



Tema 8:

Rutas de datos y controladores

Fundamentos de computadores I

José Manuel Mendías Cuadros

*Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid*





Contenidos

- ✓ Estructura.
- ✓ Modelo de cómputo.
- ✓ Metodología de diseño.
- ✓ Diseño de un multiplicador iterativo.
- ✓ Diseño de un calculador del máximo común divisor.

- ✓ Apéndice tecnológico.

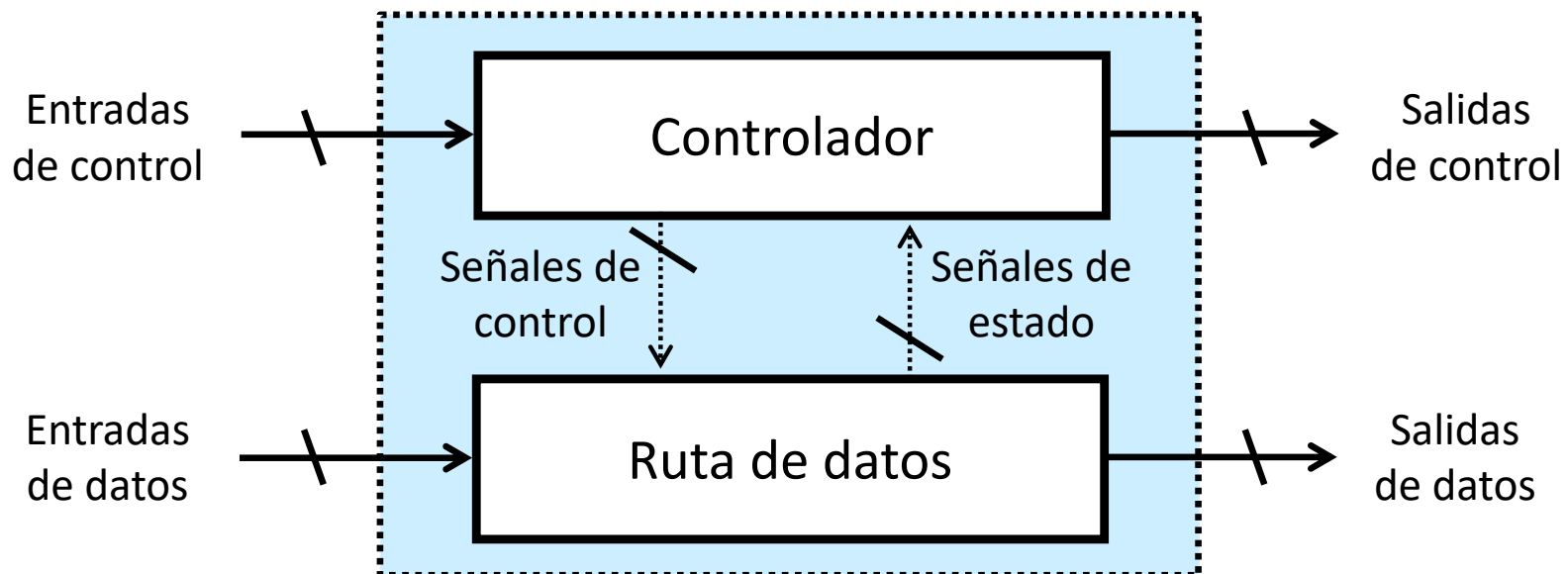
Transparencias basadas en los libros:

- R. Hermida, F. Sánchez y E. del Corral. *Fundamentos de computadores.*
- D. Gajsky. *Principios de diseño digital.*



Estructura

- Cuando un sistema secuencial realiza un **algoritmo complejo** **no es viable** especificarlo mediante un diagrama de estados.
- Los sistemas complejos se diseñan interconectando:
 - **Ruta de datos:** que realiza las operaciones y almacena resultados parciales.
 - **Controlador:** secuencia la realización de las operaciones en la ruta de datos.

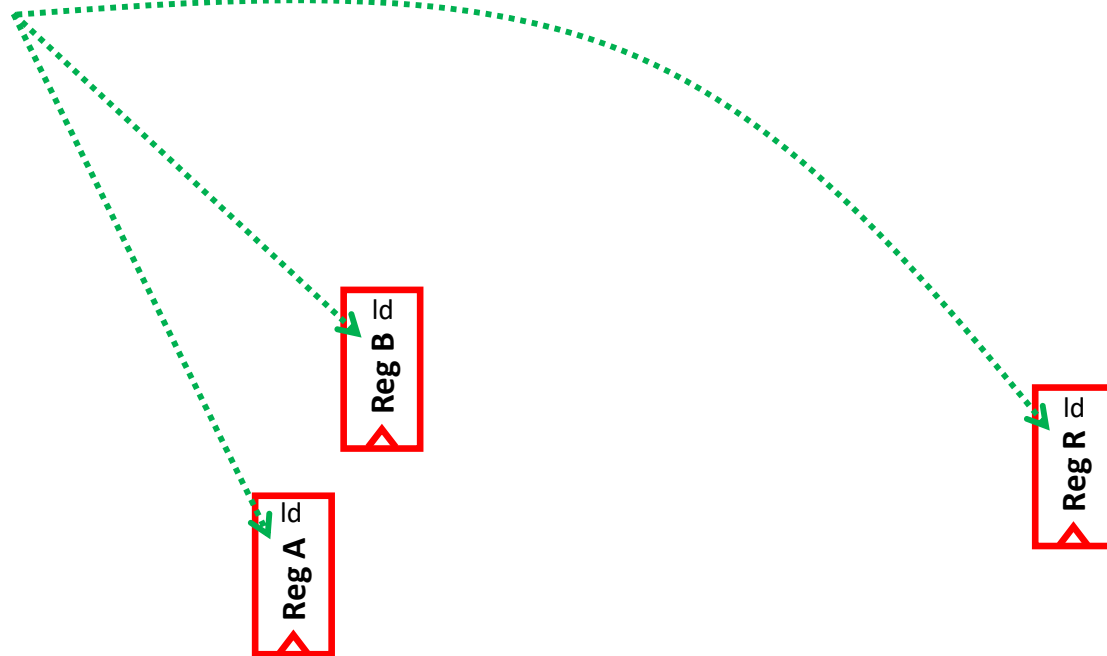




Estructura

- Toda **ruta de datos** está formada por:
 - **Módulos secuenciales**: registros, contadores, desplazadores ...

Los registros
almacenan los datos



RUTA DE DATOS

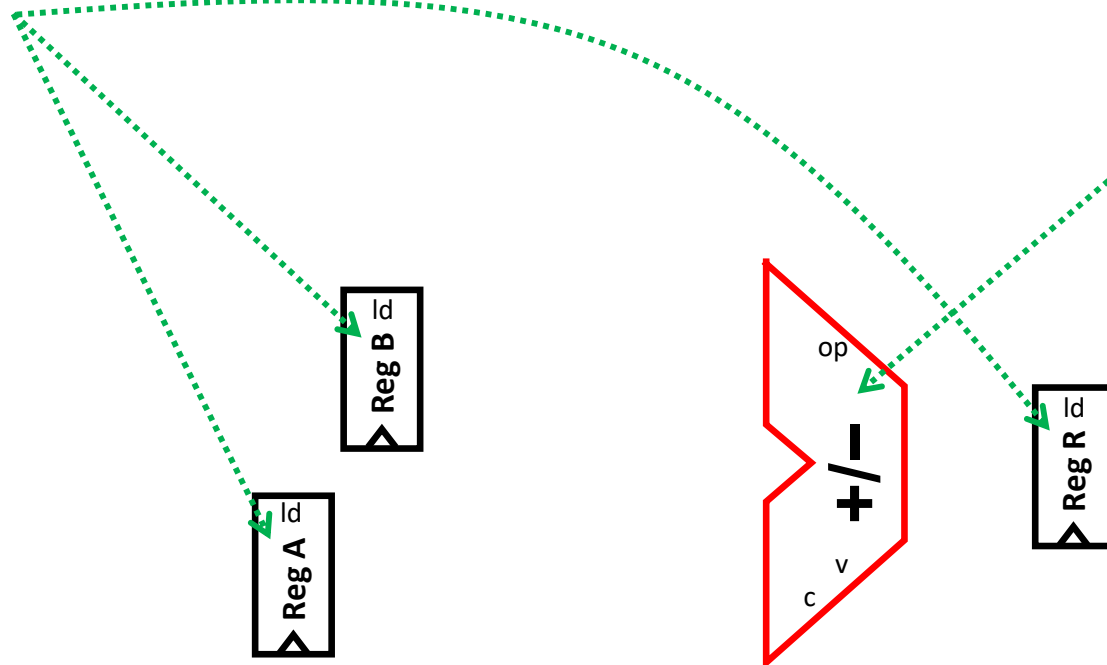


Estructura

- Toda **ruta de datos** está formada por:
 - Módulos **secuenciales**: registros, contadores, desplazadores ...
 - Módulos **combinacionales**: sumadores, ALU, MUX, lógica ...

Los registros
almacenan los datos

Las unidades funcionales
efectúan los cálculos

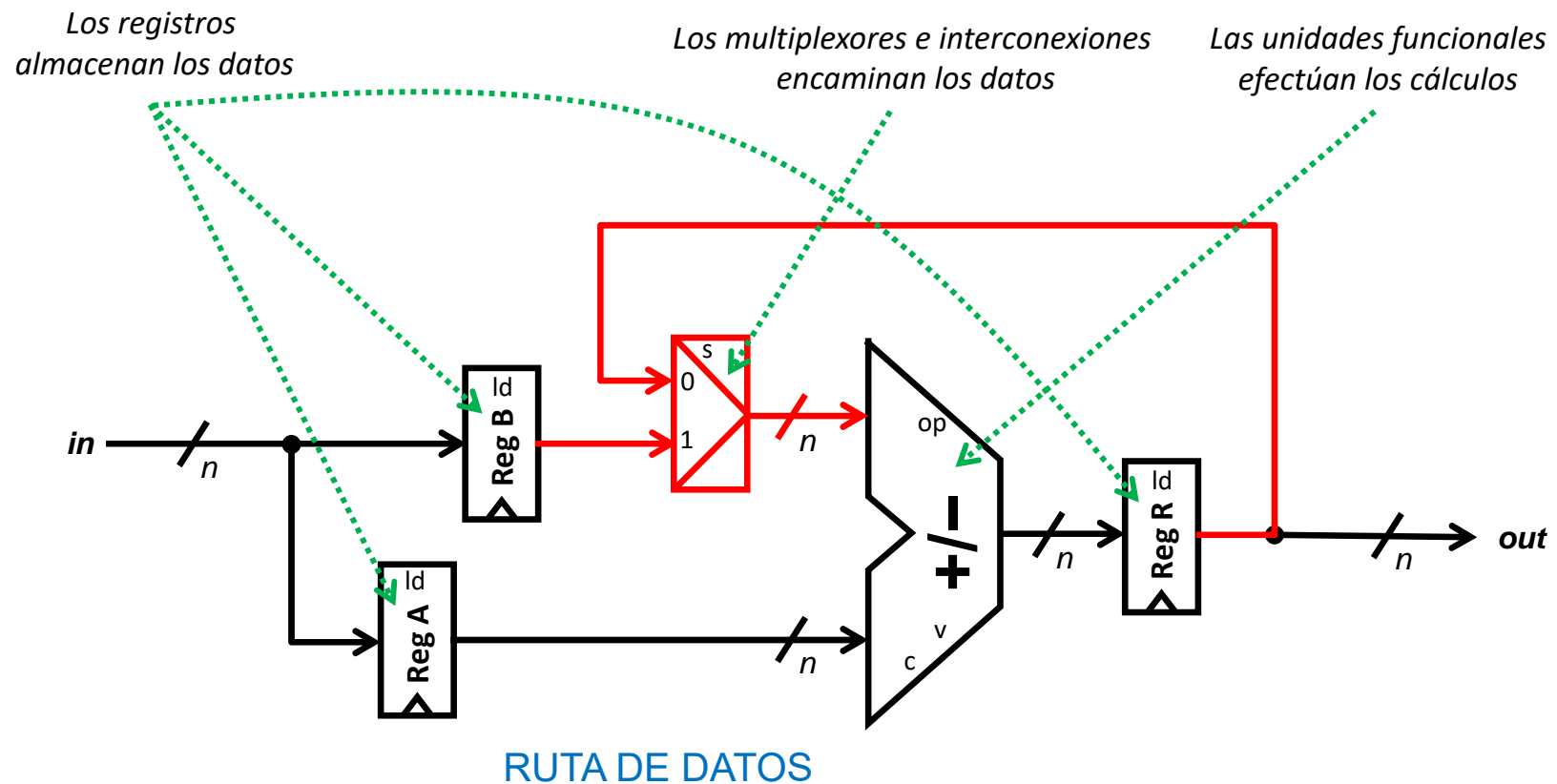


RUTA DE DATOS



Estructura

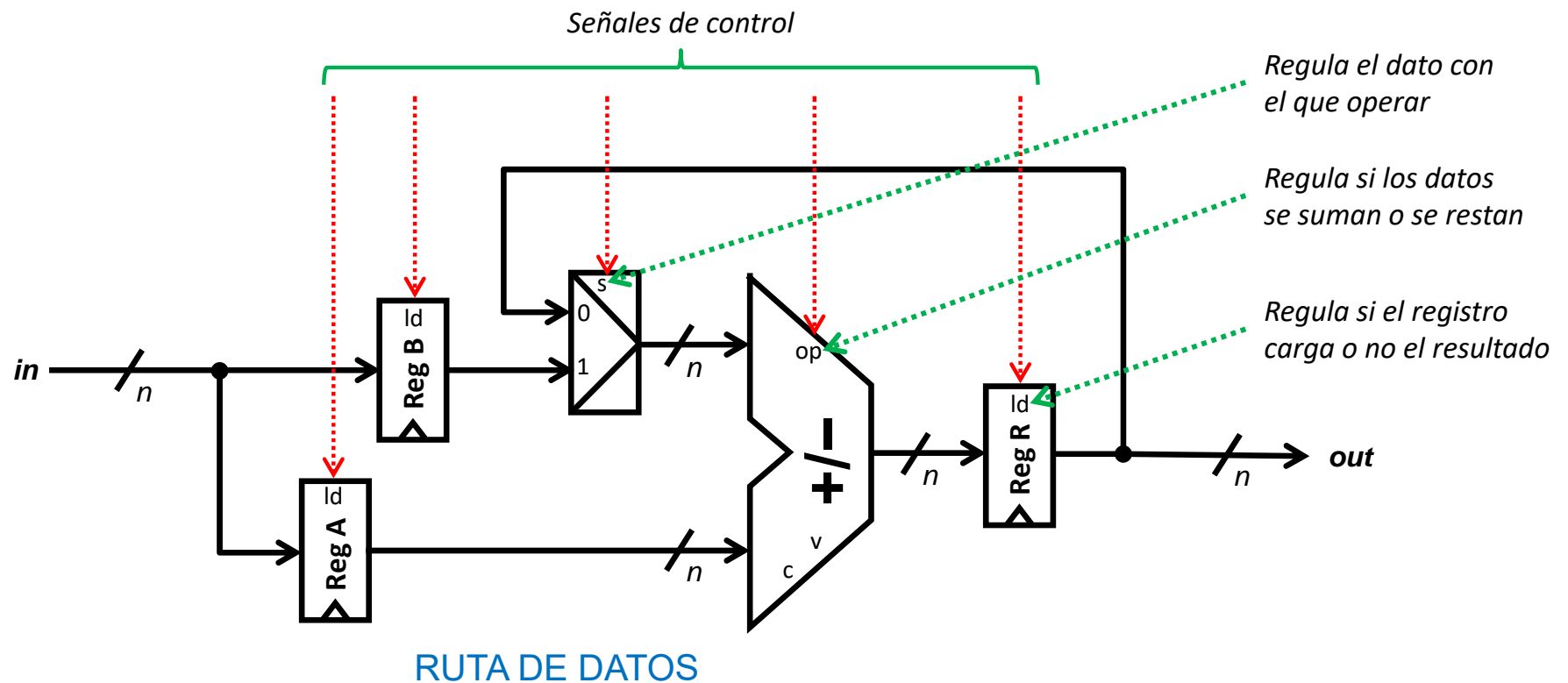
- Toda **ruta de datos** está formada por:
 - Módulos secuenciales: registros, contadores, desplazadores ...
 - Módulos combinacionales: sumadores, ALU, MUX, lógica ...





Estructura

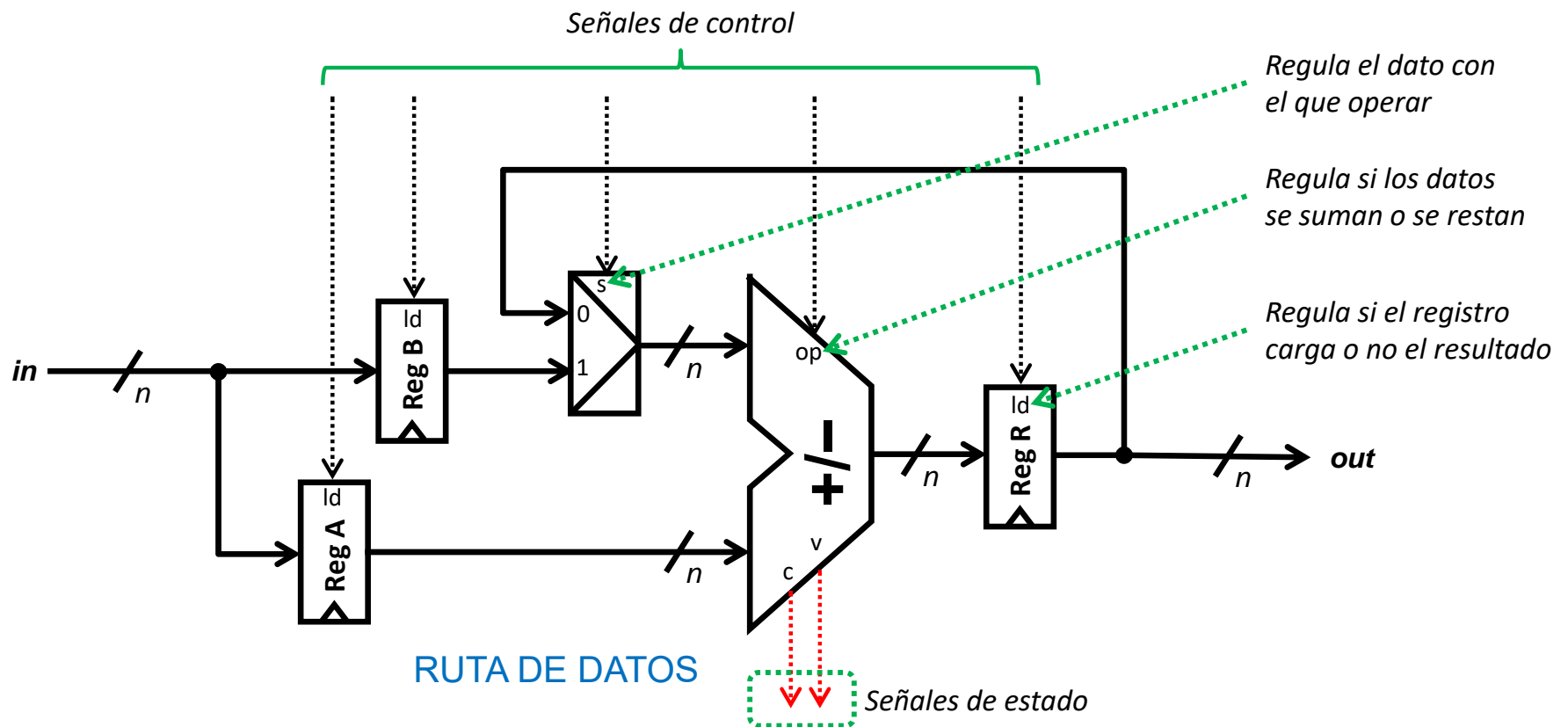
- En toda **ruta de datos** existen:
 - **Señales de control**: fijan la funcionalidad de cada módulo.





Estructura

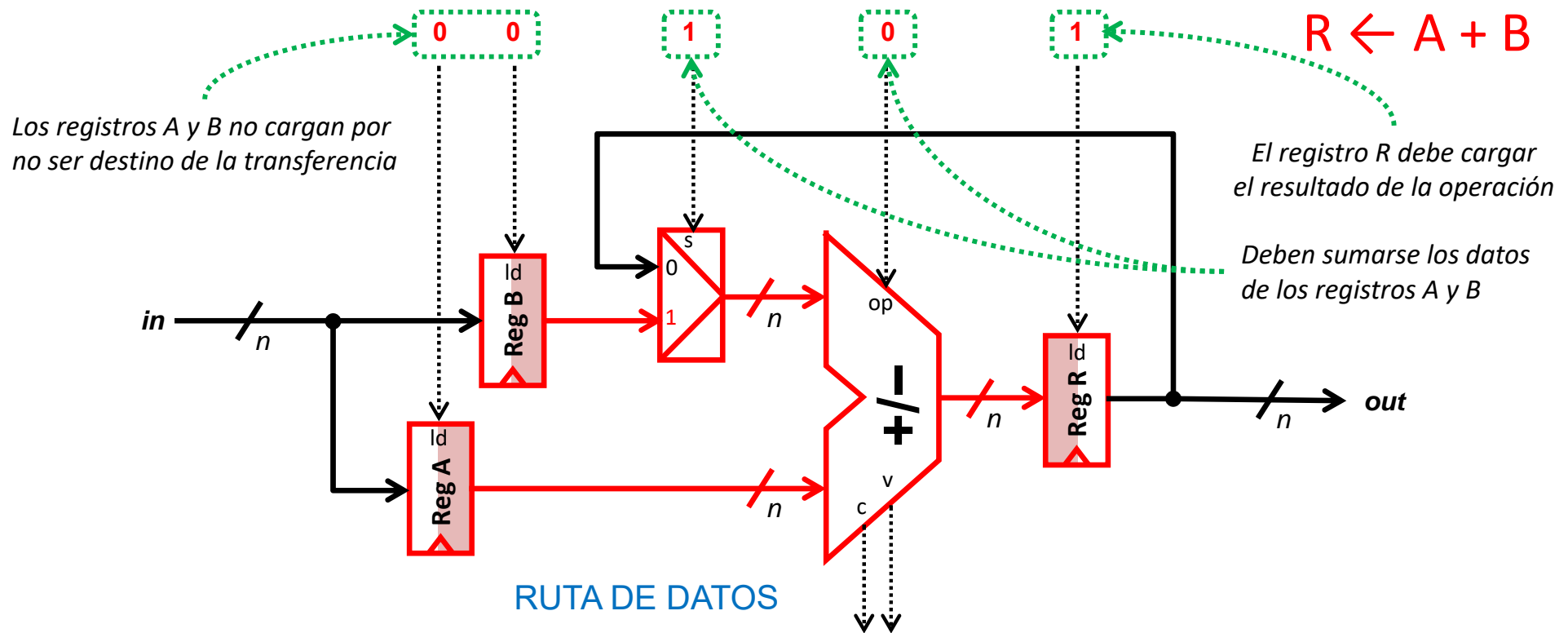
- En toda **ruta de datos** existen:
 - Señales de control: fijan la funcionalidad de cada módulo.
 - Señales de estado: informan del resultado de las operaciones.





Modelo de cómputo

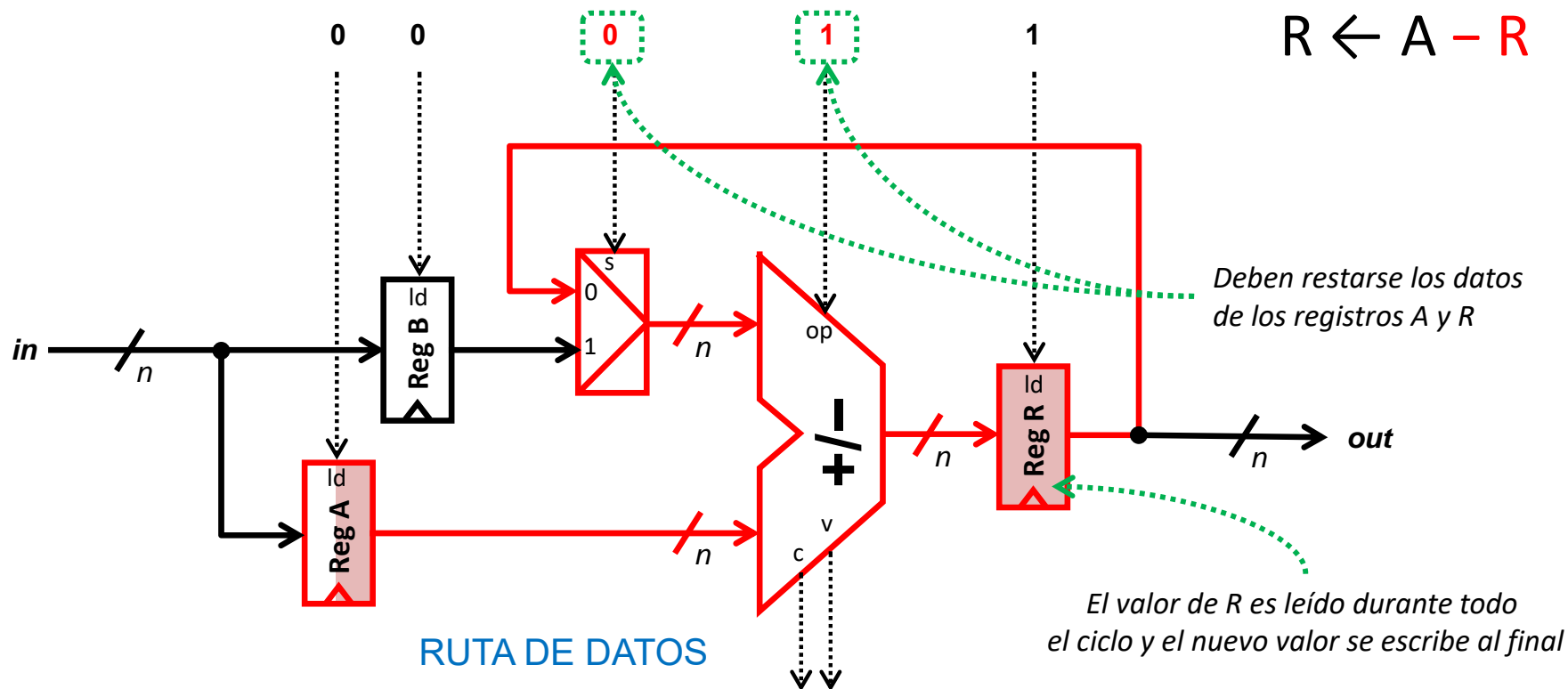
- Según su diseño, cada **ruta de datos** puede realizar un cierto conjunto de **transferencias entre registros**.
 - El valor de las **señales de control** fija cual de ellas efectúa.





Modelo de cómputo

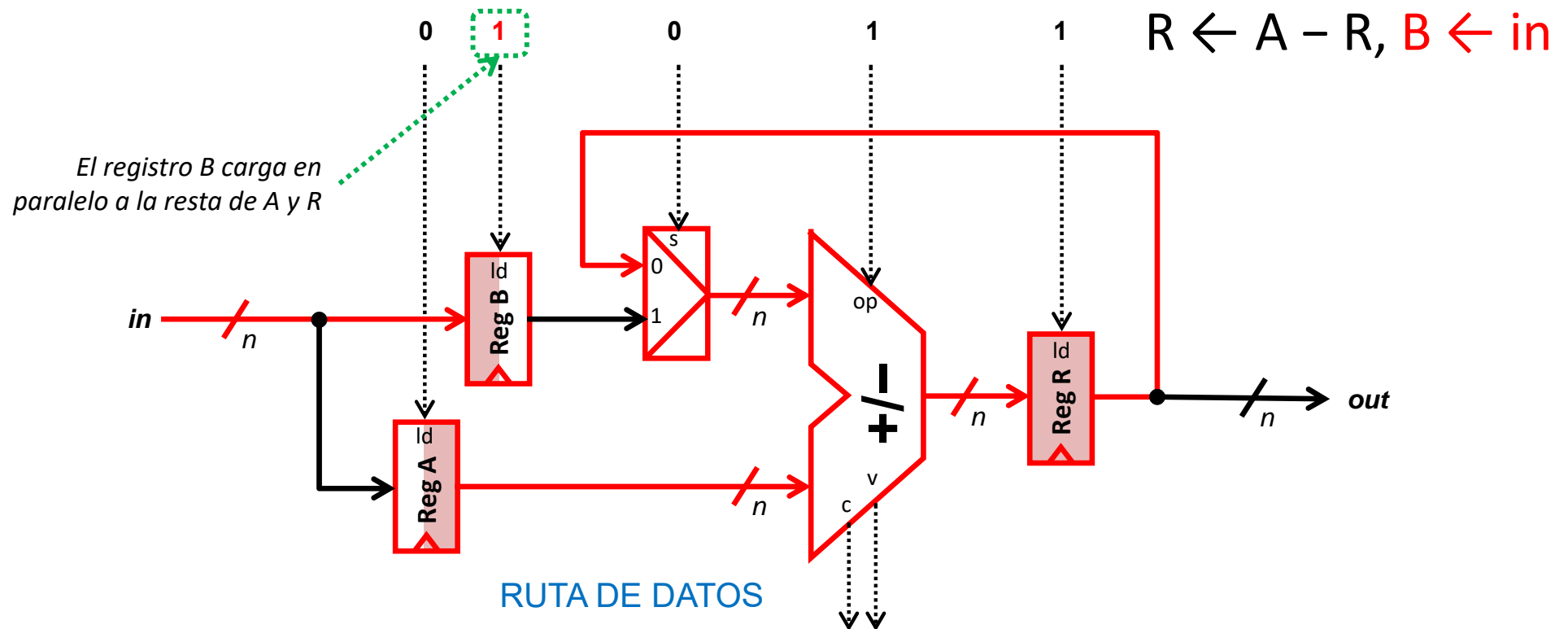
- Según su diseño, cada **ruta de datos** puede realizar un cierto conjunto de **transferencias entre registros**.
 - El valor de las **señales de control** fija cual de ellas efectúa.





Modelo de cómputo

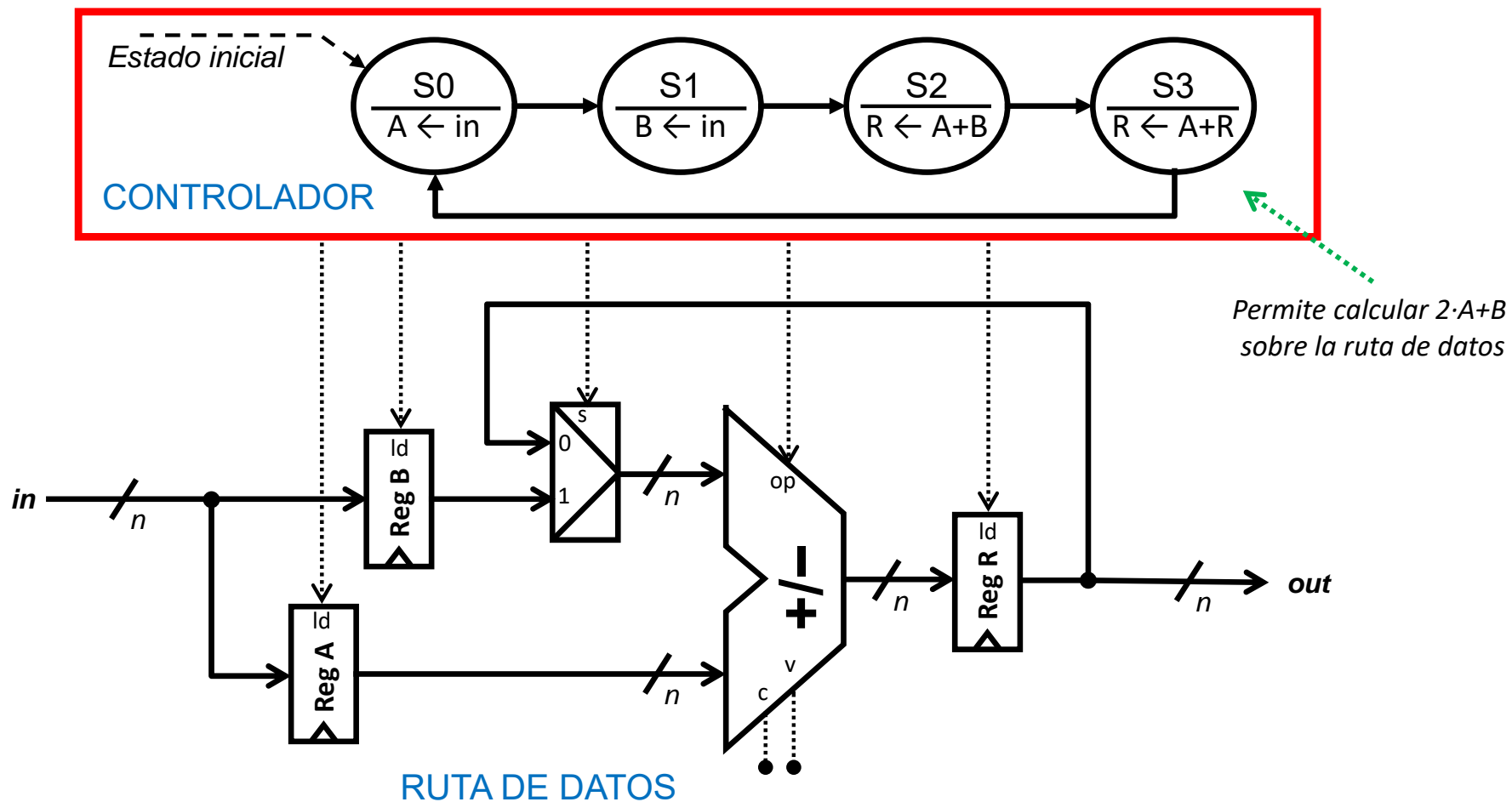
- Según su diseño, cada **ruta de datos** puede realizar un cierto conjunto de **transferencias entre registros**.
 - El valor de las **señales de control** fija cual efectúa.
 - Algunas **pueden realizarse simultáneamente** en paralelo.





Modelo de cómputo

- El **controlador** secuencia las transferencias cambiando el valor de las señales de control generado en cada estado.





Metodología de diseño

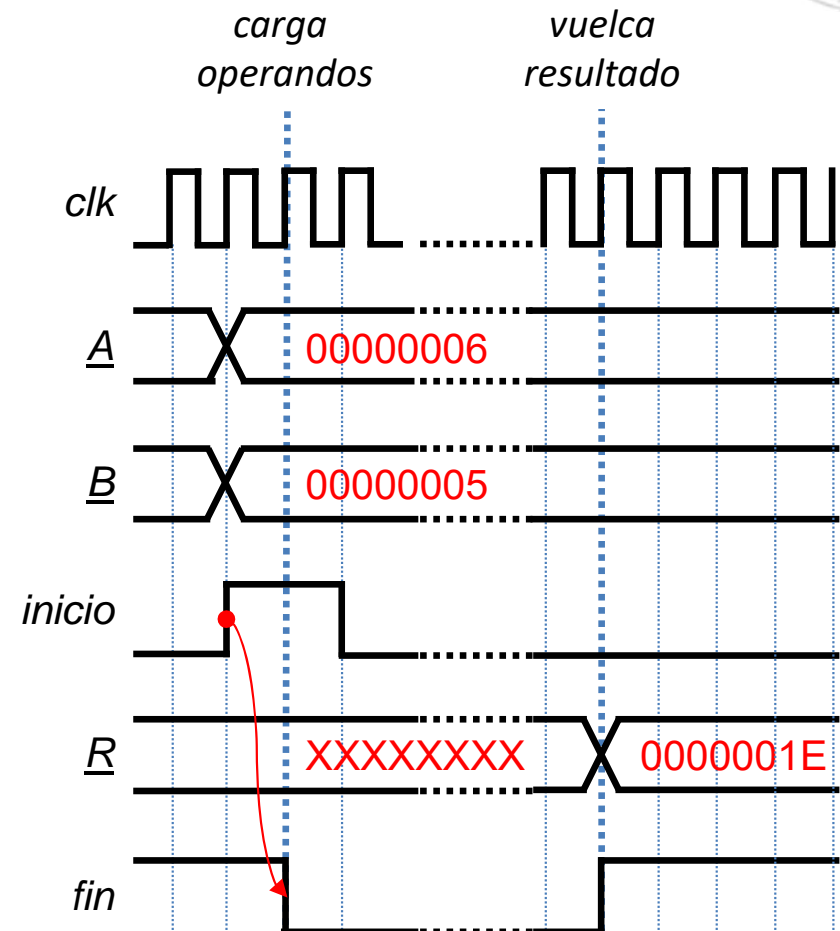
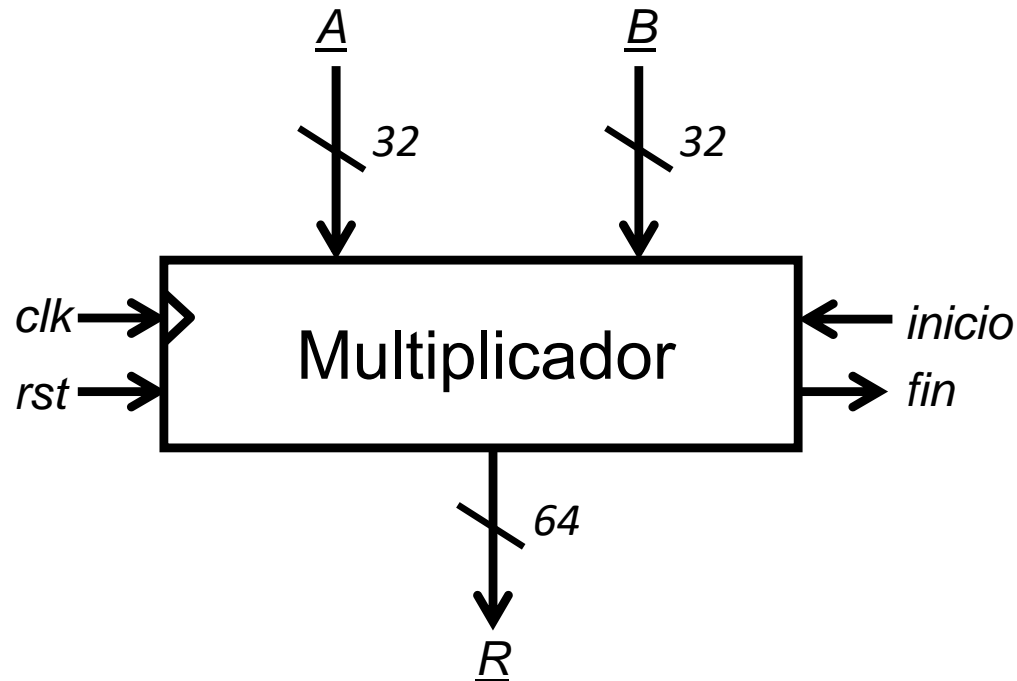
- Para diseñar sistemas complejos se aplica la metodología de **diseño algorítmico** RTL (*Register Transfer Level*) siguiente:
 1. Especificación de alto nivel de la funcionalidad requerida.
 2. Especificación de un algoritmo que la efectúe.
 3. Diseño de la ruta de datos.
 4. Identificación de las señales de control.
 5. Identificación de las señales de estado.
 6. Reformulación del algoritmo como una secuencia de transferencias entre registros.
 7. Especificación del controlador como FSM.
 8. Diseño y optimización del controlador.



Multiplicador iterativo

Especificación de la funcionalidad y el interfaz

versión 14/07/23



110	$a_2 a_1 a_0$
× 101	× $b_2 b_1 b_0$
000110	$(000a_2 a_1 a_0) \times b_0$
000000	$(00a_2 a_1 a_0 0) \times b_1$
+ 011000	+ $(0a_2 a_1 a_0 00) \times b_2$
011110	$r_5 r_4 r_3 r_2 r_1 r_0$

Algoritmo para **operandos de 3 bits**

1. Especificación de alto nivel



Multiplicador iterativo

Especificación y simulación del algoritmo

versión 14/07/23

Para operandos de 32 bits
el bucle sería de 0 a 31

```

1. A = Ain;
2. B = Bin;
3. R = 0;
   for( C=0; C<3; C++ )
   {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
   };
7. Rout = R;

```

Algoritmo para **operandos de 3 bits**

C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀

tema 8:
Rutas de datos y controladores



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
           R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;

```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-



Multiplicador iterativo

Especificación y simulación del algoritmo

```
1.  A = Ain;  
2.  B = Bin;  
3.  R = 0;  
   for( C=0; C<3; C++ )  
   {  
4.      if( B0==1 )  
         R = R+A;  
5.      A = A << 1;  
6.      B = B >> 1;  
   };  
7.  Rout = R;
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
   for( C=0; C<3; C++ )
   {
4.    if( B0==1 )
       R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
   };
7.  Rout = R;

```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;

```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;

```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0
6.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0
6.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
4.	2	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;
  
```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0
6.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
4.	2	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1
5.	2	0	1	1	1	1	0	1	1	0	0	0	0	0	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;

```

Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0
6.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
4.	2	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1
5.	2	0	1	1	1	1	0	1	1	0	0	0	0	0	0	1
6.	2	0	1	1	1	1	0	1	1	0	0	0	0	0	0	1



Multiplicador iterativo

Especificación y simulación del algoritmo

```

1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<3; C++ )
    {
4.      if( B0==1 )
          R = R+A;
5.      A = A << 1;
6.      B = B >> 1;
    };
7.  Rout = R;

```

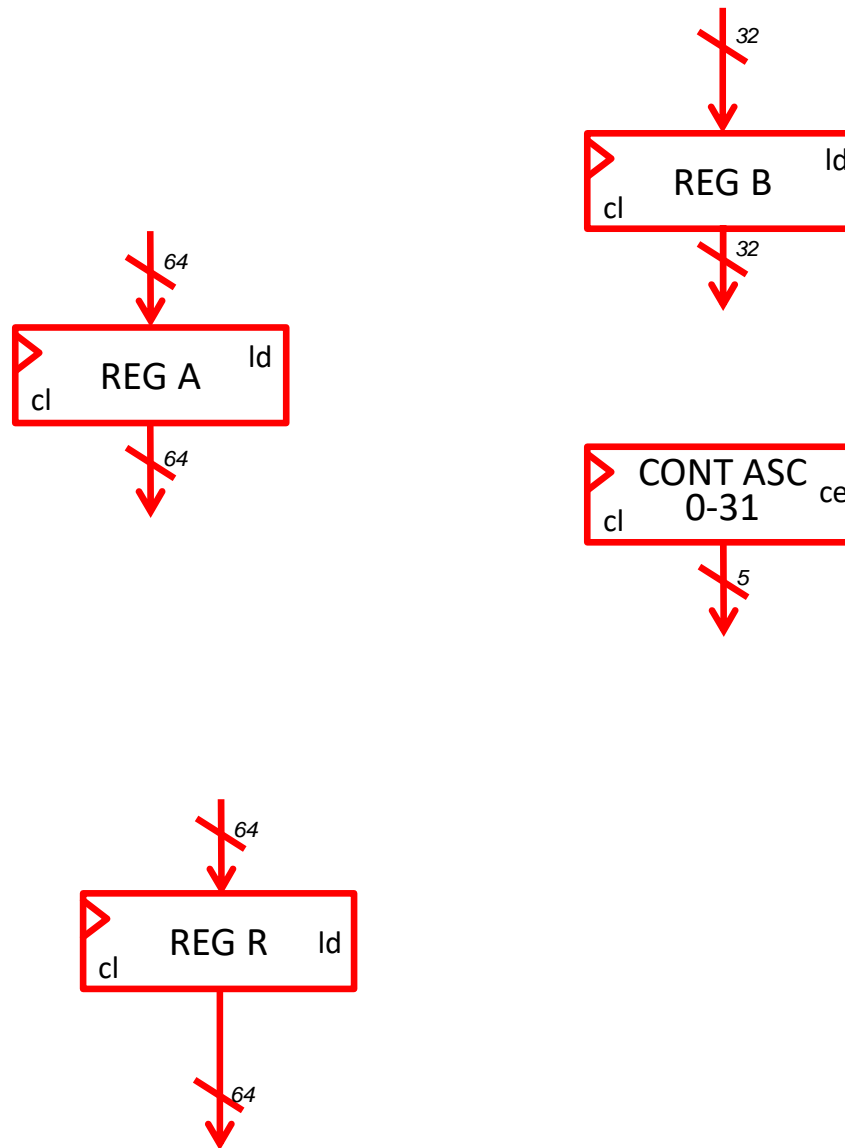
Algoritmo para **operandos de 3 bits**

	C	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
1.	-	-	-	-	-	-	-	0	0	0	1	1	0	-	-	-
2.	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0	1
3.	-	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
4.	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1
5.	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1
6.	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
4.	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0
5.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0
6.	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
4.	2	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1
5.	2	0	1	1	1	1	0	1	1	0	0	0	0	0	0	1
6.	2	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0



Multiplicador iterativo

Elementos de almacenamiento



- ```
1. A = Ain;
2. B = Bin;
3. R = 0;
 for(C=0; C<32; C++)
 {
4. if(B0==1)
 R = R+A;
5. A = A << 1;
6. B = B >> 1;
 };
7. Rout = R;
```

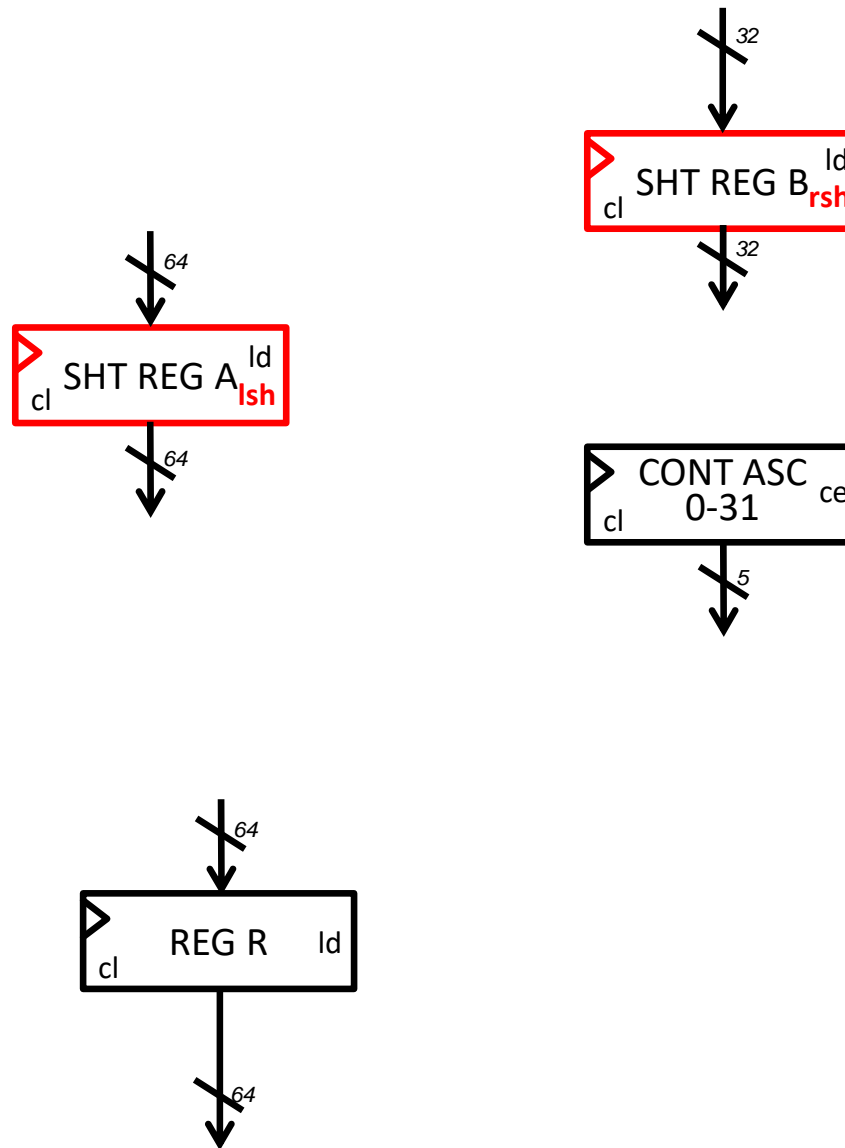
*Algoritmo para  
operandos de 32 bits*

### 3. Diseño de la ruta de datos



# Multiplicador iterativo

## Desplazamiento de operandos



- ```
1. A = Ain;
2. B = Bin;
3. R = 0;
  for( C=0; C<32; C++ )
  {
4.   if( B0==1 )
       R = R+A;
5.   A = A << 1;
6.   B = B >> 1;
  };
7. Rout = R;
```

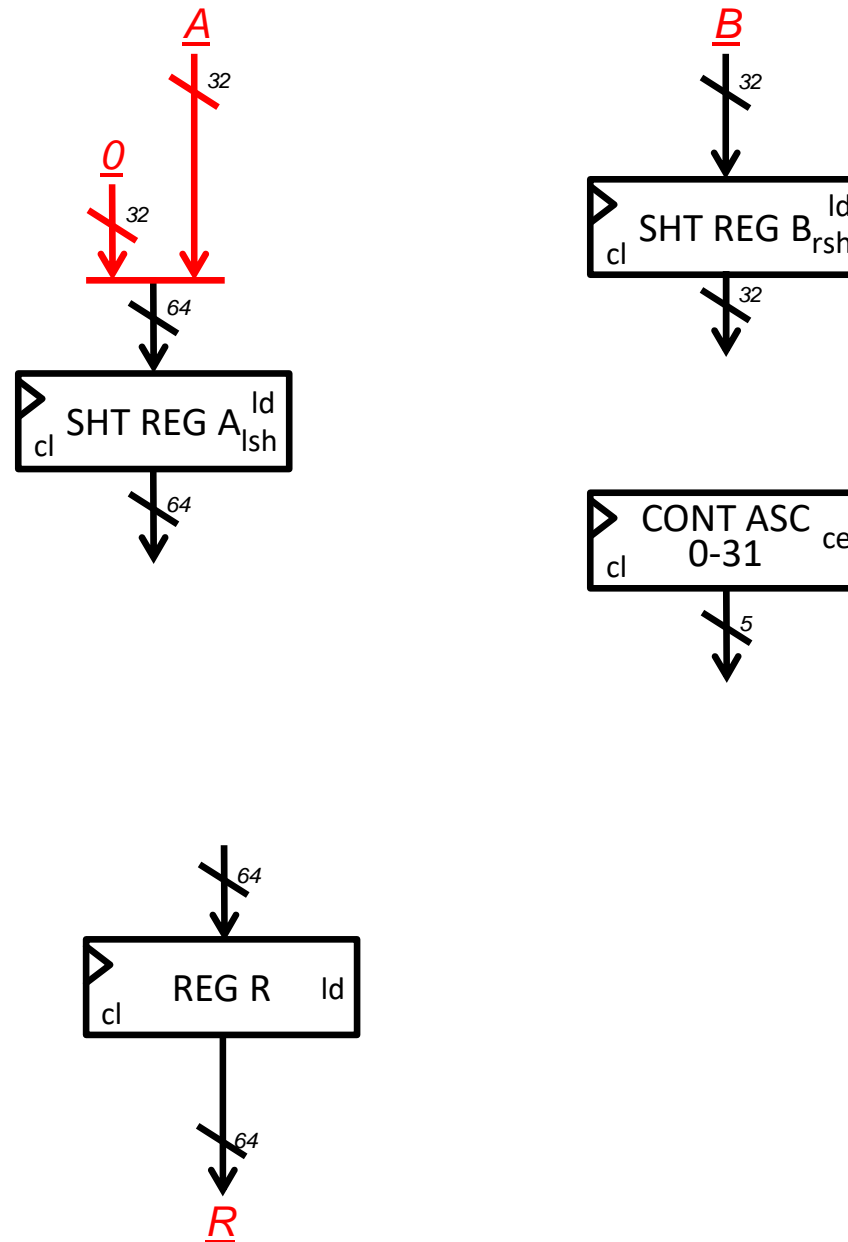
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos



Multiplicador iterativo

Conexión con el exterior



- ```
1. A = Ain;
2. B = Bin;
3. R = 0;
 for(C=0; C<32; C++)
 {
4. if(B0==1)
 R = R+A;
5. A = A << 1;
6. B = B >> 1;
 }
7. Rout = R;
```

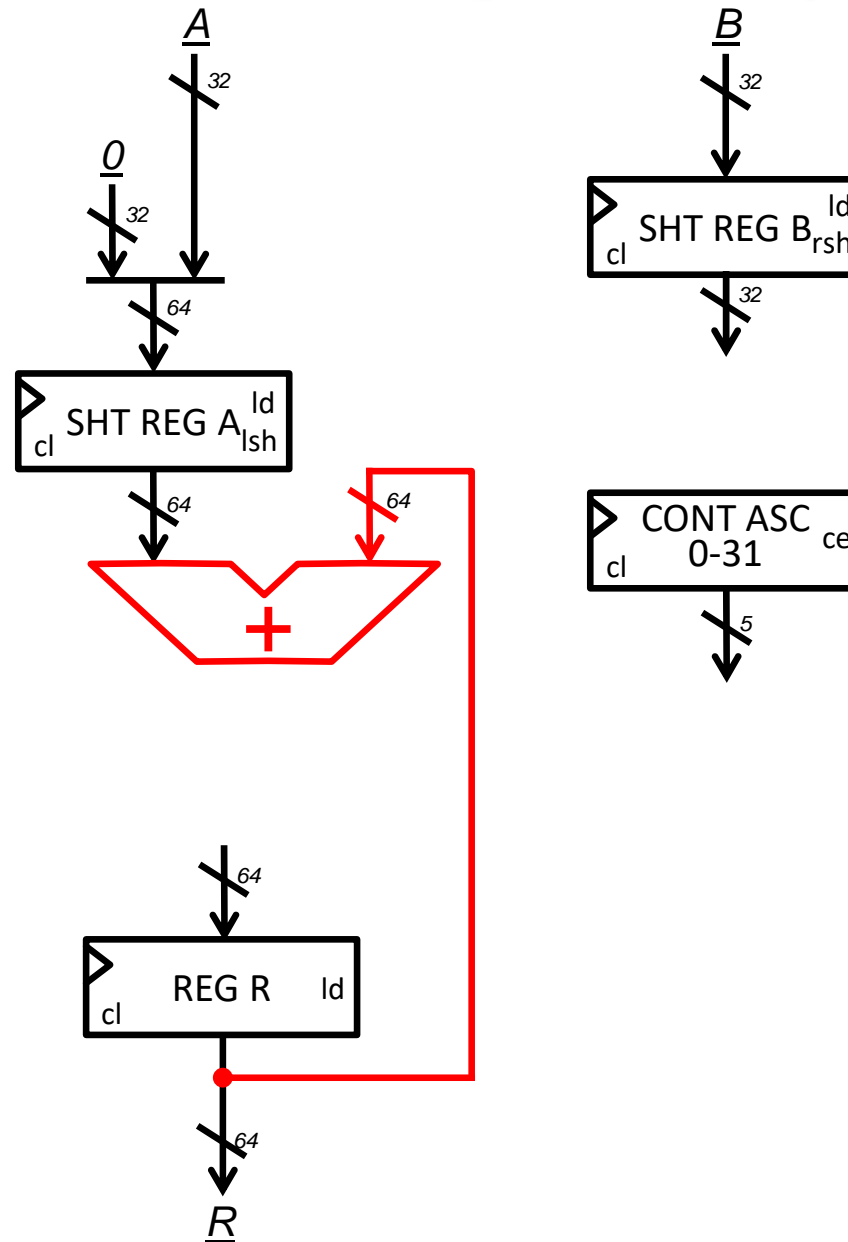
*Algoritmo para  
operandos de 32 bits*

### 3. Diseño de la ruta de datos



# Multiplicador iterativo

## Suma de productos parciales



- ```
1. A = Ain;
2. B = Bin;
3. R = 0;
for( C=0; C<32; C++ )
{
4.   if( B0==1 )
       R = R+A;
5.   A = A << 1;
6.   B = B >> 1;
};
7. Rout = R;
```

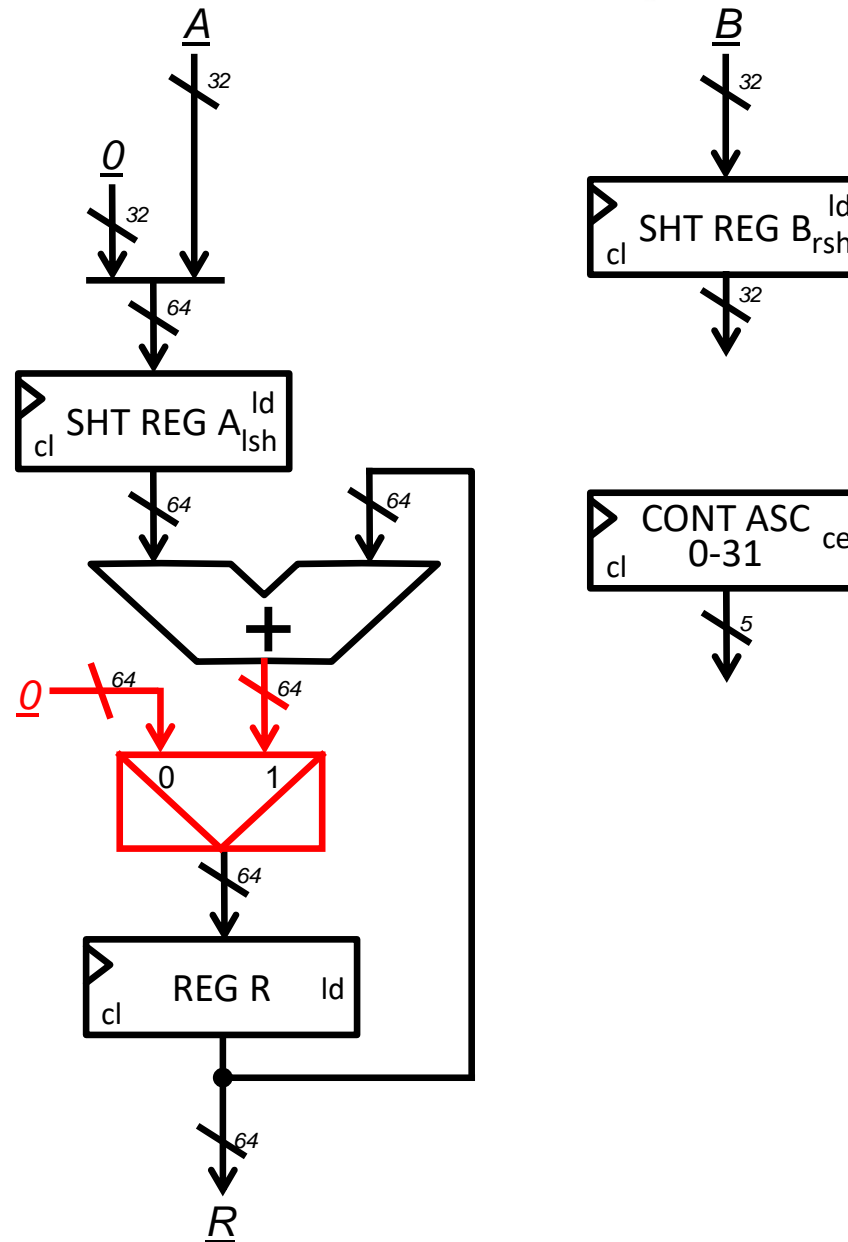
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos



Multiplicador iterativo

Inicialización y almacenaje



1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
4. `for(C=0; C<32; C++)`
 - `{`
 - 4. `if(B0==1)`
 - $R = R+A;$
 - 5. $A = A \ll 1;$
 - 6. $B = B \gg 1;$
 - `};`
7. $R_{out} = R;$

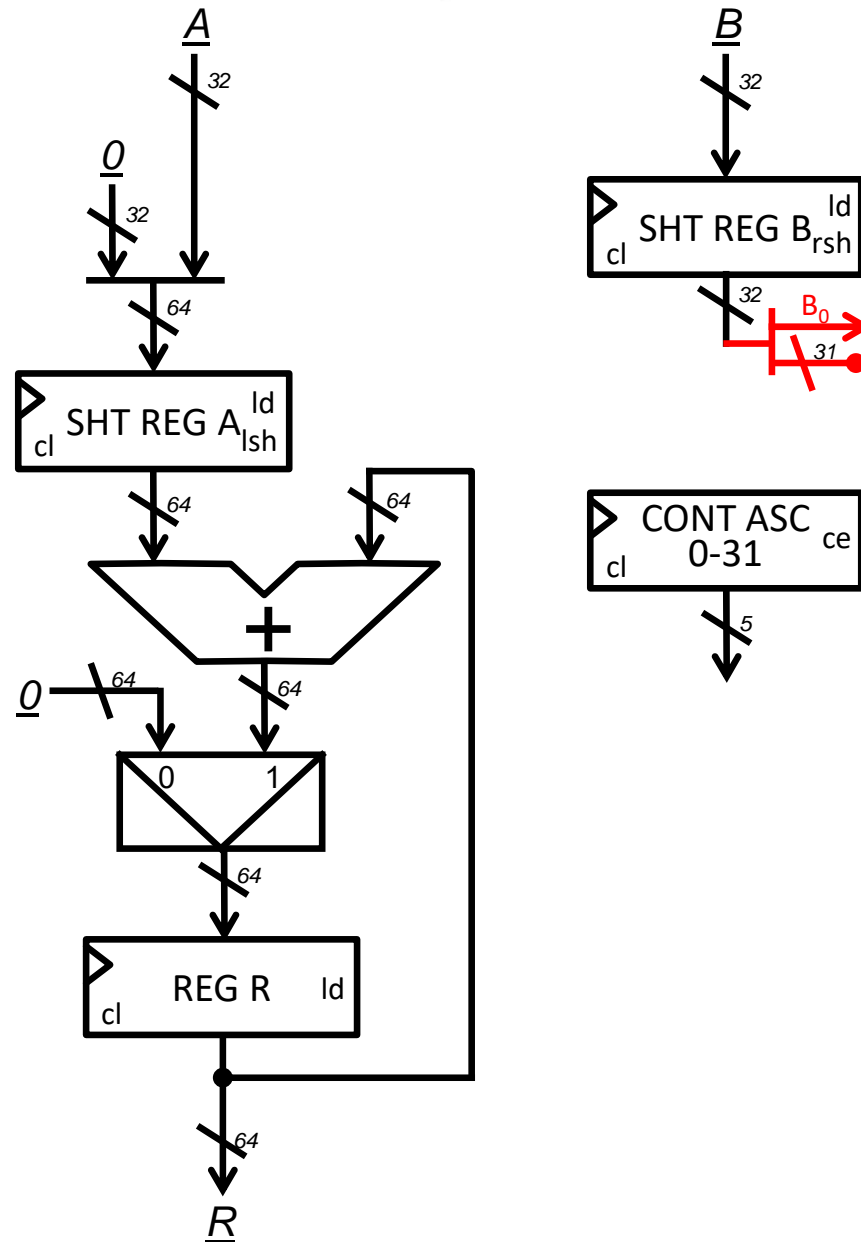
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos



Multiplicador iterativo

Chequeo de condición



```
1. A = Ain;
2. B = Bin;
3. R = 0;
for( C=0; C<32; C++ )
{
4.   if( B0==1 )
       R = R+A;
5.   A = A << 1;
6.   B = B >> 1;
};
7. Rout = R;
```

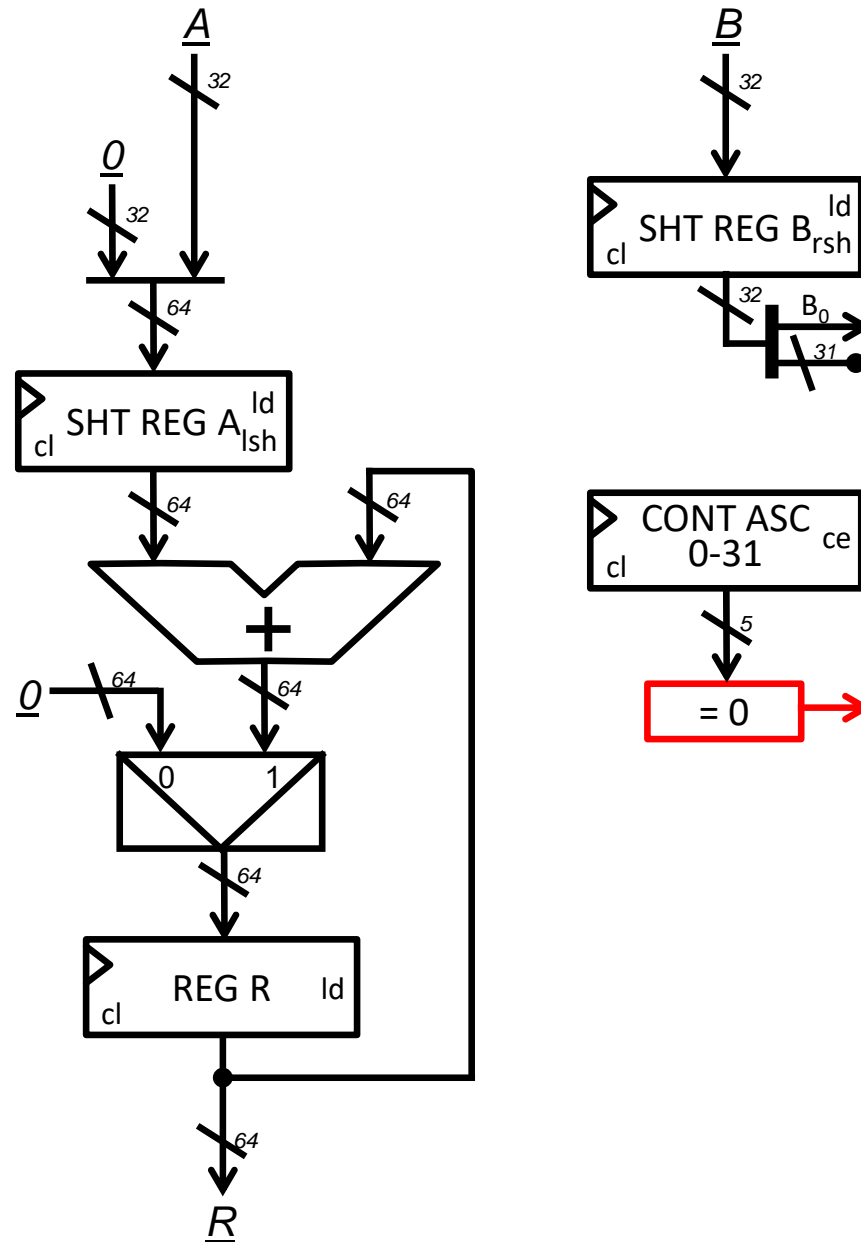
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos



Multiplicador iterativo

Detección del fin del bucle



```

1. A = Ain;
2. B = Bin;
3. R = 0;
   for( C=0; C<32; C++ )
   {
4.   if( B0==1 )
       R = R+A;
5.   A = A << 1;
6.   B = B >> 1;
   };
7. Rout = R;

```

*Algoritmo para
operandos de 32 bits*

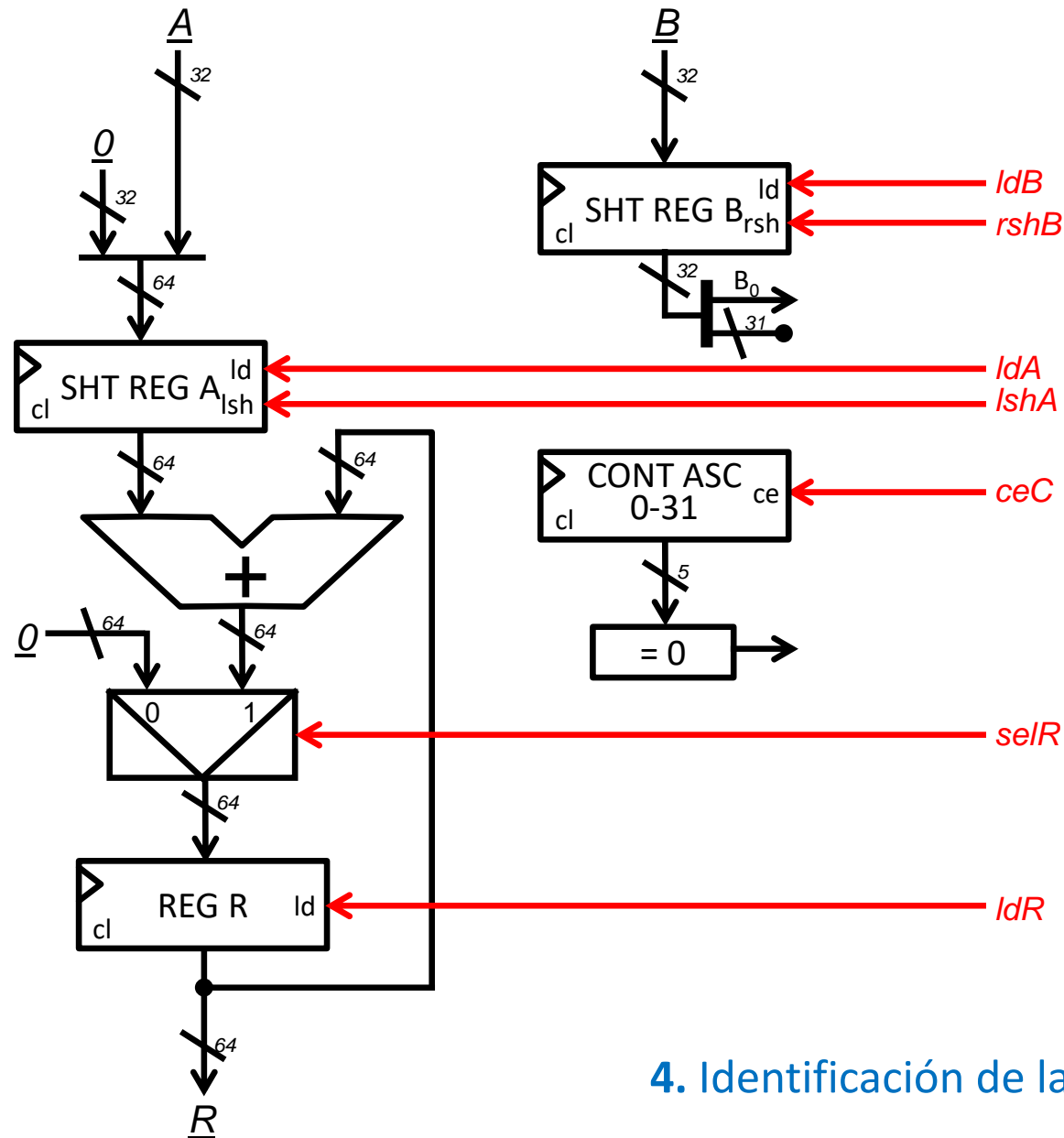


3. Diseño de la ruta de datos



Multiplicador iterativo

Señales de control

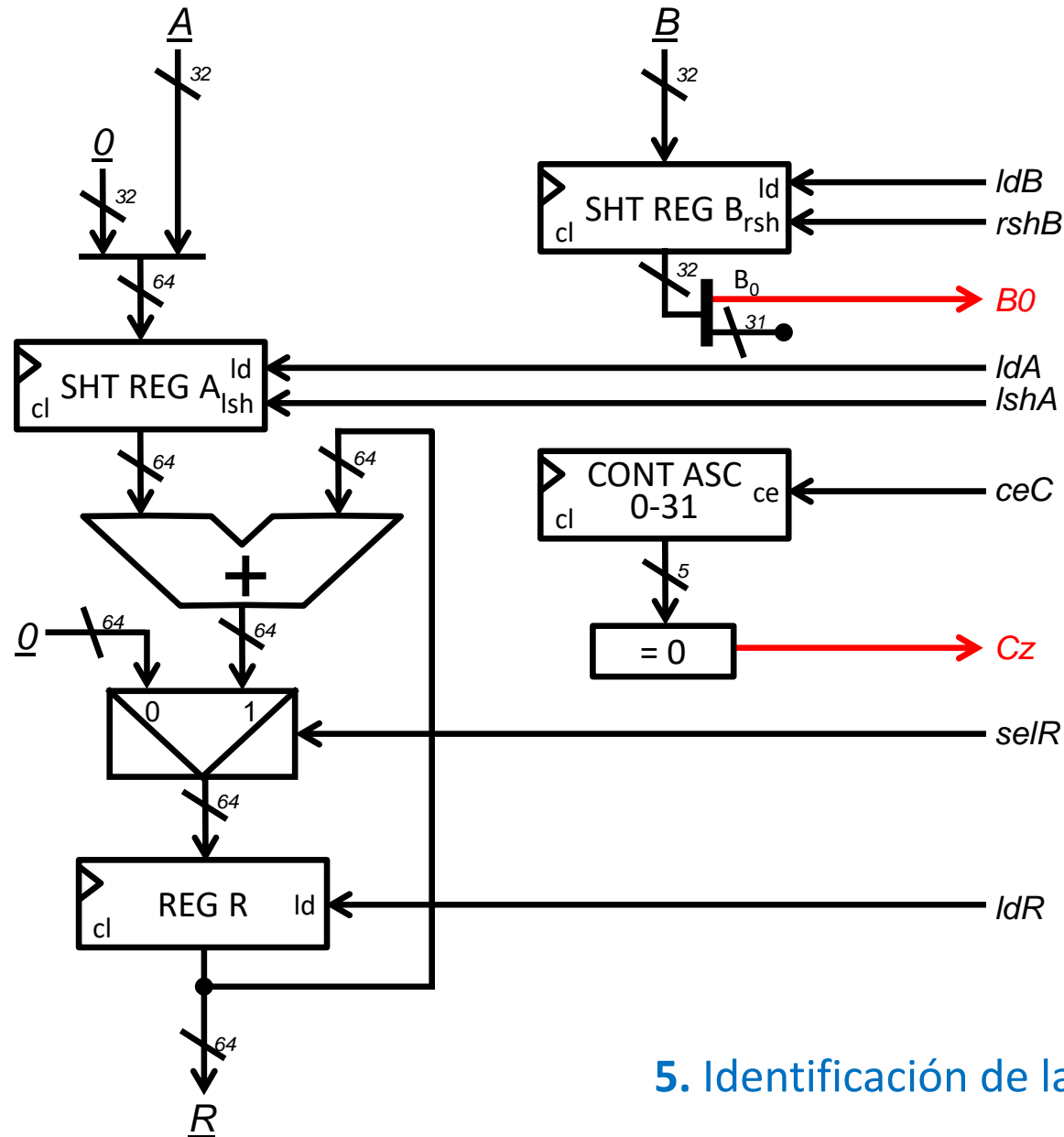


4. Identificación de las señales de control



Multiplicador iterativo

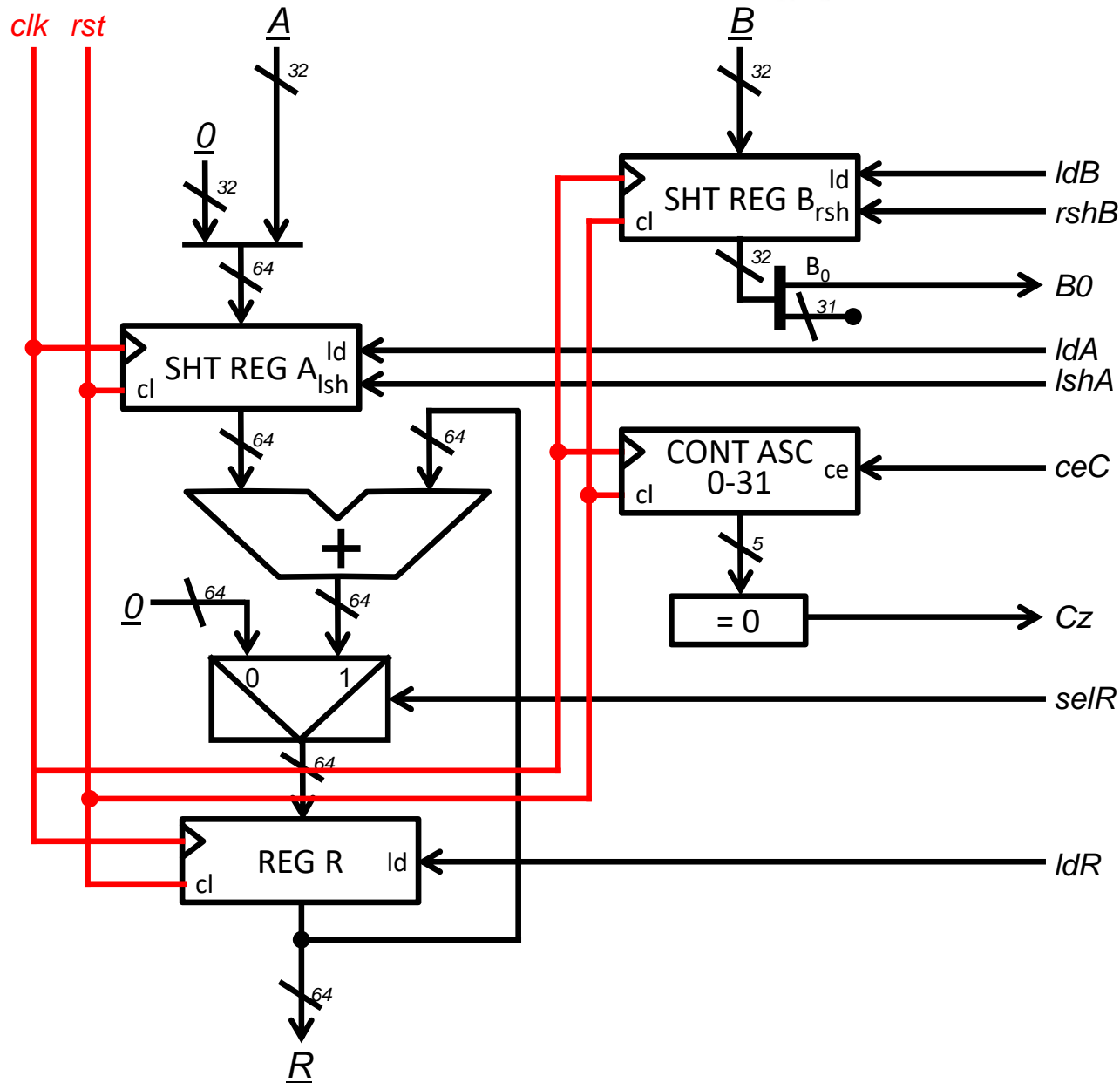
Señales de estado





Multiplicador iterativo

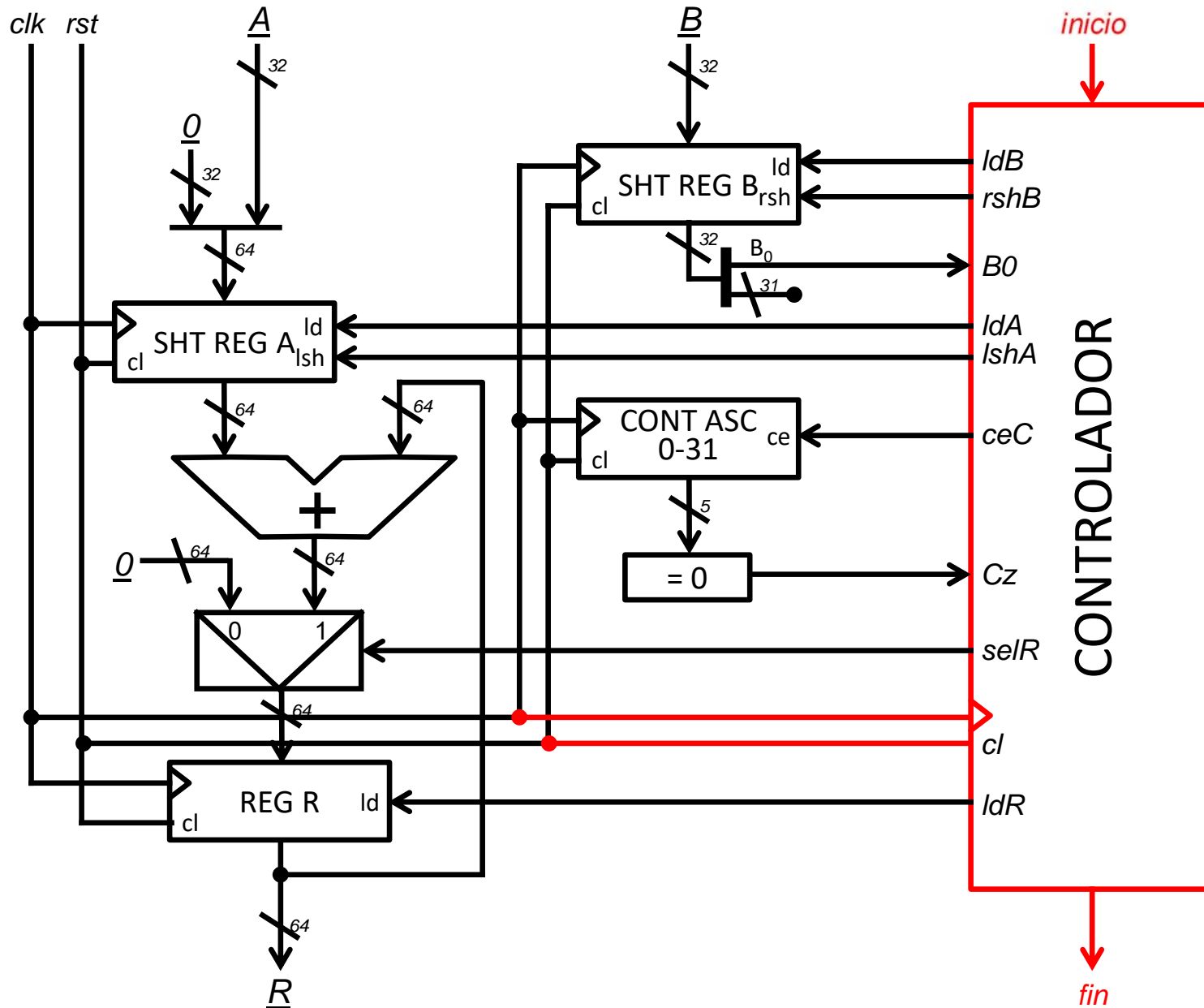
Conexión del reloj y reset





Multiplicador iterativo

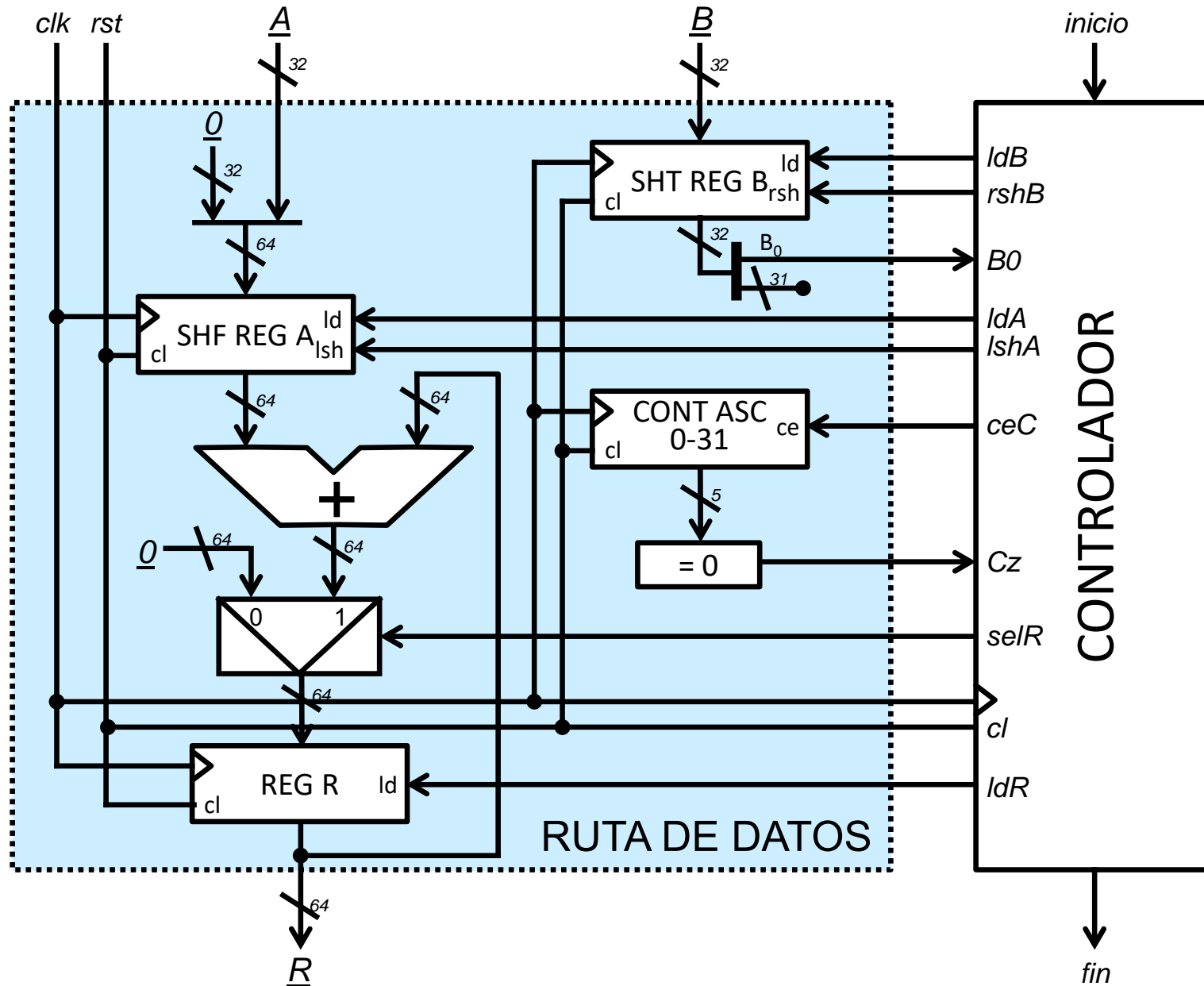
Conexión con el controlador





Multiplicador iterativo

Estructura del sistema completo





Multiplicador iterativo

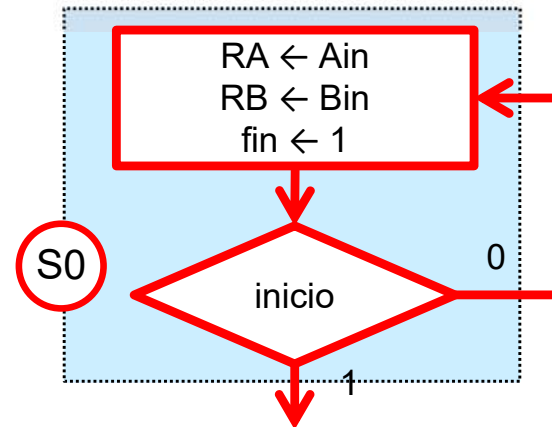
Formalismo ASM (Algorithmic State Maquina)

- Para facilitar la expresión del algoritmo como una FSM usaremos un **diagrama ASM** como formalismo intermedio.
- Un diagrama ASM es un grupo de bloques interconectados.
 - Cada **bloque** se corresponde con un **estado de la FSM** e indica las acciones que se realizan en paralelo en dicho estado.
- Todo **bloque** (recuadro azul) esta formado por:
 - **1 caja de estado** (rectangular): que indica las transferencias entre registros que se realizan incondicionalmente en el estado.
 - **0..n cajas de selección** (romboidales): indican las condiciones bajo las cuales se que se determina el estado siguiente del sistema o se realizan ciertas transferencias entre registros.
 - **0..n cajas condicionales** (ovaladas): indica las trasferencias entre registros que se realizan condicionalmente en un estado.



Multiplicador iterativo

Algoritmo como ASM



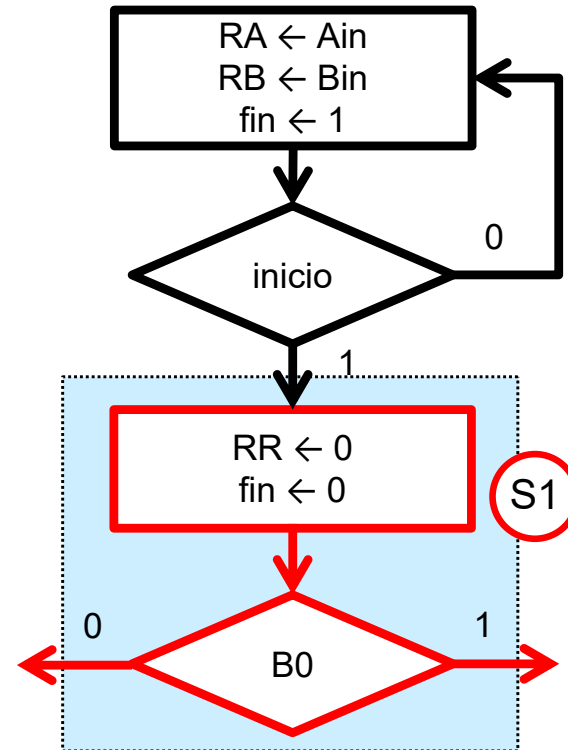
1. **A = Ain;**
2. **B = Bin;**
3. **R = 0;**
for(C=0; C<32; C++)
{
4. **if(B₀==1)**
 R = R+A;
5. **A = A << 1;**
6. **B = B >> 1;**
};
7. **Rout = R;**

6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



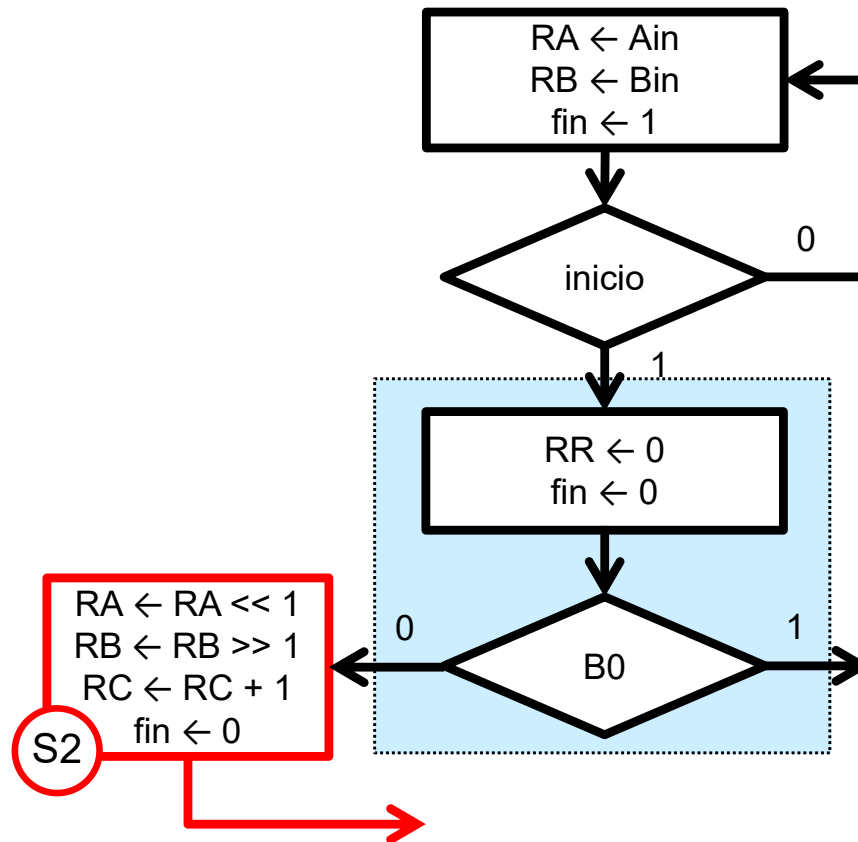
1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`for(C=0; C<32; C++)`
`{`
4. `if(B0==1)`
`R = R+A;`
5. `A = A << 1;`
6. `B = B >> 1;`
7. `};`
`Rout = R;`

6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



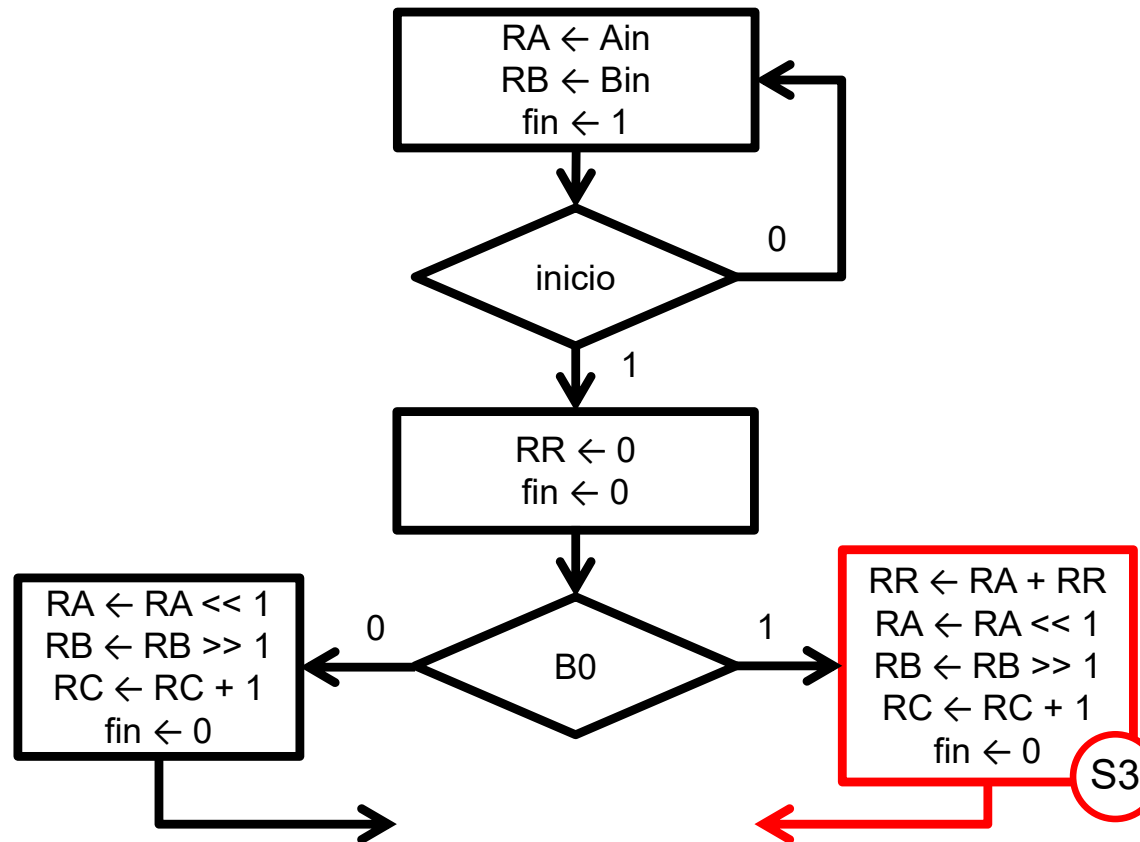
1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`for(C=0; C<32; C++)`
`{`
4. `if(B0==1)`
 `R = R+A;`
5. `A = A << 1;`
6. `B = B >> 1;`
- `};`
7. `Rout = R;`

6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



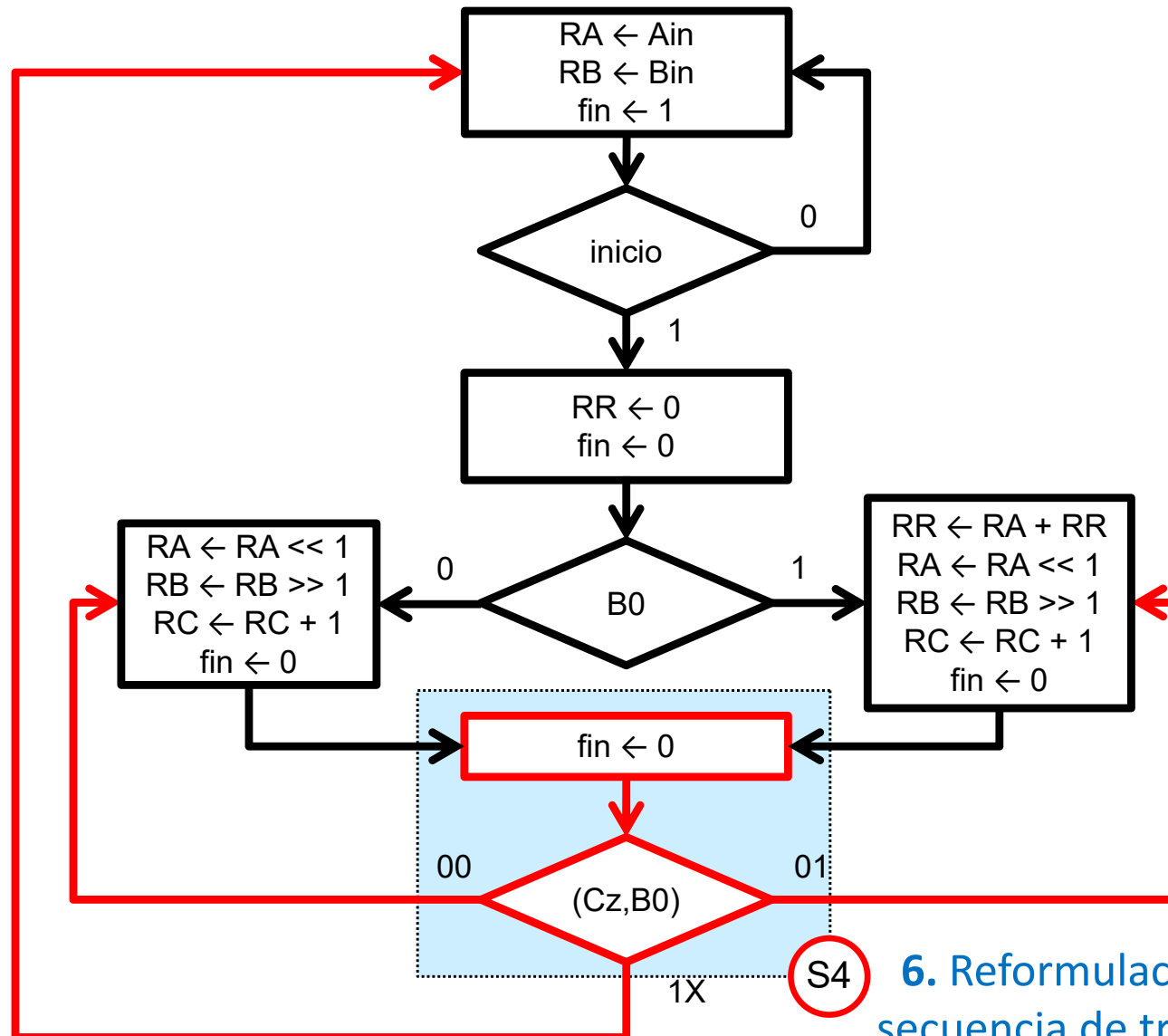
1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`for(C=0; C<32; C++)`
{
4. `if(B0==1)`
 `R = R+A;`
5. `A = A << 1;`
6. `B = B >> 1;`
7. `Rout = R;`
};

6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



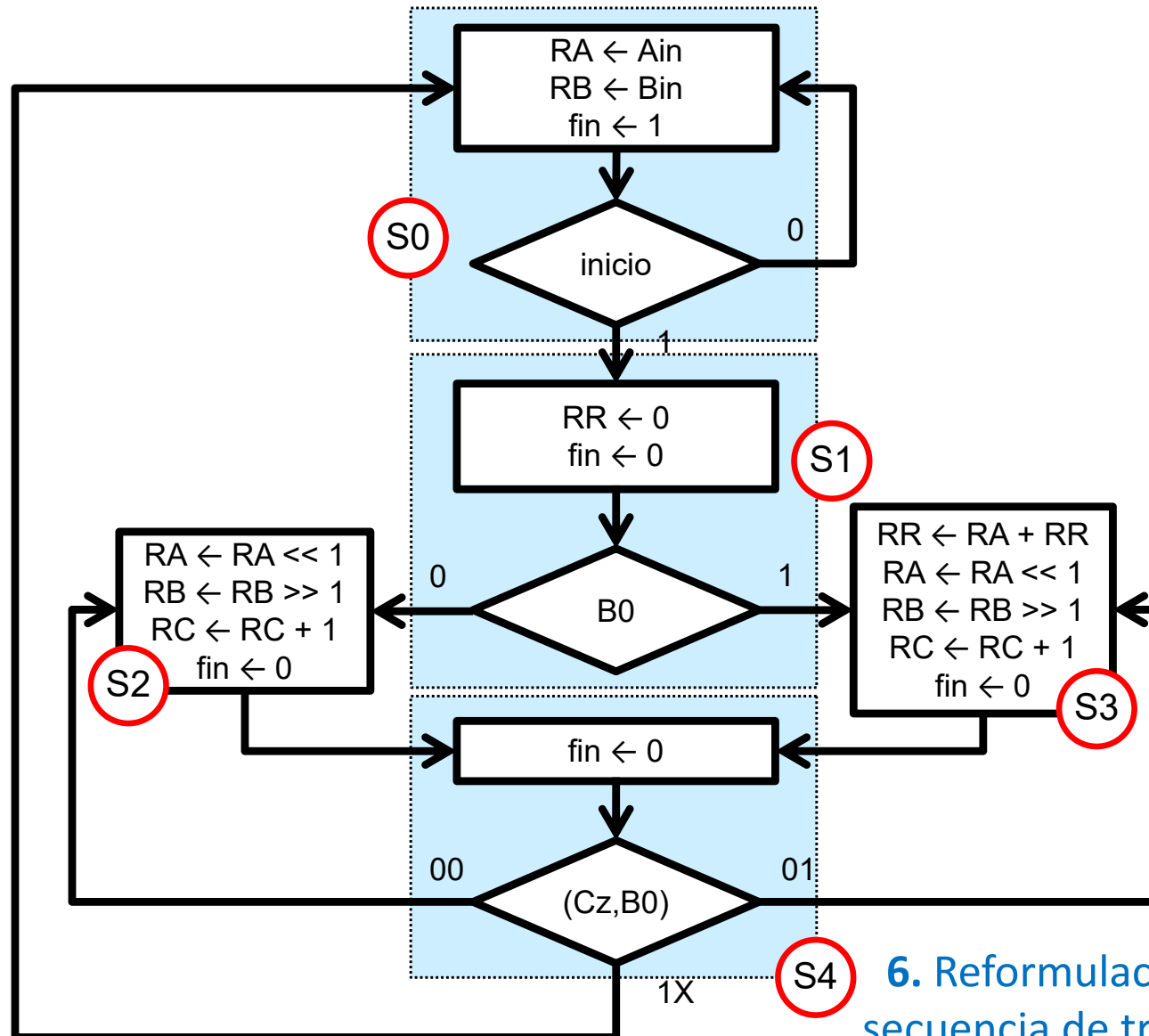
1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
`for(C=0; C<32; C++)`
{
4. `if(B0==1)`
 $R = R+A;$
5. $A = A \ll 1;$
6. $B = B \gg 1;$
7. `};`
7. $R_{out} = R;$

S4 6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



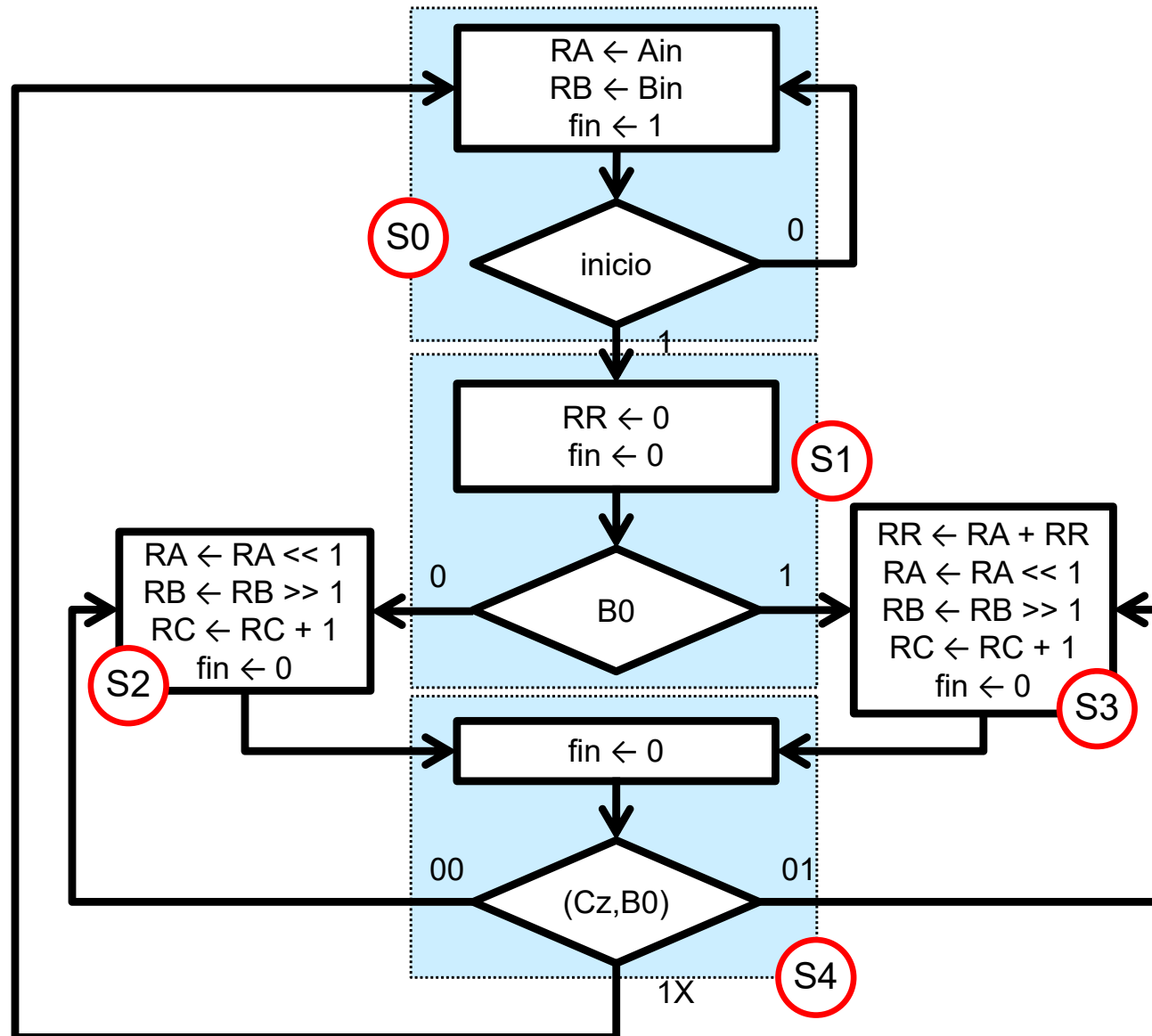
1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`for(C=0; C<32; C++)`
`{`
4. `if(B0==1)`
 `R = R+A;`
5. `A = A << 1;`
6. `B = B >> 1;`
`};`
7. `Rout = R;`

6. Reformulación del algoritmo como una secuencia de transferencias entre registros



Multiplicador iterativo

Algoritmo como ASM



S0 RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;

S1 RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;

S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

S3 RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

S4 fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;

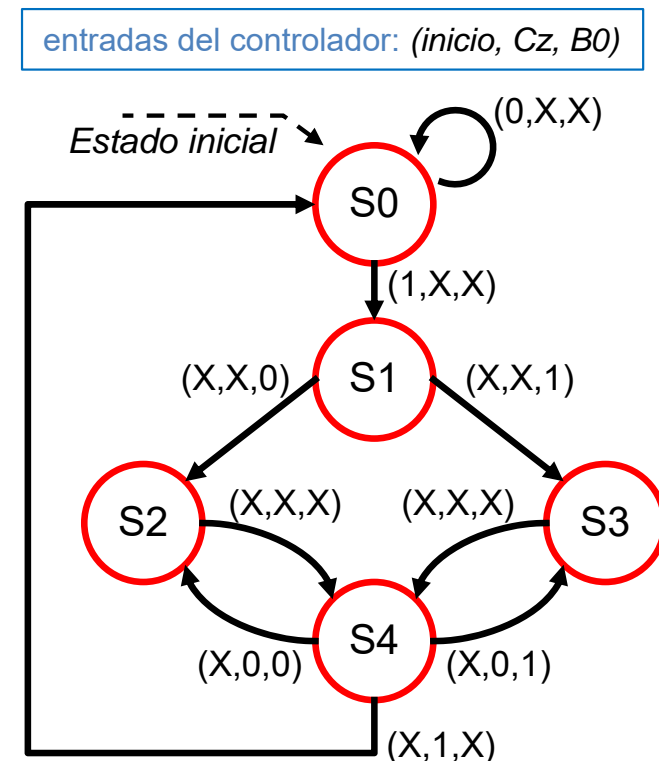
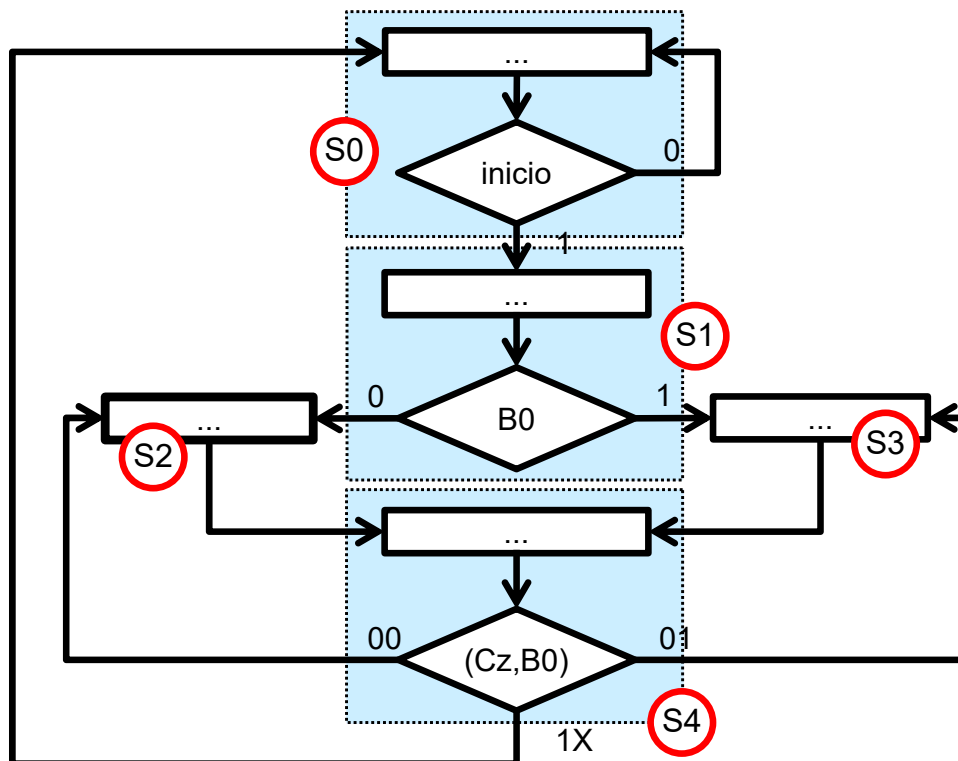
tiempo de cálculo: $2 + 32 \times 2 = 66$ ciclos



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada bloque ASM equivale a un estado de la FSM.
 - Las transiciones entre bloques equivalen a transiciones entre estados.



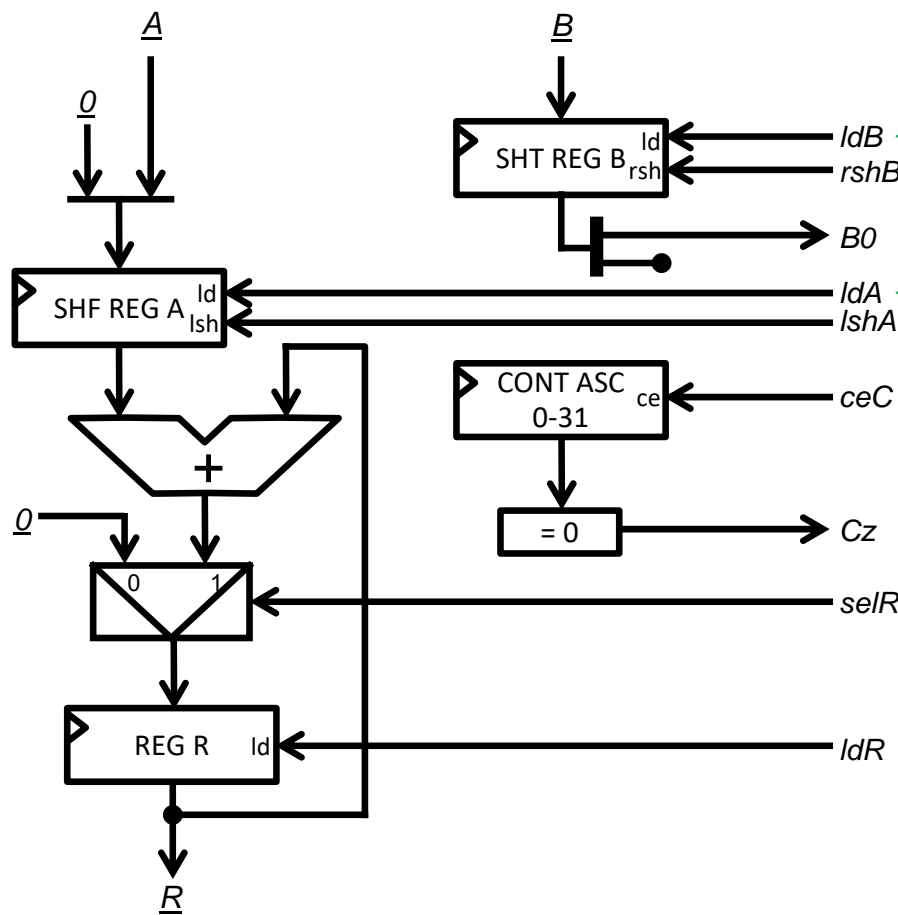
7. Especificación del controlador como FSM



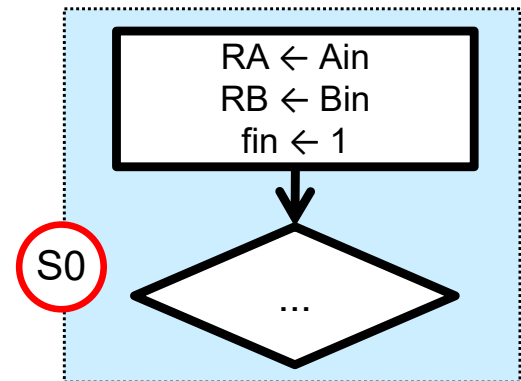
Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



Señales de carga de registros destino a 1



estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1		1					1

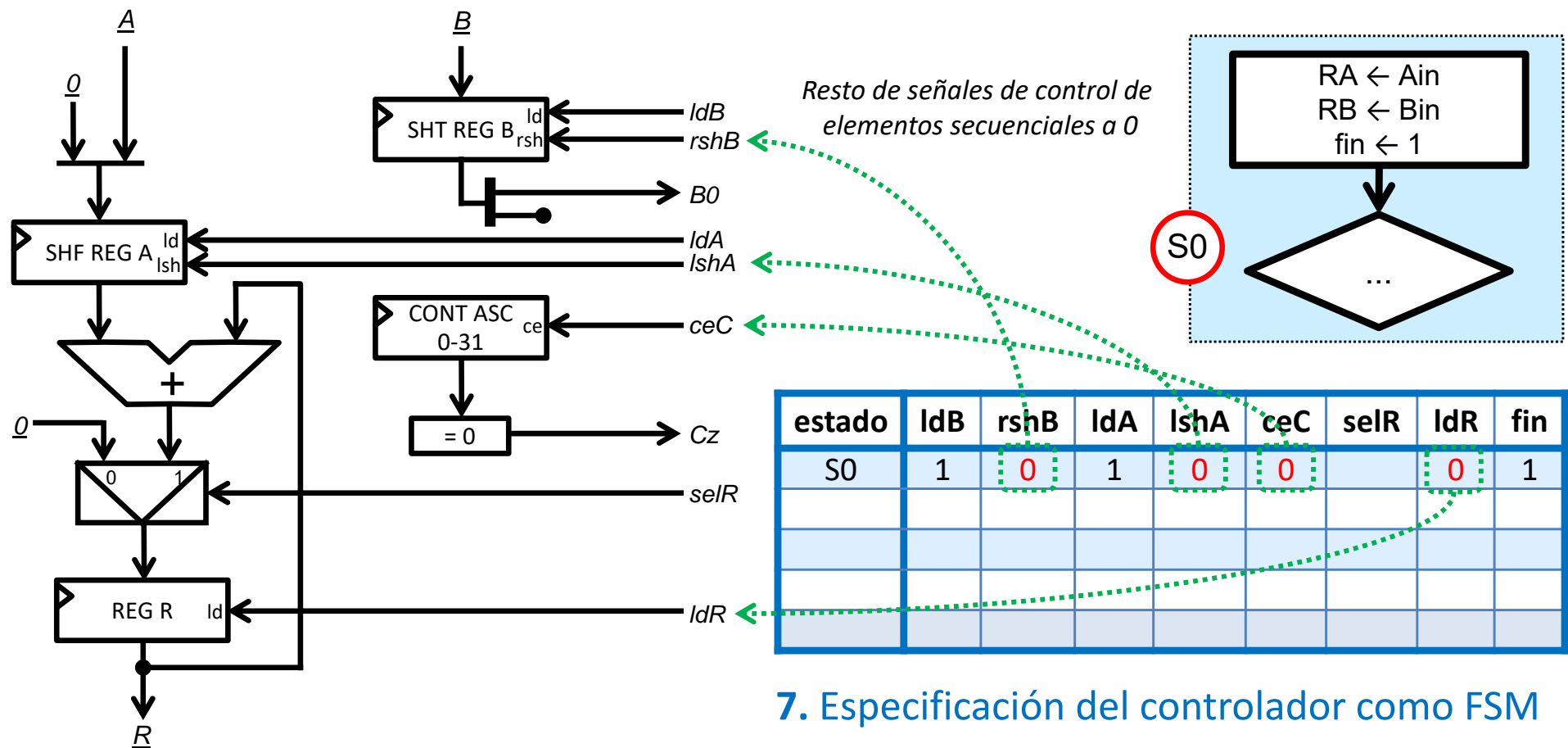
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



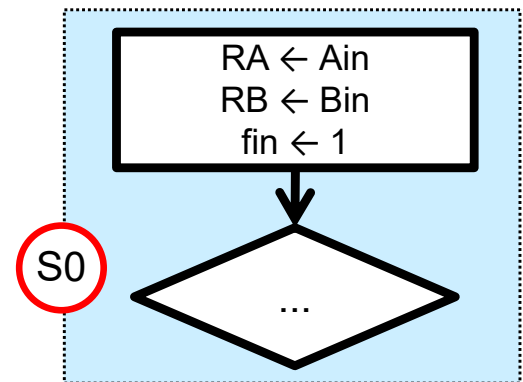
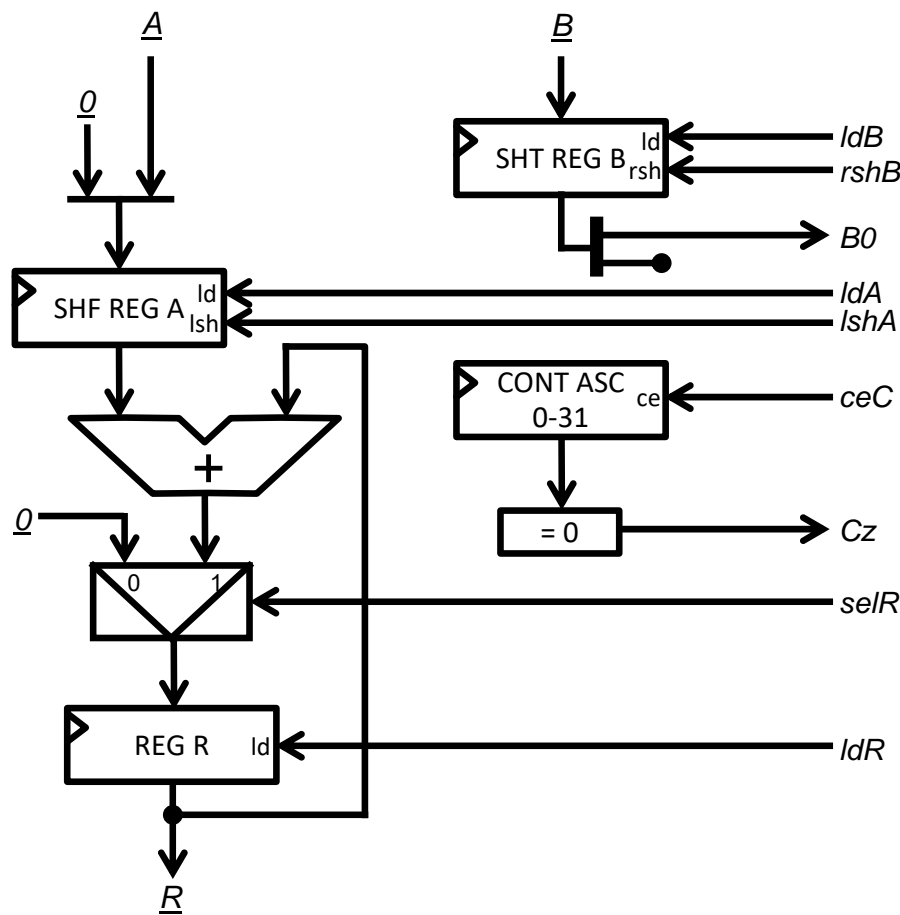
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



Si R no carga, es irrelevante el valor que selecciona el MUX

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1

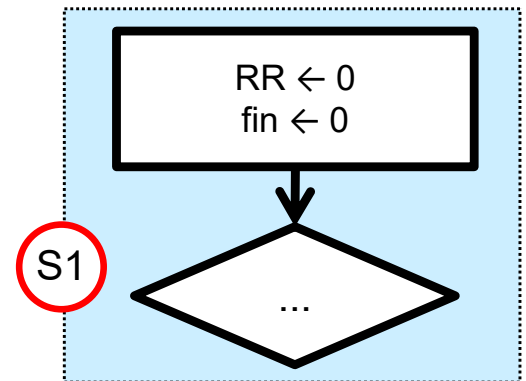
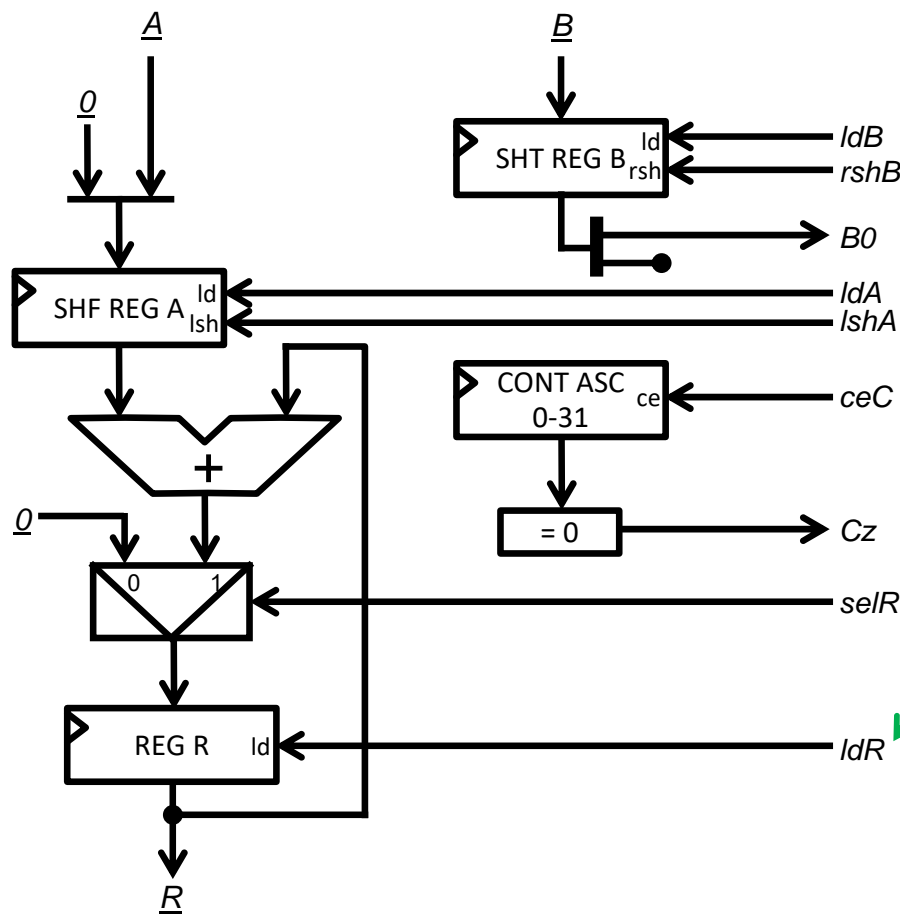
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



Señal de carga de R a 1

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1							1	0

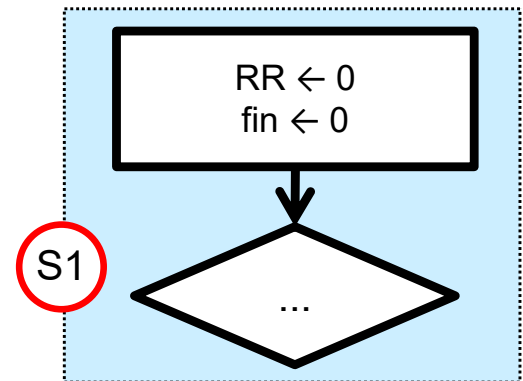
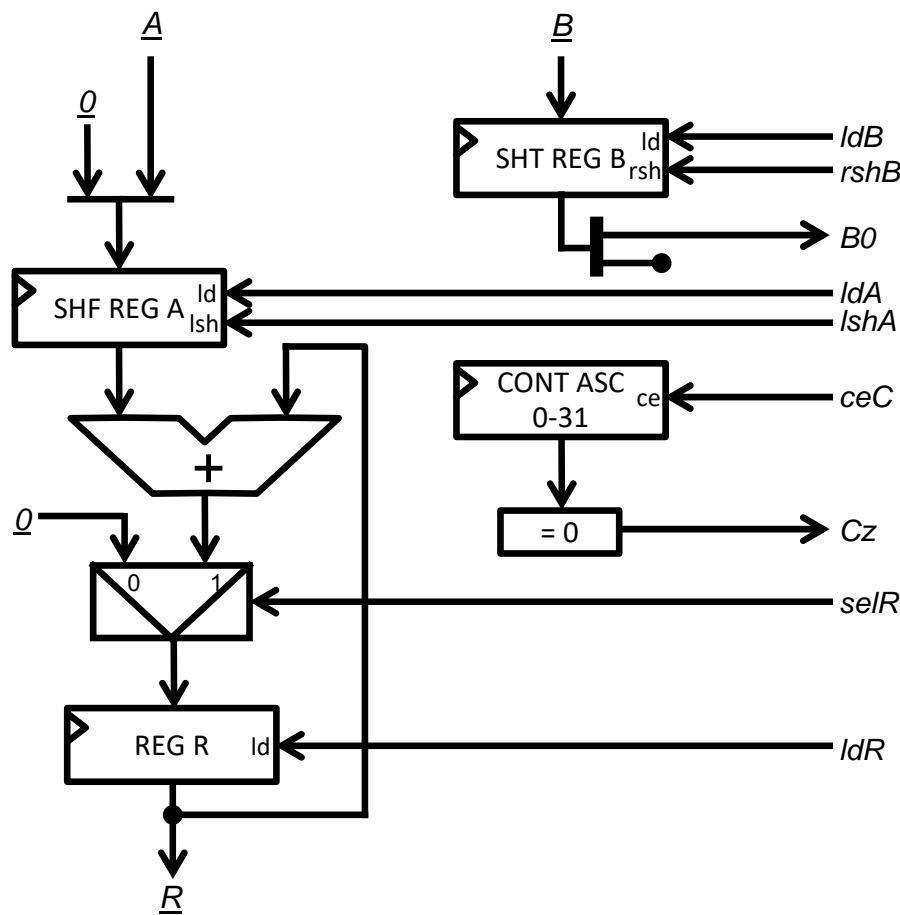
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



El MUX debe seleccionar el 0 para que R lo cargue

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1						0	1	0

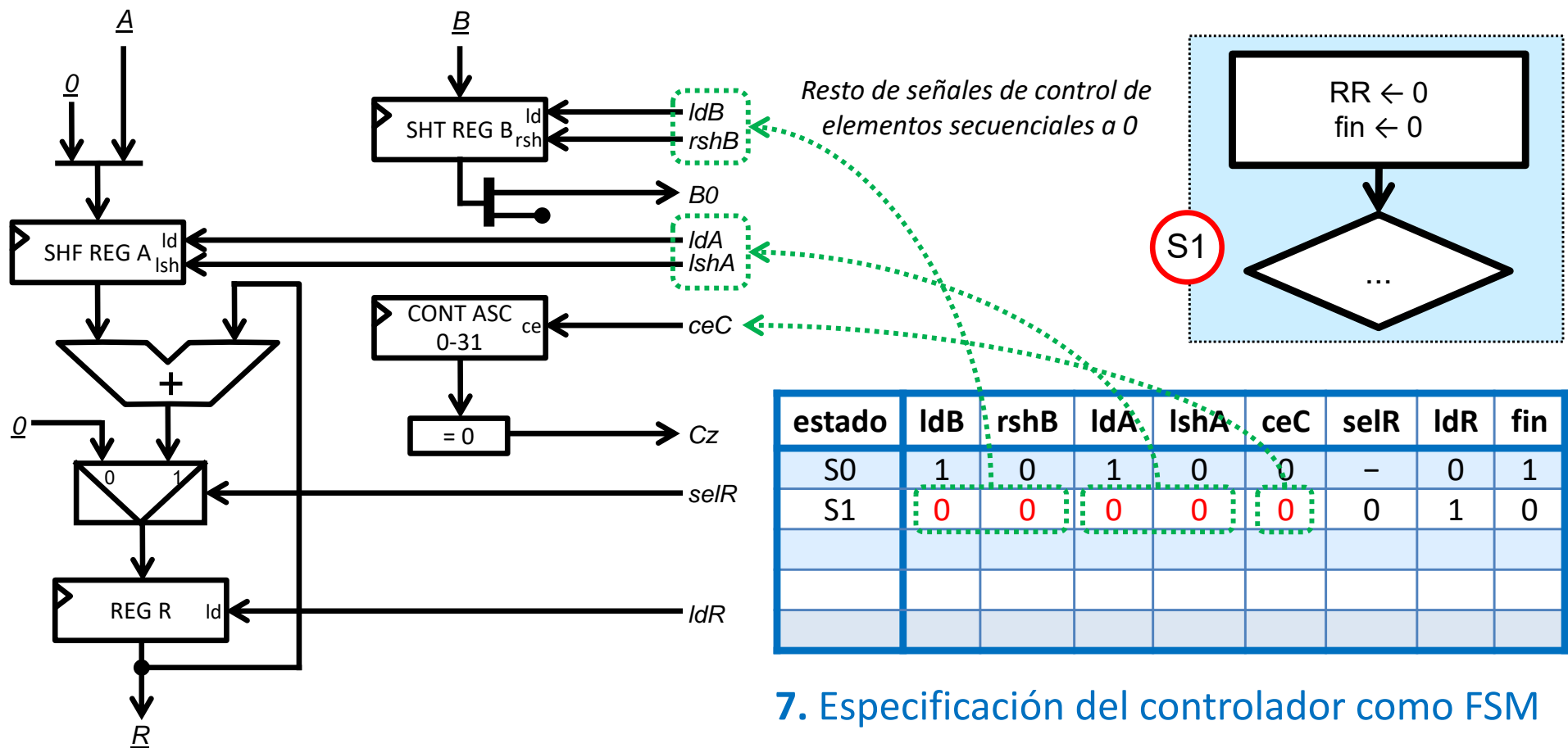
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



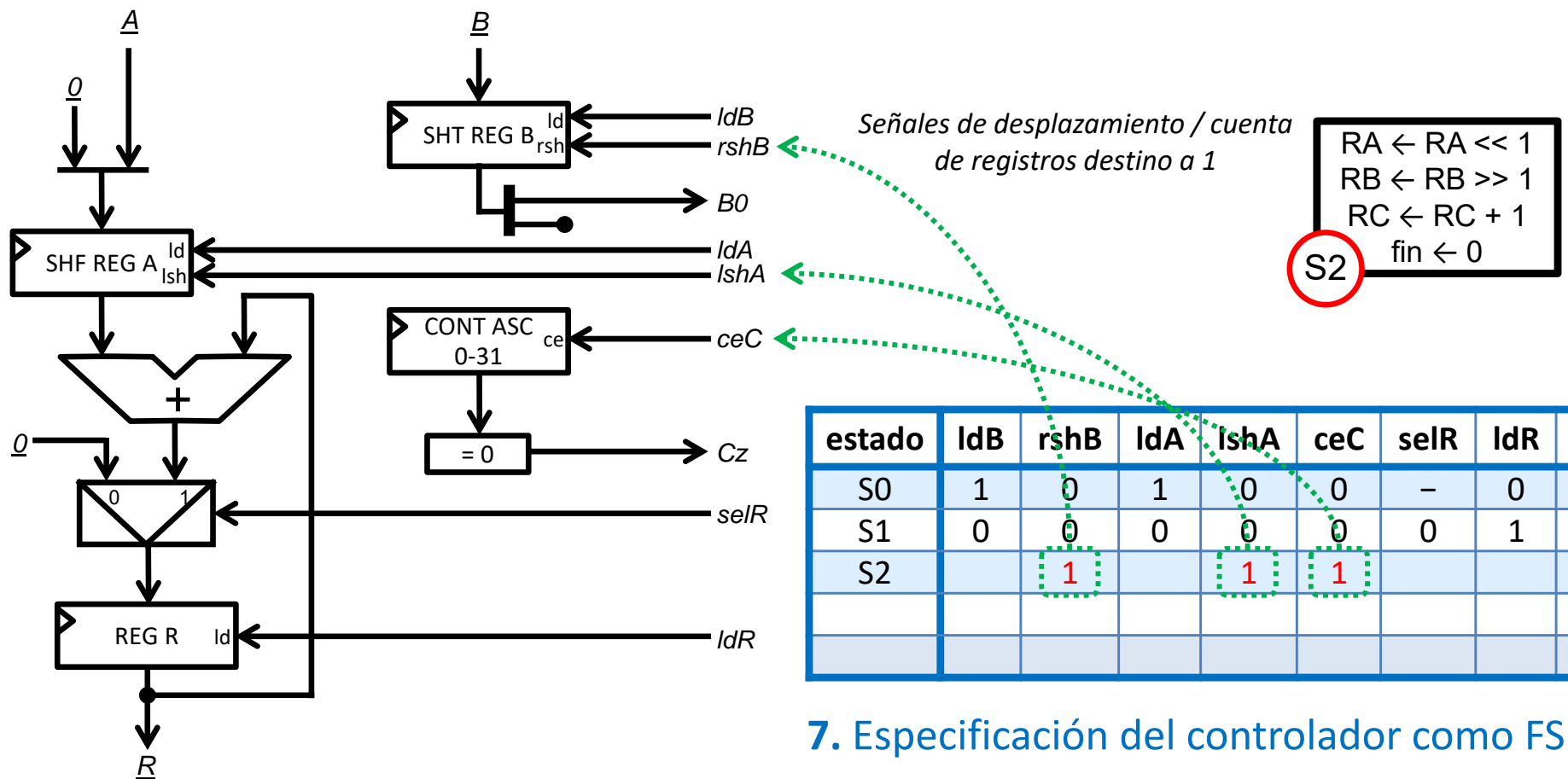
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



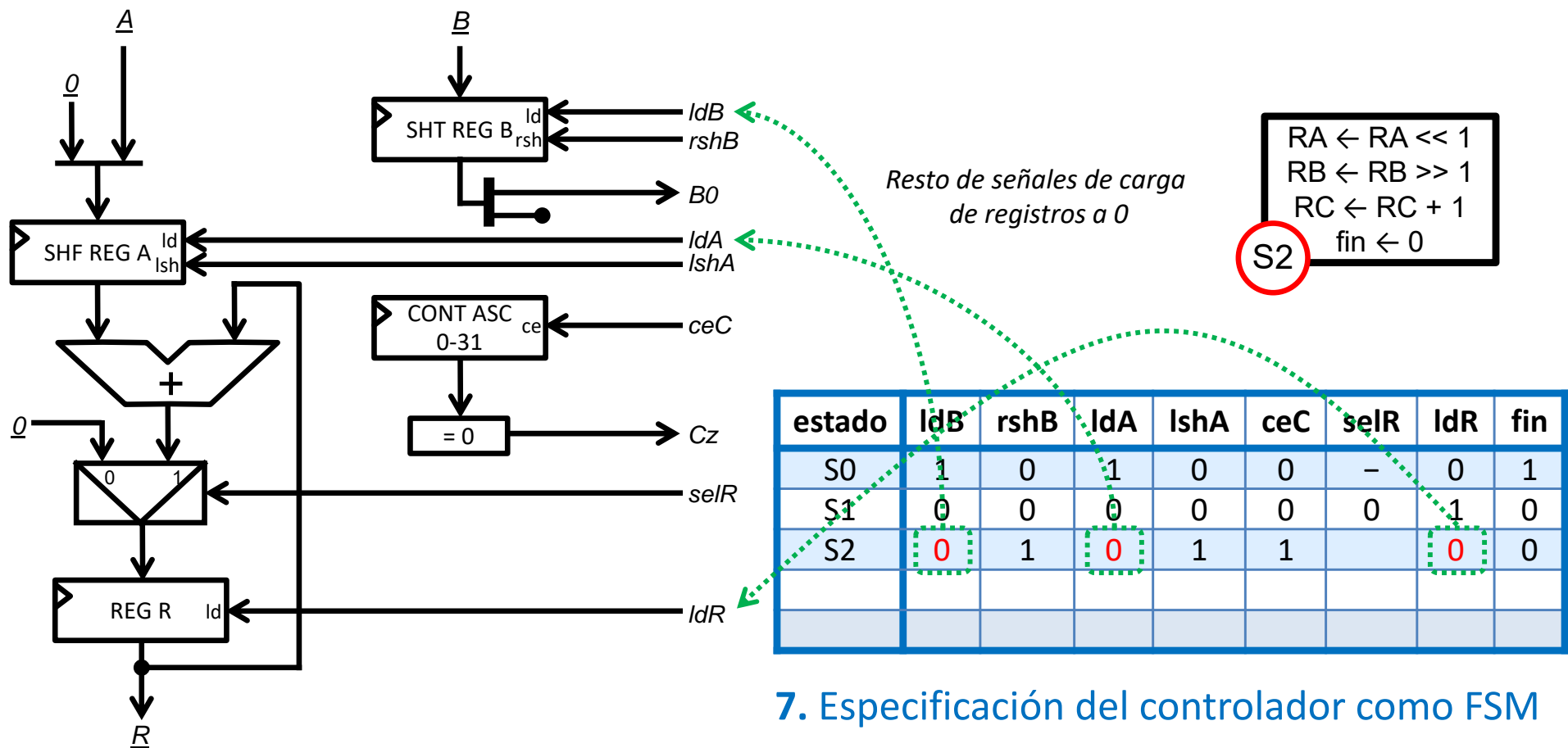
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



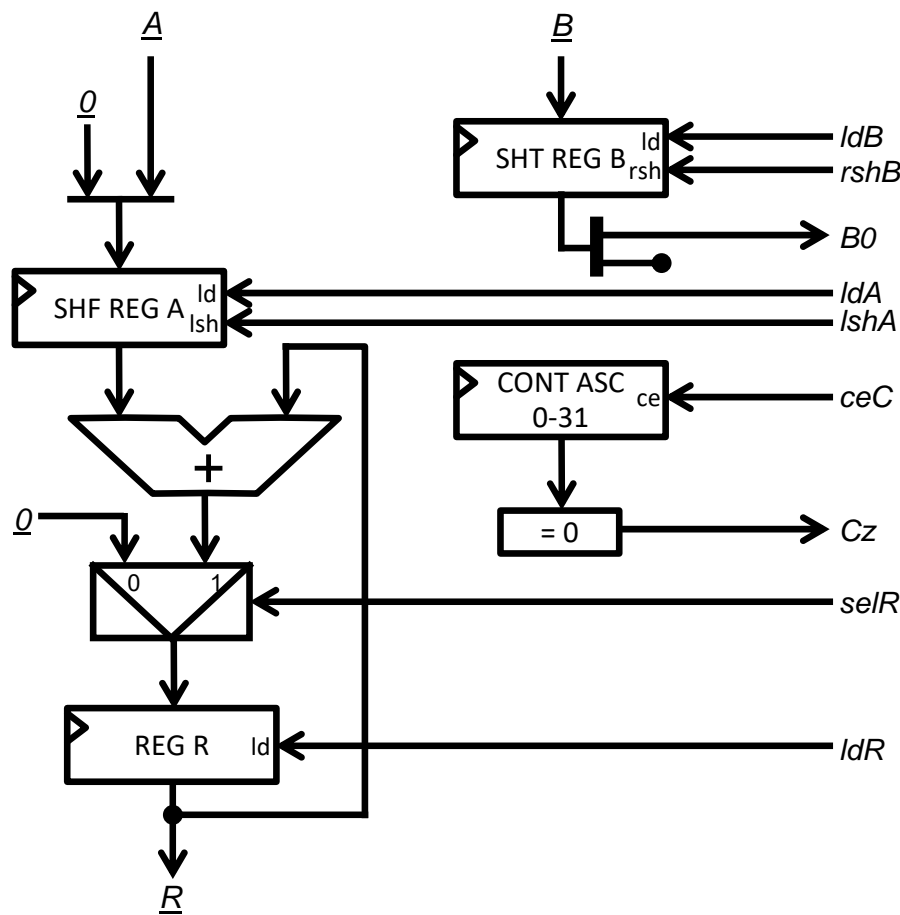
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



S2

$$RA \leftarrow RA \ll 1$$

$$RB \leftarrow RB \gg 1$$

$$RC \leftarrow RC + 1$$

$$fin \leftarrow 0$$

Si R no carga, es irrelevante el valor que selecciona el MUX

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0

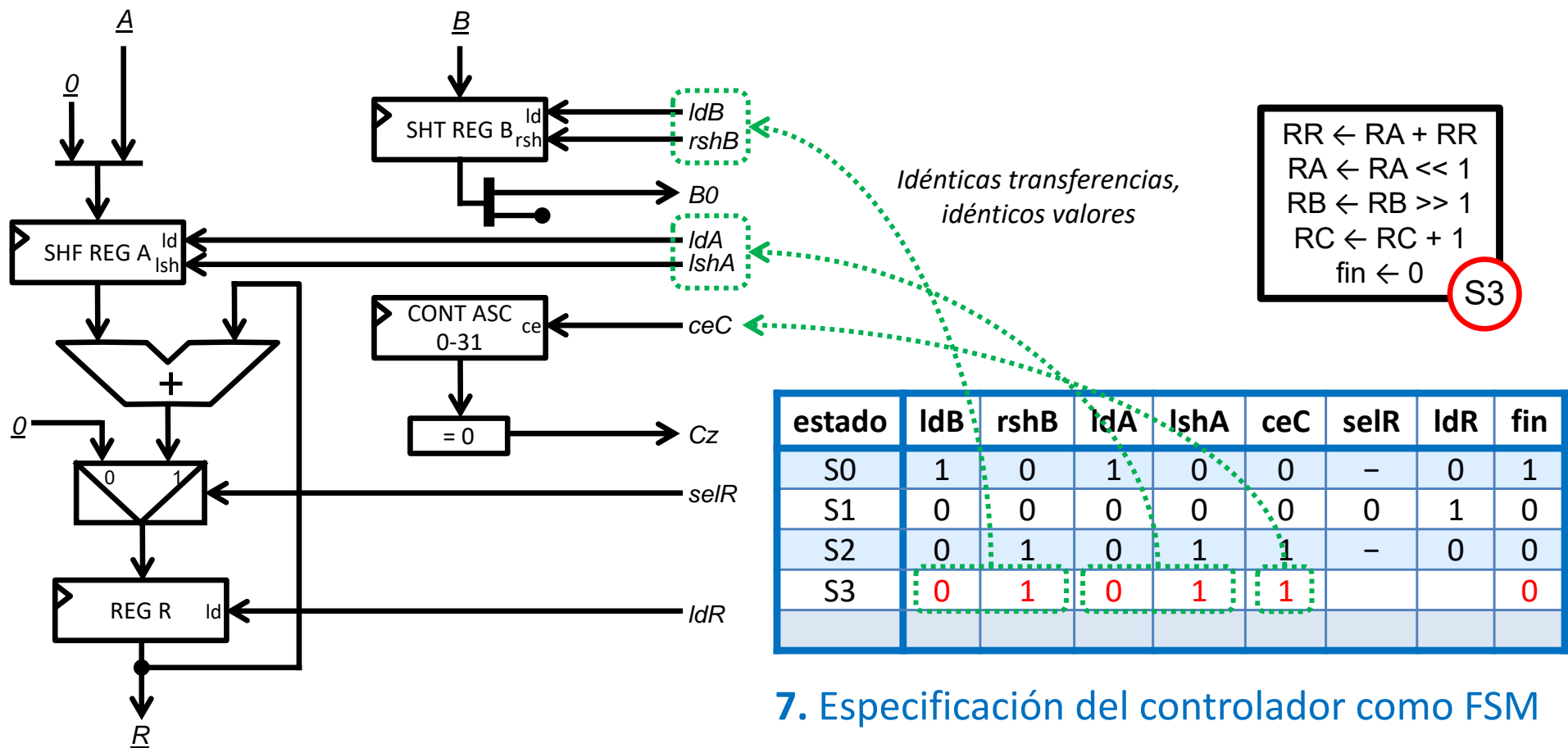
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



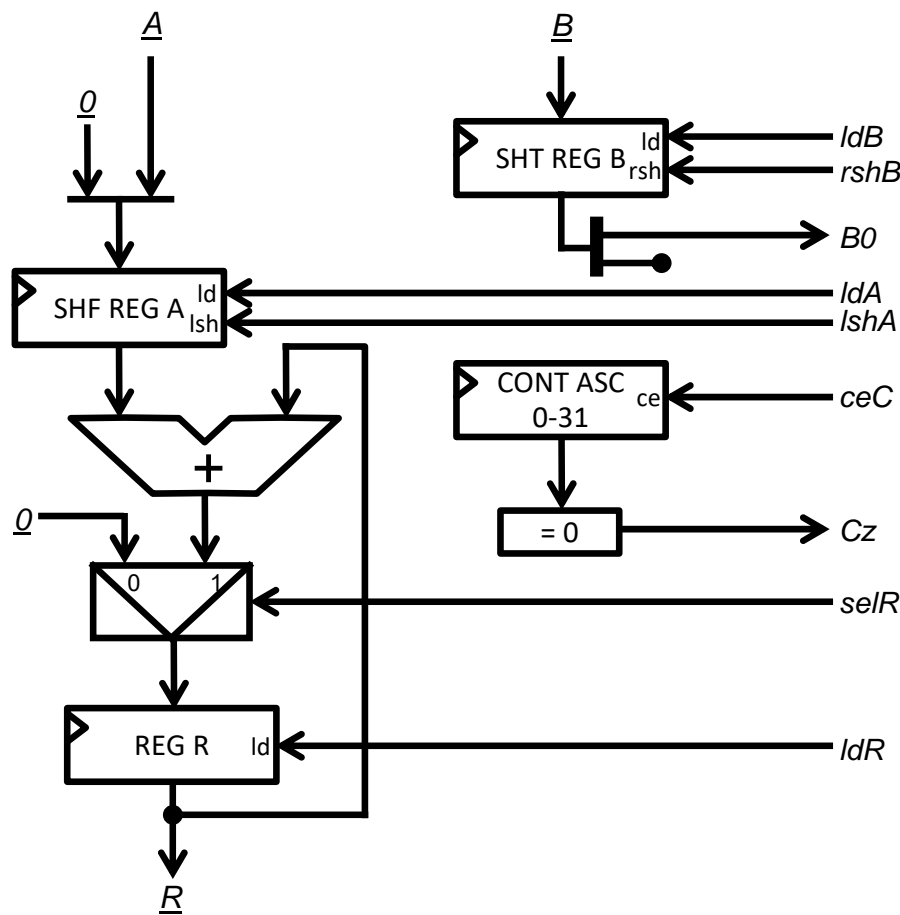
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



$RR \leftarrow RA + RR$
 $RA \leftarrow RA \ll 1$
 $RB \leftarrow RB \gg 1$
 $RC \leftarrow RC + 1$
 $fin \leftarrow 0$

S3

R carga el valor calculado por el sumador

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0

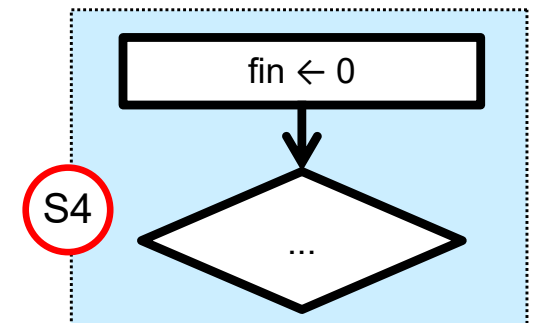
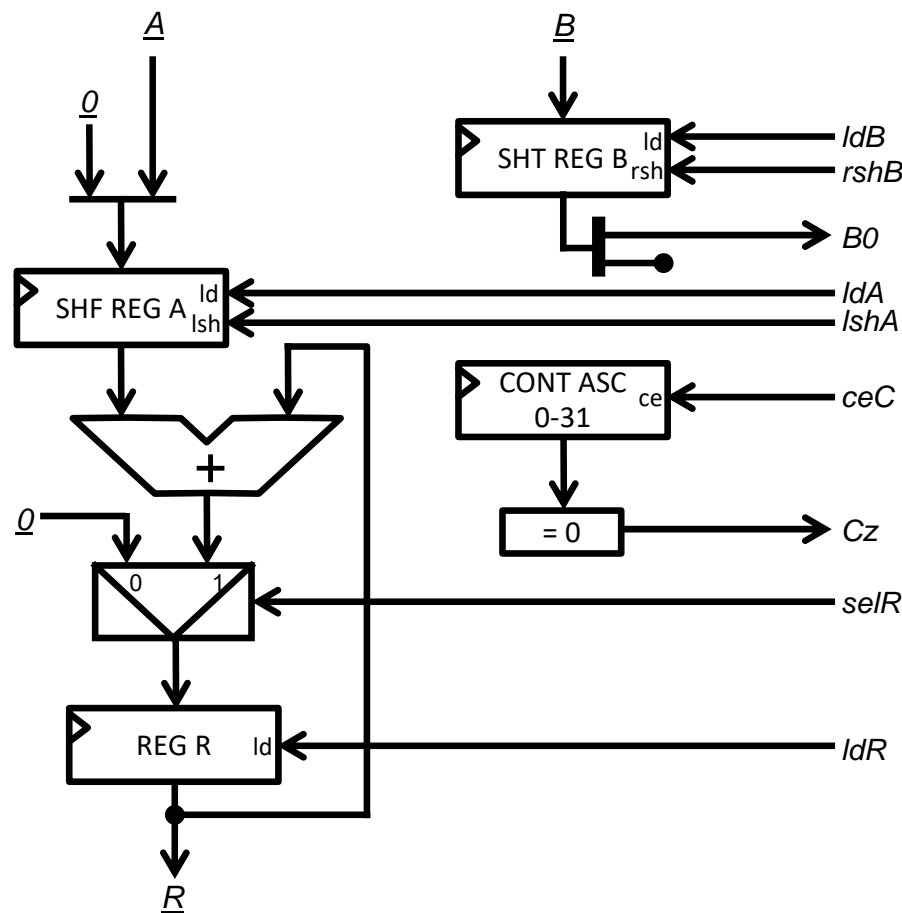
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

- Diagrama ASM = Diagrama de estados de una FSM.
 - Cada transferencia entre registros de un bloque ASM se traduce a valores de señales de control en el estado correspondiente.



estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0

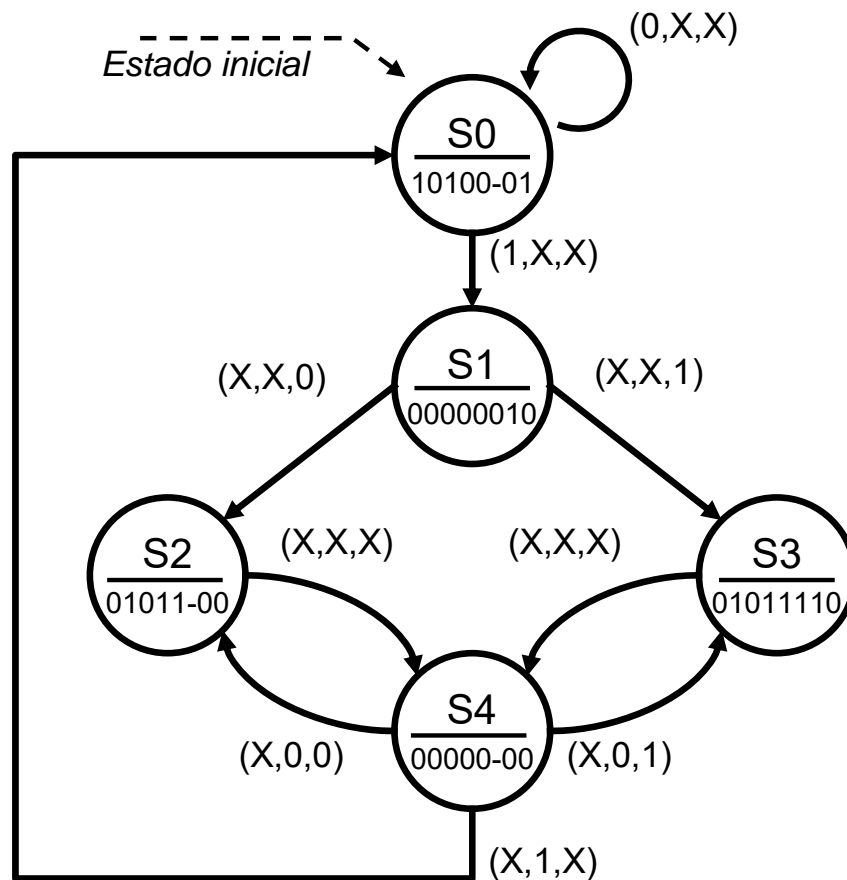
7. Especificación del controlador como FSM



Multiplicador iterativo

Algoritmo como FSM

entradas del controlador: (inicio, Cz, B0)
 salidas del controlador: (ldB, rshB, ldA, lshA, ceC, selr, LDr, fin)



Función de transición de estados

estado	inicio	Cz	B0	estado'
S0	0	X	X	S0
S0	1	X	X	S1
S1	X	X	0	S2
S1	X	X	1	S3
S2	X	X	X	S4
S3	X	X	X	S4
S4	X	0	0	S2
S4	X	0	1	S3
S4	X	1	X	S0

Función de salida

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0



Multiplicador iterativo

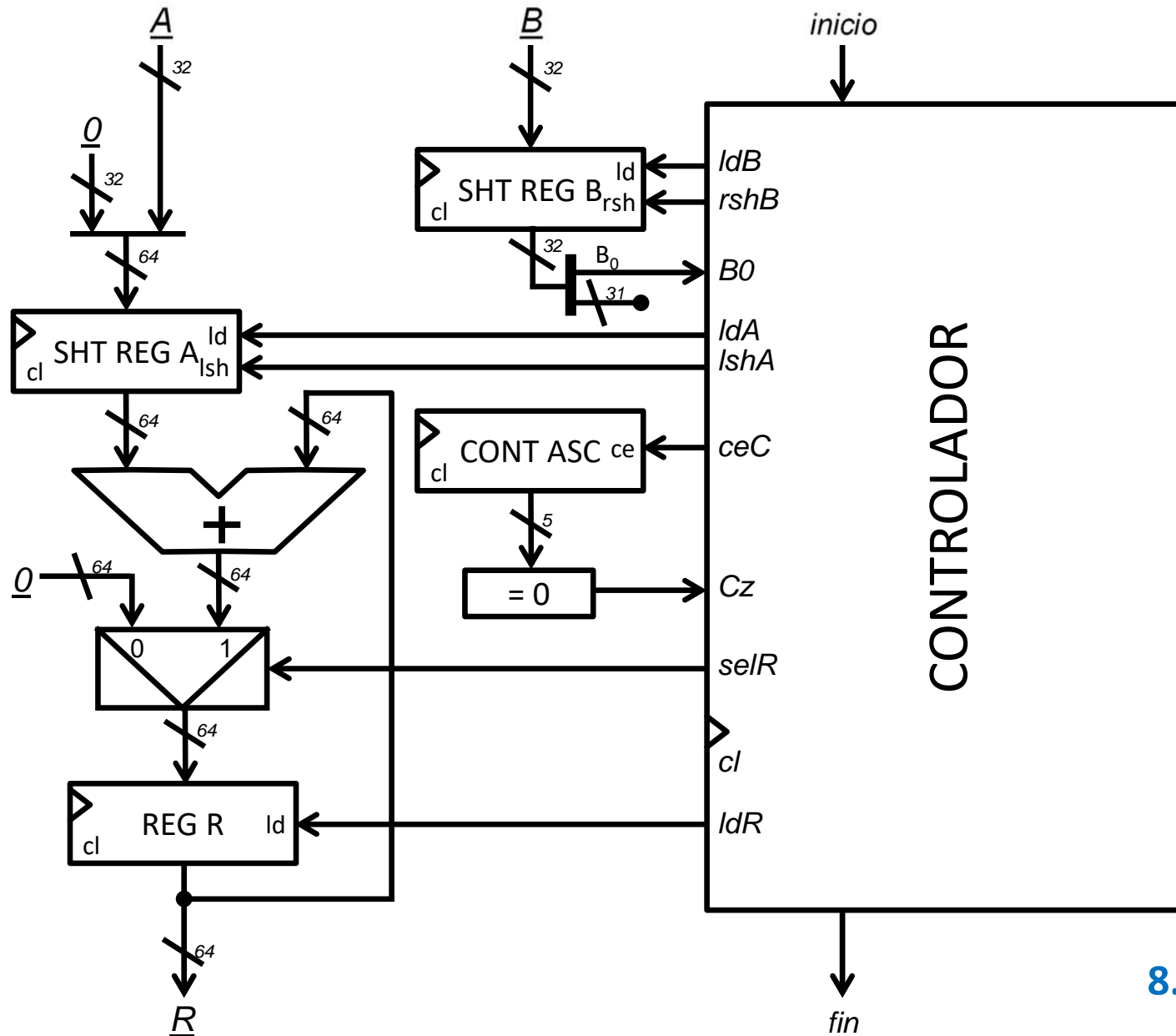
Diseño del controlador

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

63



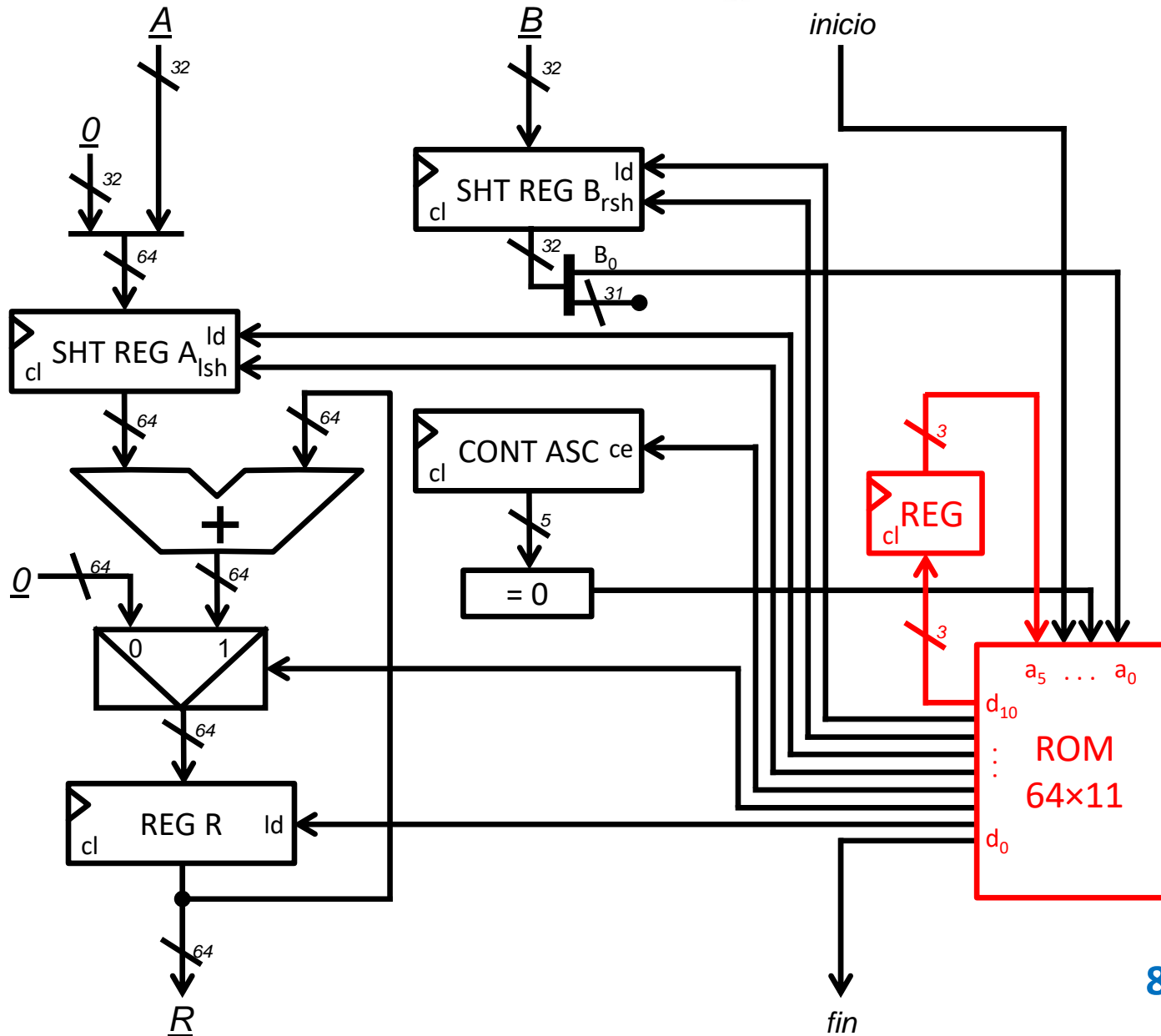
- S0** $RA \leftarrow A_{in}; RB \leftarrow B_{in};$
 $fin \leftarrow 1;$
 si ($inicio=0$) ir a S0;
 si ($inicio=1$) ir a S1;
- S1** $RR \leftarrow 0; fin \leftarrow 0;$
 si ($B_0=0$) ir a S2;
 si ($B_0=1$) ir a S3;
- S2** $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
 ir a S4;
- S3** $RR \leftarrow RA + RR;$
 $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
 ir a S4;
- S4** $fin \leftarrow 0;$
 si ($Cz=1$) ir a S0;
 si ($Cz=0$ y $B_0=0$) ir a S2;
 si ($Cz=0$ y $B_0=1$) ir a S3;

8. Diseño del controlador



Multiplicador iterativo

ROM + registro de estado



- S0** $RA \leftarrow A_{in}; RB \leftarrow B_{in};$
 $fin \leftarrow 1;$
 si ($inicio=0$) ir a S0;
 si ($inicio=1$) ir a S1;
- S1** $RR \leftarrow 0; fin \leftarrow 0;$
 si ($B_0=0$) ir a S2;
 si ($B_0=1$) ir a S3;
- S2** $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
 ir a S4;
- S3** $RR \leftarrow RA + RR;$
 $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
 ir a S4;
- S4** $fin \leftarrow 0;$
 si ($Cz=1$) ir a S0;
 si ($Cz=0$ y $B_0=0$) ir a S2;
 si ($Cz=0$ y $B_0=1$) ir a S3;

8. Diseño del controlador



Multiplicador iterativo

Generación del contenido de la ROM

q_2	q_1	q_0	inicio	Cz	B0
a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
...
1	1	1	1	1	1

dirección

DIR.	q_2'	q_1'	q_0'	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
0	0	0	0	1	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	0	1
2	0	0	0	1	0	1	0	0	0	0	1
3	0	0	0	1	0	1	0	0	0	0	1
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
...
63											

ROM 64x11

estado	inicio	Cz	B0	estado'
S0	0	X	X	S0
S0	1	X	X	S1
S1	X	X	0	S2
S1	X	X	1	S3
S2	X	X	X	S4
S3	X	X	X	S4
S4	X	0	0	S2
S4	X	0	1	S3
S4	X	1	X	S0

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0

8. Diseño del controlador



Multiplicador iterativo

Generación del contenido de la ROM

q_2	q_1	q_0	inicio	Cz	B0
a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
...					
1	1	1	1	1	1

dirección

q_2'	q_1'	q_0'	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin	
DIR.	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	1	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	0	1
2	0	0	0	1	0	1	0	0	0	0	1
3	0	0	0	1	0	1	0	0	0	0	1
4	0	0	1	1	0	1	0	0	0	0	1
5	0	0	1	1	0	1	0	0	0	0	1
6	0	0	1	1	0	1	0	0	0	0	1
7	0	0	1	1	0	1	0	0	0	0	1
8											
9											
10											
11											
12											
13											
14											
15											
...											
63											

ROM 64×11

estado	inicio	Cz	B0	estado'
S0	0	X	X	S0
S0	1	X	X	S1
S1	X	X	0	S2
S1	X	X	1	S3
S2	X	X	X	S4
S3	X	X	X	S4
S4	X	0	0	S2
S4	X	0	1	S3
S4	X	1	X	S0

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0

8. Diseño del controlador



Multiplicador iterativo

Generación del contenido de la ROM

q_2	q_1	q_0	inicio	Cz	B0
a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
...					
1	1	1	1	1	1

dirección

q_2'	q_1'	q_0'	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin	
DIR.	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	1	0	1	0	0	0	0	1
1	0	0	0	1	0	1	0	0	0	0	1
2	0	0	0	1	0	1	0	0	0	0	1
3	0	0	0	1	0	1	0	0	0	0	1
4	0	0	1	1	0	1	0	0	0	0	1
5	0	0	1	1	0	1	0	0	0	0	1
6	0	0	1	1	0	1	0	0	0	0	1
7	0	0	1	1	0	1	0	0	0	0	1
8	0	1	0	0	0	0	0	0	0	1	0
9											
10	0	1	0	0	0	0	0	0	0	1	0
11											
12	0	1	0	0	0	0	0	0	0	1	0
13											
14	0	1	0	0	0	0	0	0	0	1	0
15											
...											
63											

ROM 64×11

estado	inicio	Cz	B0	estado'
S0	0	X	X	S0
S0	1	X	X	S1
S1	X	X	0	S2
S1	X	X	1	S3
S2	X	X	X	S4
S3	X	X	X	S4
S4	X	0	0	S2
S4	X	0	1	S3
S4	X	1	X	S0

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0

8. Diseño del controlador



Multiplicador iterativo

Generación del contenido de la ROM

a_5	a_4	a_3	a_2	a_1	a_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
...
1	1	1	1	1	1

dirección

DIR.	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	1	0	1	0	0	1	1	1
1	0	0	0	1	0	1	0	0	1	1	1
2	0	0	0	1	0	1	0	0	1	1	1
3	0	0	0	1	0	1	0	0	1	1	1
4	0	0	1	1	0	1	0	0	1	1	1
5	0	0	1	1	0	1	0	0	1	1	1
6	0	0	1	1	0	1	0	0	1	1	1
7	0	0	1	1	0	1	0	0	1	1	1
8	0	1	0	0	0	0	0	0	0	1	0
9	0	1	1	0	0	0	0	0	0	1	0
10	0	1	0	0	0	0	0	0	0	1	0
11	0	1	1	0	0	0	0	0	0	1	0
12	0	1	0	0	0	0	0	0	0	1	0
13	0	1	1	0	0	0	0	0	0	1	0
14	0	1	0	0	0	0	0	0	0	1	0
15	0	1	1	0	0	0	0	0	0	1	0
...
63											

ROM 64×11

estado	inicio	Cz	B0	estado'
S0	0	X	X	S0
S0	1	X	X	S1
S1	X	X	0	S2
S1	X	X	1	S3
S2	X	X	X	S4
S3	X	X	X	S4
S4	X	0	0	S2
S4	X	0	1	S3
S4	X	1	X	S0

estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	-	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	-	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	-	0	0

8. Diseño del controlador



Multiplicador iterativo

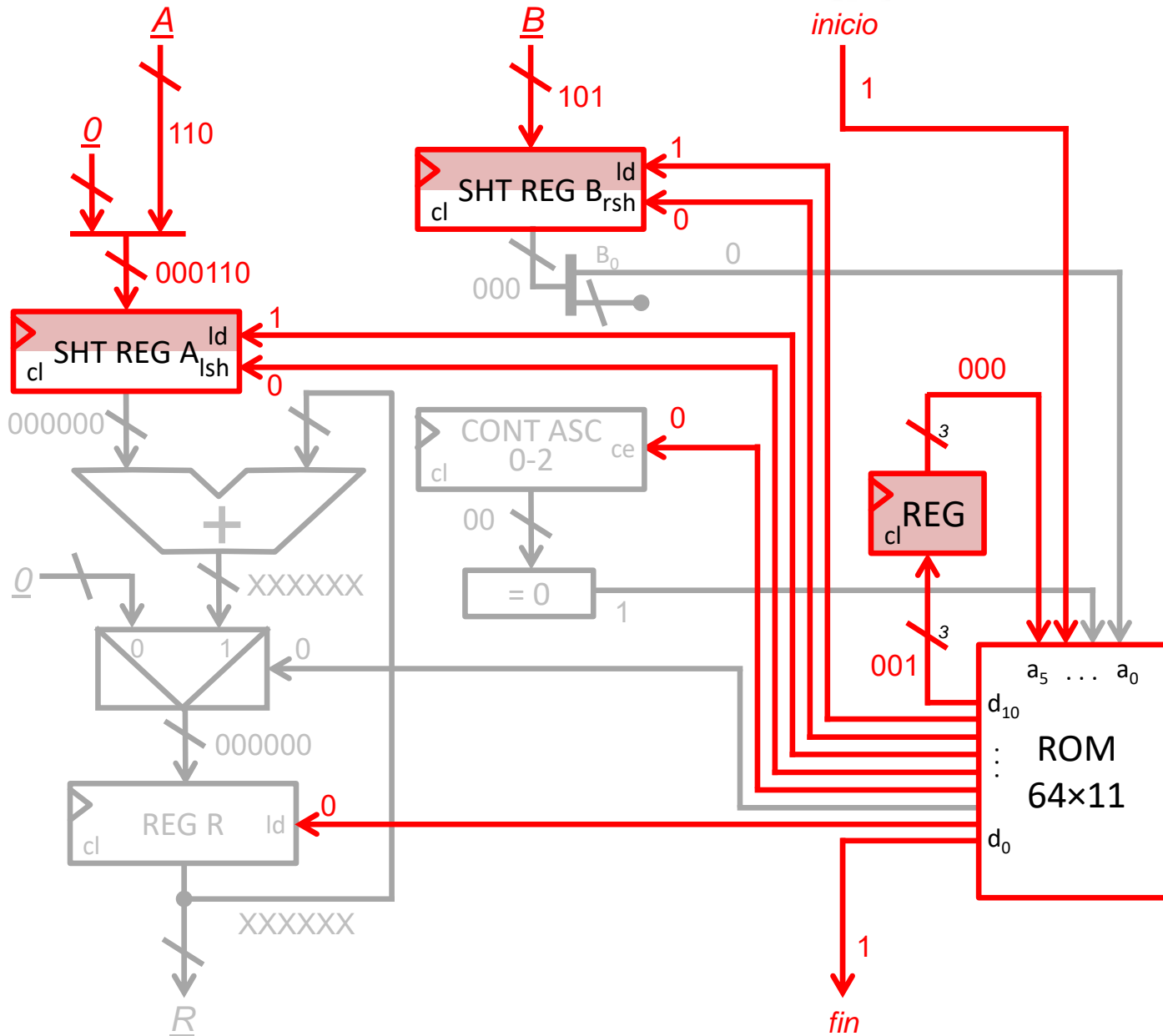
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

69



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

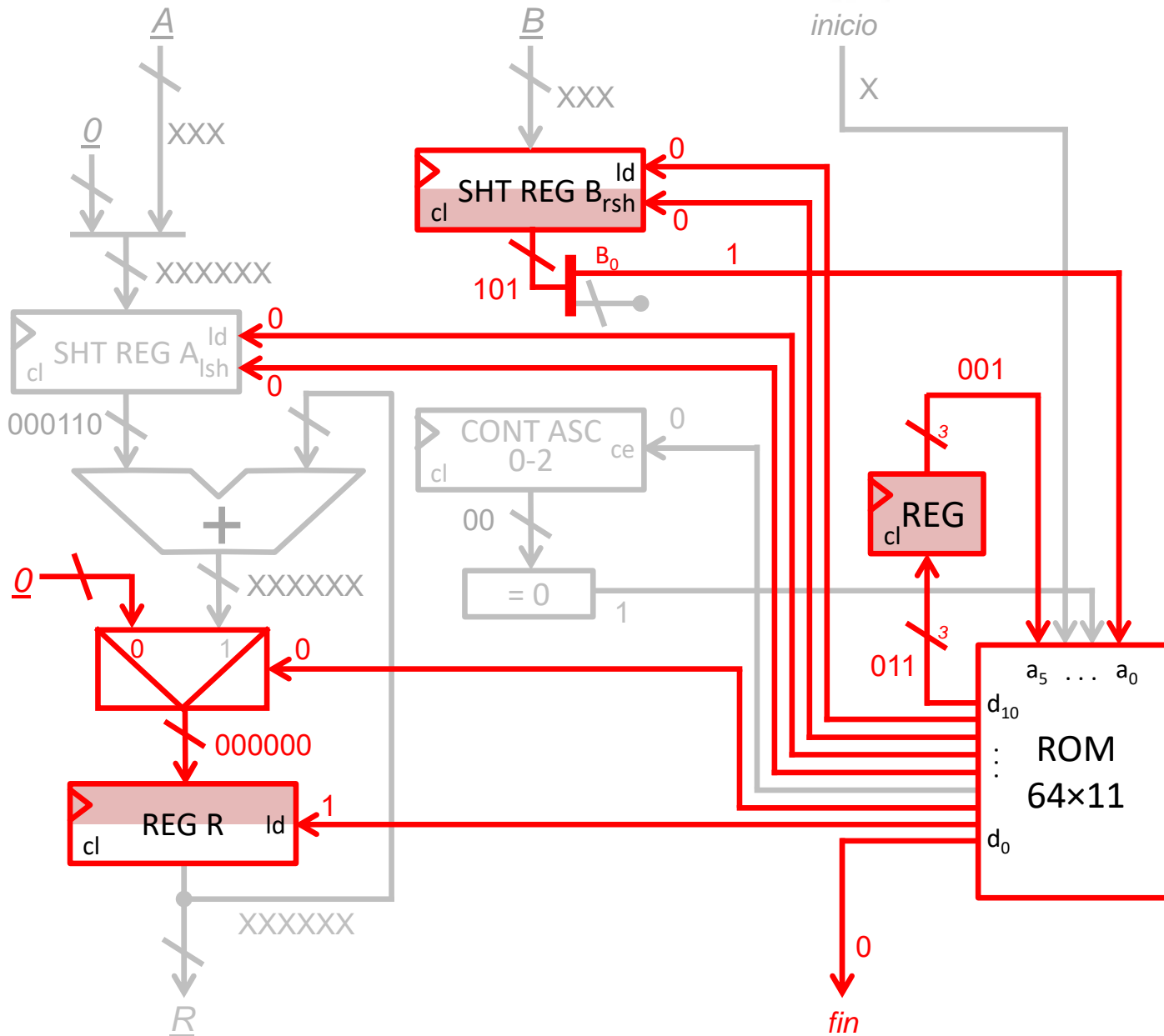
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

70



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

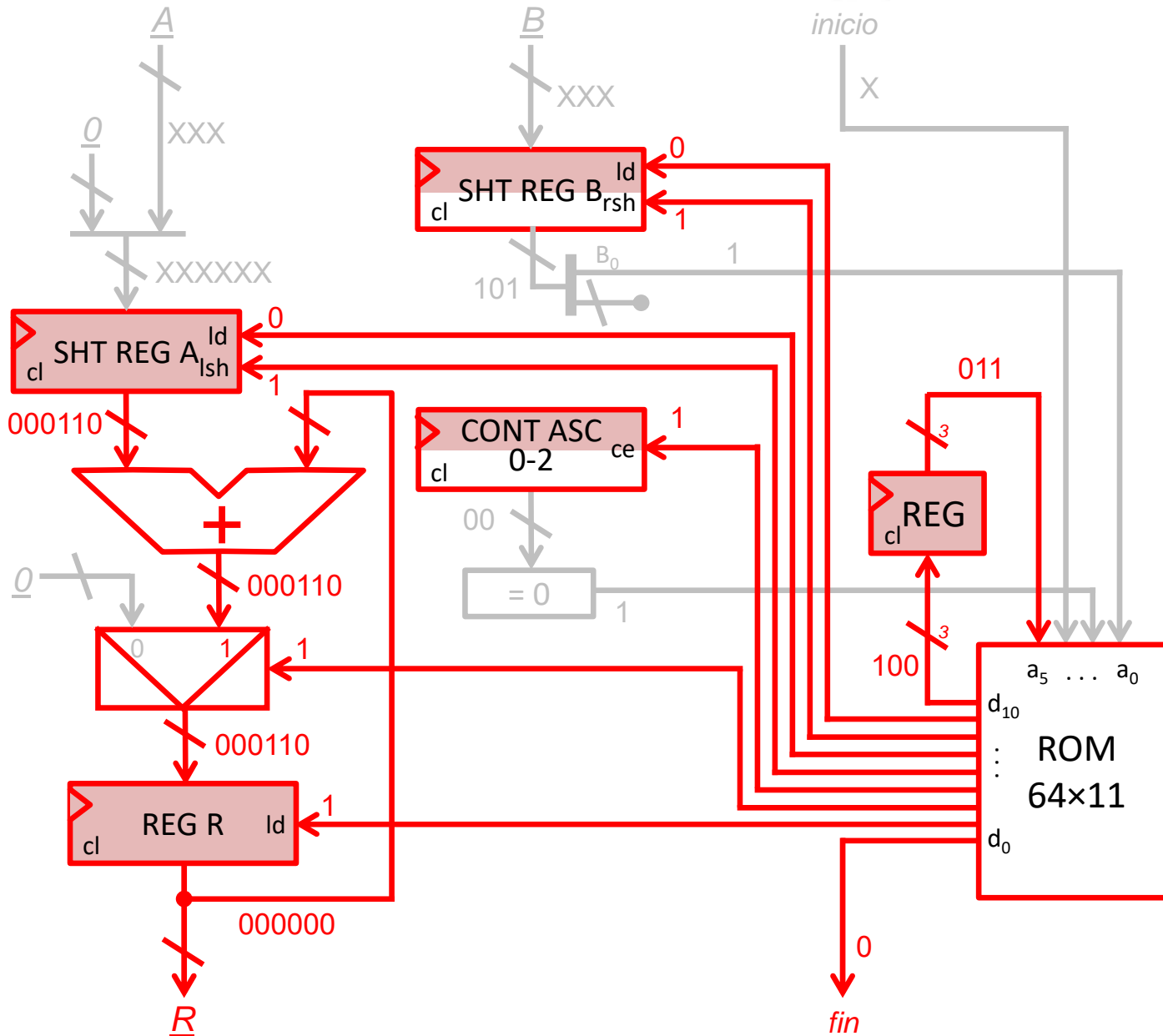
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

71



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

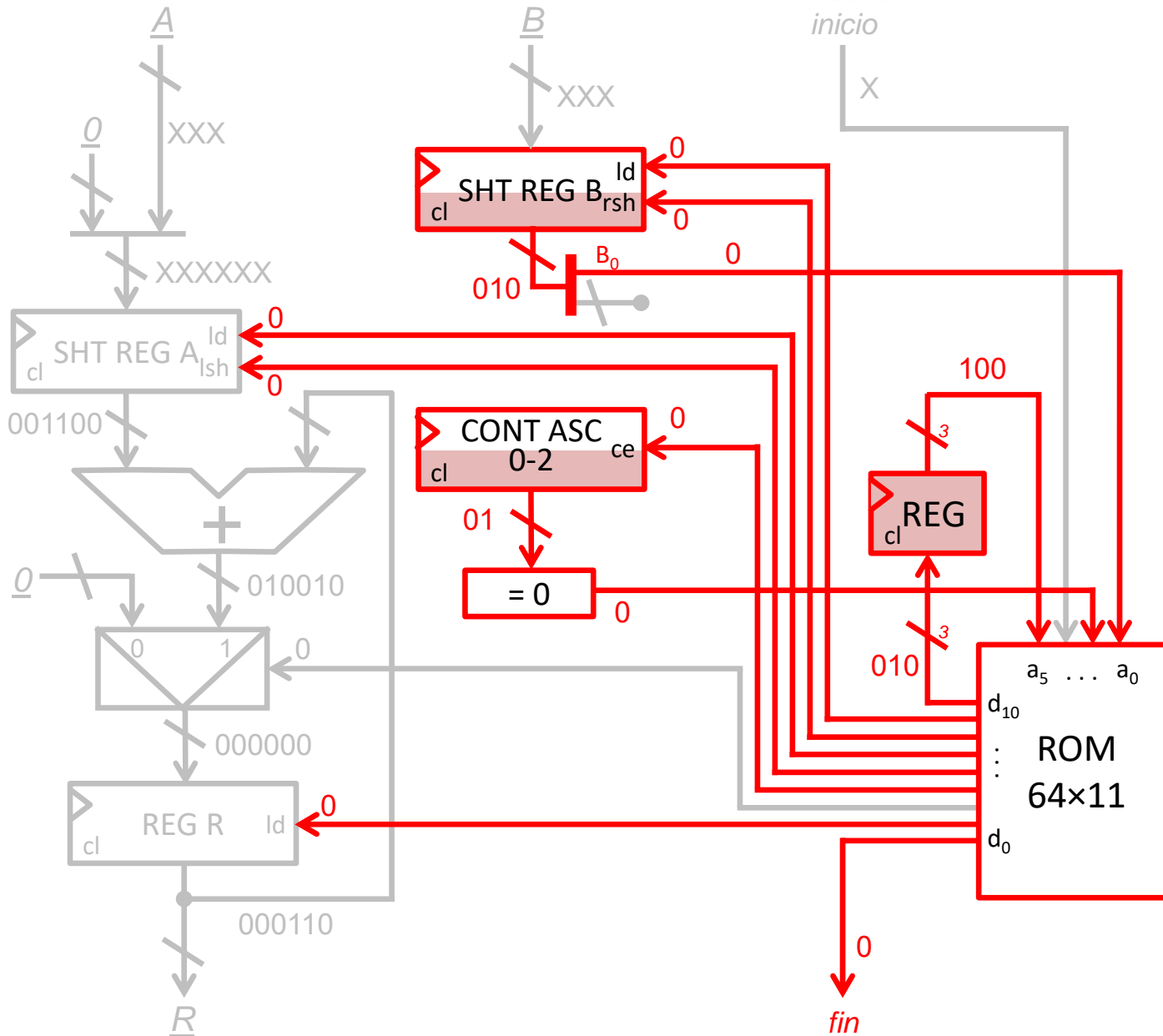
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

72



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

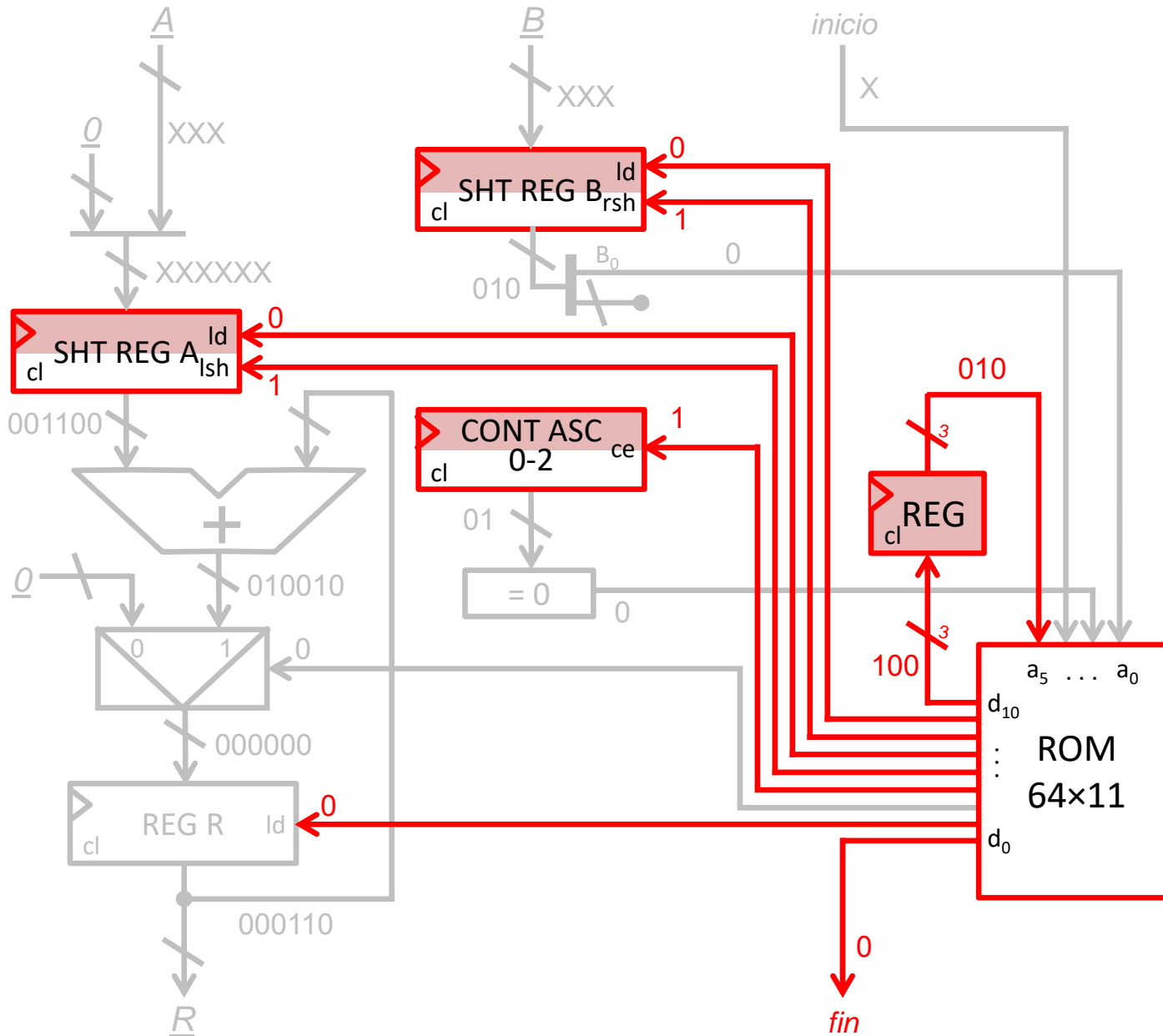
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

73



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

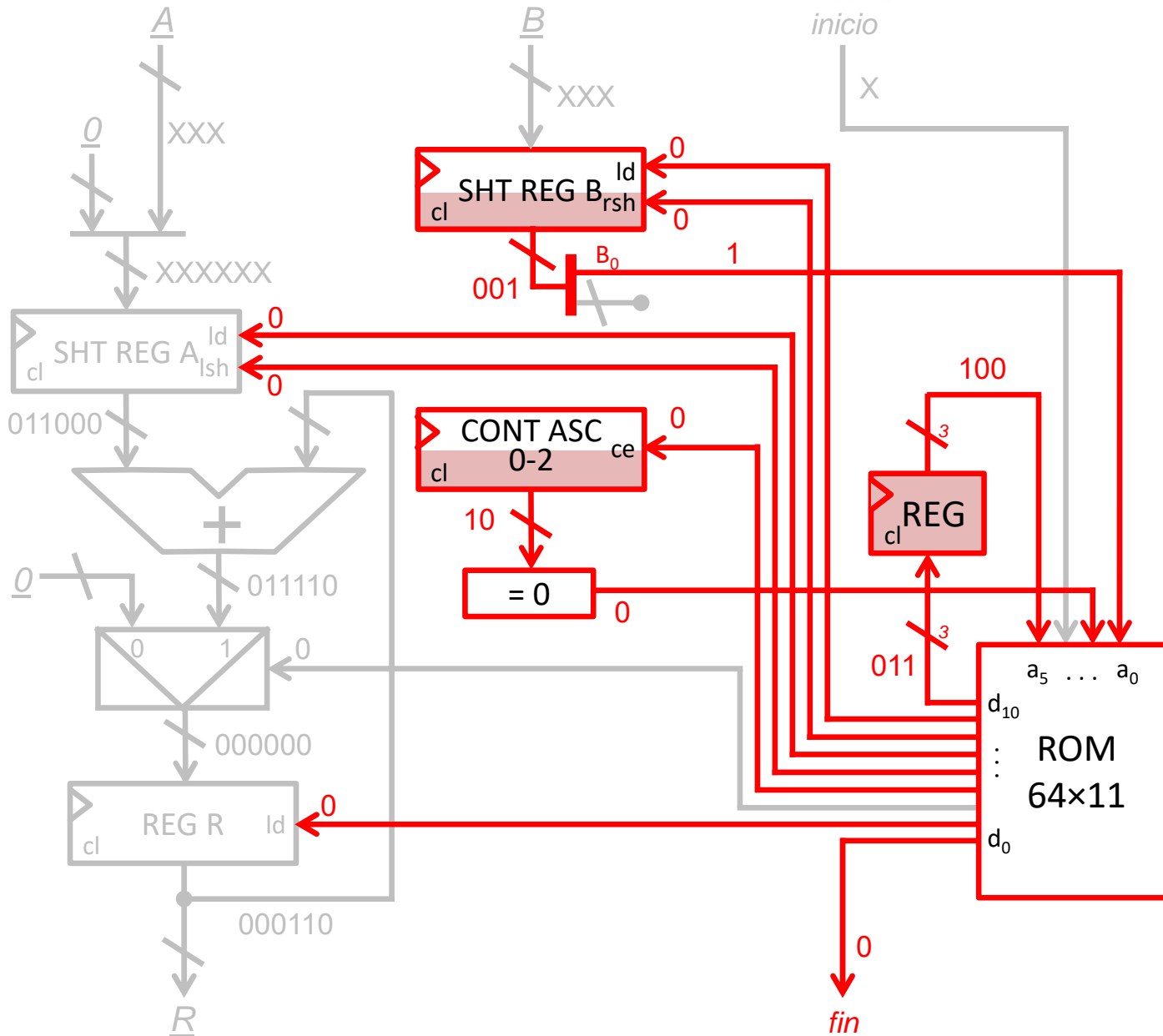
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

74



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

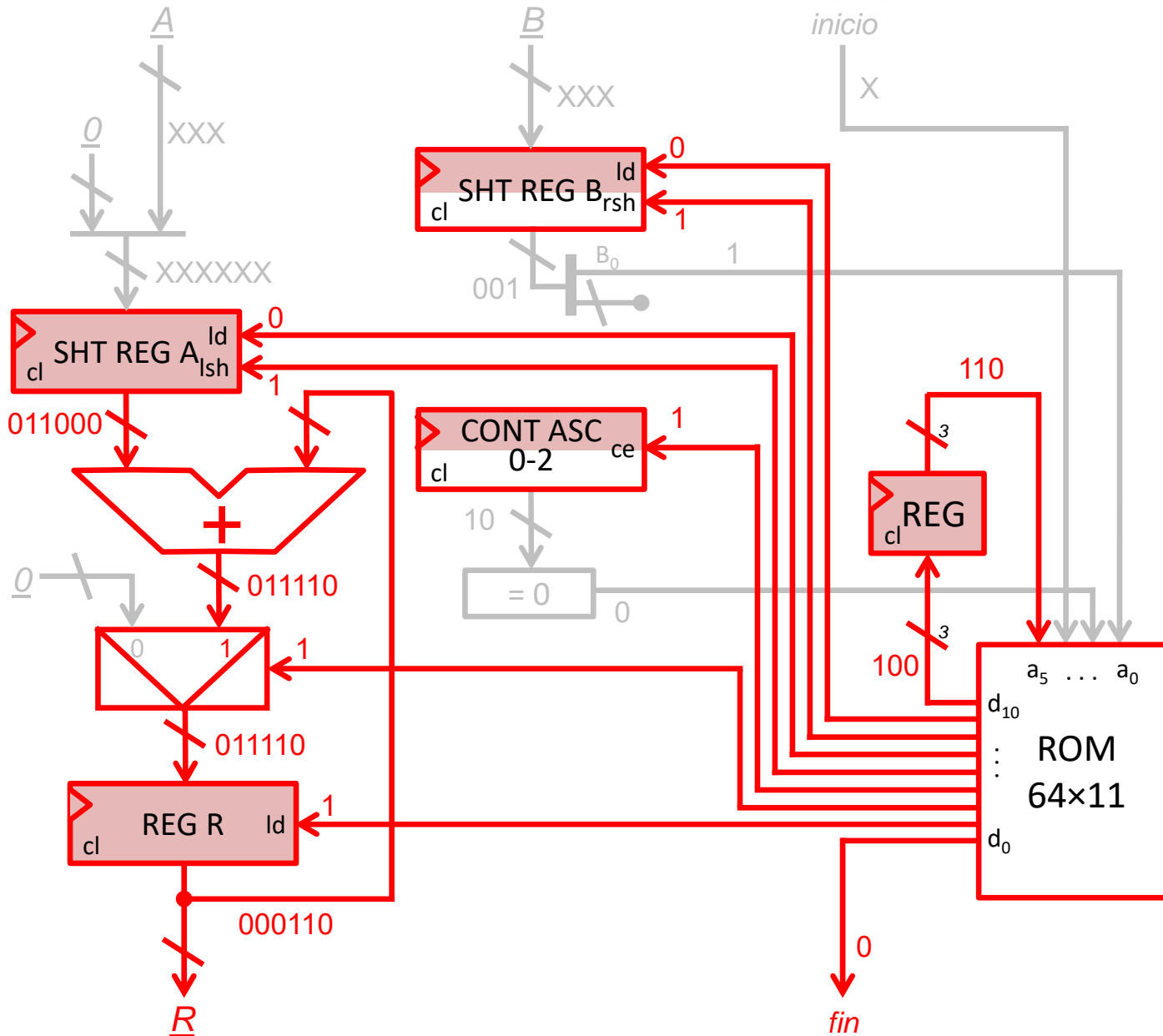
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

75



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

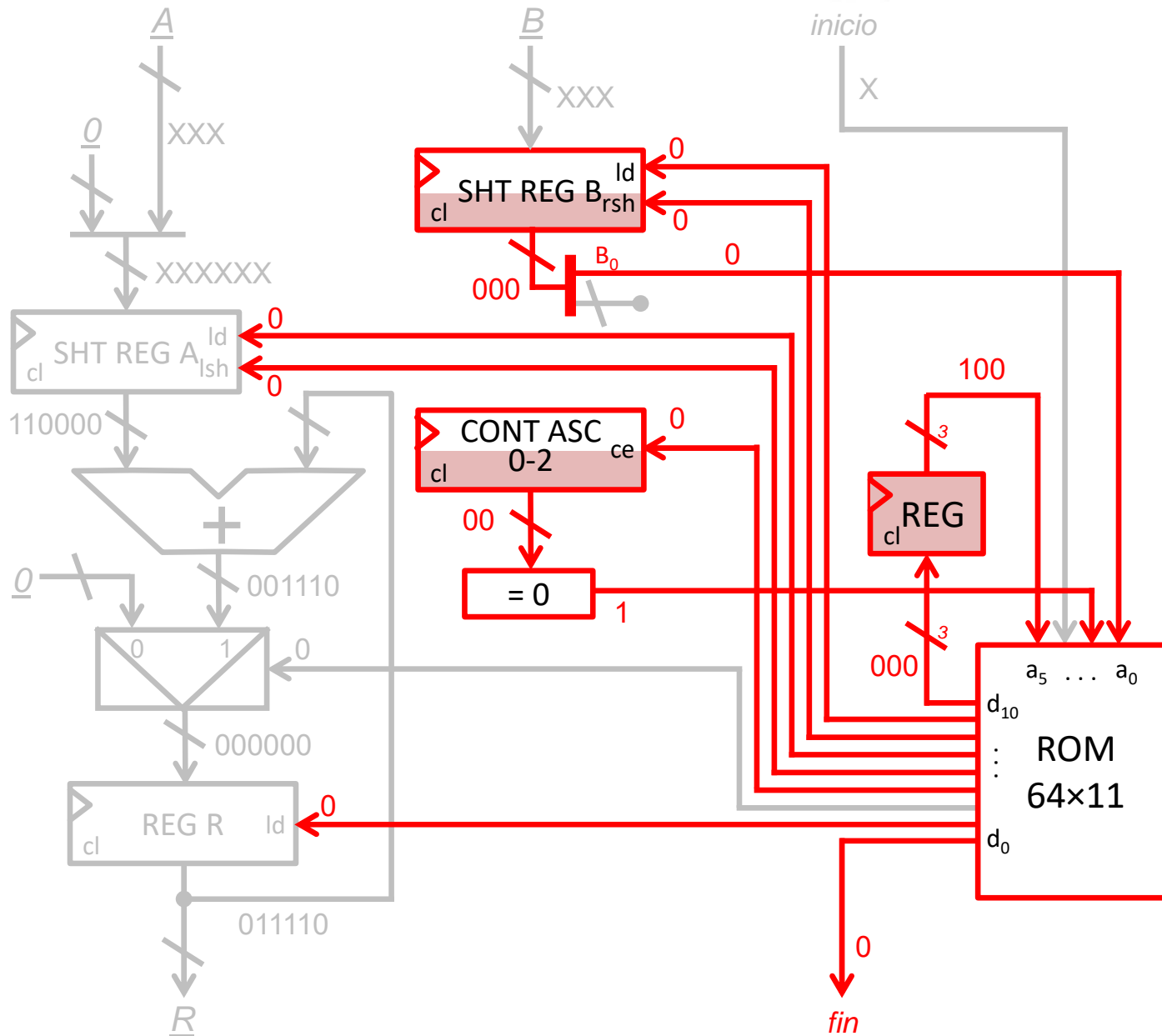
Simulación del sistema (operandos de 3 bits)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

76

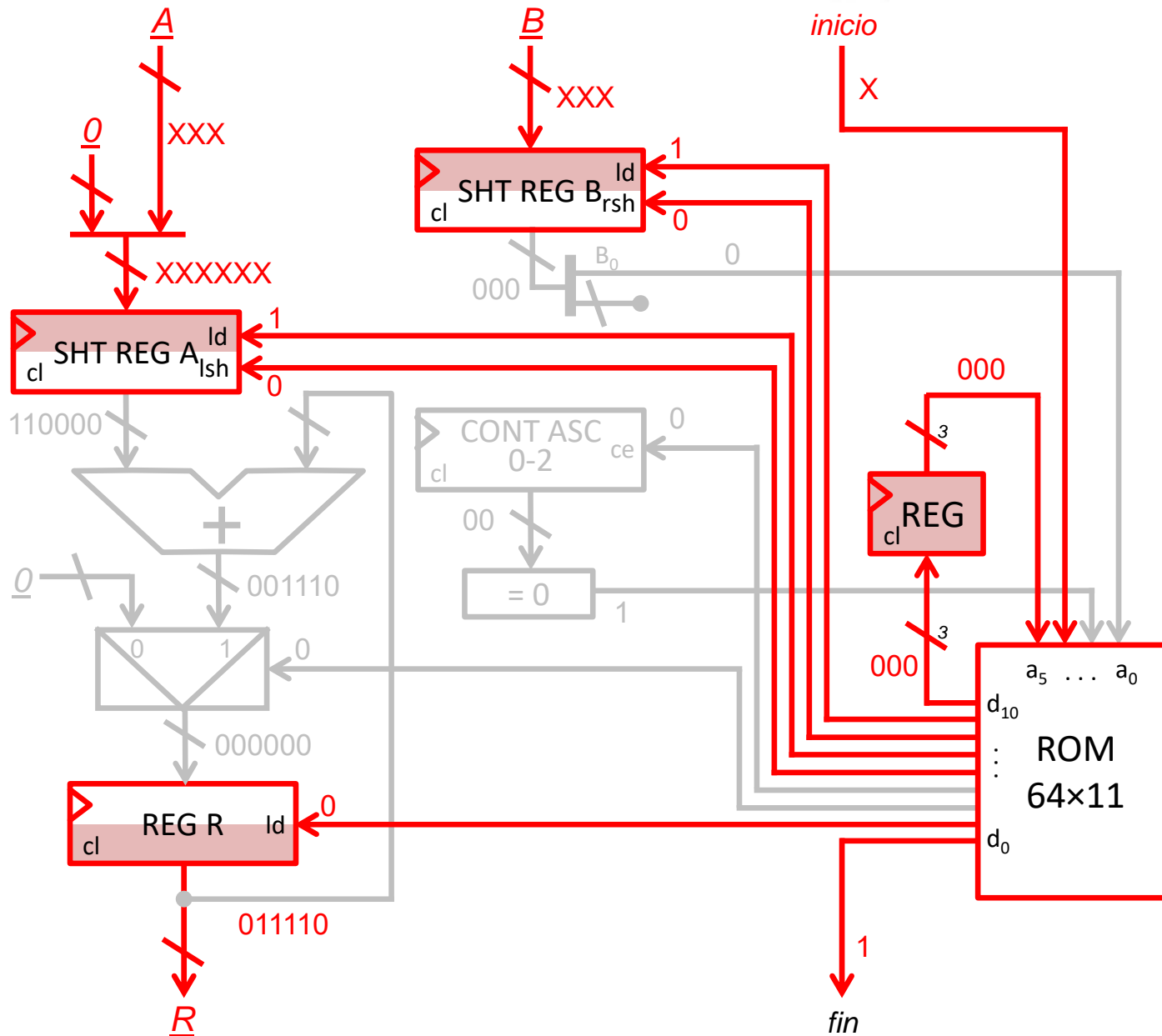


- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

Simulación del sistema (operandos de 3 bits)



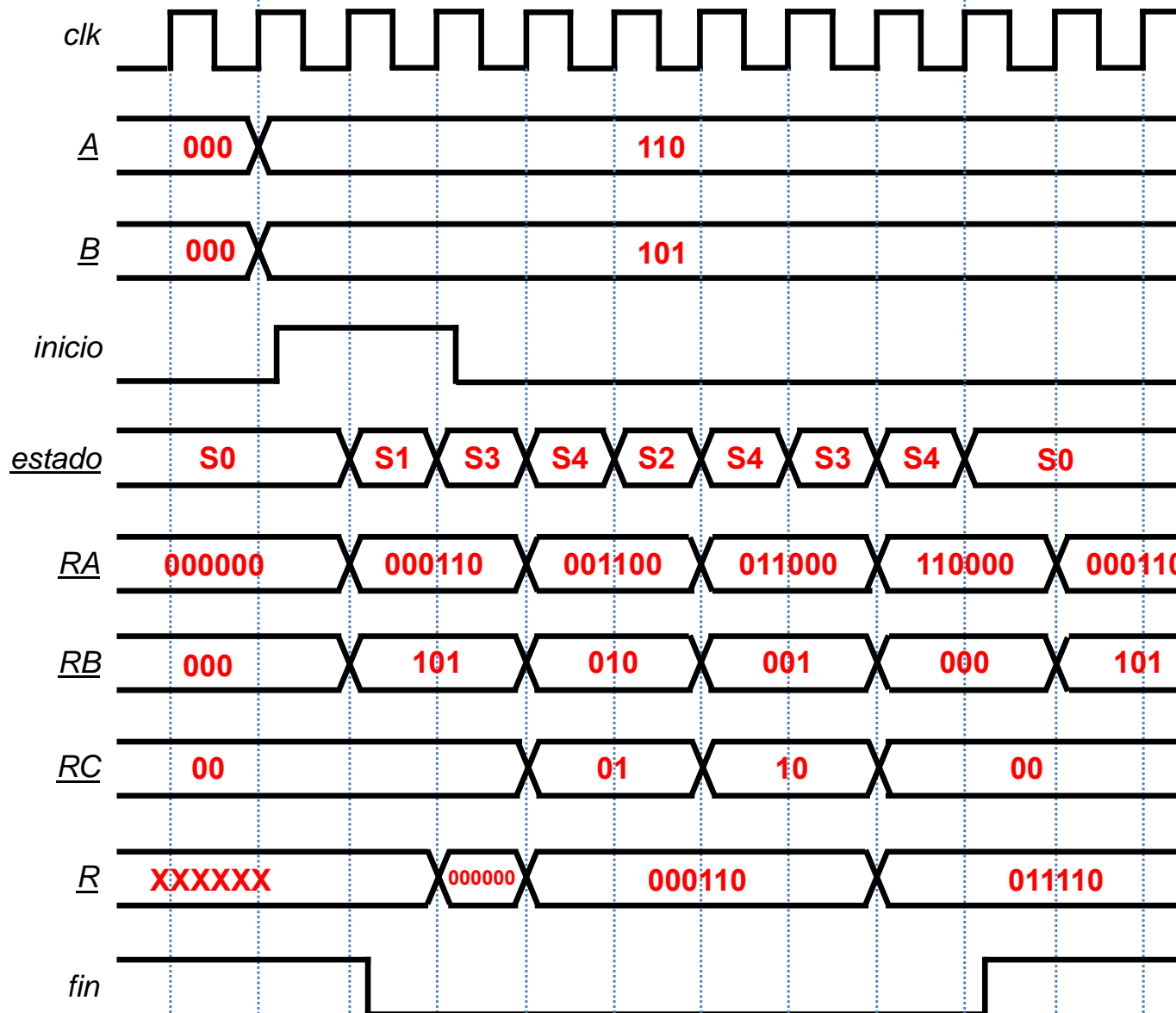
- S0** $RA \leftarrow A_{in}; RB \leftarrow B_{in};$
 $fin \leftarrow 1;$
si ($inicio=0$) ir a S0;
si ($inicio=1$) ir a S1;
- S1** $RR \leftarrow 0; fin \leftarrow 0;$
si ($B_0=0$) ir a S2;
si ($B_0=1$) ir a S3;
- S2** $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
ir a S4;
- S3** $RR \leftarrow RA + RR;$
 $RA \leftarrow RA \ll 1;$
 $RB \leftarrow RB \gg 1;$
 $RC \leftarrow RC + 1; fin \leftarrow 0;$
ir a S4;
- S4** $fin \leftarrow 0;$
si ($Cz=1$) ir a S0;
si ($Cz=0$ y $B_0=0$) ir a S2;
si ($Cz=0$ y $B_0=1$) ir a S3;



Multiplicador iterativo

Simulación del sistema (operandos de 3 bits)

← tiempo de cálculo: $2+3 \times 2 = 8$ ciclos →



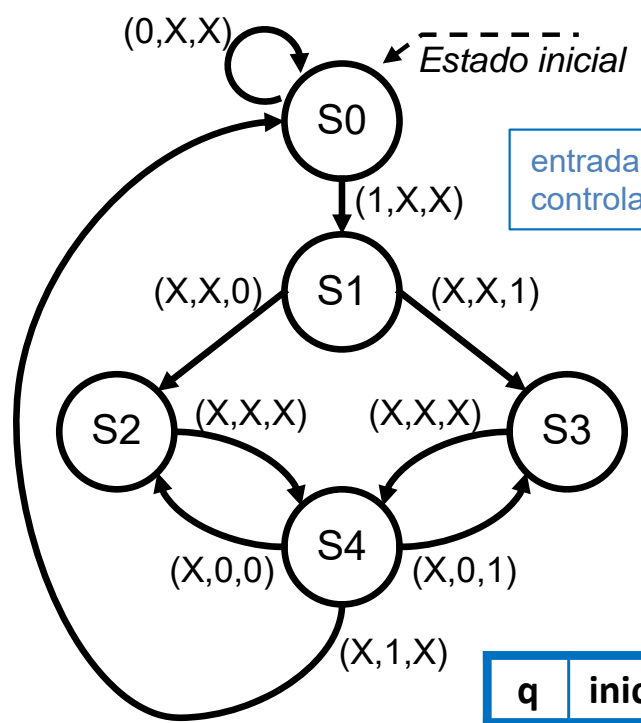
- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (B0=0) ir a S2;
si (B0=1) ir a S3;
- S2** RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S3** RR ← RA + RR;
RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;
- S4** fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;



Multiplicador iterativo

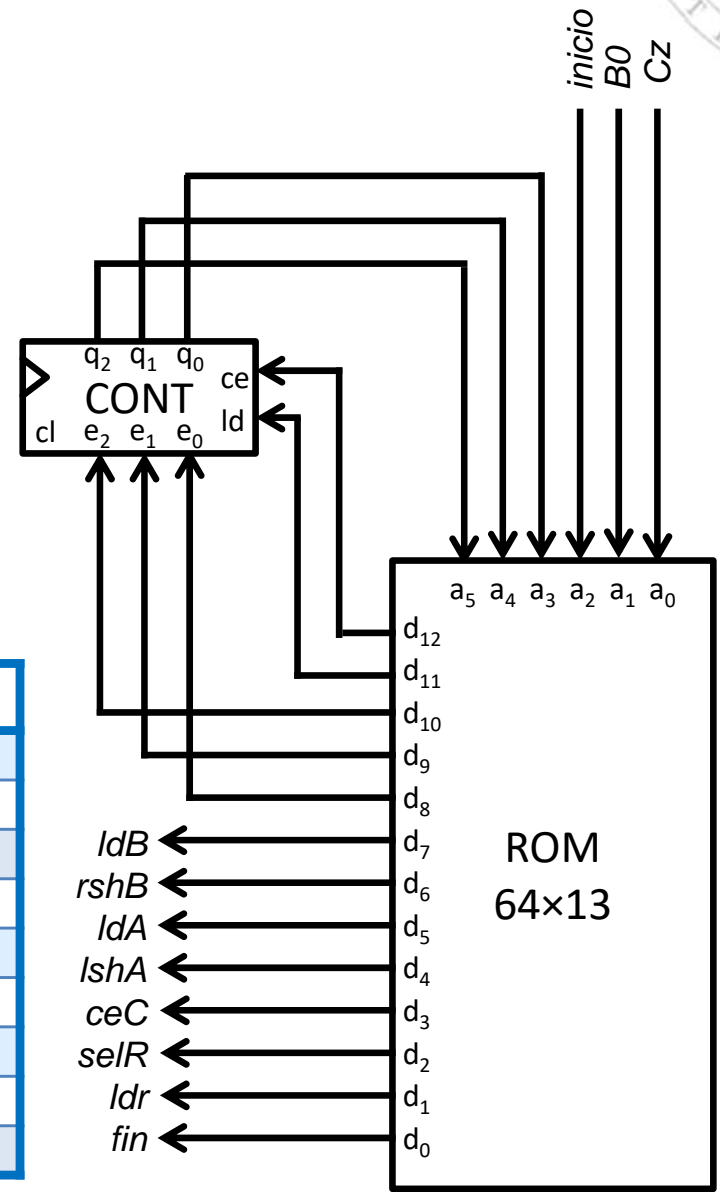
Otro posible controlador: ROM + contador

versión 14/07/23



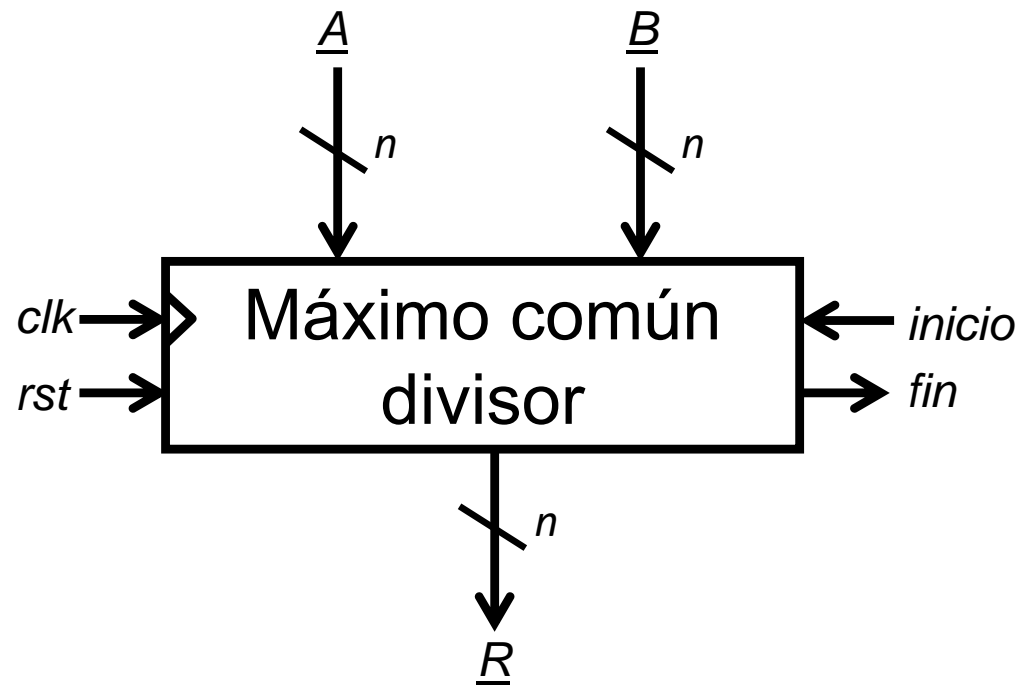
entradas del controlador: (inicio, Cz, B0)

q	inicio	Cz	B0	e	ld	ce
S0	0	X	X	-	0	0
S0	1	X	X	-	0	1
S1	X	X	0	-	0	1
S1	X	X	1	S3	1	0
S2	X	X	X	S4	1	0
S3	X	X	X	-	0	1
S4	X	0	0	S2	1	0
S4	X	0	1	S3	1	0
S4	X	1	X	S0	1	0





Máximo común divisor



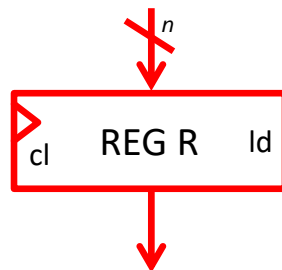
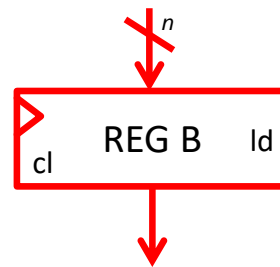
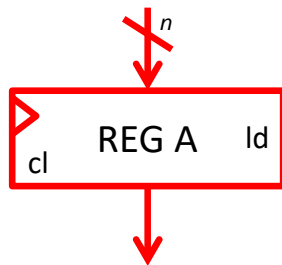
1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`if(A!=0 && B!=0)`
`{`
`while(A!=B)`
`if(A>B)`
`A = A-B;`
`else`
`B = B-A;`
`R = A;`
`};`
7. `Rout = R;`

algoritmo de Euclides

	A ₃	A ₂	A ₁	A ₀	B ₃	B ₂	B ₁	B ₀	R ₃	R ₂	R ₁	R ₀
1.	1	0	0	1	-	-	-	-	-	-	-	-
2.	1	0	0	1	0	1	1	0	-	-	-	-
3.	1	0	0	1	0	1	1	0	0	0	0	0
4.	0	0	1	1	0	1	1	0	0	0	0	0
5.	0	0	1	1	0	0	1	1	0	0	0	0
6.	0	0	1	1	0	0	1	1	0	0	1	1



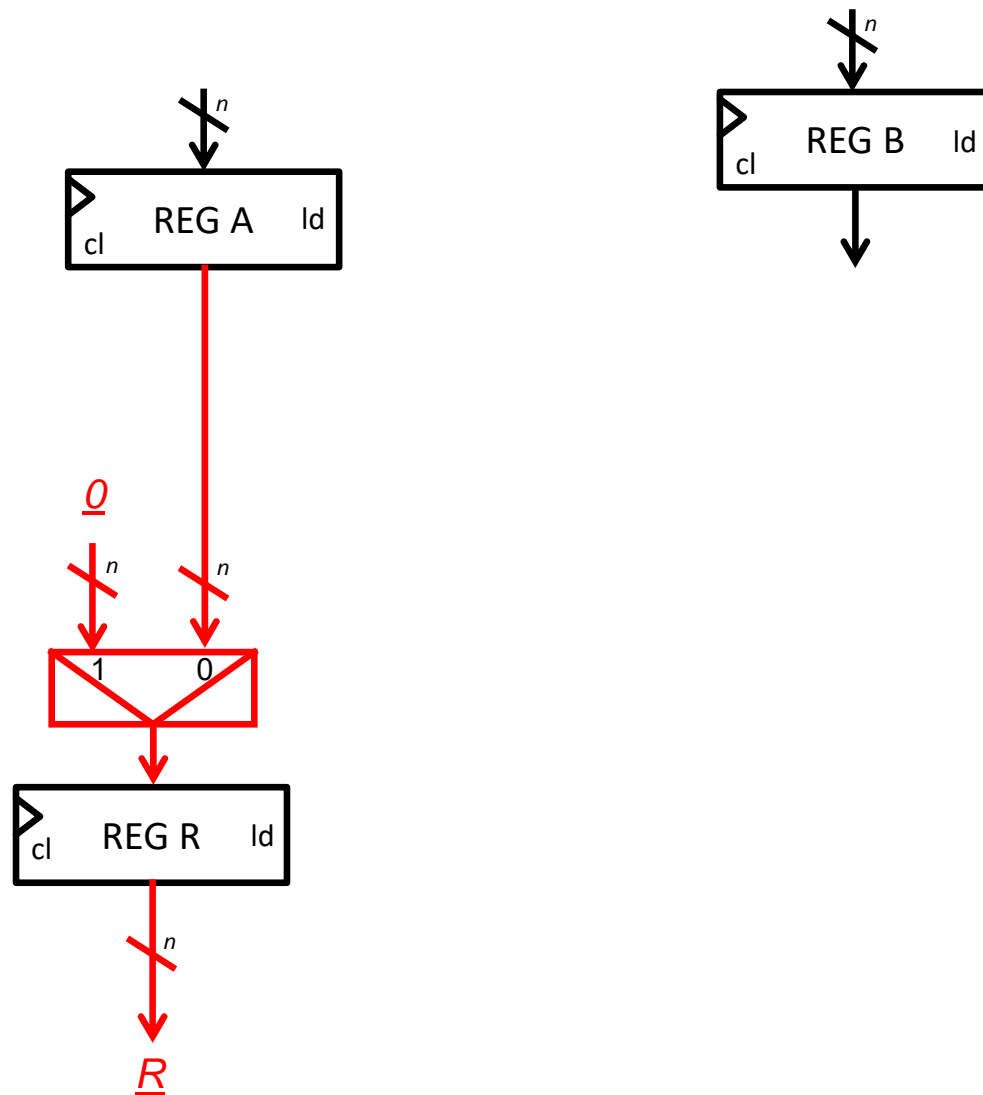
Máximo común divisor



1. **A** = Ain;
2. **B** = Bin;
3. **R** = 0;
`if(A!=0 && B!=0)`
{
 `while(A!=B)`
 `if(A>B)`
4. **A** = A-B;
 `else`
5. **B** = B-A;
6. **R** = A;
 };
7. **Rout** = R;



Máximo común divisor

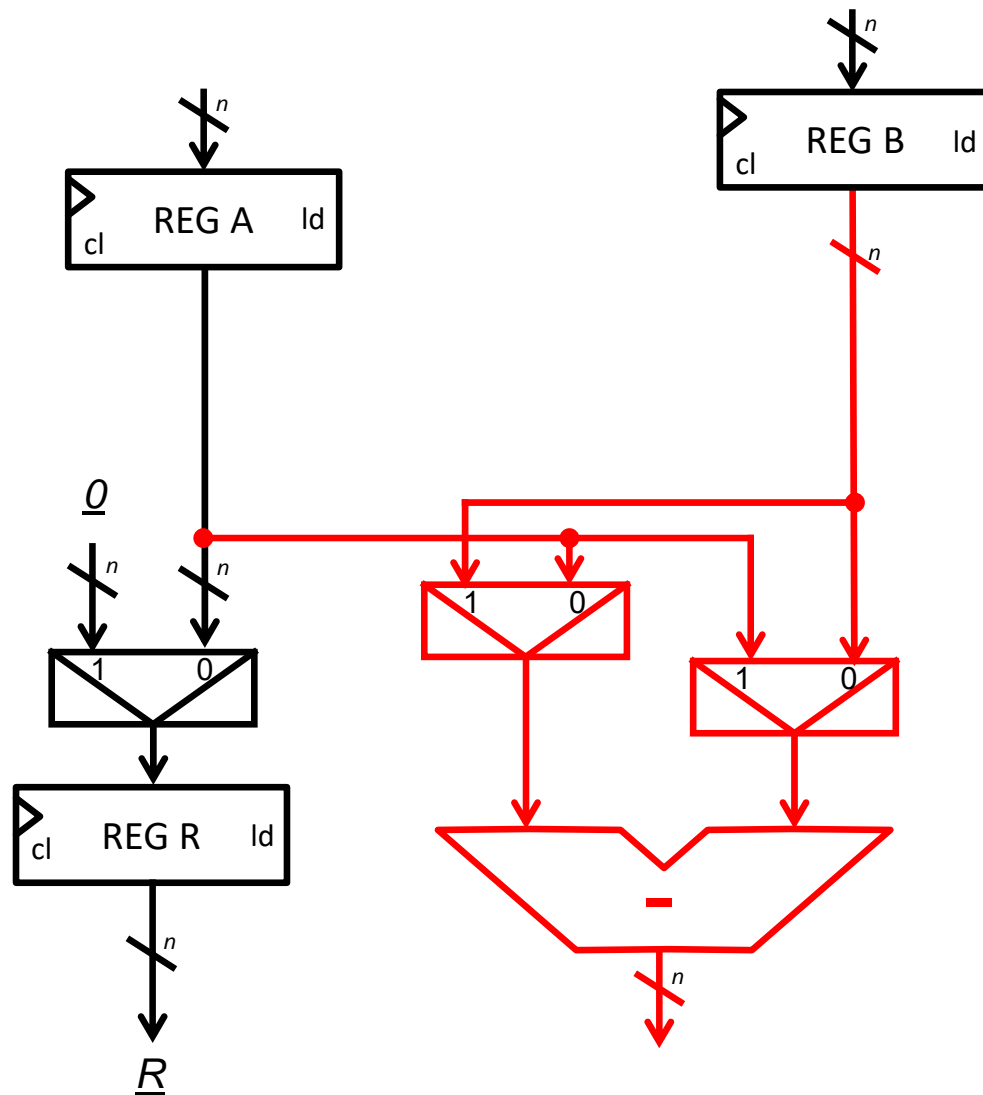


1. `A = Ain;`
2. `B = Bin;`
3. `R = 0;`
`if(A!=0 && B!=0)`
{
 `while(A!=B)`
 `if(A>B)`
 `A = A-B;`
 `else`
 `B = B-A;`
 `R = A;`
};
4. `Rout = R;`
- 5.
- 6.
- 7.



Máximo común divisor

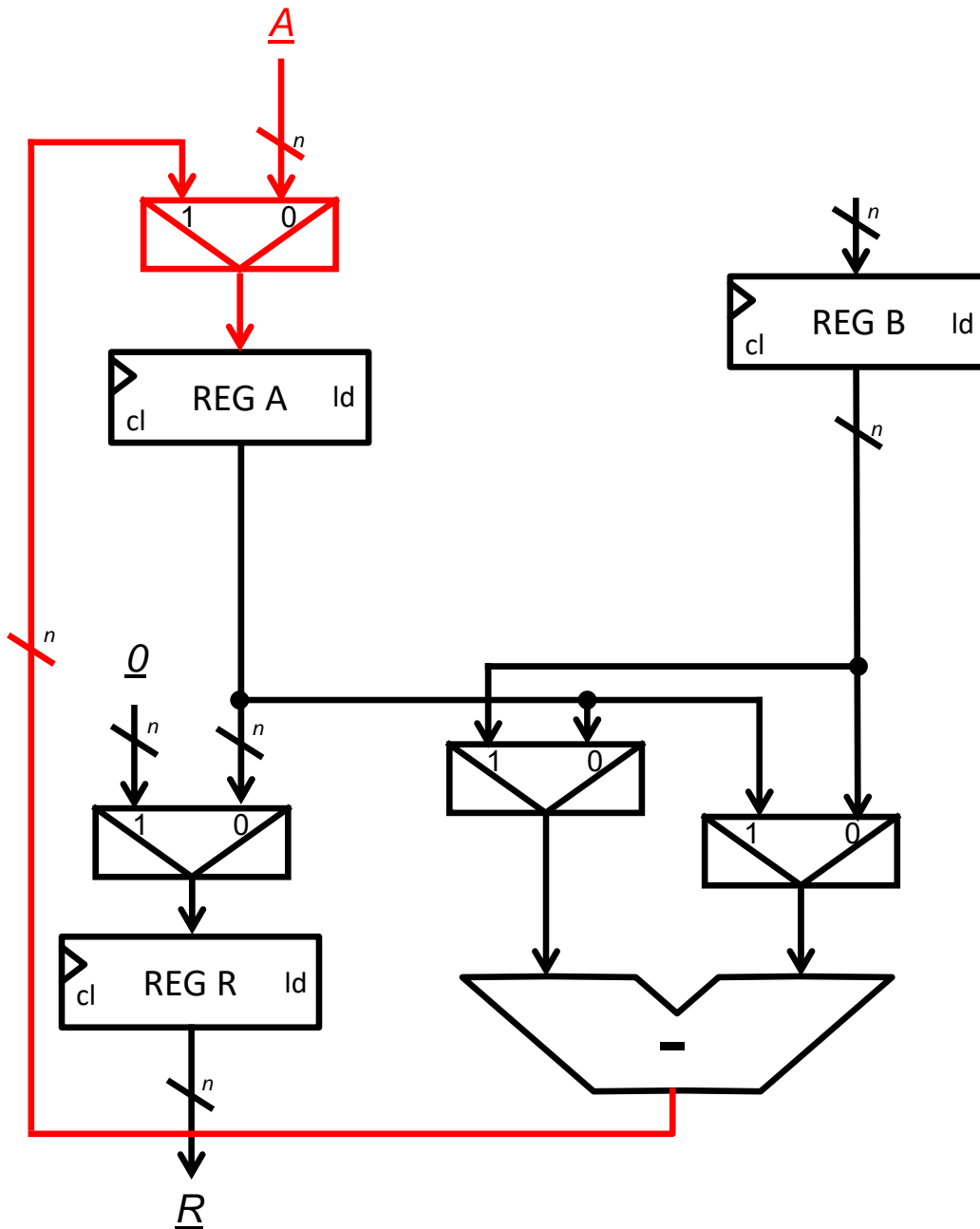
Suma/resta parcial



1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
 $\text{if}(A \neq 0 \ \&\& \ B \neq 0)$
{
 $\text{while}(A \neq B)$
 $\text{if}(A > B)$
 $A = A - B;$
 else
 $B = B - A;$
 $R = A;$
 };
4. $A = A - B;$
5. $B = B - A;$
6. $R = A;$
7. $R_{out} = R;$



Máximo común divisor



1. **A = Ain;**
2. **B = Bin;**
3. **R = 0;**
if(A!=0 && B!=0)
{
 while(A!=B)
 if(A>B)
 A = A-B;
 else
 B = B-A;
 R = A;
 };
7. **Rout = R;**



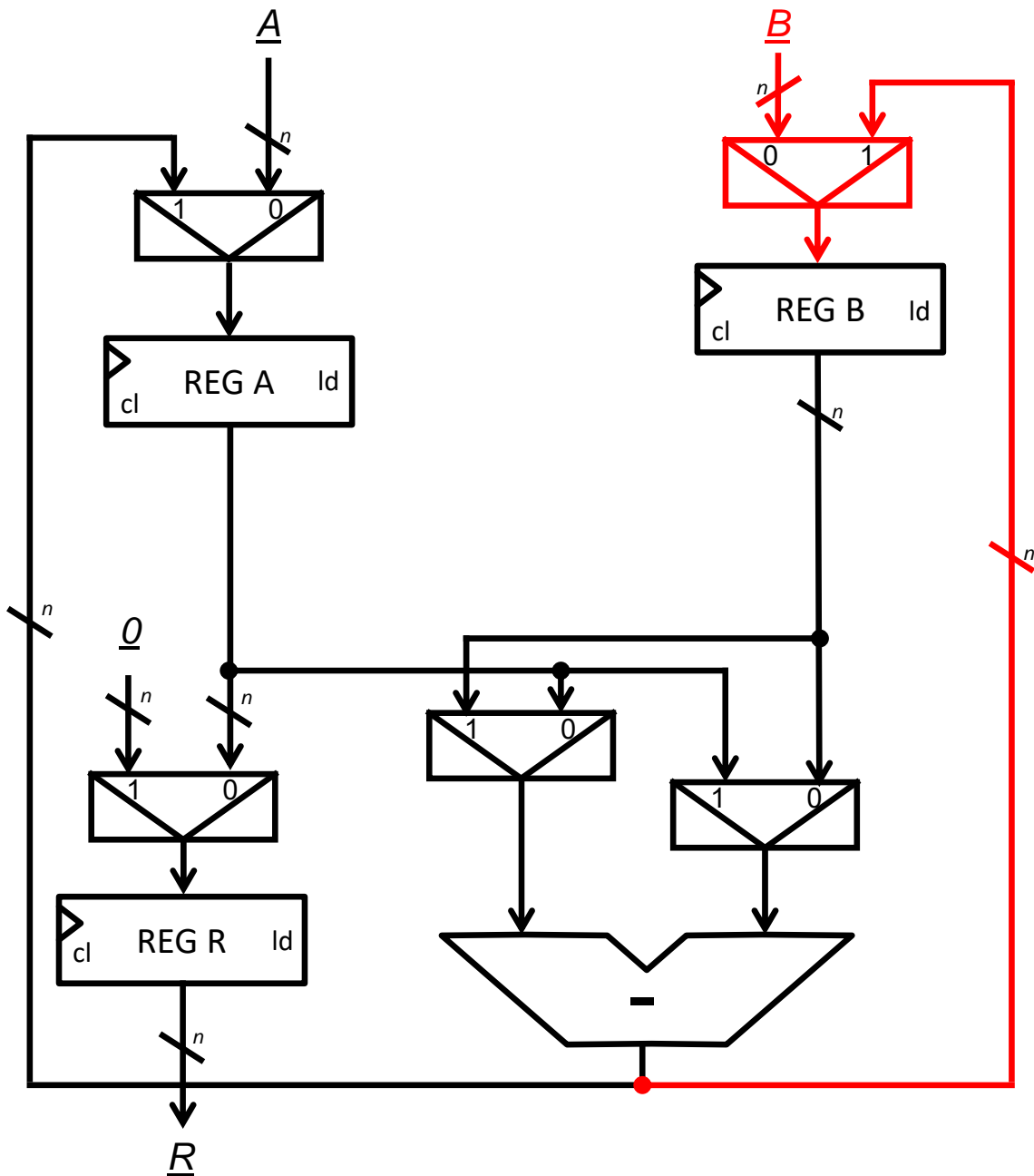
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

85



1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
 $\text{if}(A \neq 0 \ \&\& \ B \neq 0)$
{
 $\text{while}(A \neq B)$
 $\text{if}(A > B)$
 $A = A - B;$
 else
 $B = B - A;$
 $R = A;$
 };
4. $R_{out} = R;$
- 5.
- 6.
- 7.



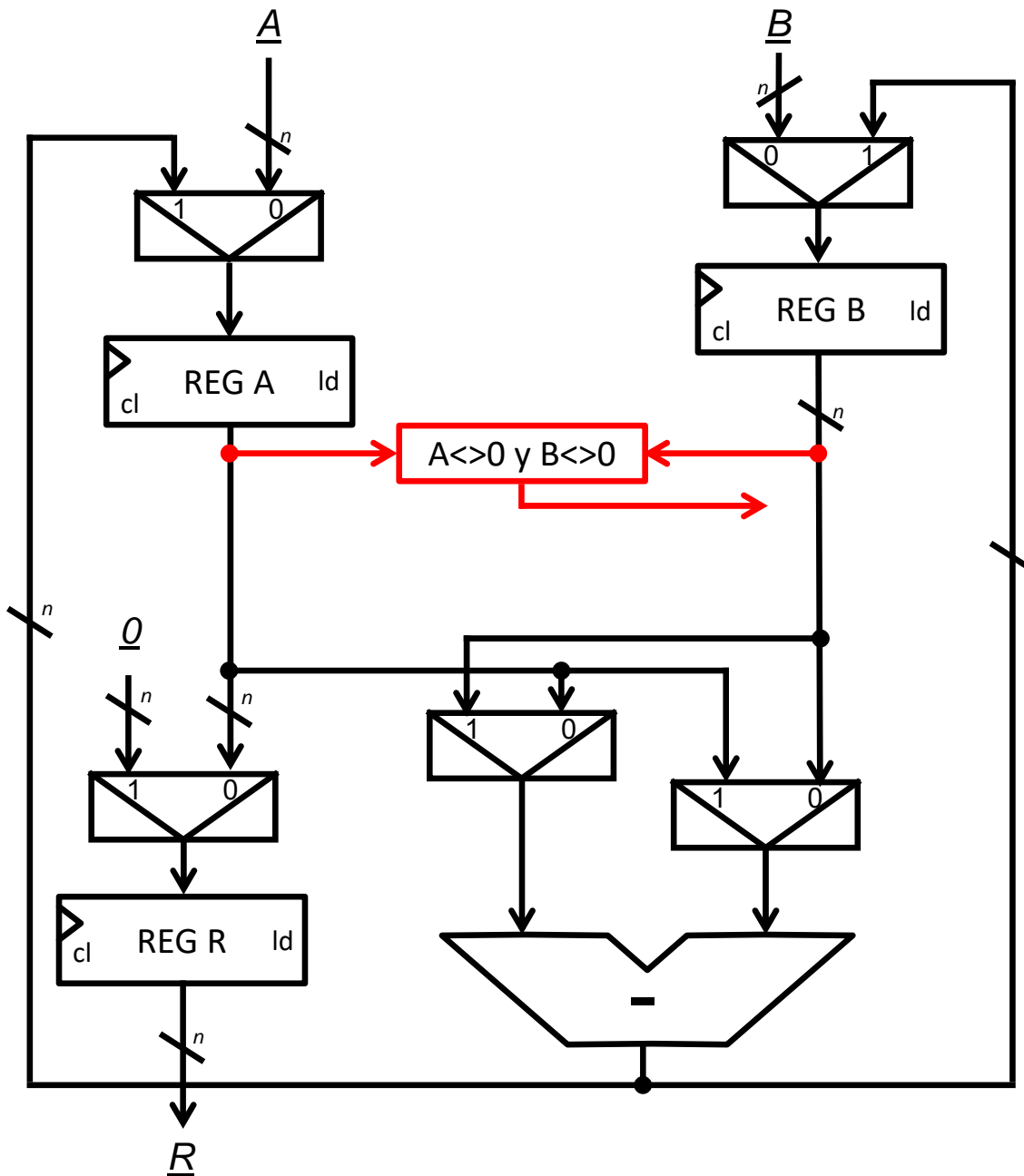
Máximo común divisor

versión 14/07/23

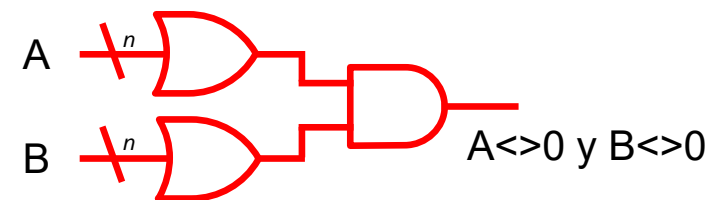
tema 8:
Rutas de datos y controladores

FC-1

86

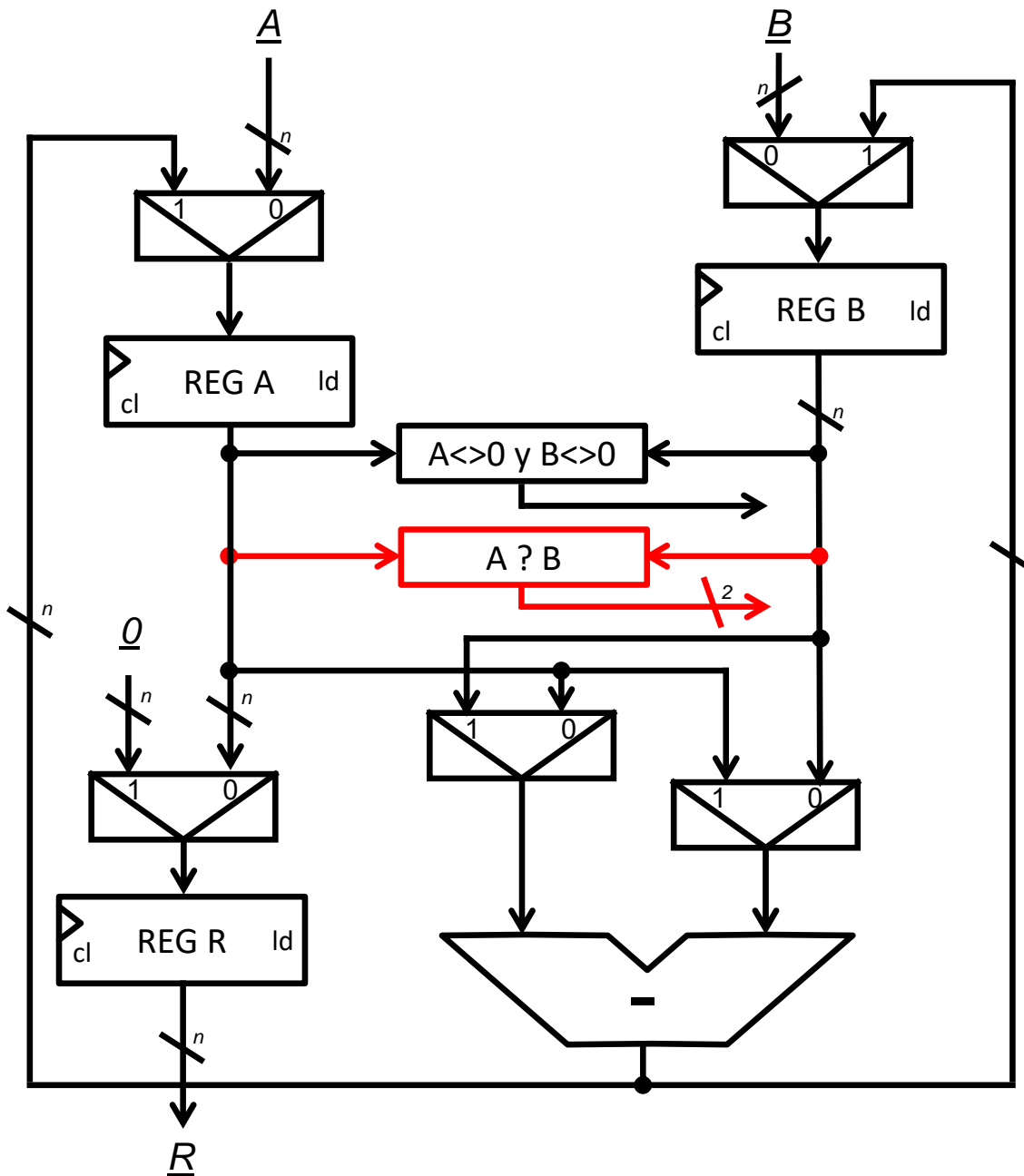


1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
 $\text{if}(A \neq 0 \ \&\& \ B \neq 0)$
 $\{$
 $\quad \text{while}(A \neq B)$
 $\quad \quad \text{if}(A > B)$
 $\quad \quad \quad A = A - B;$
 $\quad \quad \text{else}$
 $\quad \quad \quad B = B - A;$
 $\quad \quad R = A;$
 $\quad \quad \};$
7. $R_{out} = R;$





Máximo común divisor



1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
`if($A \neq 0 \ \&\& \ B \neq 0$)`
`{`
`while($A \neq B$)`
`if($A > B$)`
`$A = A - B;$`
`else`
`$B = B - A;$`
`$R = A;$`
`};`
7. $R_{out} = R;$

	z
$A=B$	00
$A>B$	10
$A<B$	01



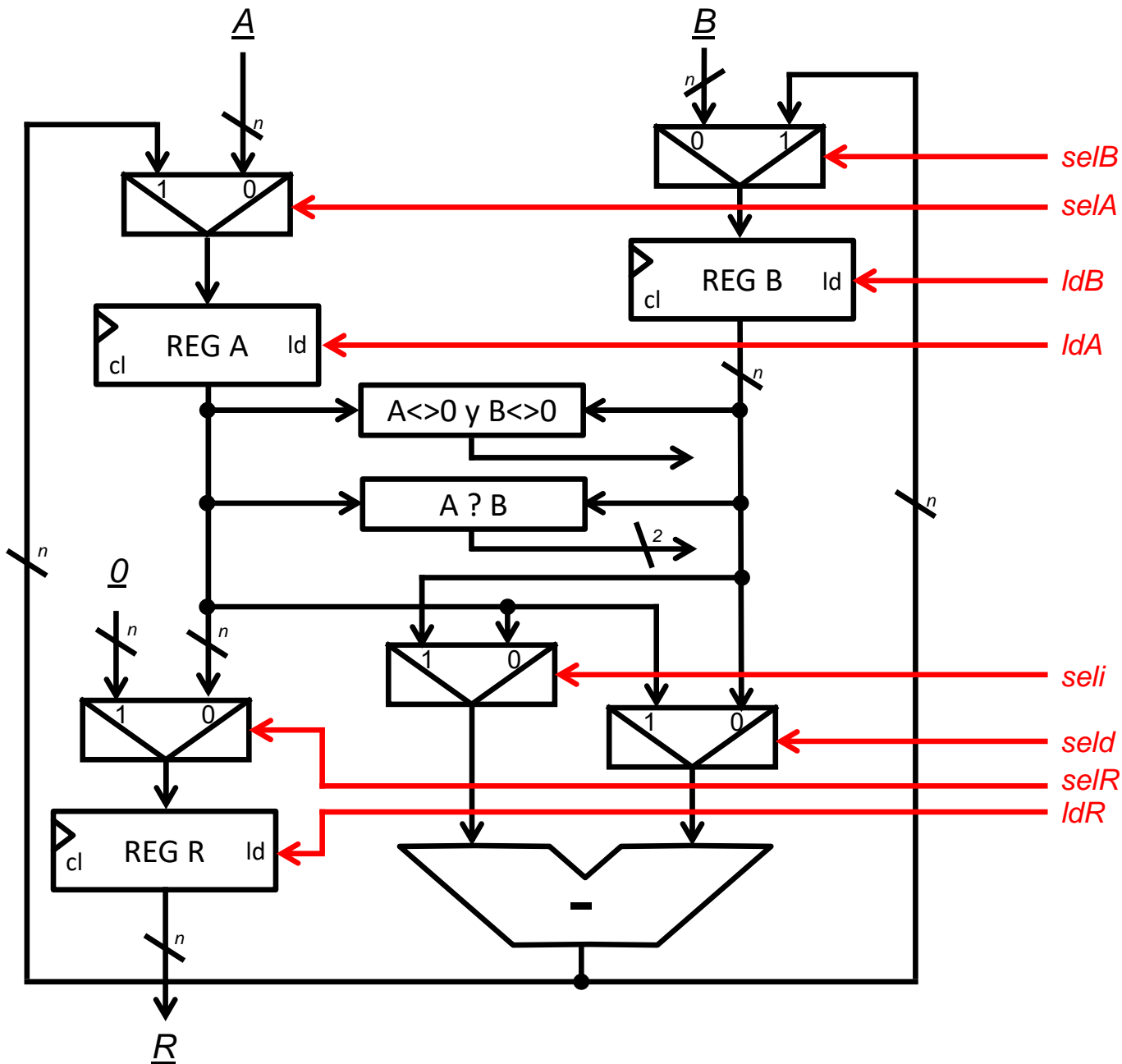
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

88





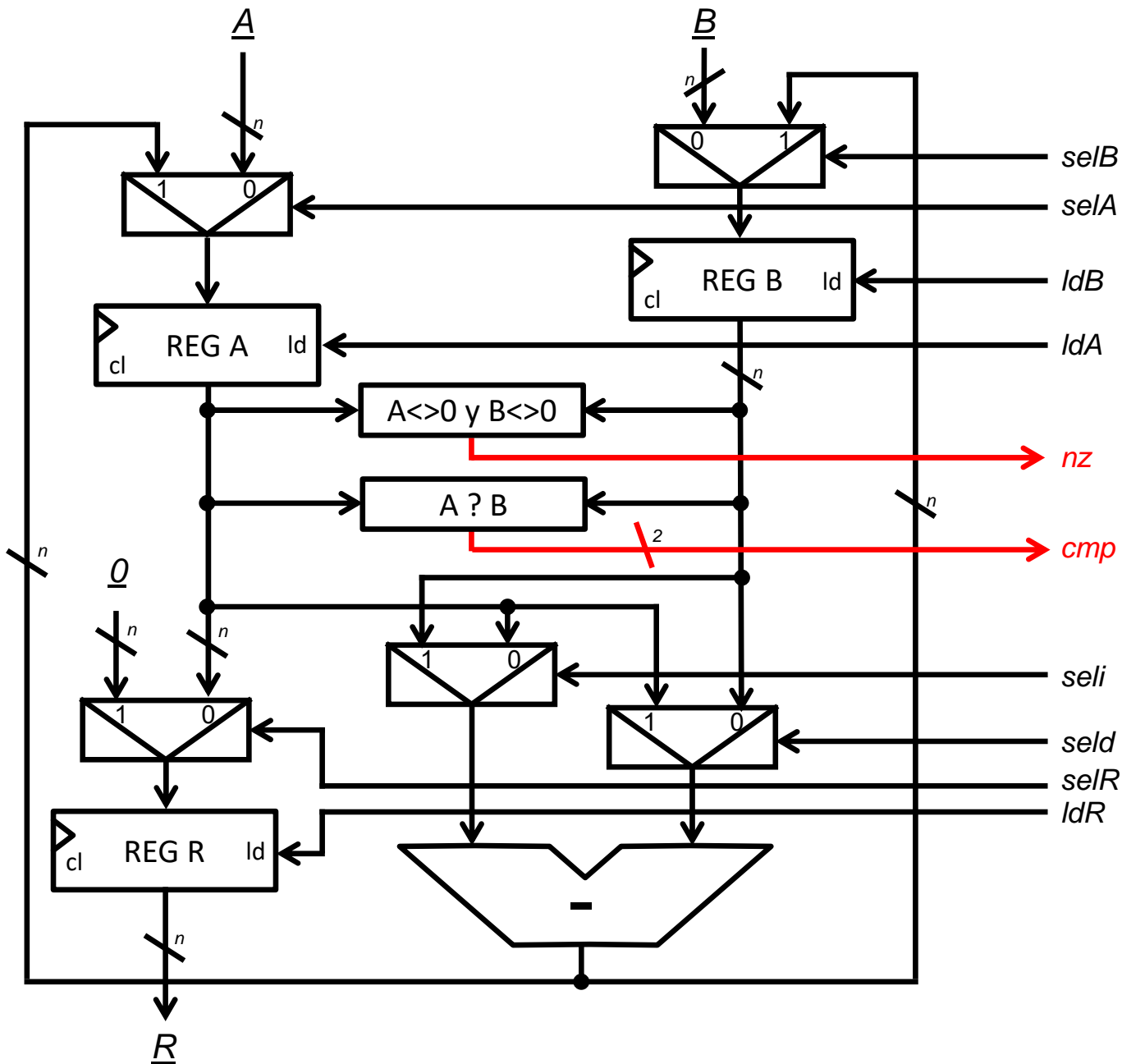
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

89





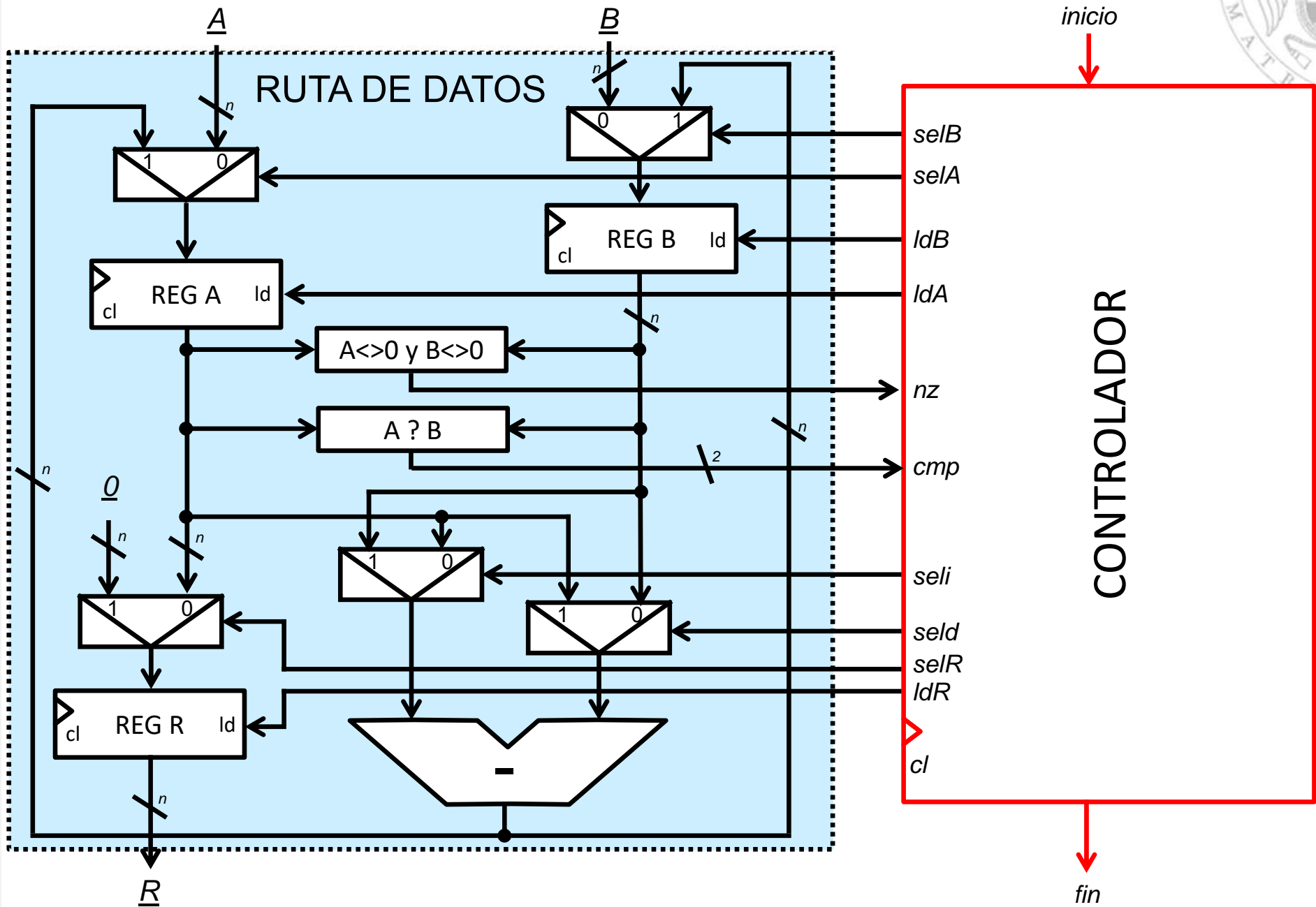
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

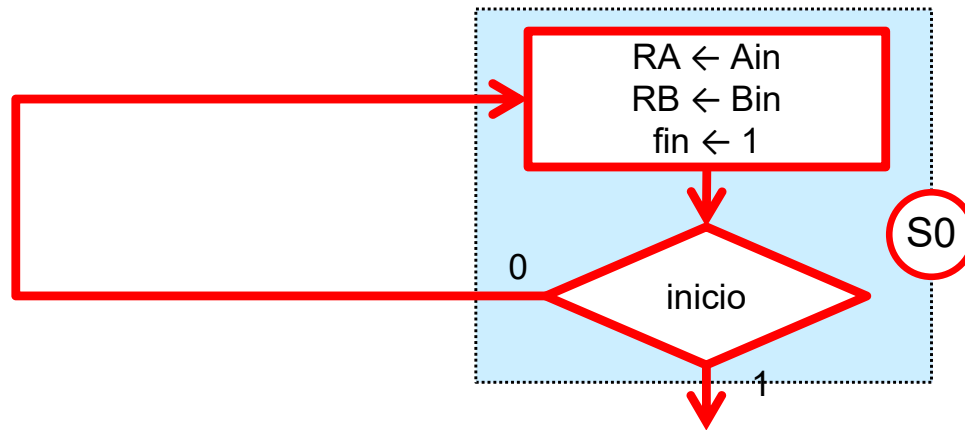
FC-1

90





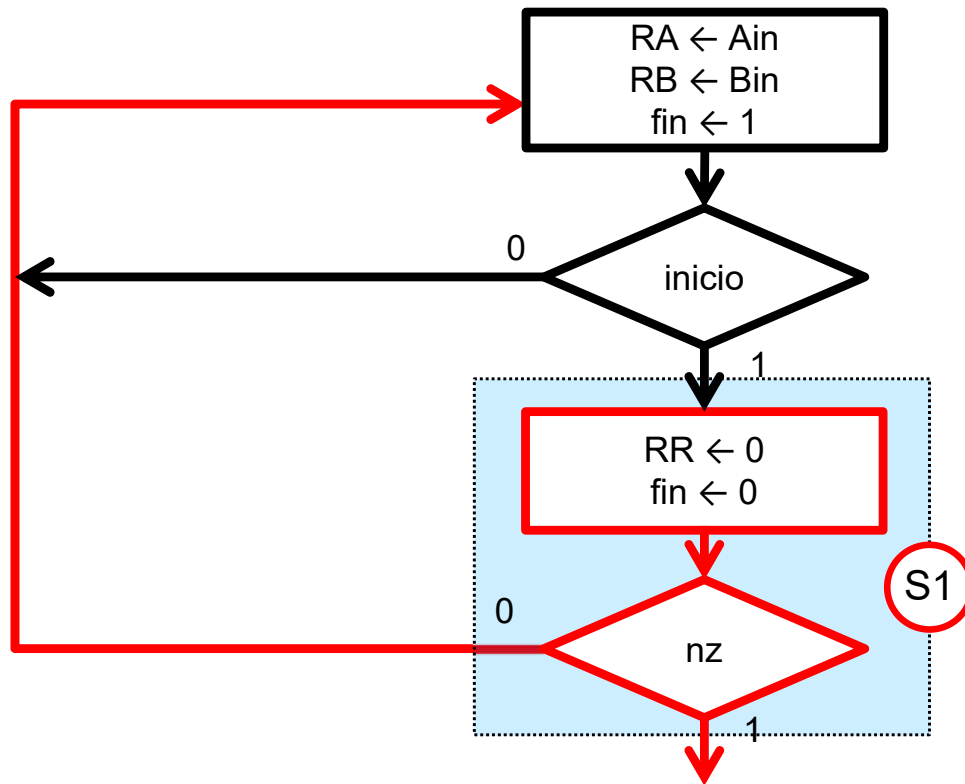
Máximo común divisor



1. **A = Ain;**
2. **B = Bin;**
3. **R = 0;**
if(A!=0 && B!=0)
{
 while(A!=B)
 if(A>B)
 A = A-B;
 else
 B = B-A;
 R = A;
};
4. **A = A-B;**
5. **B = B-A;**
6. **R = A;**
7. **Rout = R;**



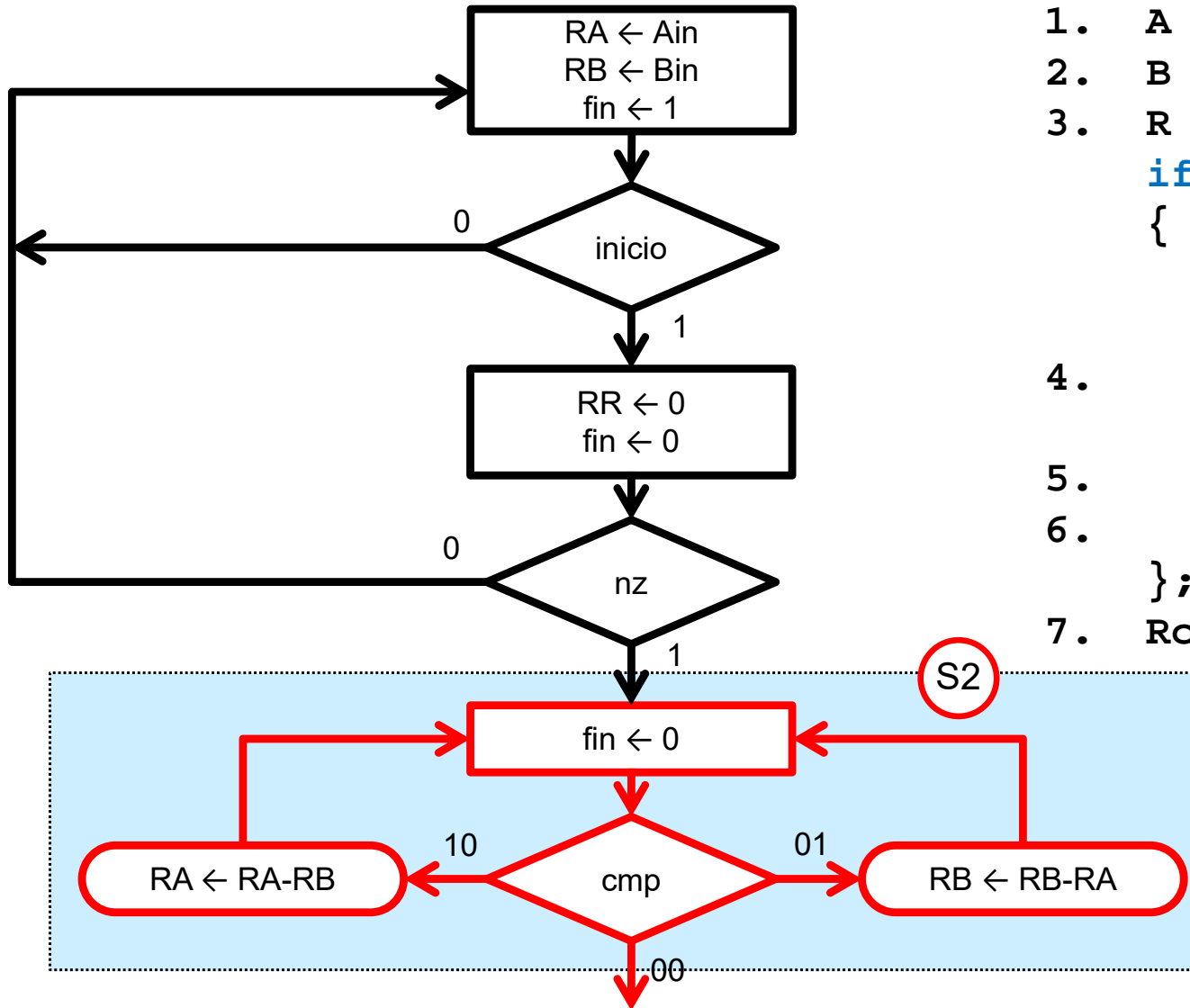
Máximo común divisor



1. $A = A_{in};$
2. $B = B_{in};$
3. $R = 0;$
 $\text{if}(A \neq 0 \ \&\& \ B \neq 0)$
{
 $\text{while}(A \neq B)$
 $\text{if}(A > B)$
 $A = A - B;$
 else
 $B = B - A;$
 $R = A;$
 };
4. $A = A - B;$
5. $B = B - A;$
6. $R = A;$
7. $R_{out} = R;$



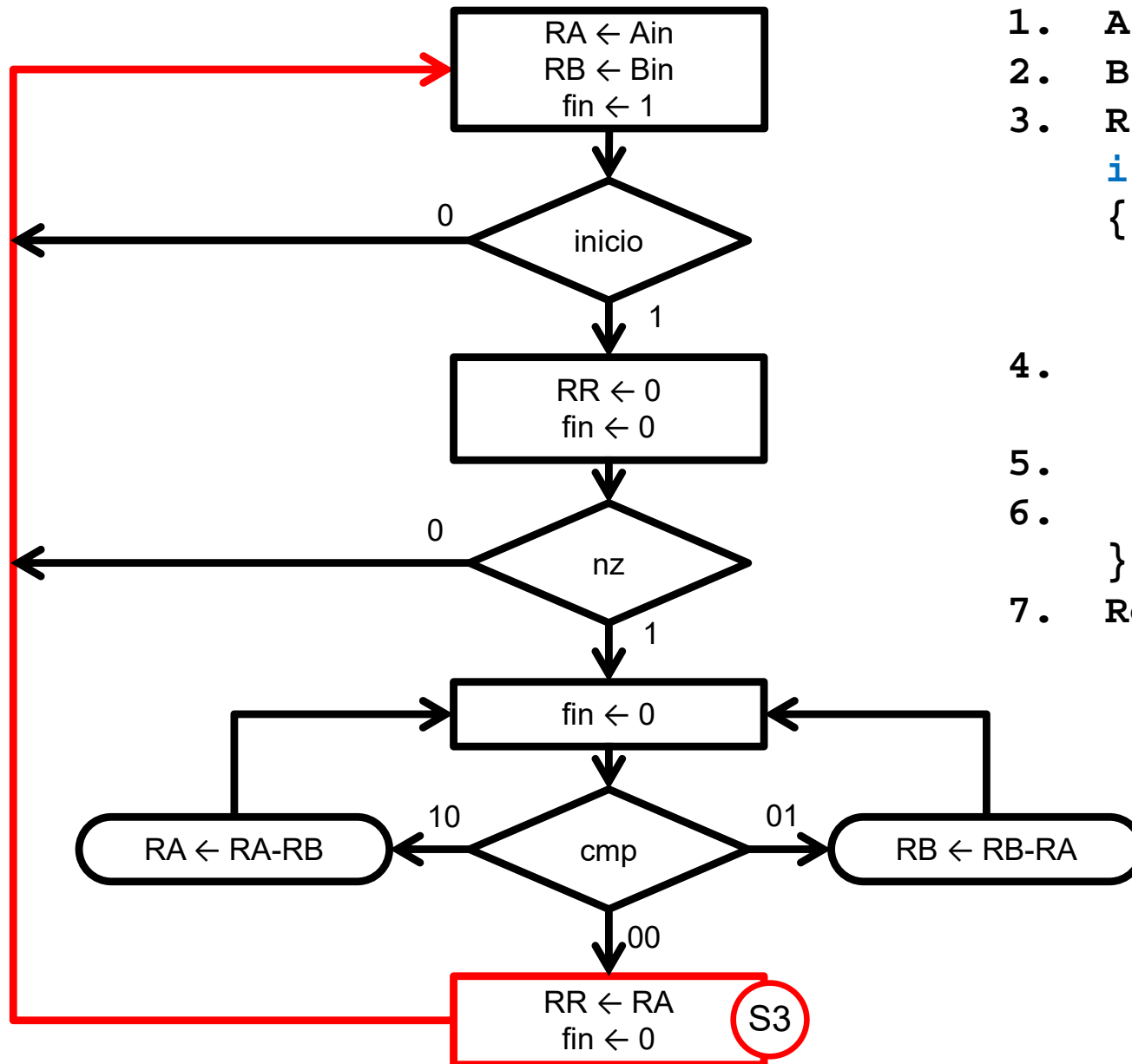
Máximo común divisor



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
   if( A!=0 && B!=0 )
   {
       while( A!=B )
           if( A>B )
               A = A-B;
           else
               B = B-A;
       R = A;
   };
7.  Rout = R;
```



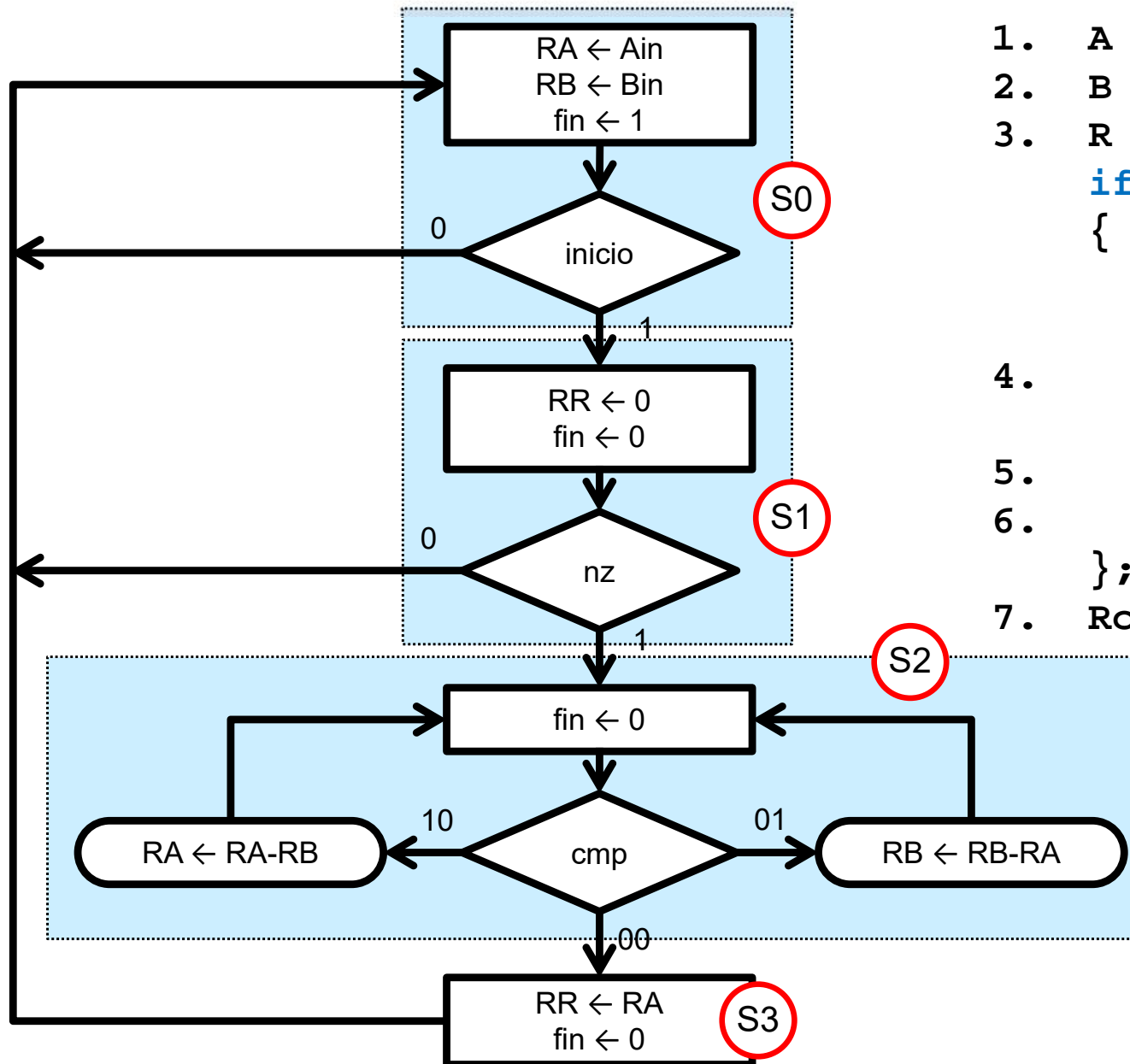
Máximo común divisor



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
   if( A!=0 && B!=0 )
   {
       while( A!=B )
           if( A>B )
               A = A-B;
           else
               B = B-A;
           R = A;
   };
7.  Rout = R;
```



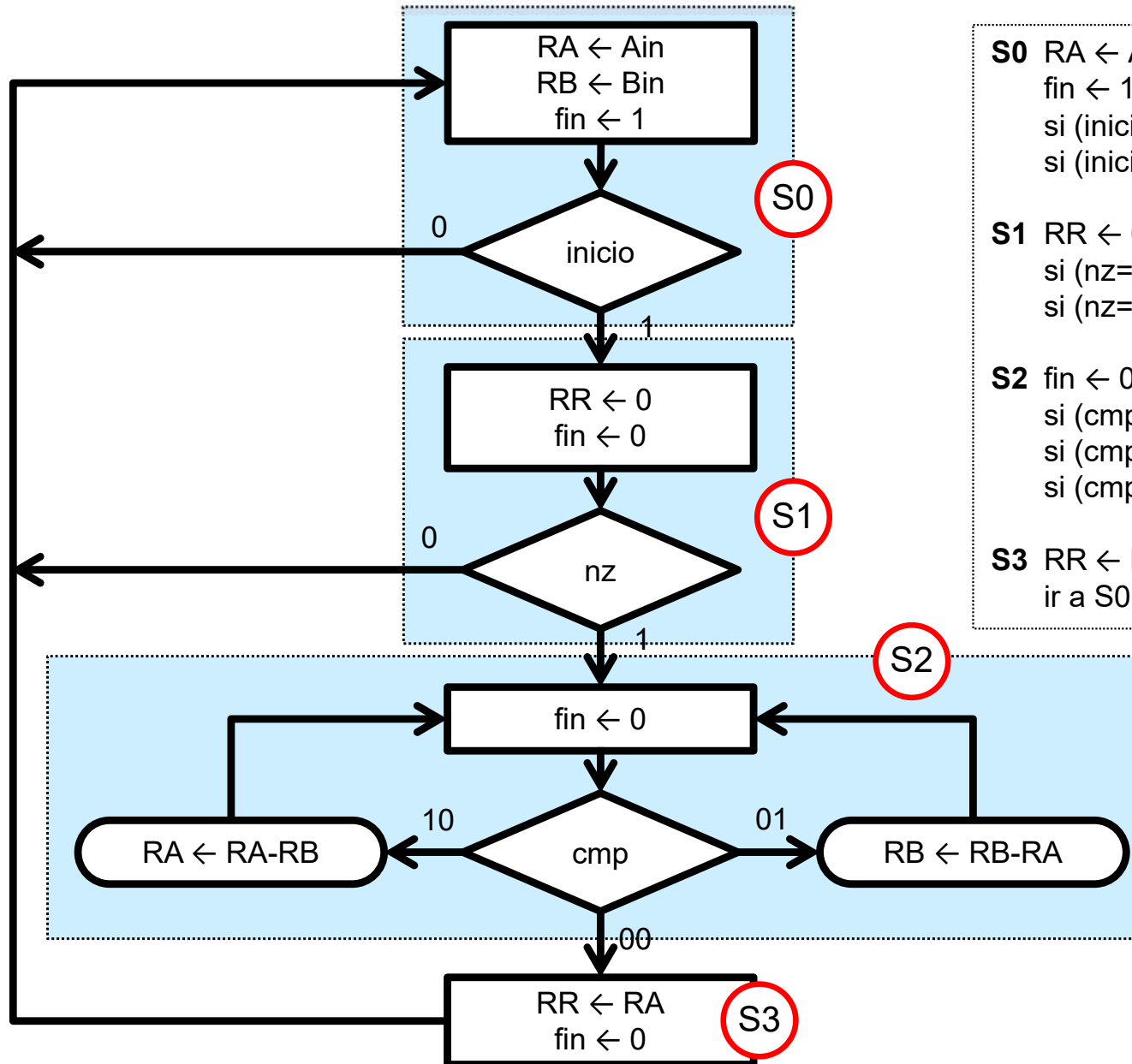
Máximo común divisor



1. $A = \text{Ain};$
2. $B = \text{Bin};$
3. $R = 0;$
`if($A \neq 0 \ \&\& \ B \neq 0$)`
{
 `while($A \neq B$)`
 `if($A > B$)`
 $A = A - B;$
 `else`
 $B = B - A;$
 $R = A;$
};
- 4.
- 5.
6. $R = A;$
7. $\text{Rout} = R;$



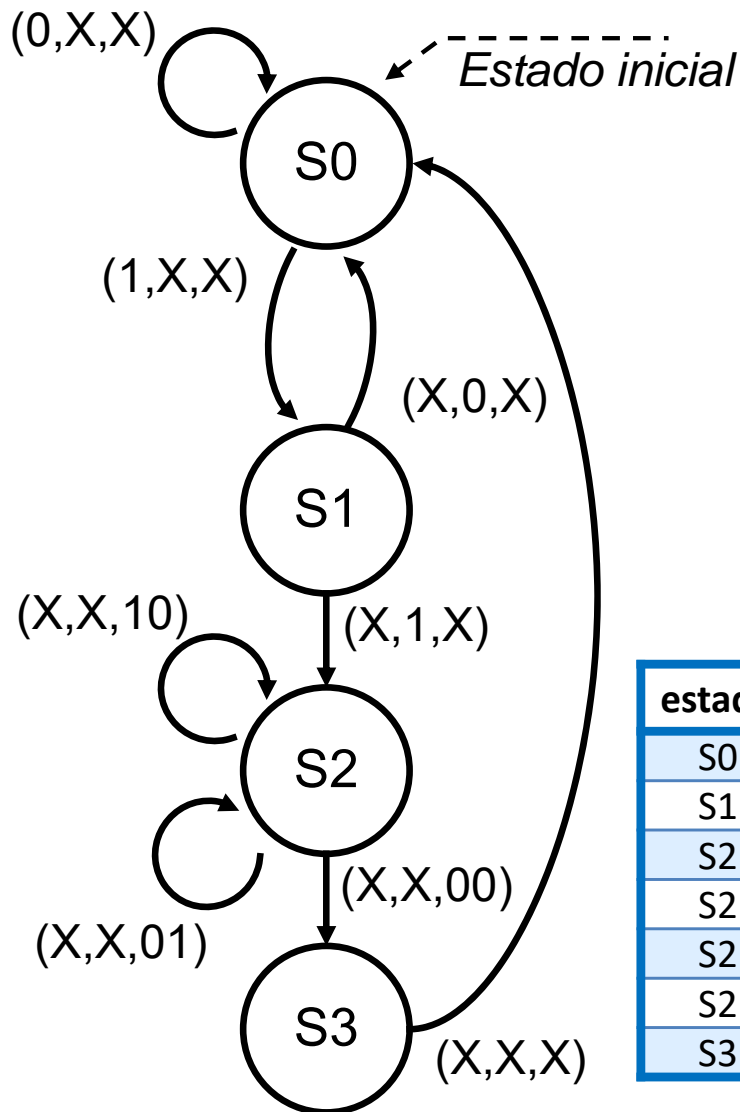
Máximo común divisor



- S0** RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;
- S1** RR ← 0; fin ← 0;
si (nz=0) ir a S0;
si (nz=1) ir a S2;
- S2** fin ← 0;
si (cmp=00) ir a S3;
si (cmp=01) RB ← RB-RA, ir a S2;
si (cmp=10) RA ← RA-RB, ir a S2;
- S3** RR ← RA; fin ← 0;
ir a S0;



Máximo común divisor



estado	inicio	nz	cmp ₁	cmp ₀	estado'
S0	0	X	X	X	S0
S0	1	X	X	X	S1
S1	X	0	X	X	S0
S1	X	1	X	X	S2
S2	X	X	0	0	S3
S2	X	X	0	1	S2
S2	X	X	1	0	S2
S2	X	X	1	1	-
S3	X	X	X	X	S0

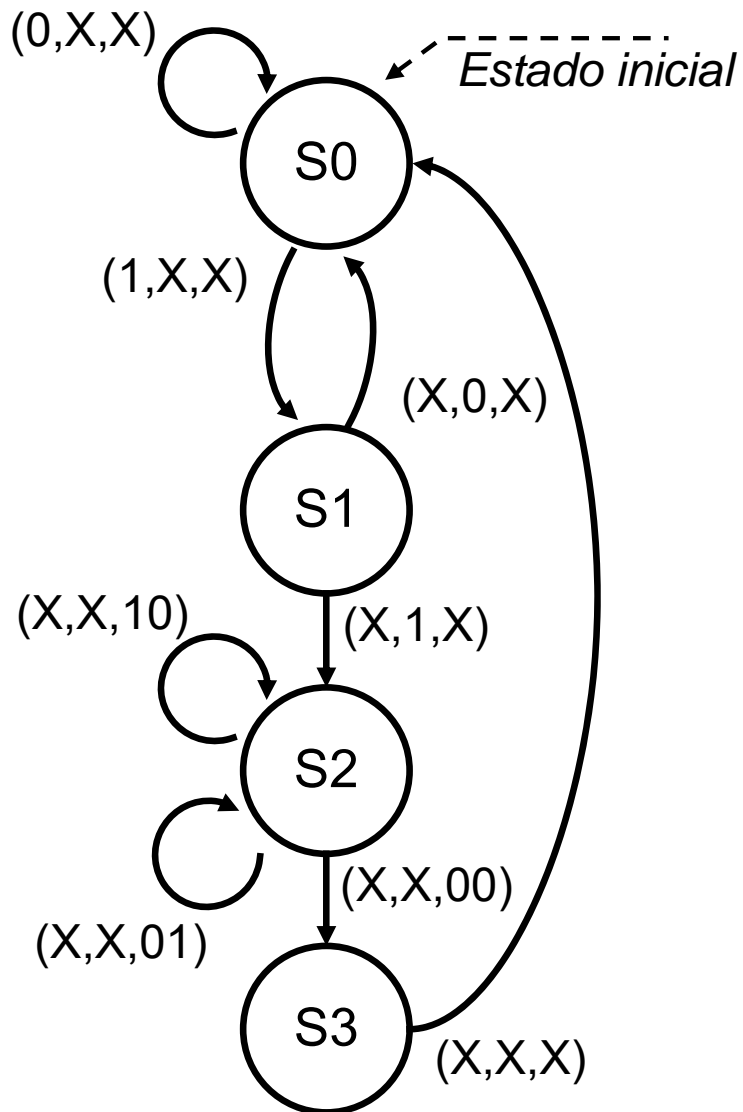
estado	cmp ₁	cmp ₀	selA	selB	ldA	ldB	seli	seld	selR	ldR	fin
S0	X	X	0	0	1	1	-	-	-	0	1
S1	X	X	-	-	0	0	-	-	1	1	0
S2	0	0	-	-	0	0	-	-	-	0	0
S2	0	1	-	1	0	1	1	1	-	0	0
S2	1	0	1	-	1	0	0	0	-	0	0
S2	1	1	-	-	-	-	-	-	-	-	-
S3	X	X	0	0	0	0	-	-	0	1	0



entradas del controlador: (inicio, nz, cmp)



Máximo común divisor



entradas del controlador: (inicio, nz, cmp)

estado	inicio	nz	cmp ₁	cmp ₀	estado'
S0	0	X	X	X	S0
S0	1	X	X	X	S1
S1	X	0	X	X	S0
S1	X	1	X	X	S2
S2	X	X	0	0	S3
S2	X	X	0	1	S2
S2	X	X	1	0	S2
S2	X	X	1	1	-
S3	X	X	X	X	S0

estado	cmp ₁	cmp ₀	ldA	ldB seli seld	selA selB selR	ldR	fin
S0	X	X	1	1	0	0	1
S1	X	X	0	0	1	1	0
S2	0	0	0	0	-	0	0
S2	0	1	0	1	1	0	0
S2	1	0	1	0	1	0	0
S2	1	1	-	-	-	-	-
S3	X	X	0	0	0	1	0



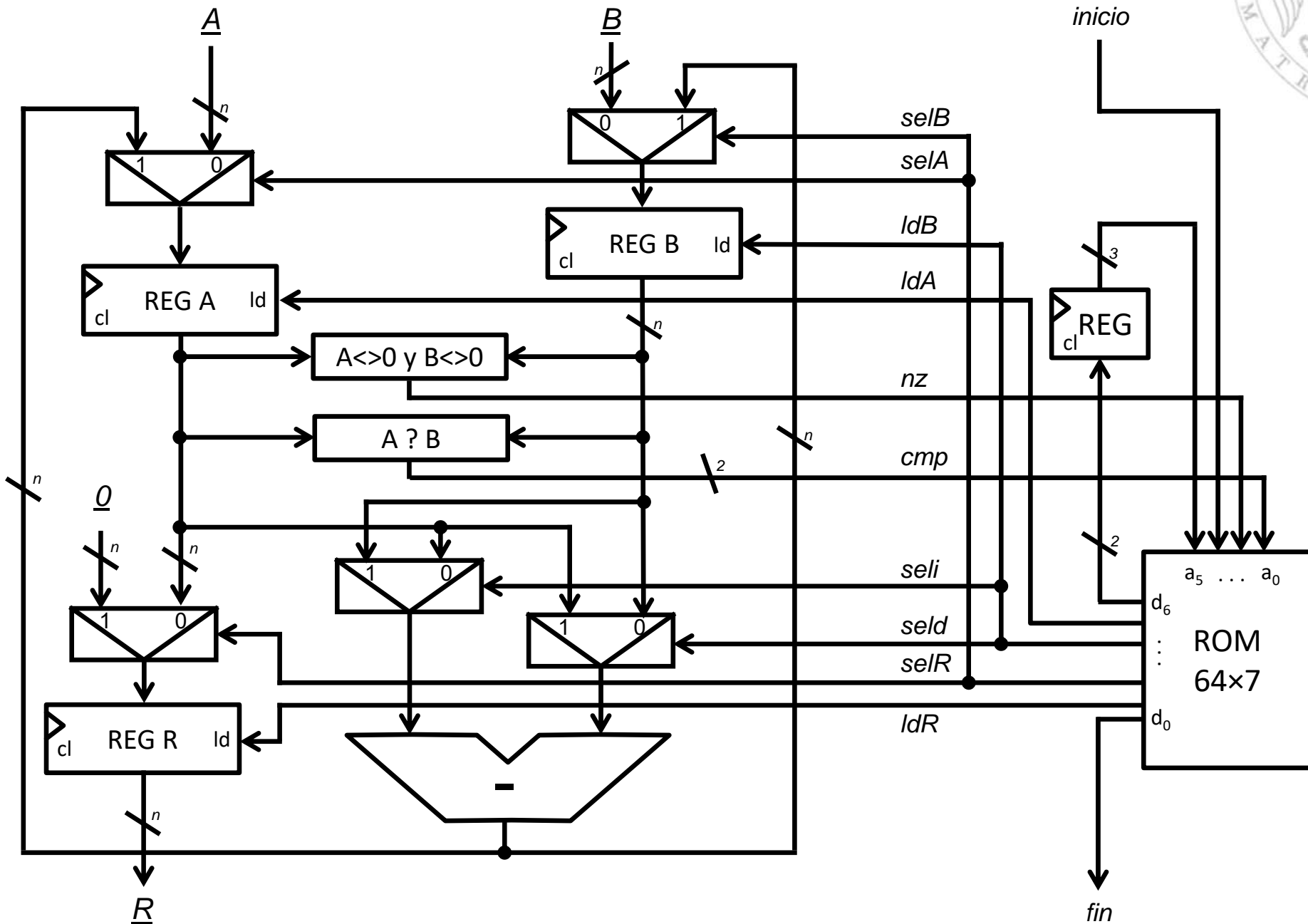
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

99





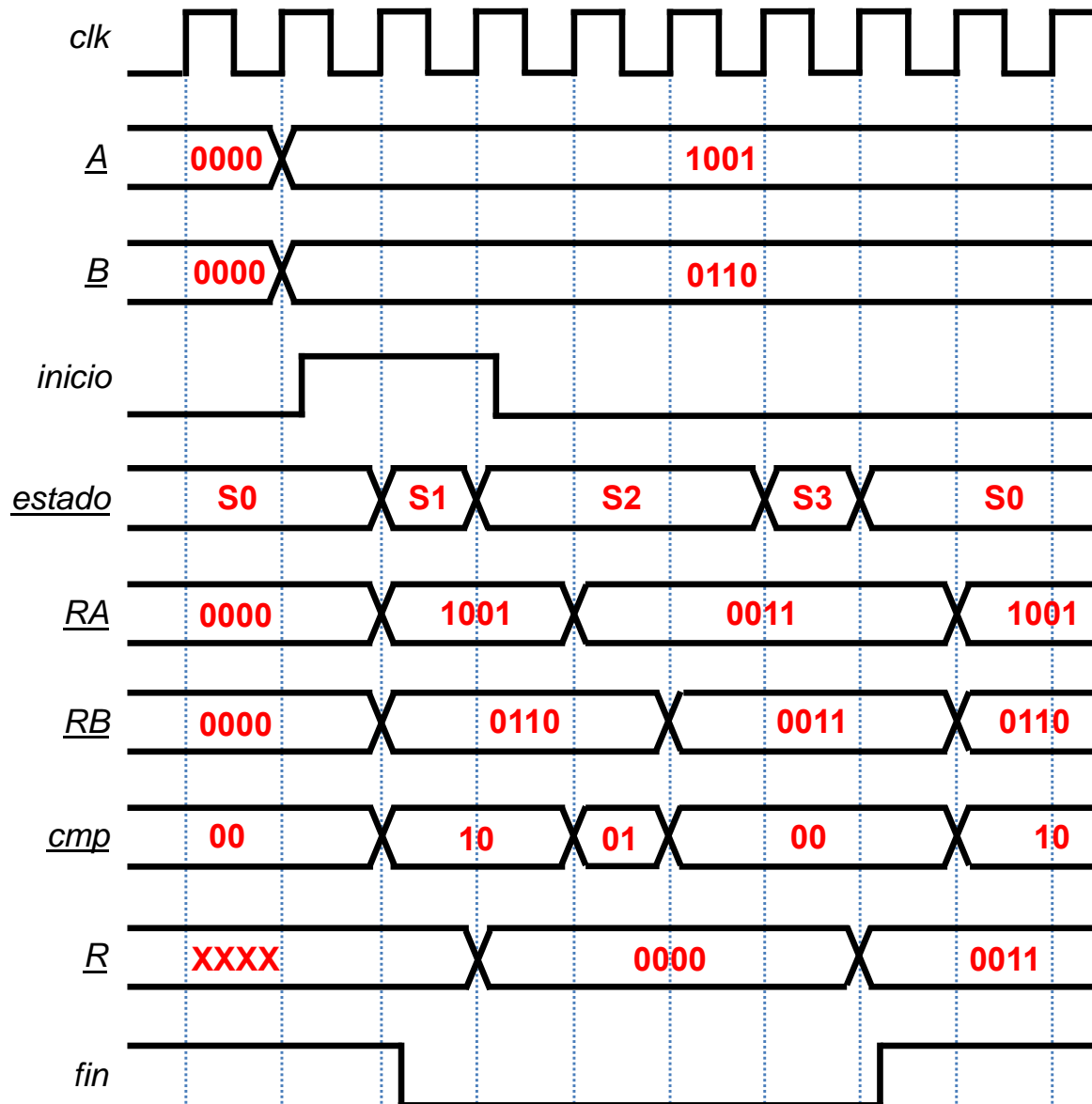
Máximo común divisor

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

100



S0 RA \leftarrow Ain; RB \leftarrow Bin;
fin \leftarrow 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;

S1 RR \leftarrow 0; fin \leftarrow 0;
si (nz=0) ir a S0;
si (nz=1) ir a S2;

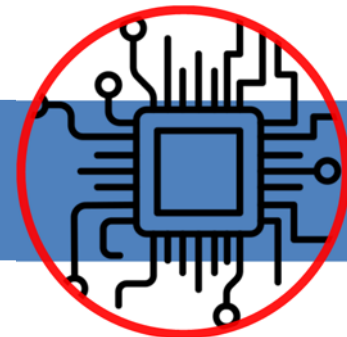
S2 fin \leftarrow 0;
si (cmp=00) ir a S3;
si (cmp=01) RB \leftarrow RB-RA, ir a S2;
si (cmp=10) RA \leftarrow RA-RB, ir a S2;

S3 RR \leftarrow RA; fin \leftarrow 0;
ir a S0;



- Cálculo del coste.
- Cálculo del tiempo de ciclo.

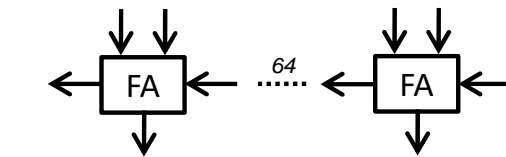
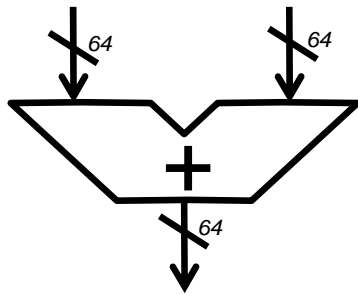
Apéndice tecnológico



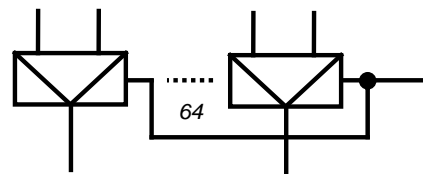
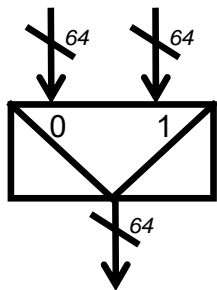


Multiplicador iterativo

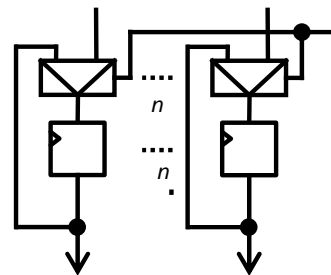
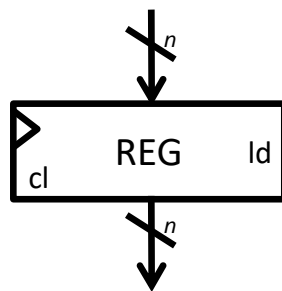
Cálculo del coste y tiempo de ciclo



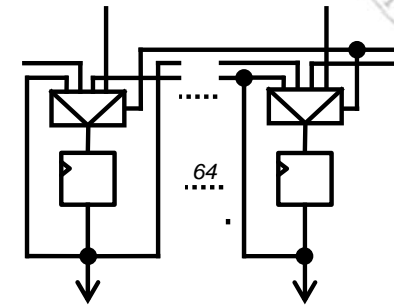
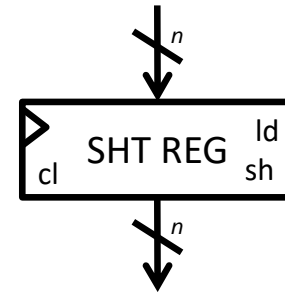
área: $64 \times 29.49 = 1887 \mu\text{m}^2$
retardo: $64 \times 226 = 14464 \text{ ps}$



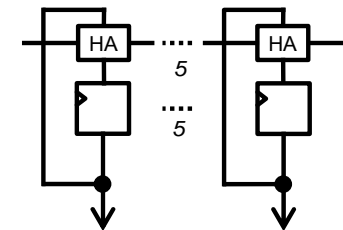
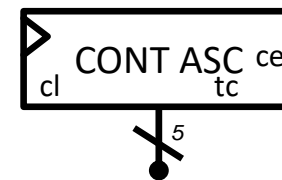
área: $64 \times 11.05 = 707.2 \mu\text{m}^2$
retardo: $1 \times 223 = 223 \text{ ps}$



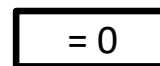
área: $n \times 11.05 + n \times 32.26 = n \times 43.31 \mu\text{m}^2$
retardo CLK→Q: $1 \times 167 = 167 \text{ ps}$
retardo in→D: $1 \times 223 = 223 \text{ ps}$



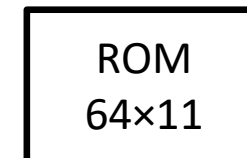
área: $n \times 23.04 + n \times 32.26 = n \times 55.3 \mu\text{m}^2$
retardo CLK→Q: $1 \times 167 = 167 \text{ ps}$
retardo in→D: $1 \times 250 = 250 \text{ ps}$



área: $5 \times 15.77 + 5 \times 32.26 = 240.6 \mu\text{m}^2$
retardo CLK→Q: $1 \times 167 = 167 \text{ ps}$
retardo in→D: $5 \times 114 = 570 \text{ ps}$



área: $15,67 \mu\text{m}^2$
retardo: 126 ps



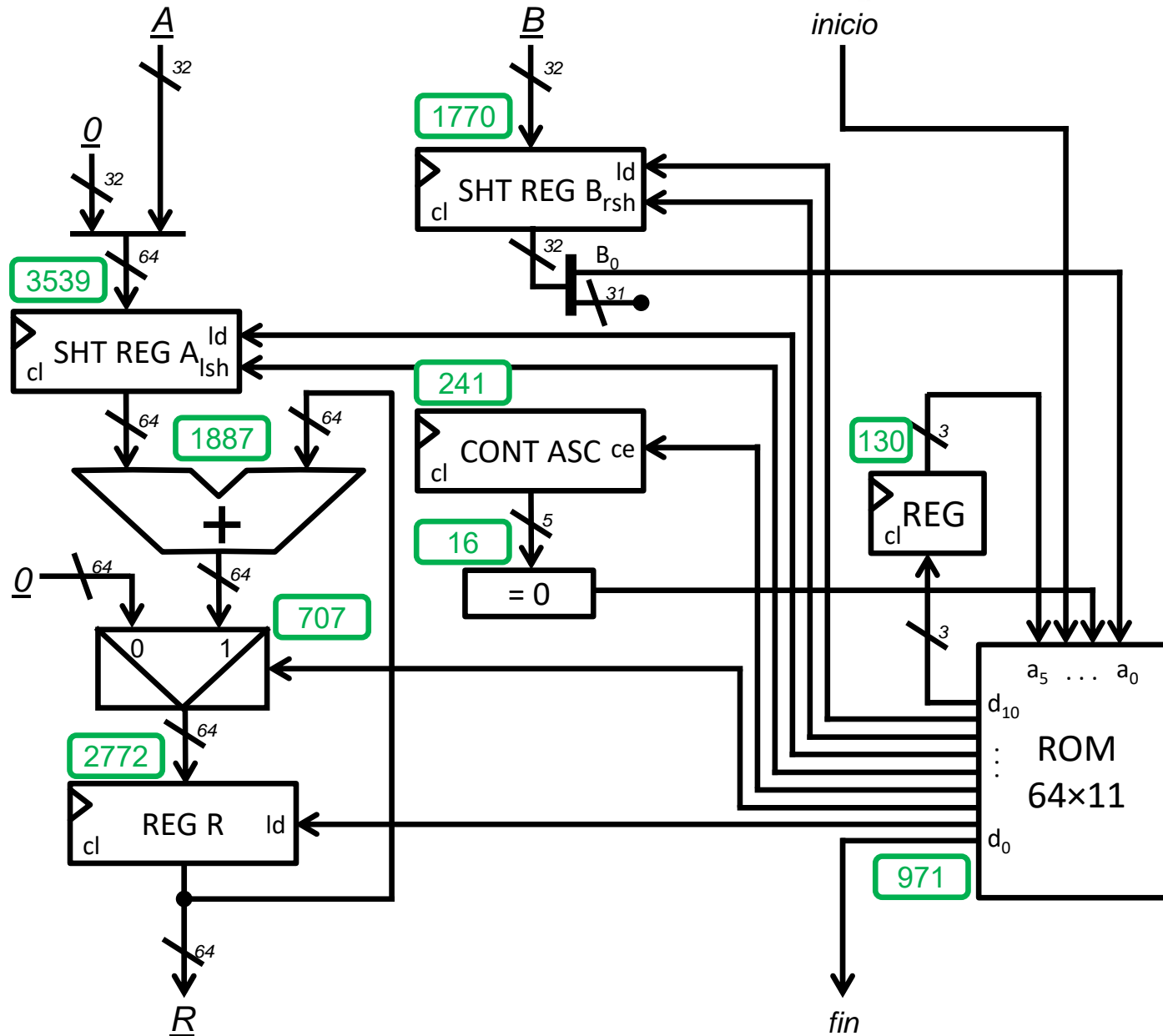
área: $971 \mu\text{m}^2$
retardo: 573 ps



Multiplicador iterativo

Cálculo del coste (CMOS 90 nm)

área: 12033 μm^2

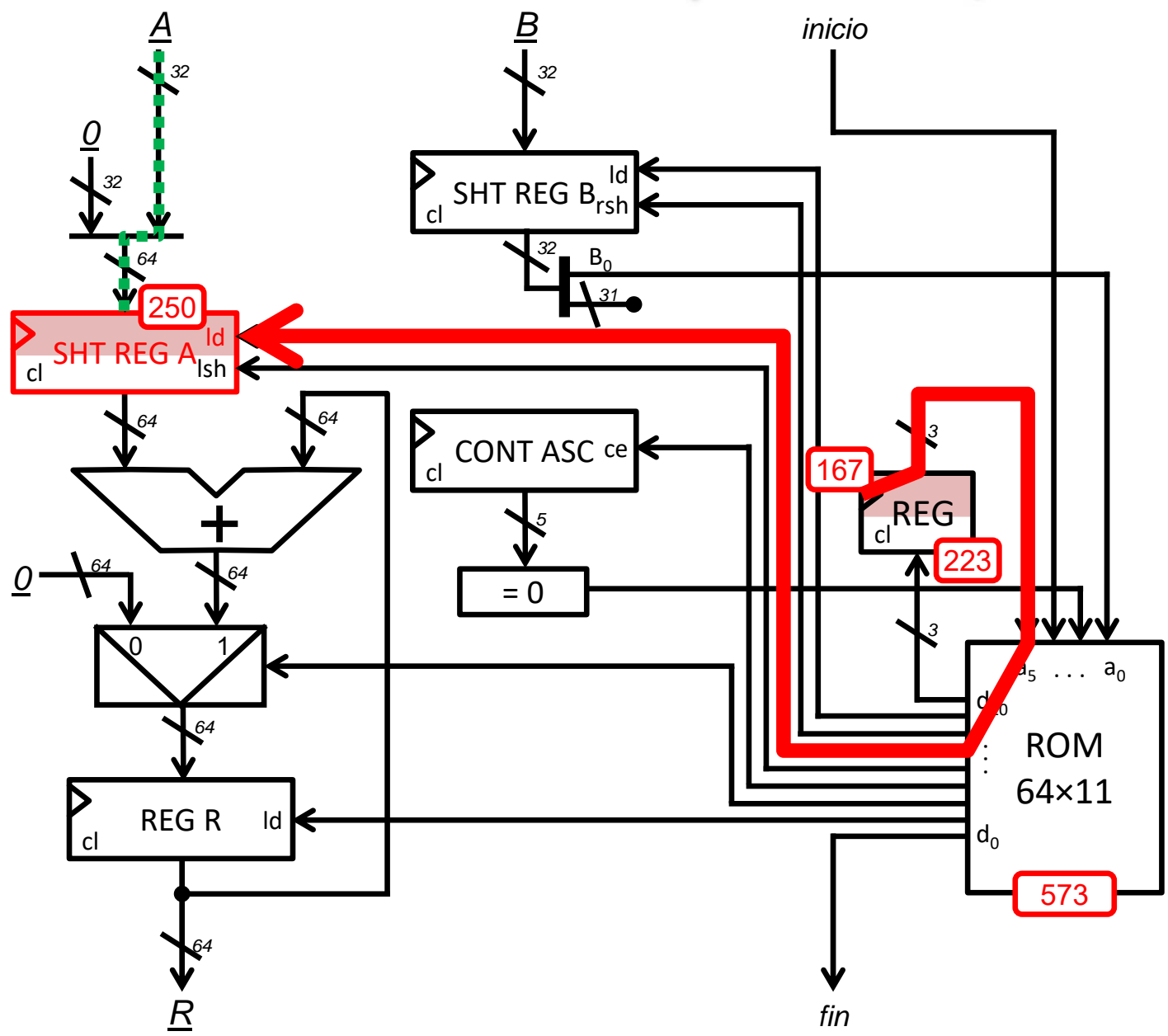




Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



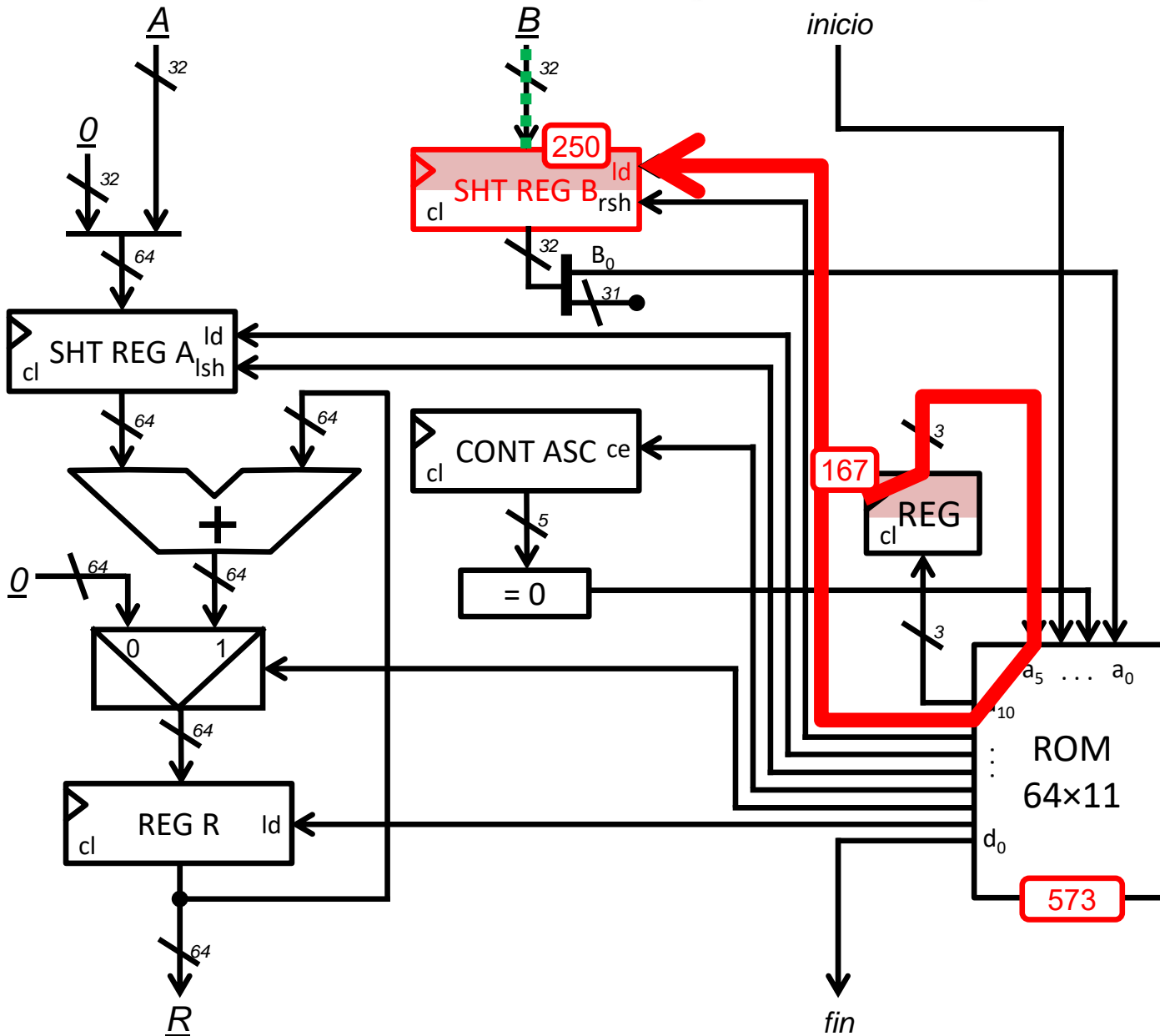
transferencia	retardo
RA ← Ain	990 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



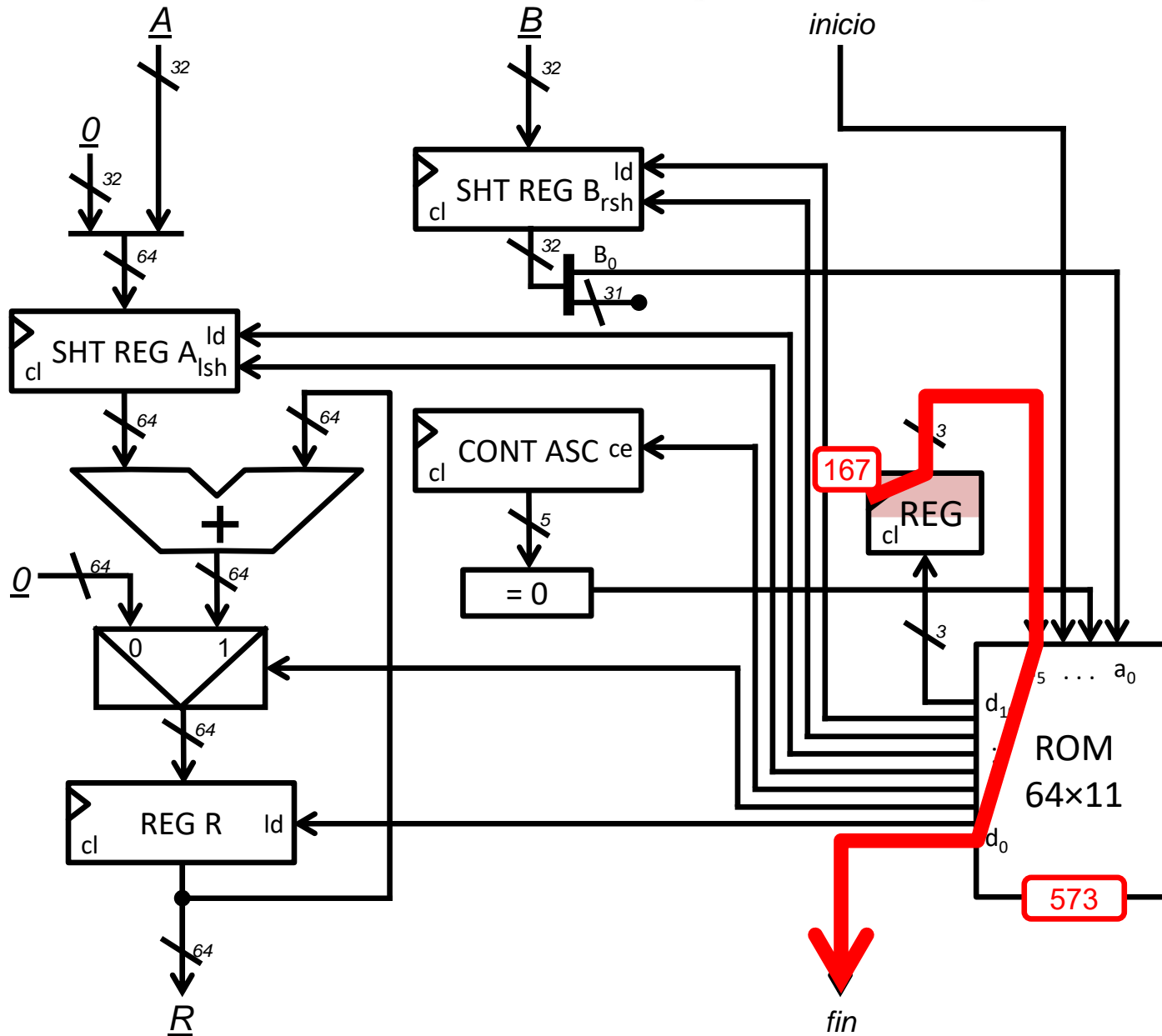
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



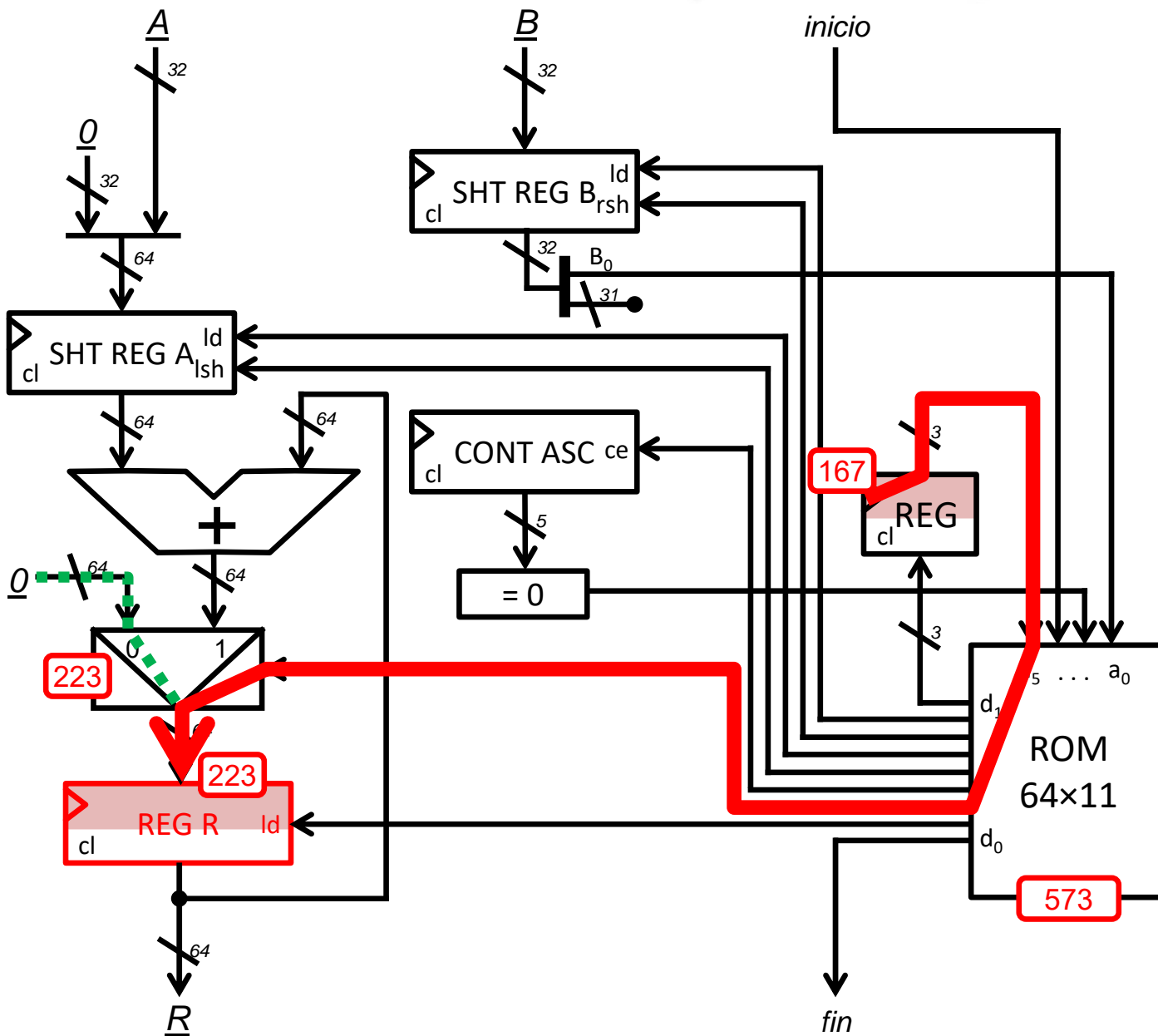
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



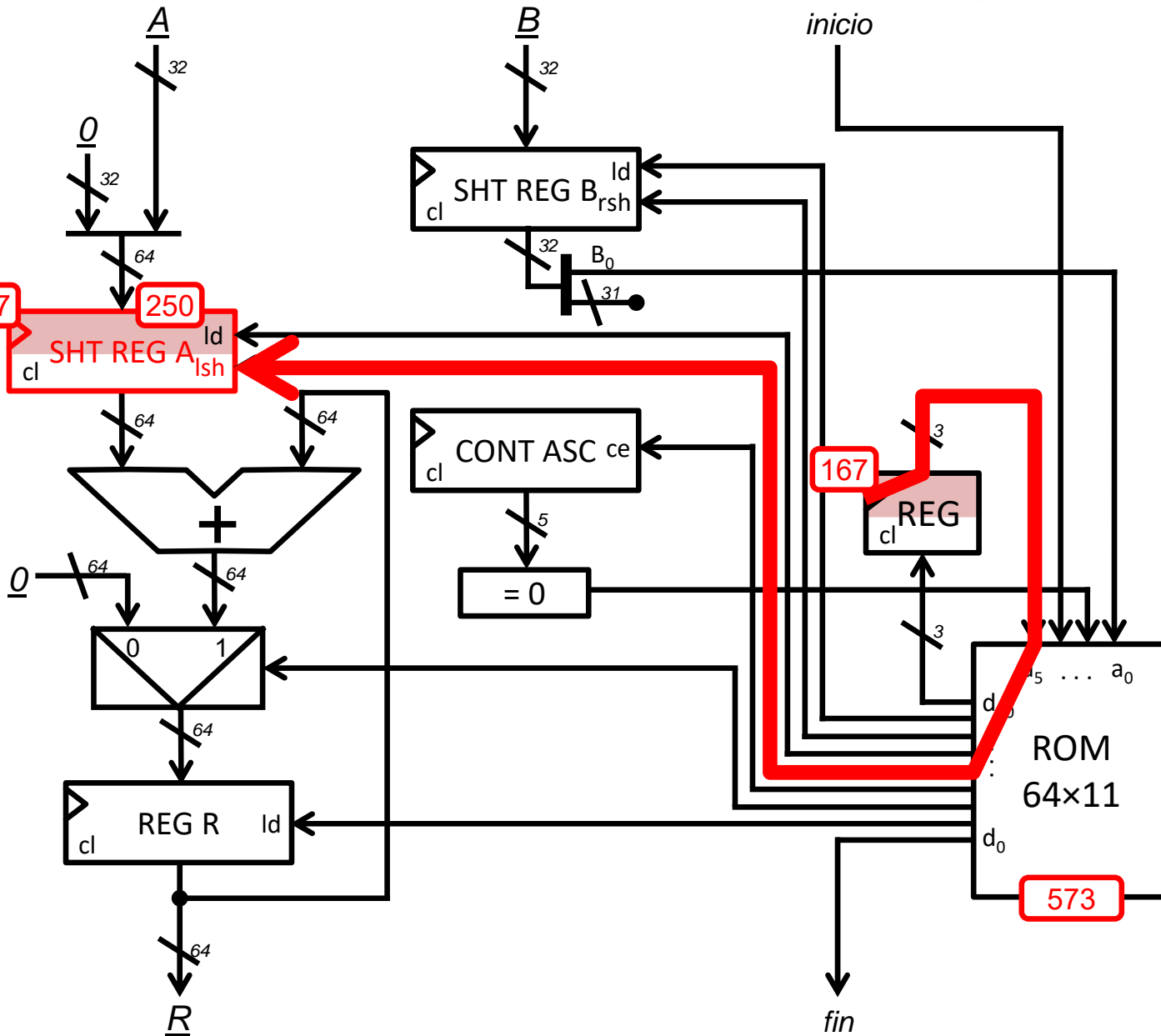
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



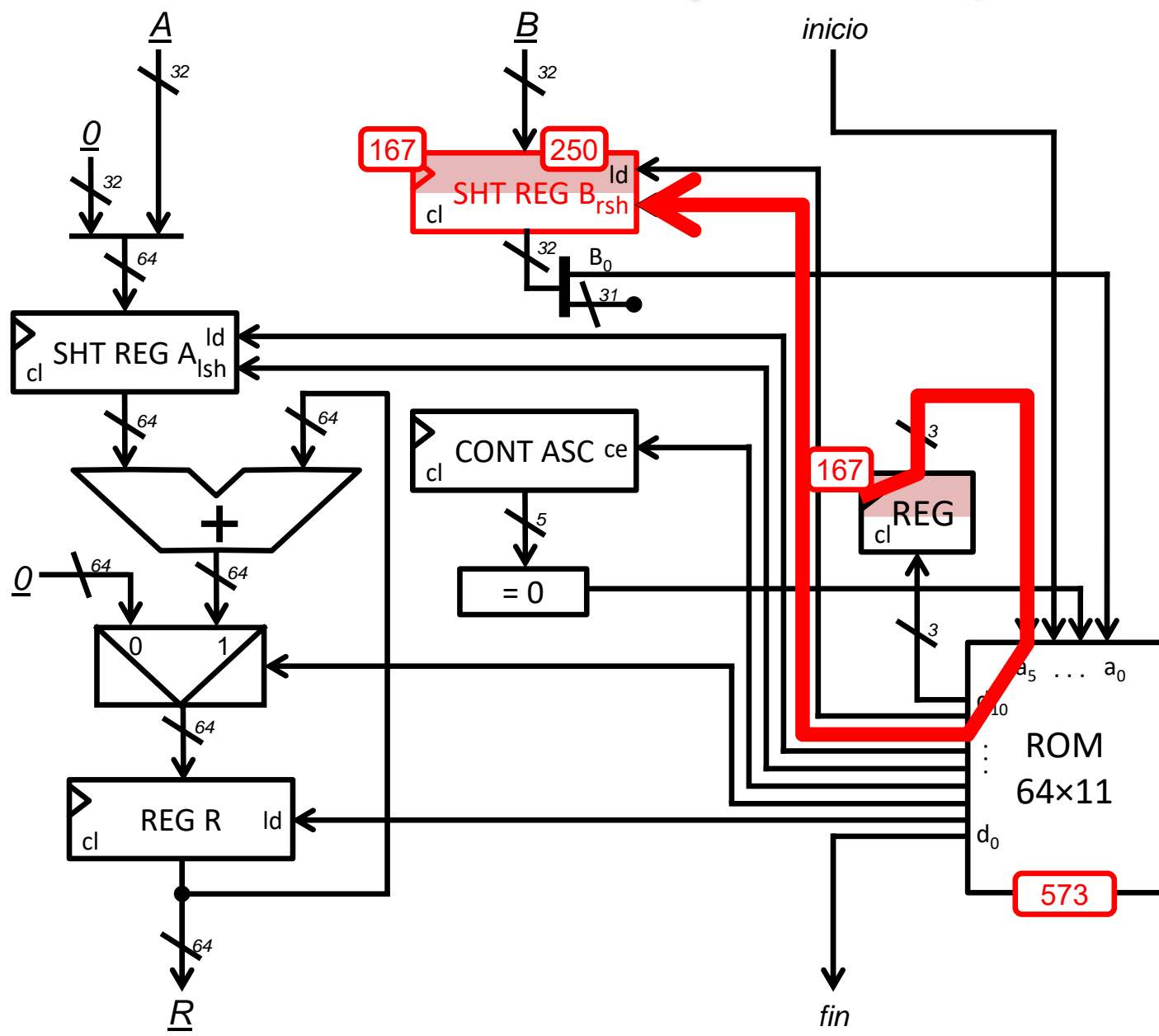
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



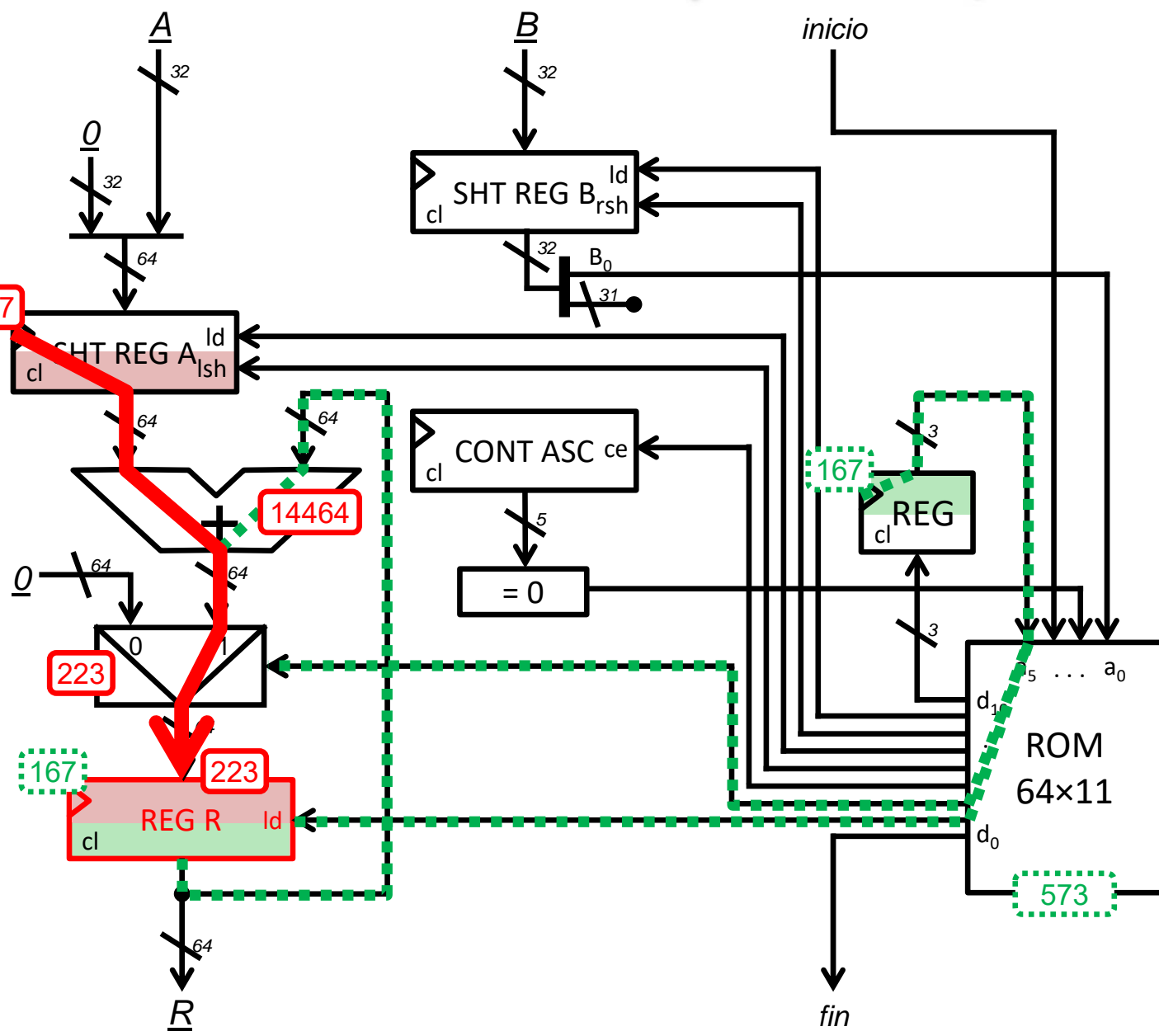
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps
RB \leftarrow RB \gg 1	990 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



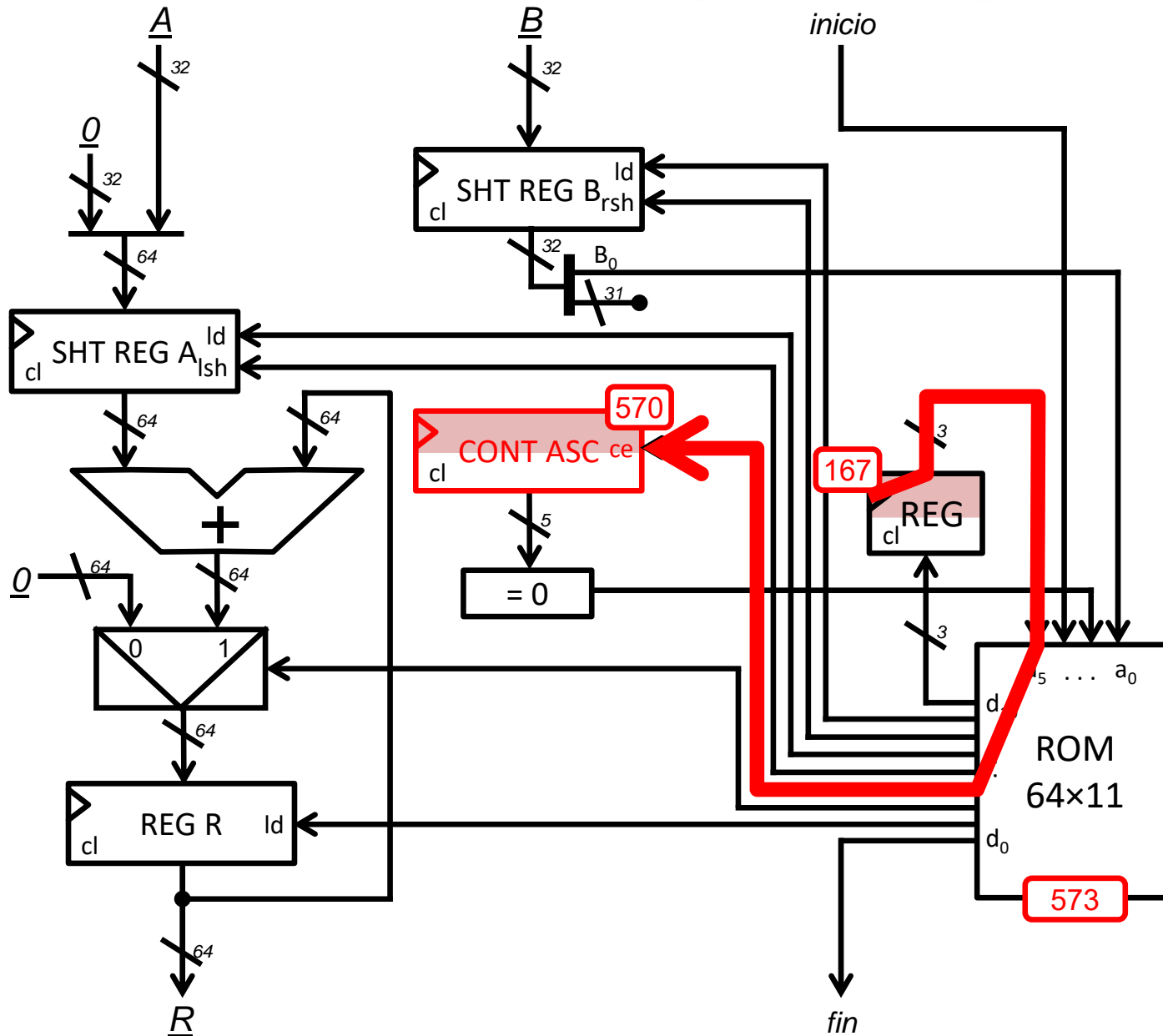
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps
RB \leftarrow RB \gg 1	990 ps
RR \leftarrow RA + RR	15077ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



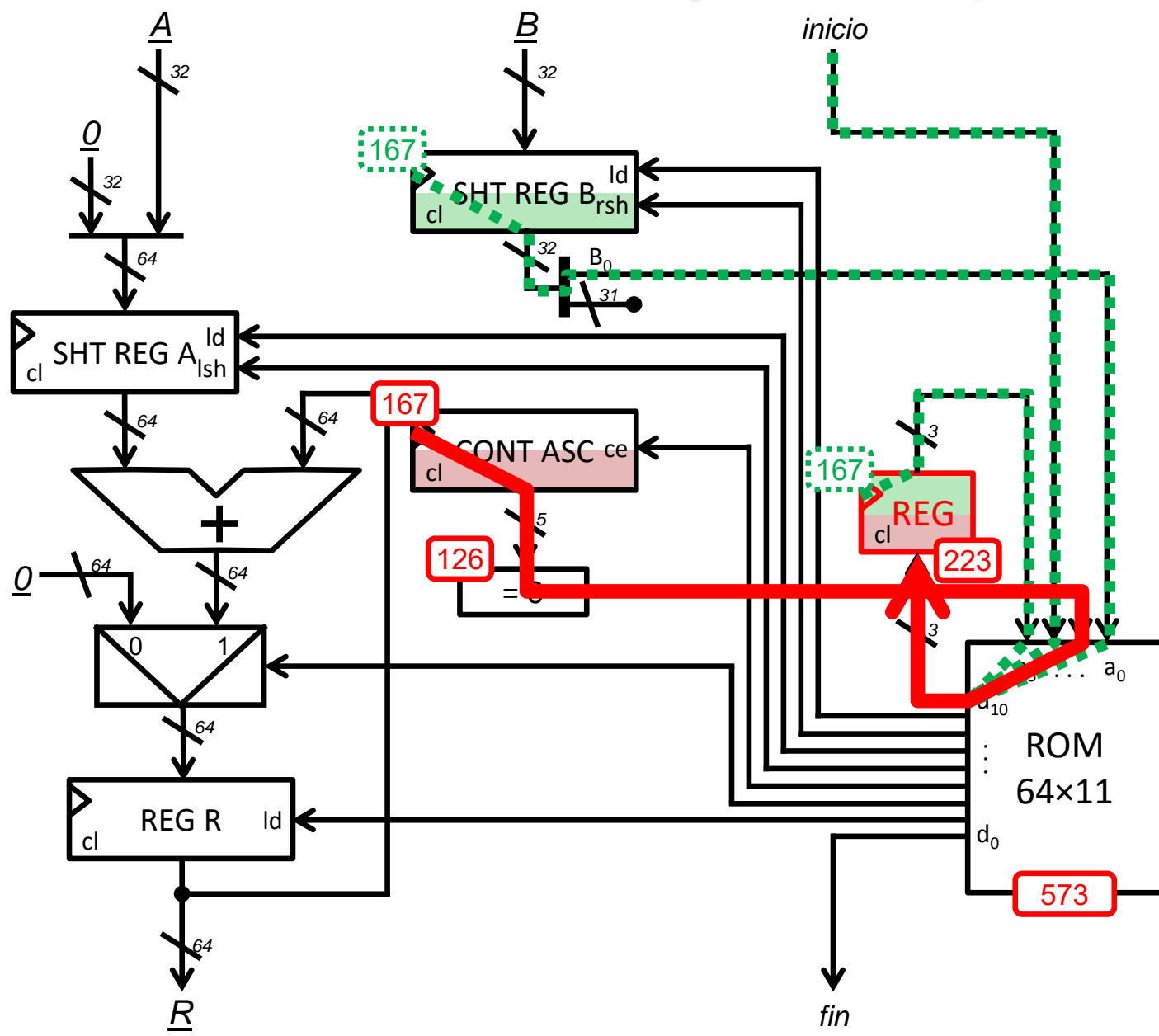
transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps
RB \leftarrow RB \gg 1	990 ps
RR \leftarrow RA + RR	15077ps
RC \leftarrow RC + 1	1350 ps



Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

área: 12033 μm^2



transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps
RB \leftarrow RB \gg 1	990 ps
RR \leftarrow RA + RR	15077ps
RC \leftarrow RC + 1	1350 ps
cálculo de estado	1089 ps



Multiplicador iterativo

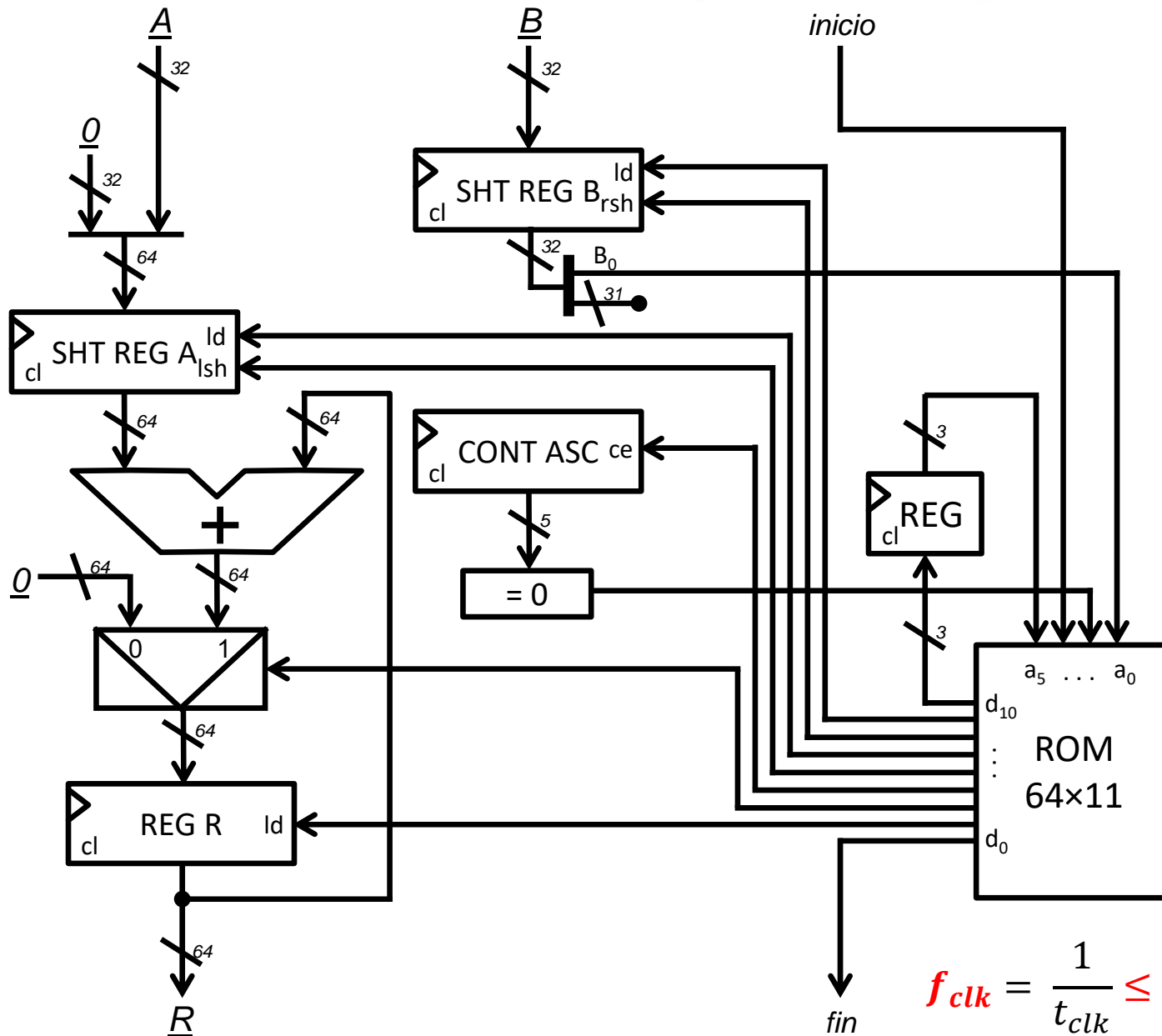
Cálculo del tiempo de ciclo (CMOS 90 nm)

versión 14/07/23

tema 8:
Rutas de datos y controladores

FC-1

113



área: 12033 μm^2

tiempo de cálculo: 1 μs
(66 ciclos \times 15077 ps/ciclo)

transferencia	retardo
RA \leftarrow Ain	990 ps
RB \leftarrow Bin	990 ps
fin \leftarrow X	740 ps
RR \leftarrow 0	1186 ps
RA \leftarrow RA \ll 1	990 ps
RB \leftarrow RB \gg 1	990 ps
RR \leftarrow RA + RR	15077ps
RC \leftarrow RC + 1	1350 ps
cálculo de estado	1089 ps
máximo	15077ps

$$f_{clk} = \frac{1}{t_{clk}} \leq \frac{1}{15077 \cdot 10^{-12}\text{s}} = \mathbf{66 \text{ MHz}}$$

Acerca de *Creative Commons*



■ Licencia CC (**Creative Commons**)

- Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:



Reconocimiento (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



No comercial (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



Compartir igual (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: <https://creativecommons.org/licenses/by-nc-sa/4.0/>