Clase 9

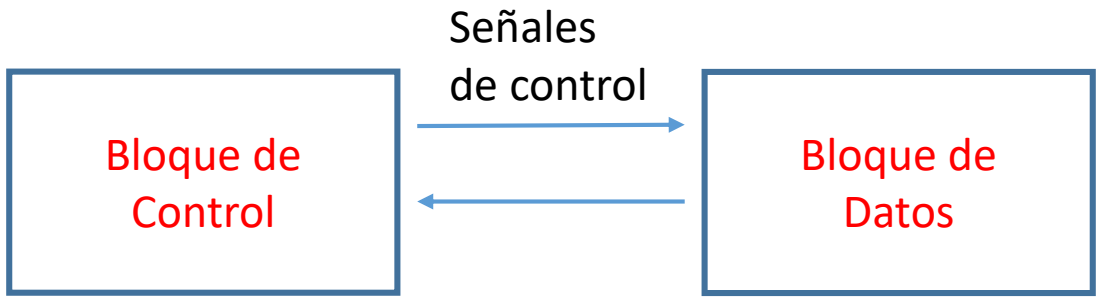
Práctico de Diseño Lógico

Clase 9 – RTL (Introducción)

- Lenguaje de especificación de Hardware
- Estructura de circuito RTL
- Descripción RTL
- Tiempos

Estructura de circuito RTL (Repaso)

- 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 (como los FF para el laboratorio).



Descripción RTL

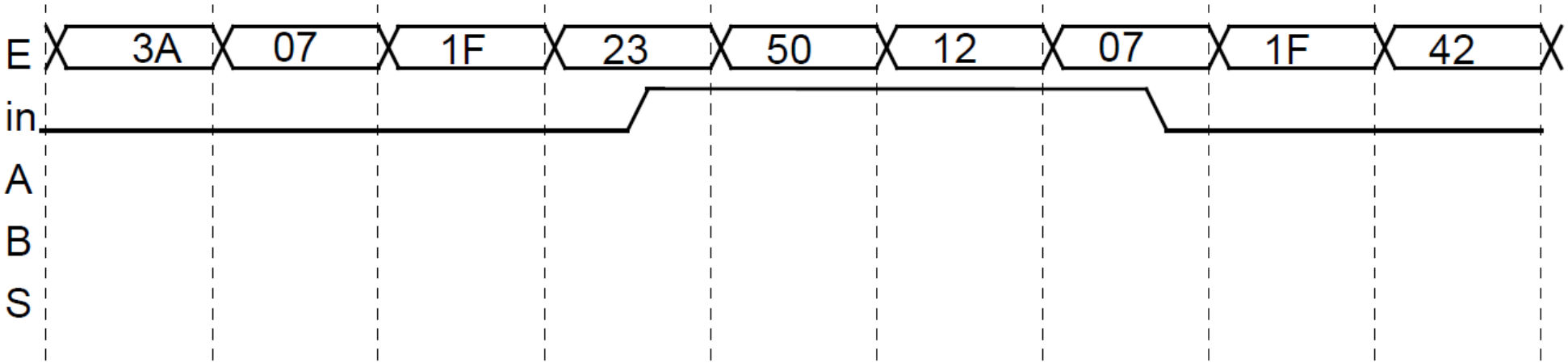
- Encabezado (nombre y declaración inputs, outputs, nodes, memory)
- Lo que está fuera del ENDSEQUENCE está fijo, es válido para todos los pasos.
- Control Reset indica el paso inicial.

```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
       1. A ← E
       2. A ← A+1, S=A, B ← E
       3. A ← A+1, S = A+B
       4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```

Ejercicio 3 - Práctico 9

Ejercicio 3. (ex. Marzo 1996)
Dada la siguiente secuencia RTL construir el bloque de datos, el bloque de control y completar el siguiente diagrama de tiempos, suponiendo que en el primer período se está en el paso 1:

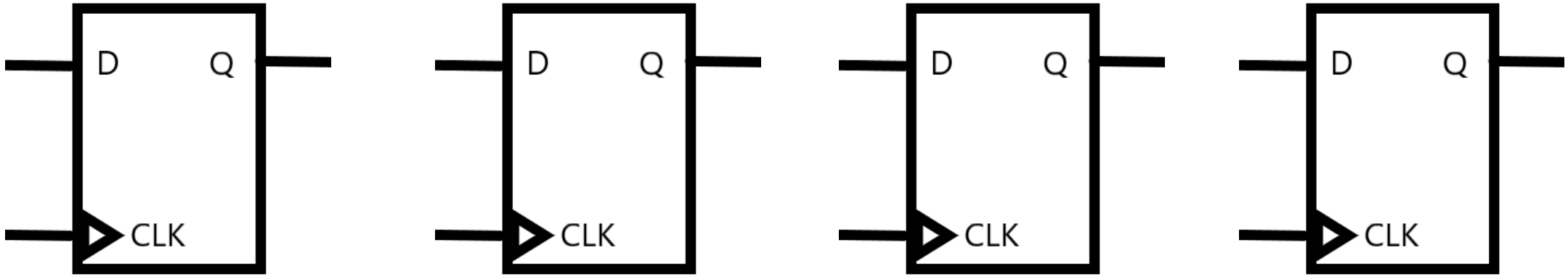
```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
       1. A ← E
       2. A ← A+1, S=A, B ← E
       3. A ← A+1, S = A+B
       4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```



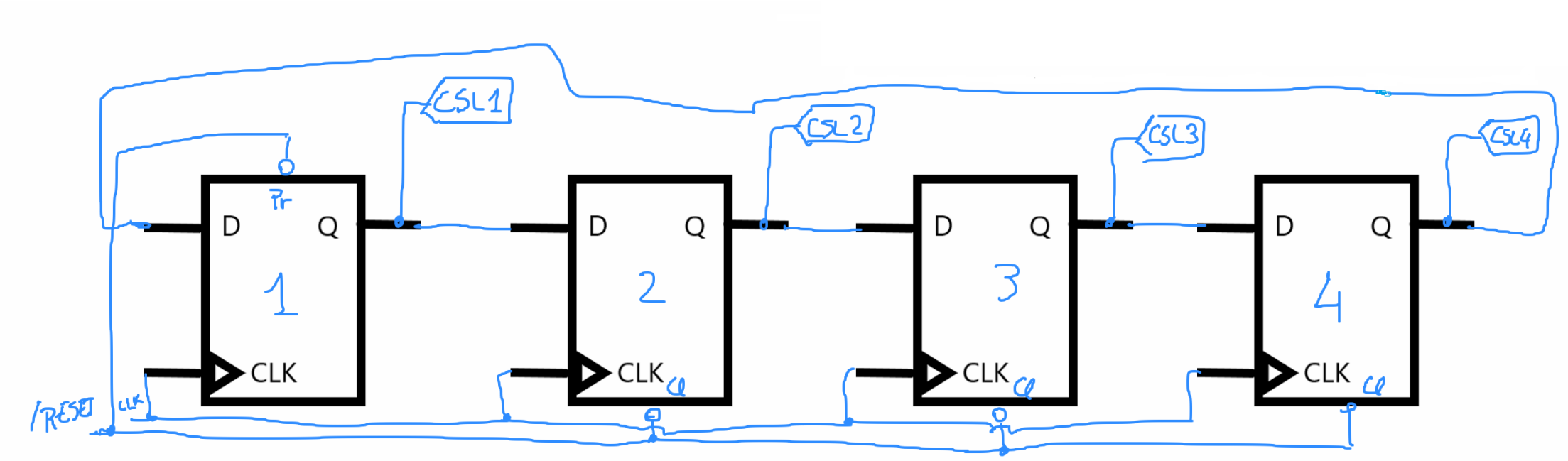
Bloque de control

- Necesitamos 1 FF por paso
- La salida de cada FF i va a ser la señal de control CSLi.
- La señal de /reset se conecta al preset del FF del paso de inicio indicado en Controlreset y al clear del resto de los FF.

```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
       1. A ← E
       2. A ← A+1, S=A, B ← E
       3. A ← A+1, S = A+B
       4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```

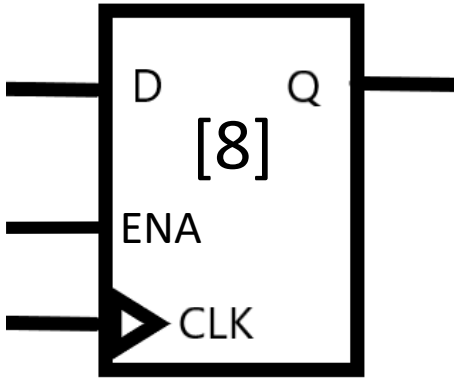


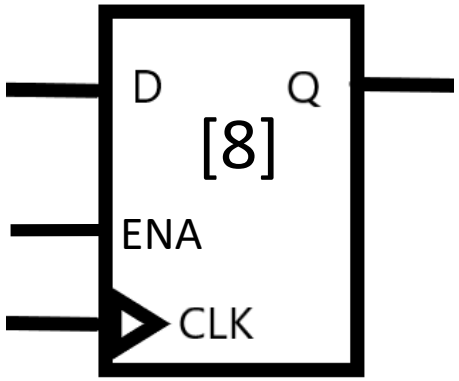
Bloque de control



Bloque de datos

- Necesitamos 1 regsitro CON ENABLE para cada memoria.





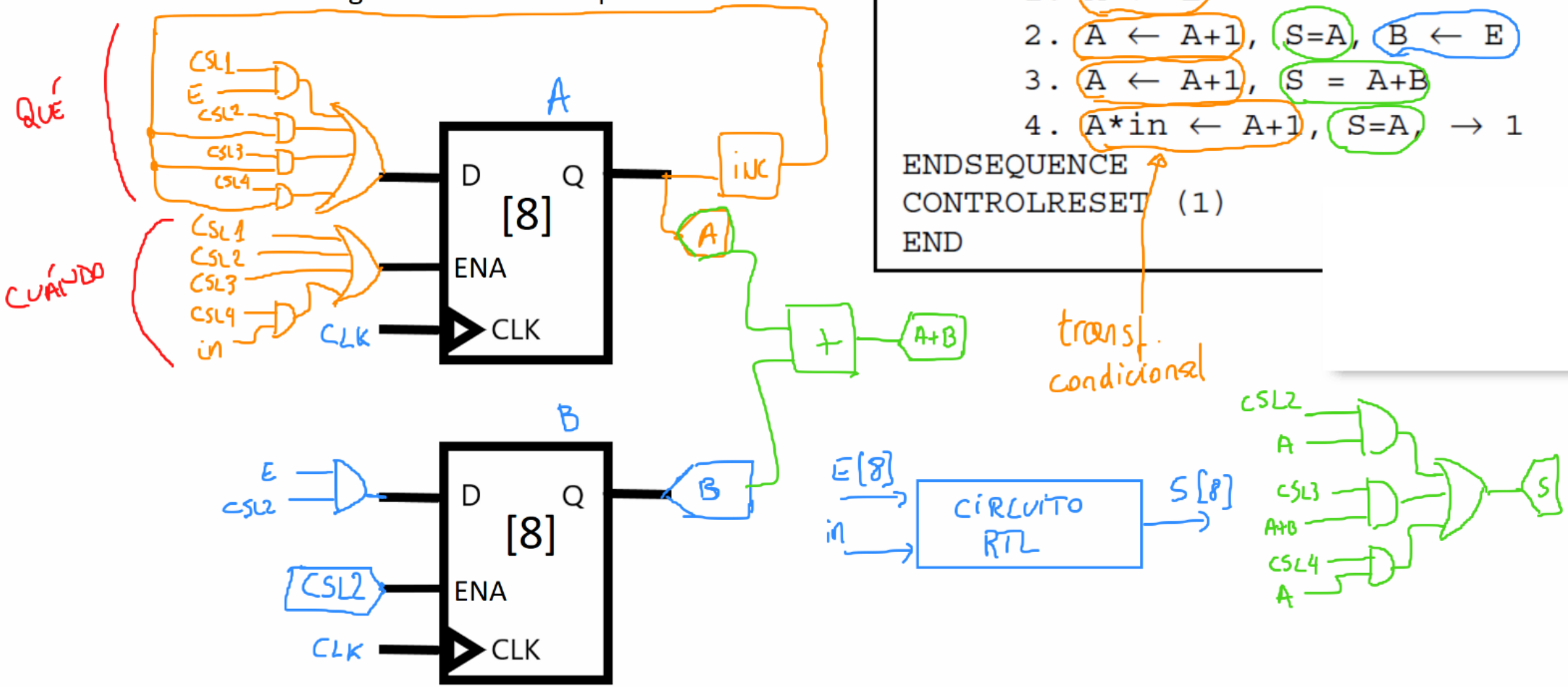

```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
      1. A ← E
      2. A ← A+1, S=A, B ← E
      3. A ← A+1, S = A+B
      4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```

- Generamos la salida

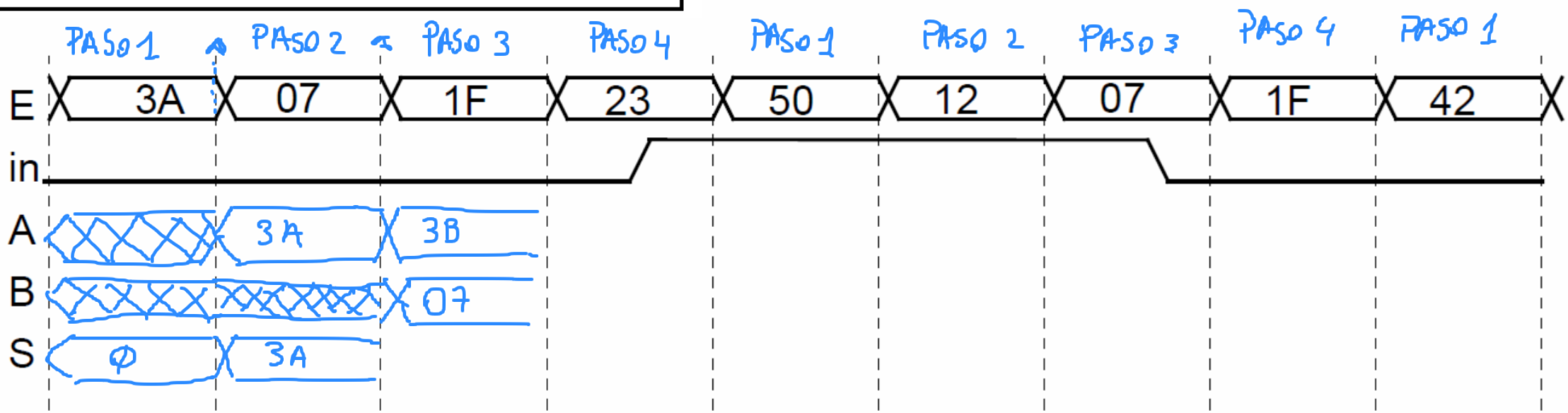
Bloque de

- Necesitamos 1 regsitro CON ENABLE para cada memoria.

```
INPUTS:     A[8], B[8]
            E[8], in
OUTPUTS:    S[8]
        1. A ← E
        2. A ← A+1, S=A, B ← E
        3. A ← A+1, S = A+B
        4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```



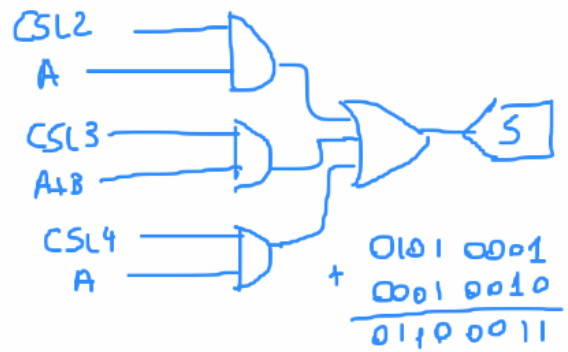
```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
       1. A ← E
       2. A ← A+1, S=A, B ← E
       3. A ← A+1, S = A+B
       4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```

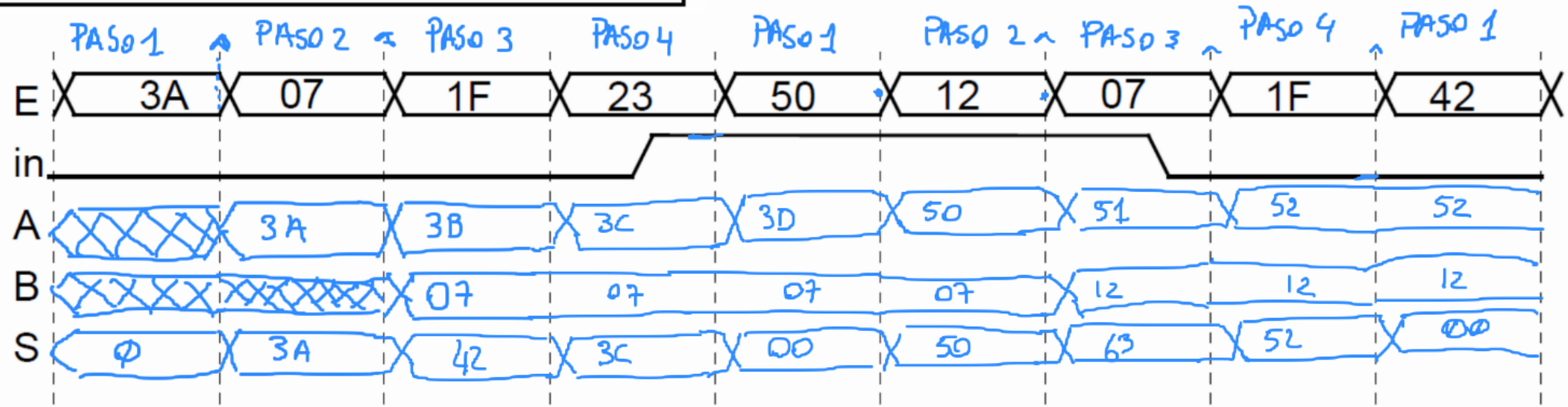


```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
       1. A ← E
       2. A ← A+1, S=A, B ← E
       3. A ← A+1, S = A+B
       4. A*in ← A+1, S=A, → 1
ENDSEQUENCE
CONTROLRESET (1)
END
```

S=0

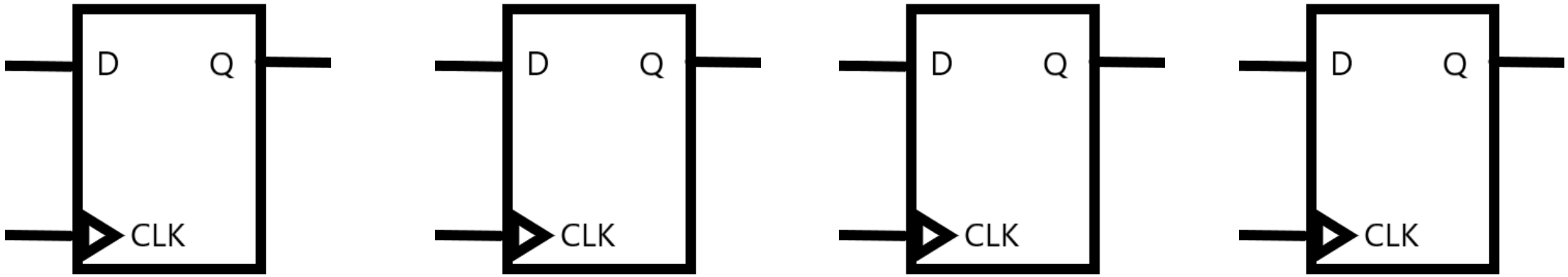
CSL1 = 1





Modificamos paso 4:

```
MODULE Ej_3
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
      1. A ← E
      2. A ← A+1, S=A, B ← E
      3. A ← A+1, S = A+B
      4. A*in <-- A+1, S=A, --> (A=2,!(A=2))/(4,1)
ENDSEQUENCE
CONTROLRESET (1)
END
```



Modificamos paso 4:

comp (en B.D)

→ (A=2 , $\overline{A=2}$) / (4,1)

```
sharing   Stop Share
MEMORY:      A[8], B[8]
INPUTS:      E[8], in
OUTPUTS:     S[8]
      1. A ← E
      2. A ← A+1, S=A, B ← E
      3. A ← A+1, S = A+B
      4. A*in <-- A+1, S=A, --> (A=2,!(A=2))/(4,1)
ENDSEQUENCE
CONTROLRESET (1)
END
```