siguiente bit en control indicará la dirección de los datos serie, 0 para el MSB primero, 1 para el LSB primero. Los 8 bits de datos

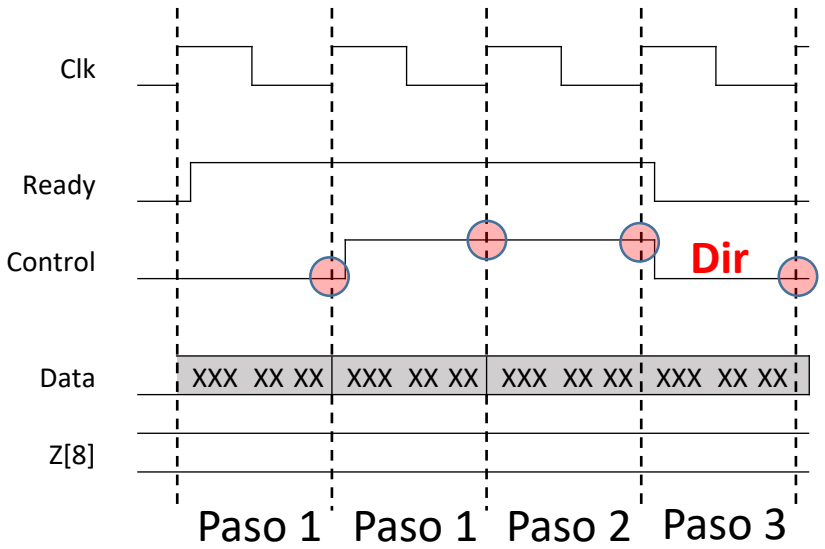
- Tenemos que subir Ready y esperar 2 Unos consecutivos en Control
- Vamos a necesitar una memoria para guardar el sentido en que vienen los datos.




```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir
OUTPUTS: Z[8], ready
1.
```

Esperamos los dos Unos en Control y guardamos la dirección de los datos

- Tenemos que subir Ready y esperar 2 Unos consecutivos en Control




```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir
OUTPUTS: Z[8], ready

1. Ready = 1
   →(Control, !Control)/(2,1)
2. Ready = 1
   →(Control, !Control)/(3,1)
3. Dir ← Control
```

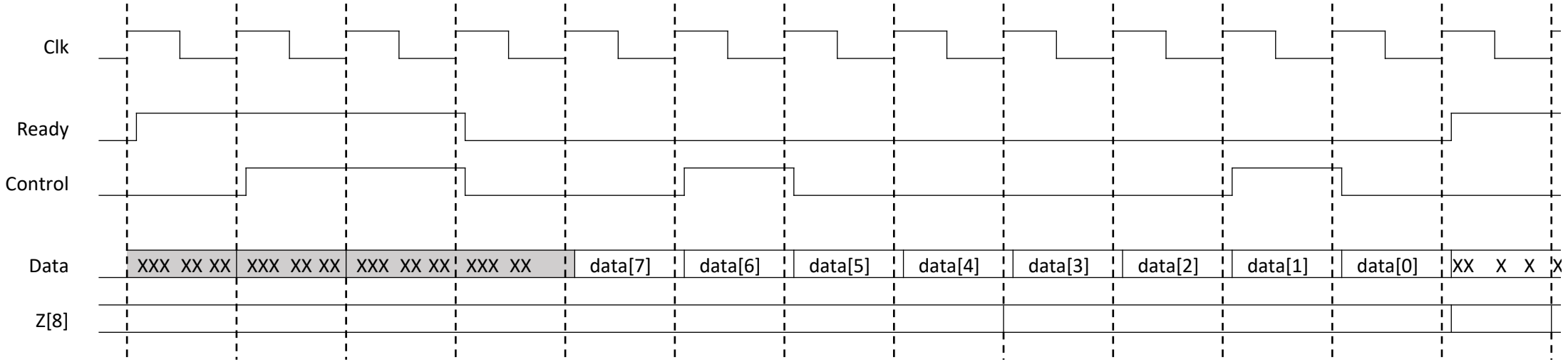
aparecerán a continuación en **data** durante los siguientes 8 períodos de reloj. **Ready** deberá bajar cuando los dos unos consecutivos en **control** se detecten y deberá volver a 1 cuando los 8 bits hayan sido recibidos. La palabra paralelo deberá presentarse en las líneas **Z** siempre que **ready** esté alta.

```
End Sequence
Control Reset (1)
```

Recepción de datos

- Necesitamos una memoria de tamaño 8 para ir guardando los bits que vamos recibiendo de a 1 con un shift-register.
- Y un contador para recibir 8 bits.

```
4. Palabra[7..0] ←
   cont ←
   →(  ,  )/(  ,  )
```



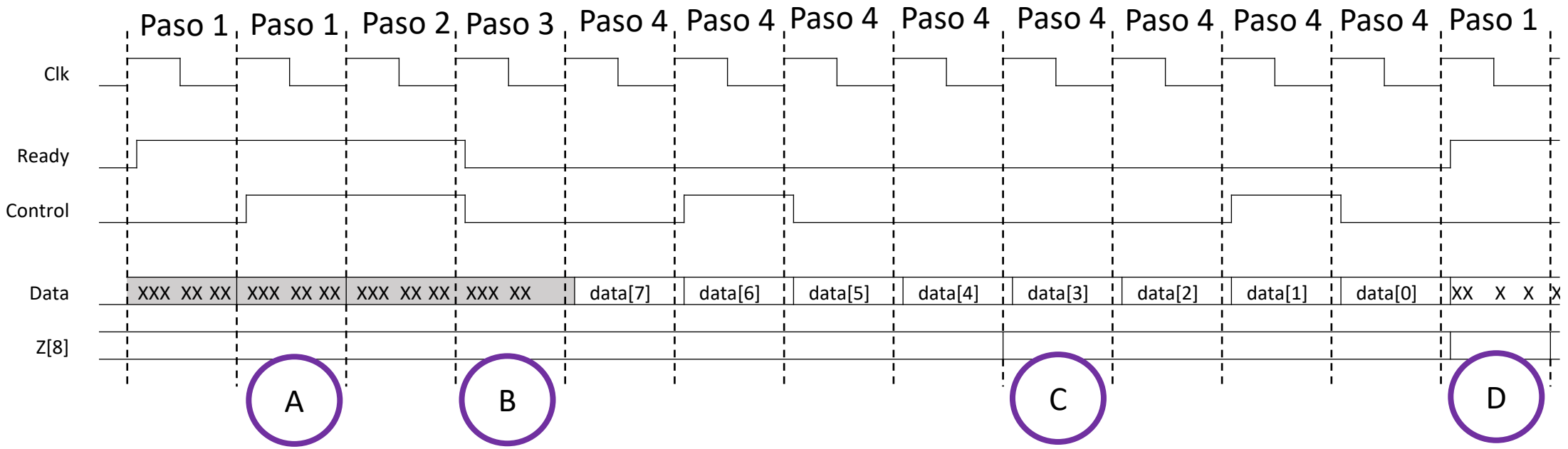
Secuencia

```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir, Palabra[8], cont[3]
OUTPUTS: Z[8], ready

1. Ready = 1
   →(Control, !Control)/(2,1)
   cont ← 000
2. Ready = 1
   →(Control, !Control)/(3,1)
3. Dir ← Control
4. Palabra[7..0] ← !Dir.{Palabra[6..0],data} + Dir.{data, Palabra[7..1]}
   cont ← inc(cont)
   →(cont=111, !(cont=111))/(1,4)

End Sequence
Z = Palabra
Control Reset (1)
End
```

PASOS:

1. Ready = 1
   → (Control, !Control)/(2,1)
   cont ← 000
2. Ready = 1
   → (Control, !Control)/(3,1)
3. Dir ← Control
4. Palabra[7..0] ← !Dir.{Palabra[6..0],data} + Dir.{data, Palabra[7..1]}
   cont ← inc(cont)
   → (cont=111, !(cont=111))/(1,4)

Y en todos los pasos: Z = Palabra



A partir de la secuencia se sintetiza el circuito

```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir, Palabra[8], cont[3]
OUTPUTS: Z[8], ready

1. Ready = 1
   →(Control, !Control)/(2,1)
   cont ← 000
2. Ready = 1
   →(Control, !Control)/(3,1)
3. Dir ← Control
4. Palabra[7..0] ← !Dir.{Palabra[6..0],data} + Dir.{data, Palabra[7..1]}
   cont ← inc(cont)
   →(cont=111, !(cont=111))/(1,4)

End Sequence
Z = Palabra
Control Reset (1)
End
```

Bloque de control

```
1. →(Control, !Control)/(2,1)
2. →(Control, !Control)/(3,1)
3.
4. →(cont=111, !(cont=111))/(1,4)
Control Reset (1)
```

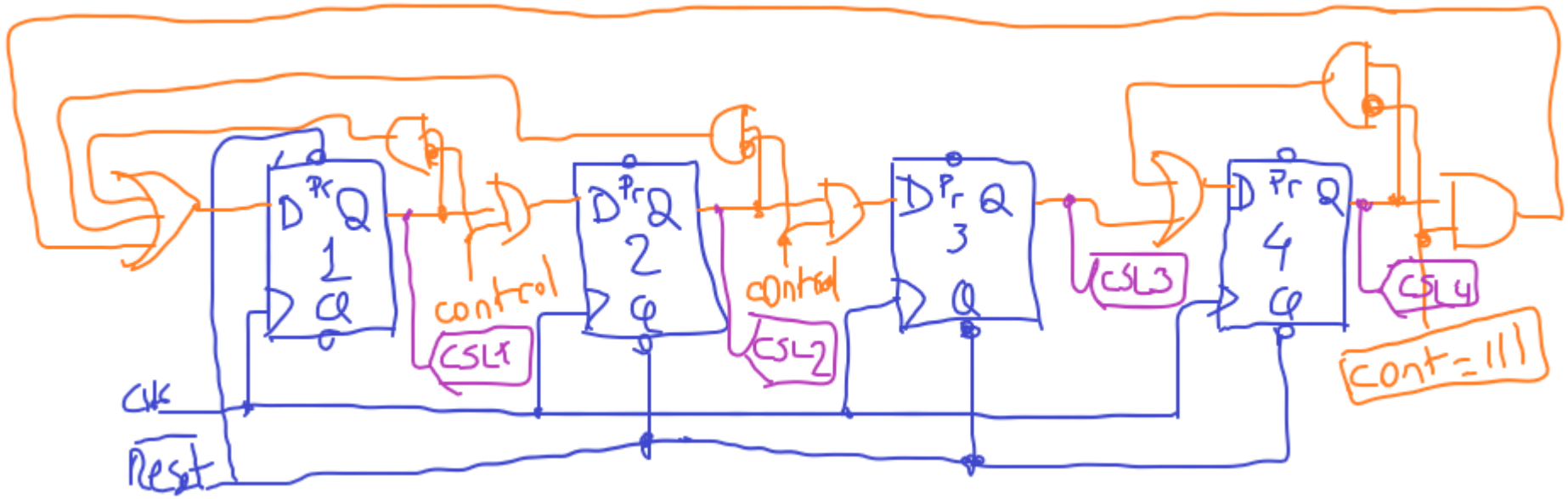
- 1 FF por paso
- Comenzamos en el paso 1 luego de un reset.
- Se generan señales de control

Bloque de control

1. →(Control, !Control)/(2,1)
2. →(Control, !Control)/(3,1)
3.
4. →(cont=111, !(cont=111))/(1,4)

Control Reset (1)

- 1 FF por paso
- Comenzamos en el paso 1 luego de un reset.
- Se generan señales de control



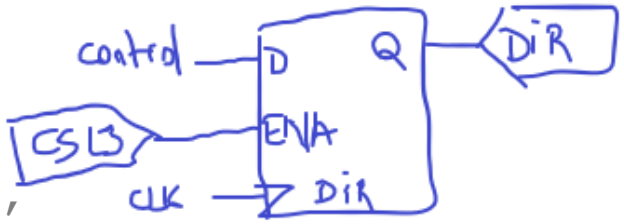
Bloque de datos

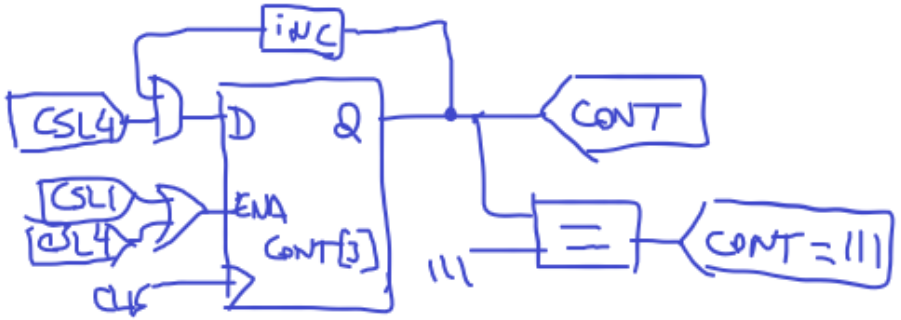
```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir, Palabra[8],
cont[3]
OUTPUTS: Z[8], ready

1. Ready = 1
   cont ← 000
2. Ready = 1
3. Dir ← Control
4. Palabra[7..0] ←
!Dir.{Palabra[6..0],data} +
Dir.{data, Palabra[7..1]}
   cont ← inc(cont)
En todos los pasos:
Z = Palabra
```

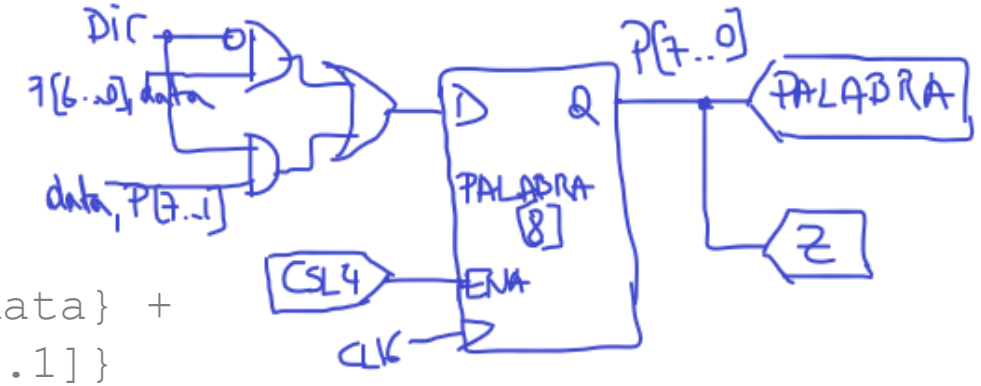
- FF con enable
- Un registro por cada memoria
- El = es un cable
- Se generan señales de control

Bloque de datos

```
MODULE Serie_a_Paralelo
INPUTS: control, data
MEMORY: Dir, Palabra[8],
cont[3]
OUTPUTS: Z[8], ready

1. Ready = 1
   cont ← 000
2. Ready = 1
3. Dir ← Control
4. Palabra[7..0] ←
!Dir.{Palabra[6..0],data} +
Dir.{data, Palabra[7..1]}
   cont ← inc(cont)
En todos los pasos:
Z = Palabra
```







- FF con enable
- Un registro por cada memoria
- El = es un cable
- Se generan señales de control