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

BY NC SA

Contenidos

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

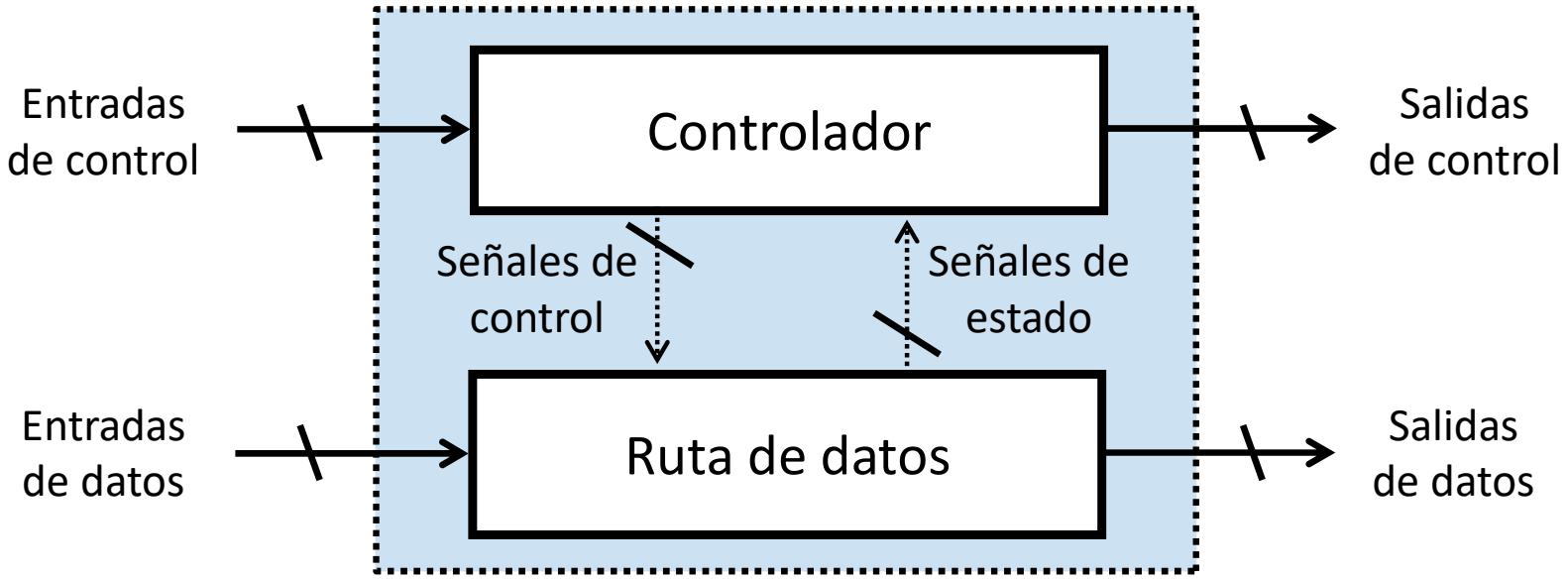
- ✓ Apéndice tecnológico.

Transparencias basadas en los libros:

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

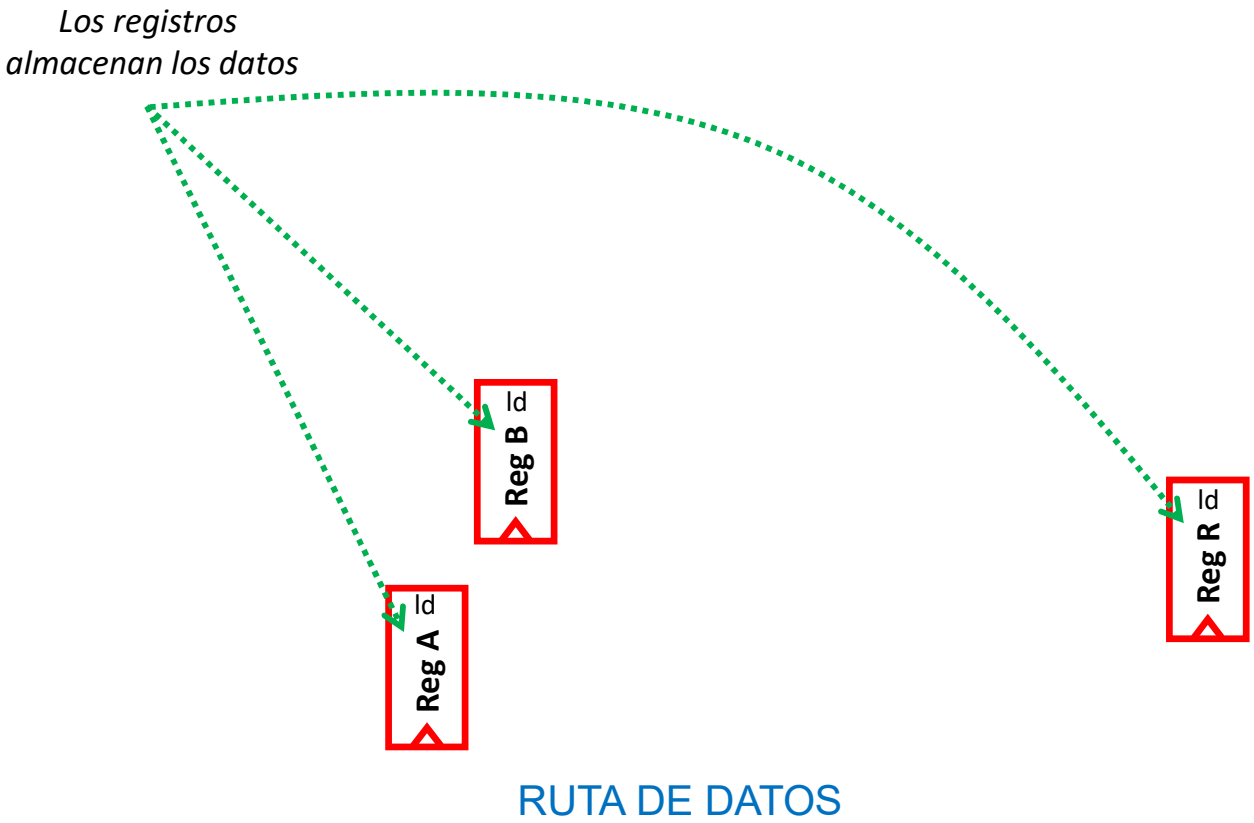
Estructura

- 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.



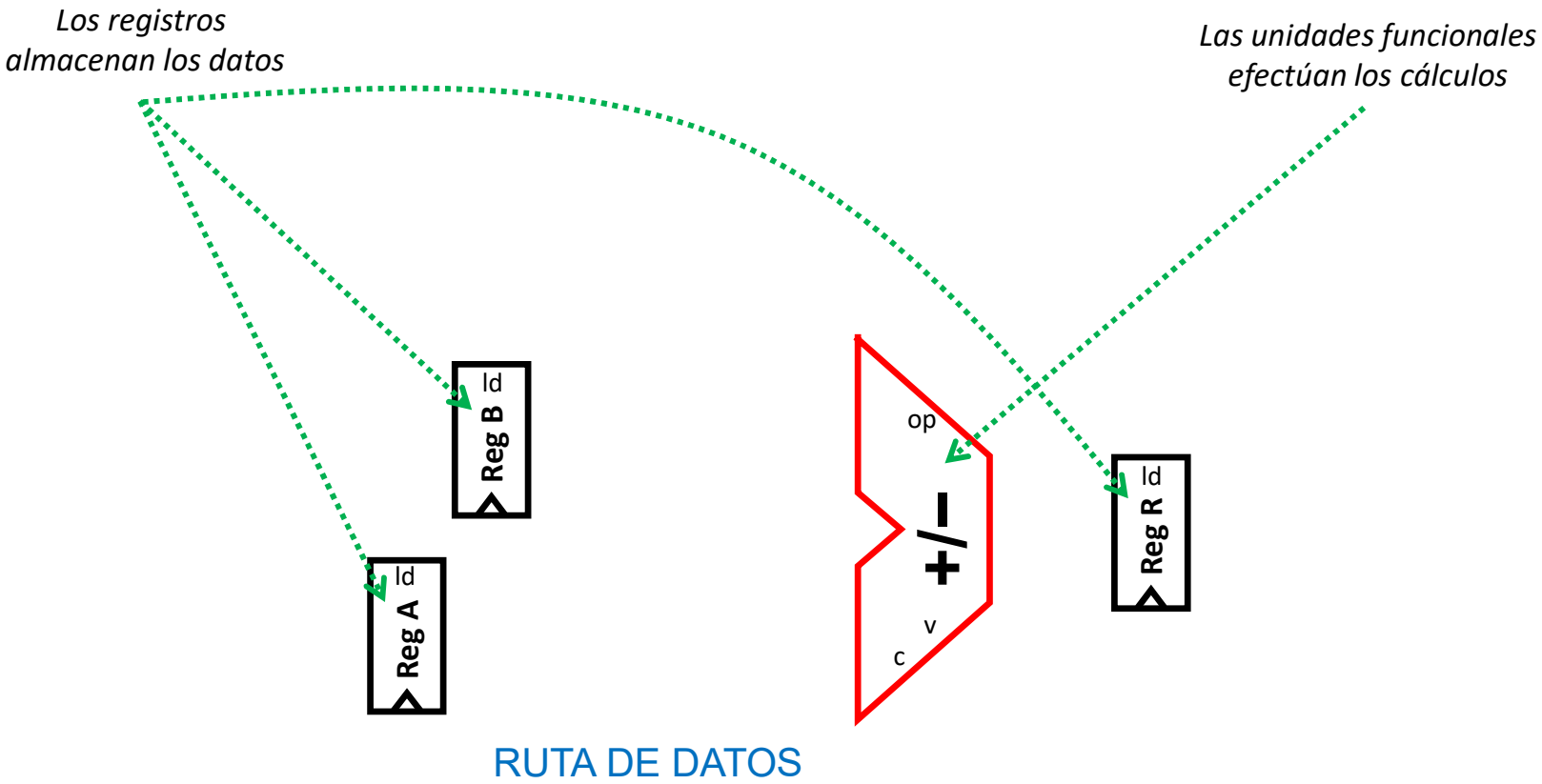
Estructura

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



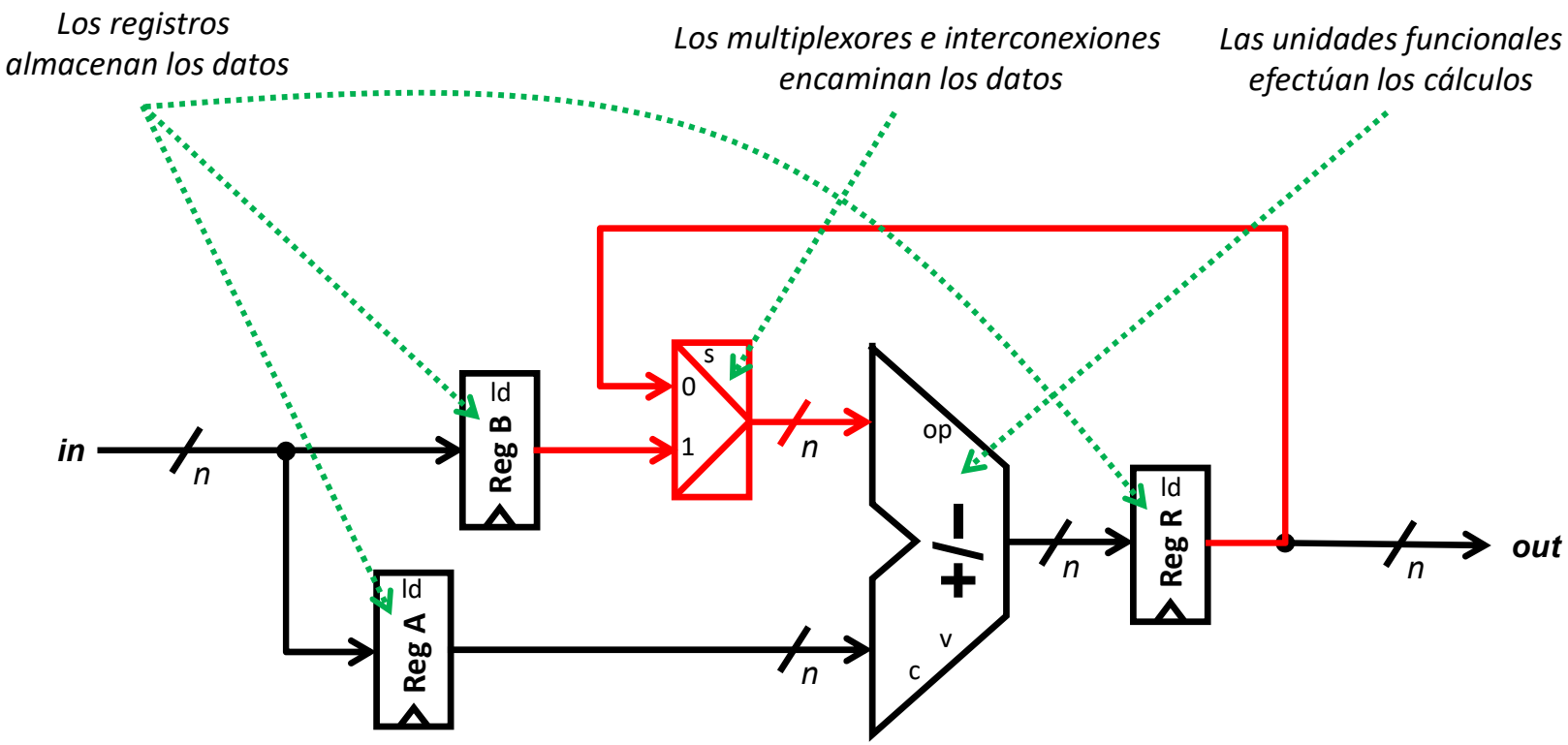
Estructura

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



Estructura

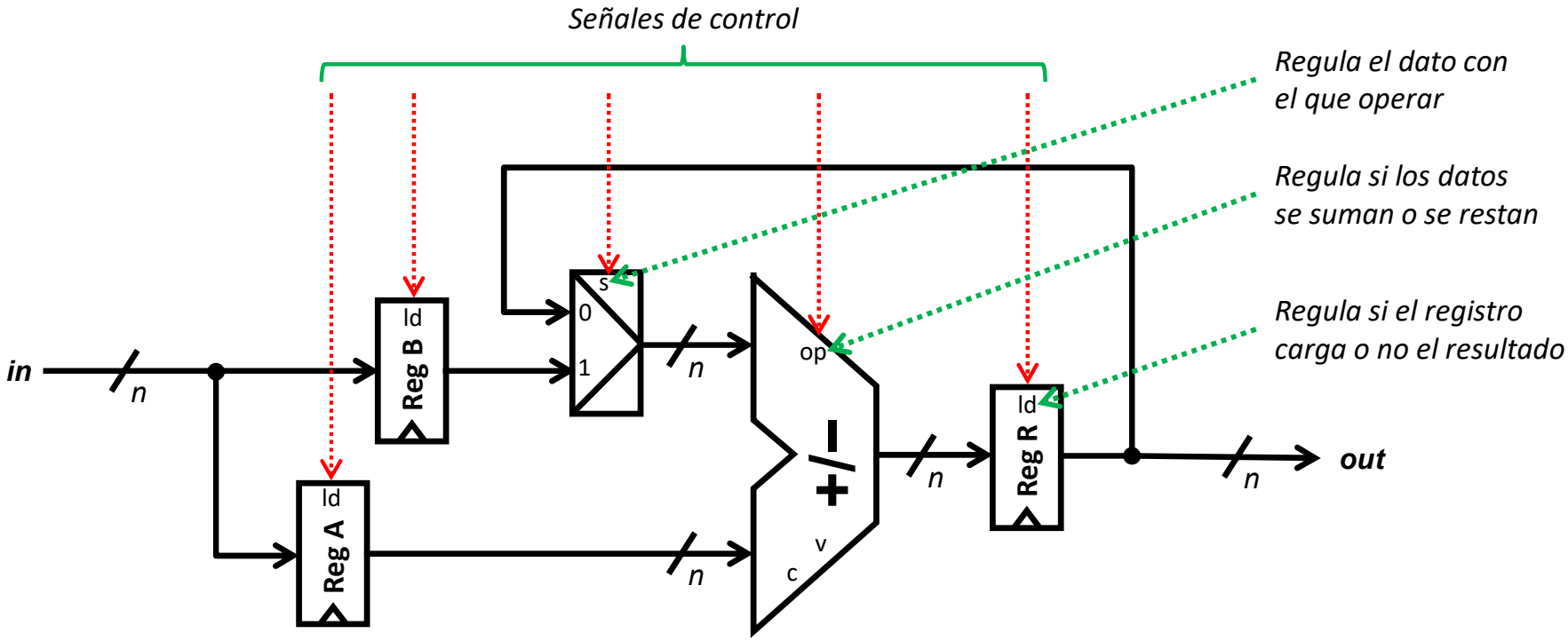
- Toda ruta de datos está formada por:
  - Módulos secuenciales: registros, contadores, desplazadores ...
  - Módulos combinacionales: sumadores, ALU, MUX, lógica ...



RUTA DE DATOS

Estructura

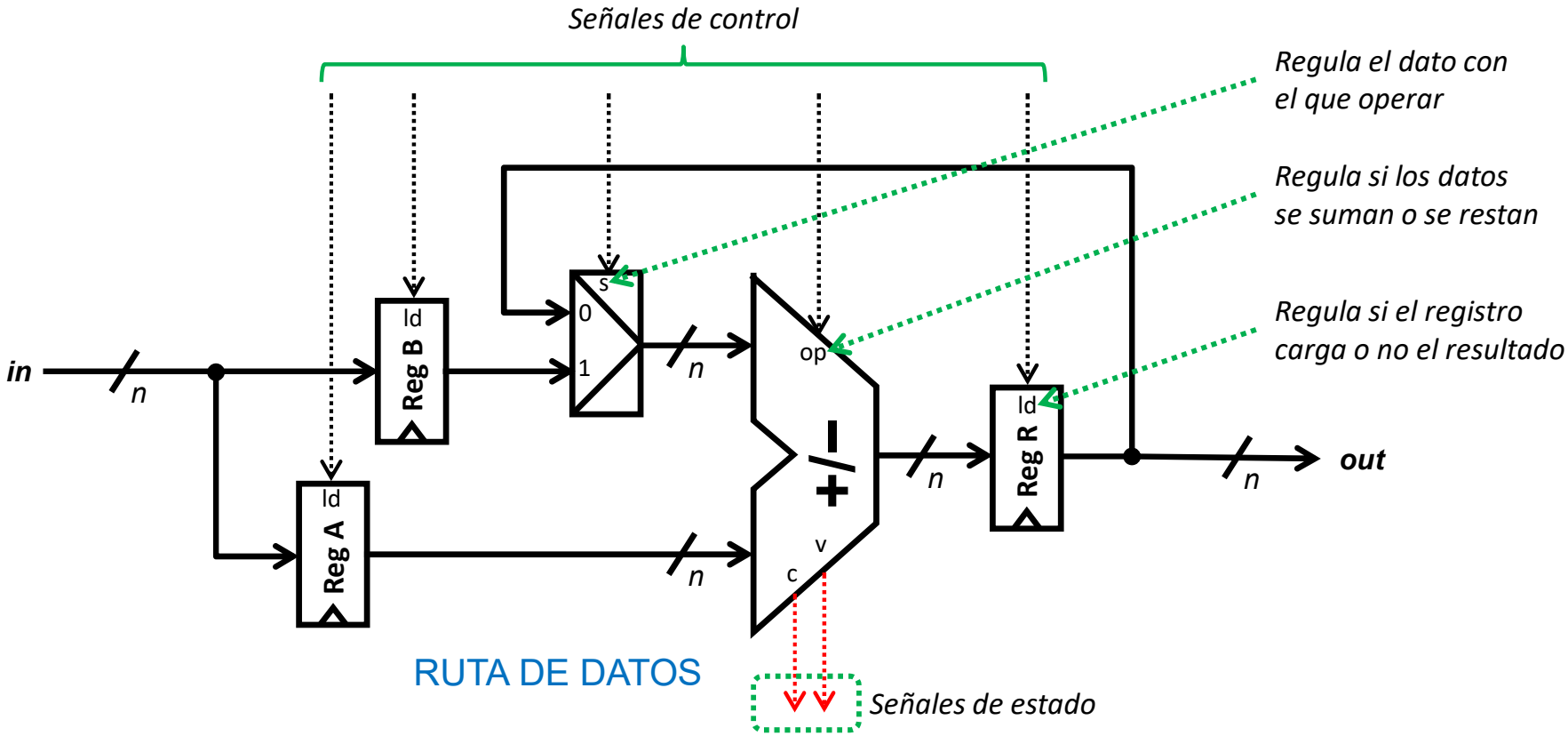
- En toda ruta de datos existen:
  - Señales de control: fijan la funcionalidad de cada módulo.



RUTA DE DATOS

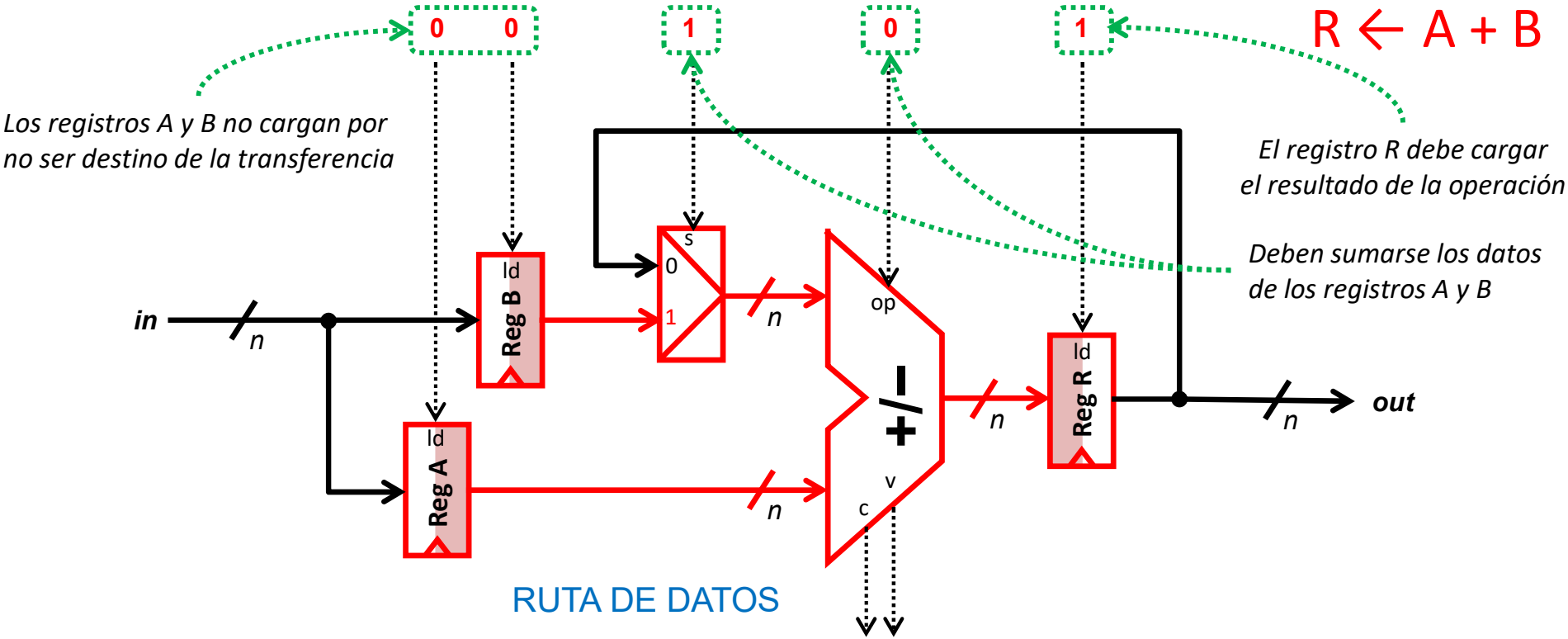
Estructura

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



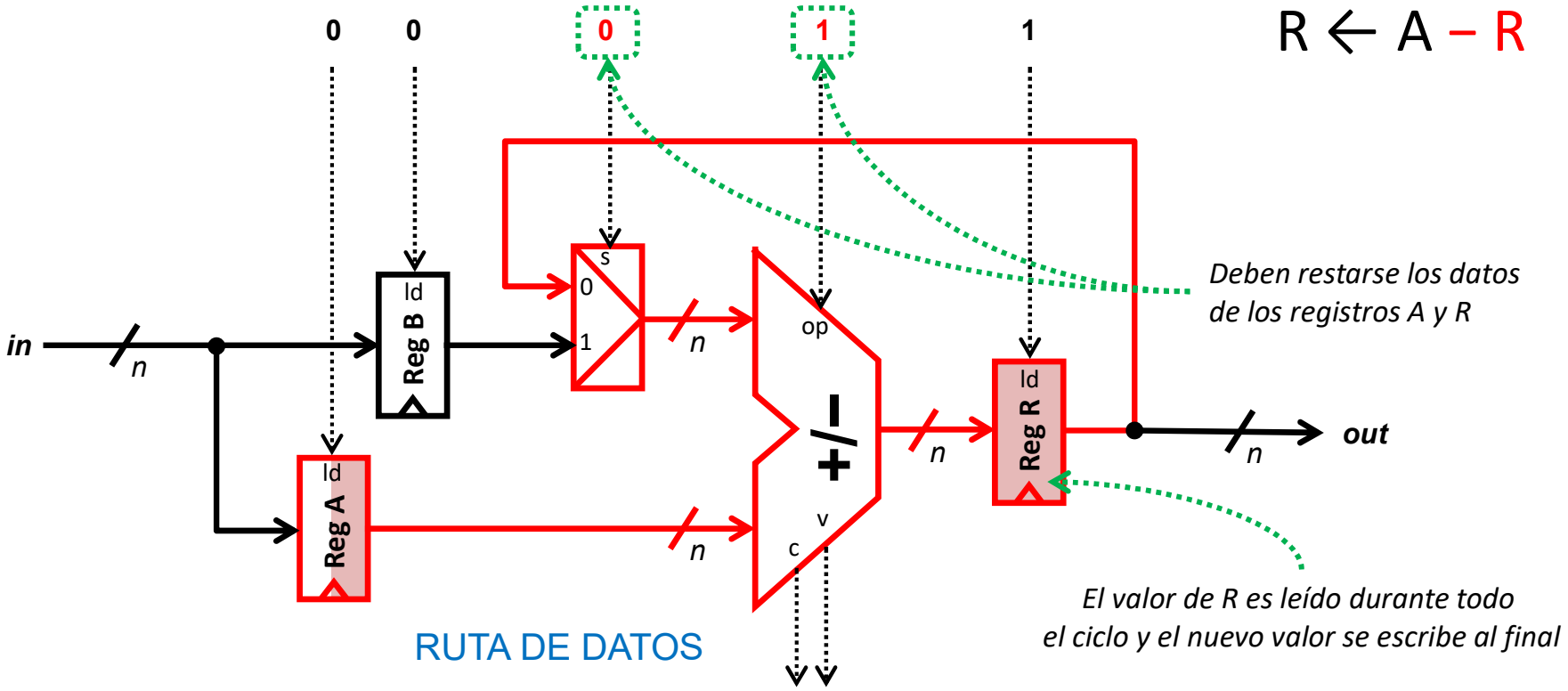
Modelo de cómputo

- 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.



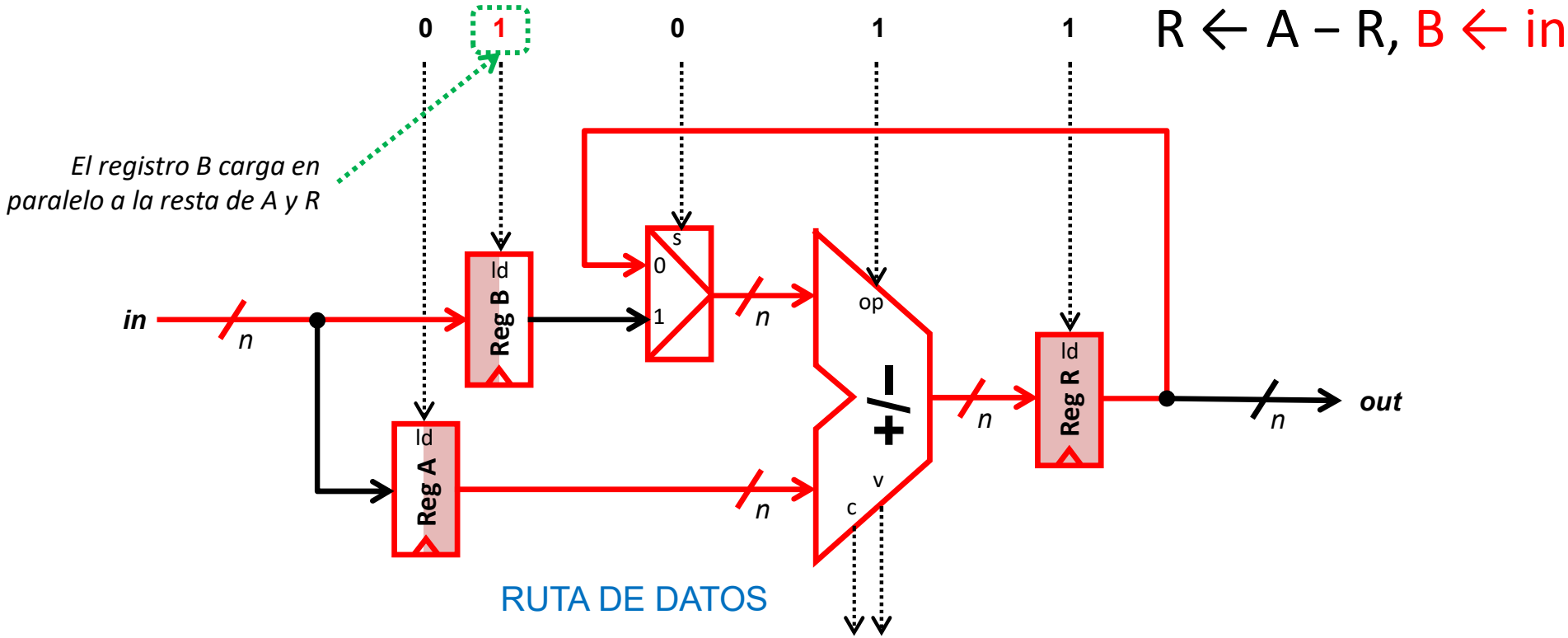
Modelo de cómputo

- 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.



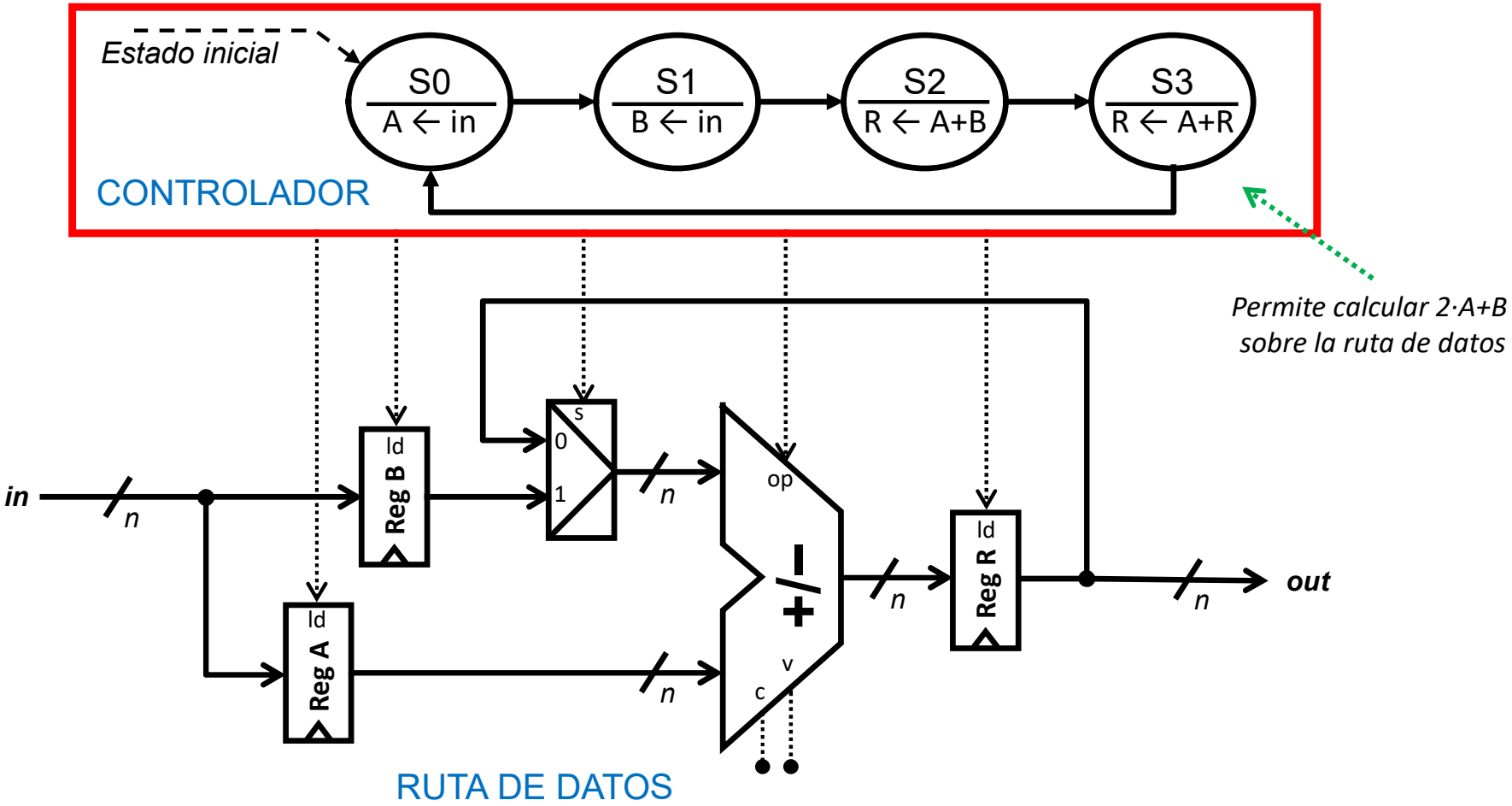
Modelo de cómputo

- 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.



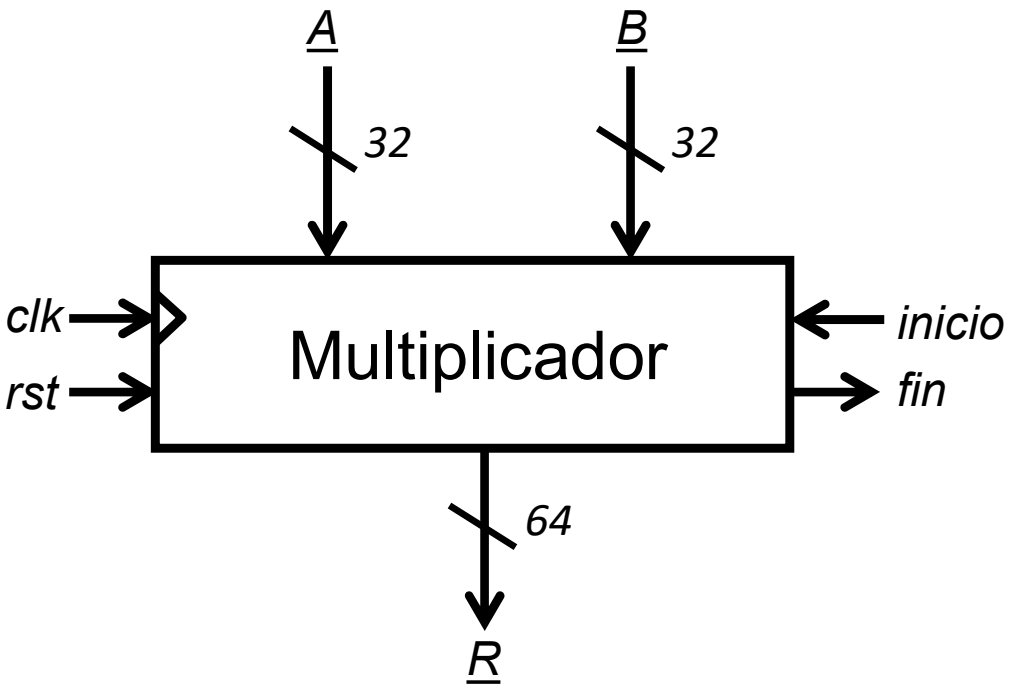
Metodología de diseño

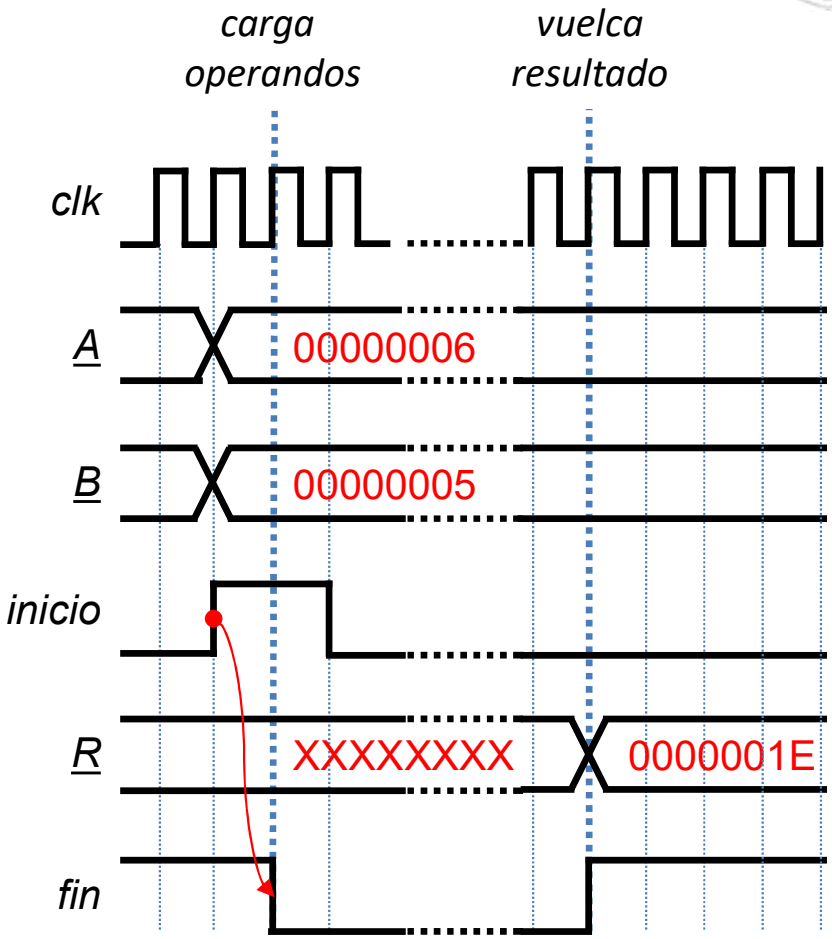
- Para diseñar sistemas complejos se aplica la metodología de diseño algorítmico RTL (*Register Transfer Level*) siguiente:

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





110	$a_2a_1a_0$
× 101	× $b_2b_1b_0$
000110	$(000a_2a_1a_0) \times b_0$
000000	$(00a_2a_1a_00) \times b_1$
+ 011000	+ $(0a_2a_1a_000) \times b_2$
011110	$r_5r_4r_3r_2r_1r_0$

*Algoritmo para **operandos de 3 bits***

1. Especificación de alto nivel

Multiplicador iterativo

Especificación y simulación del algoritmo

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

2. Especificación del algoritmo

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. Especificación del algoritmo

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

2. Especificación del algoritmo

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**

2. Especificación del algoritmo

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**

2. Especificación del algoritmo

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**	**0**

Multiplicador iterativo

Especificación y simulación del algoritmo

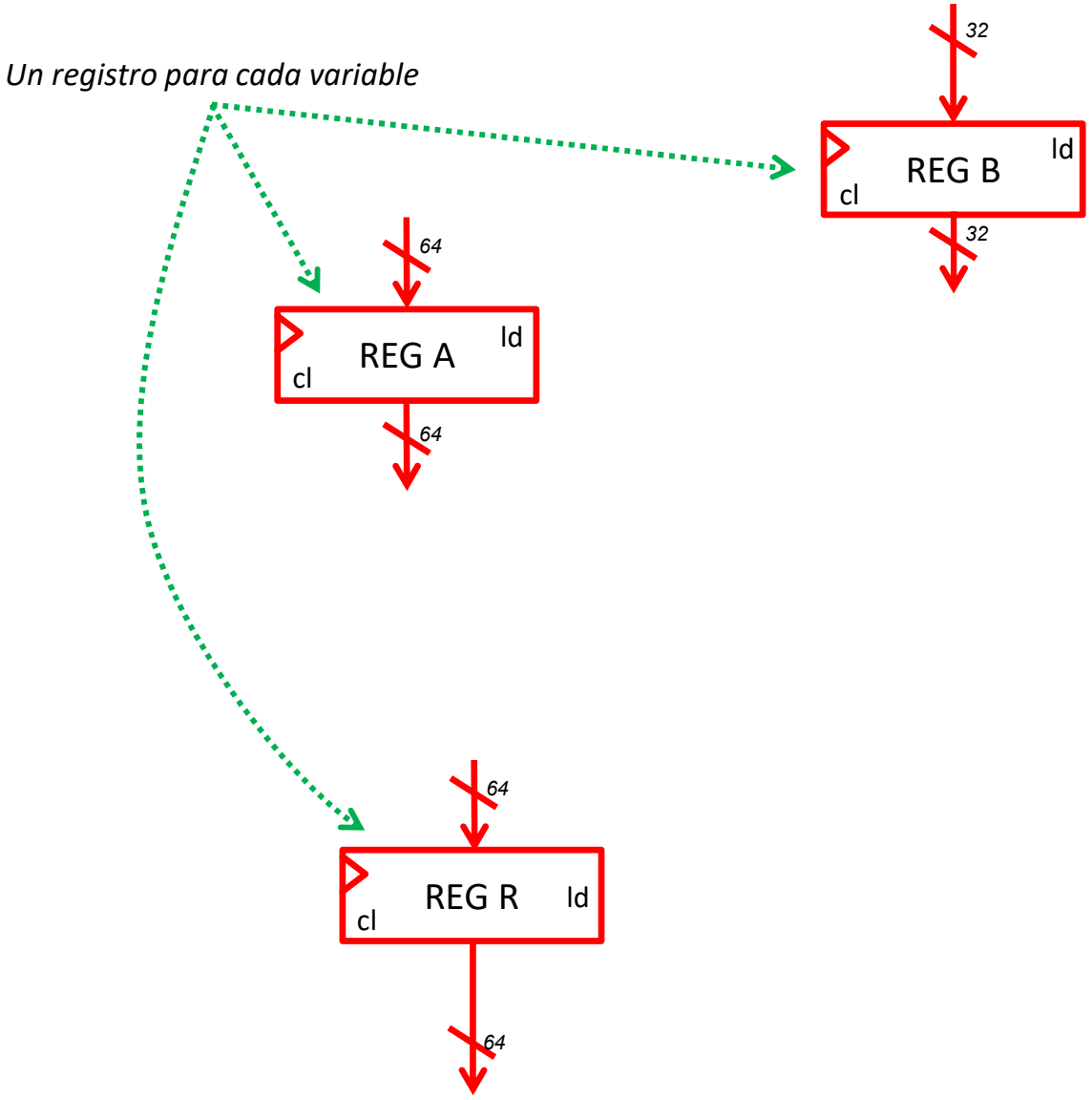
```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<3; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

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
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;
```

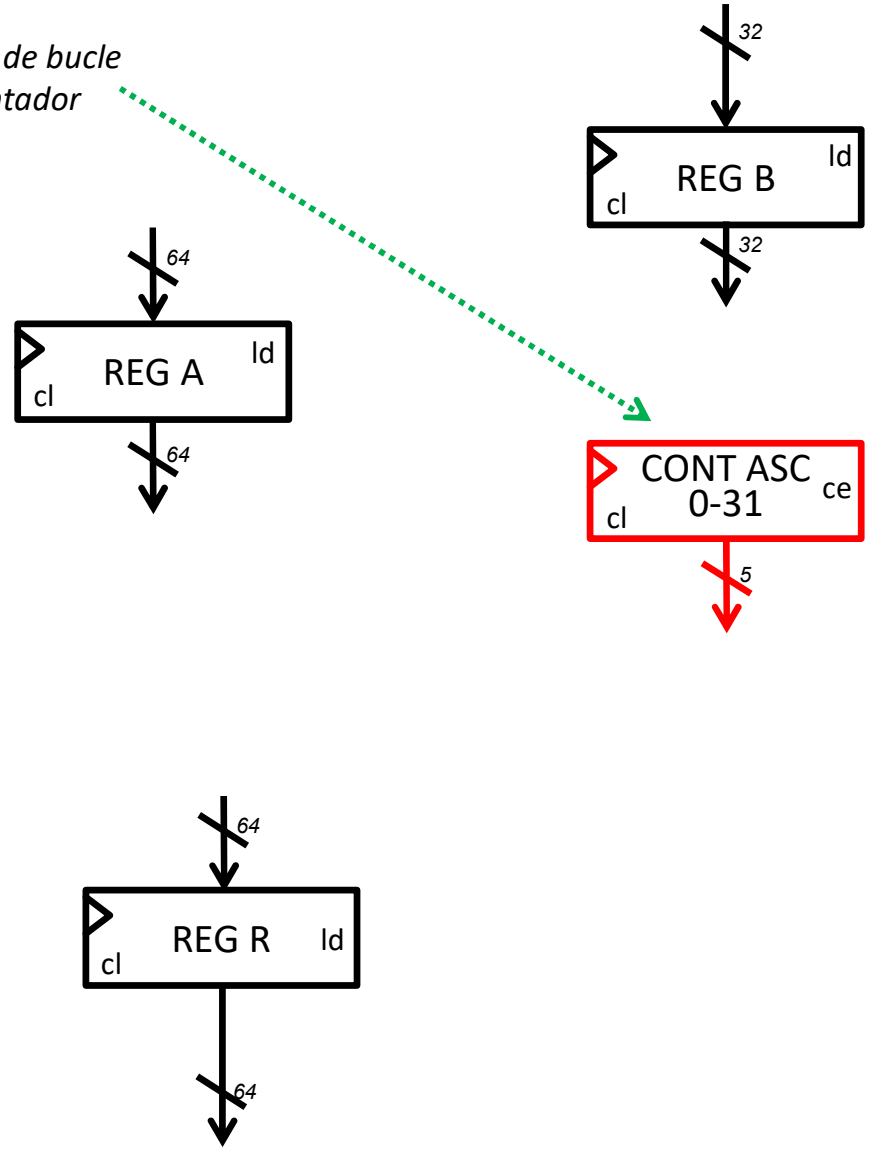
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

Multiplicador iterativo

Elementos de almacenamiento

Para los índices de bucle mejor un contador



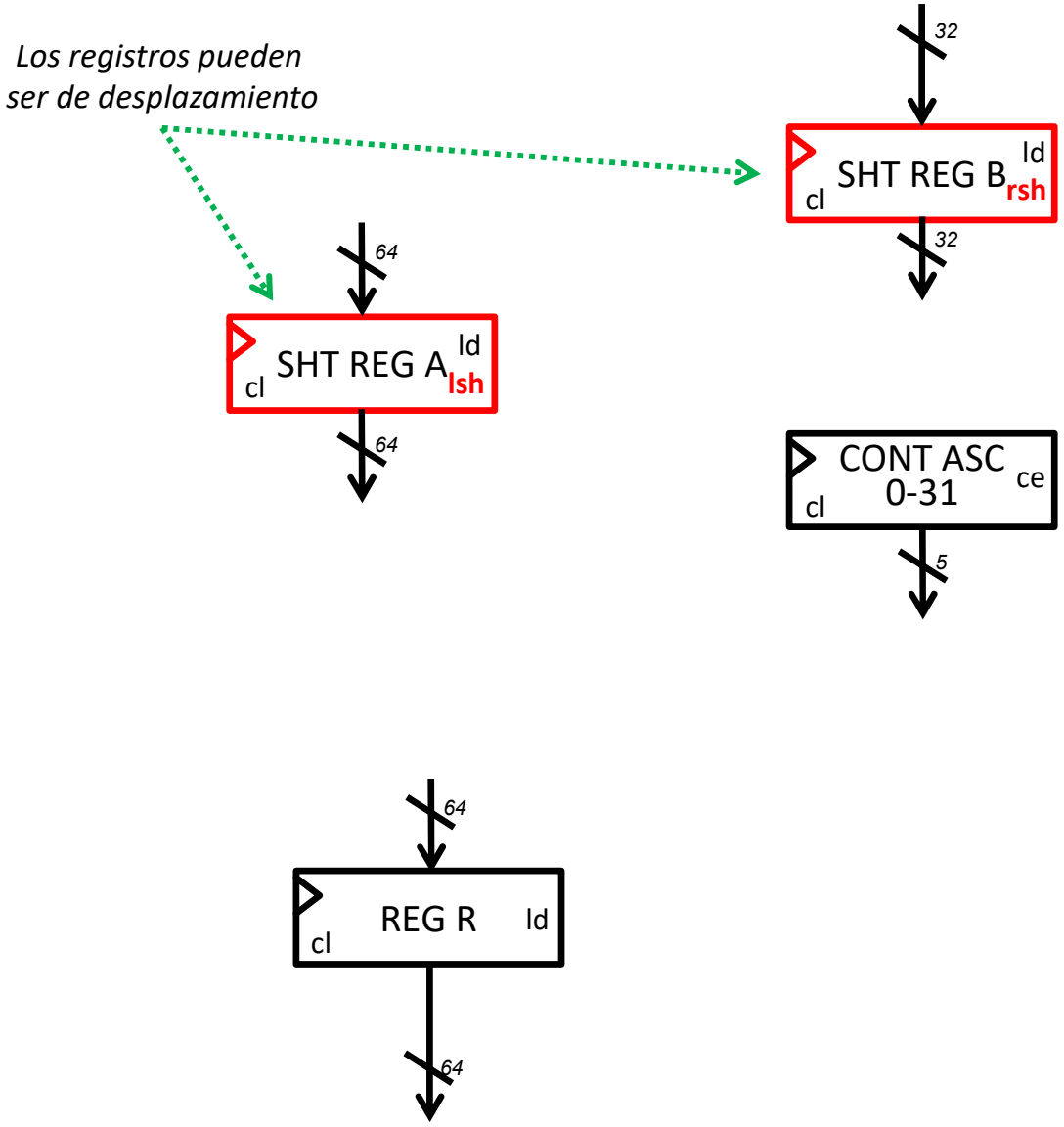

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<32; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

Algoritmo para ***operandos de 32 bits***

3. Diseño de la ruta de datos

Multiplicador iterativo

Desplazamiento de operandos



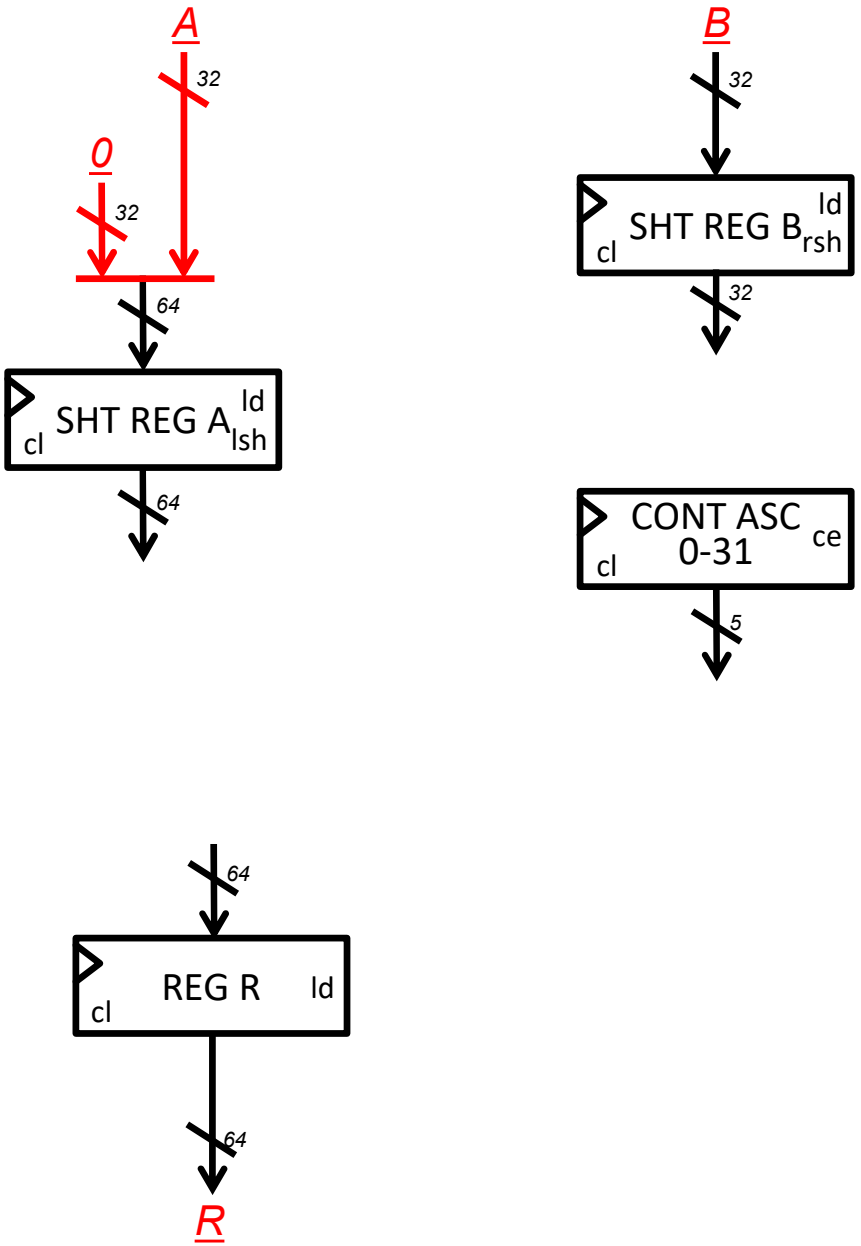

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<32; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

Multiplicador iterativo

Conexión con el exterior



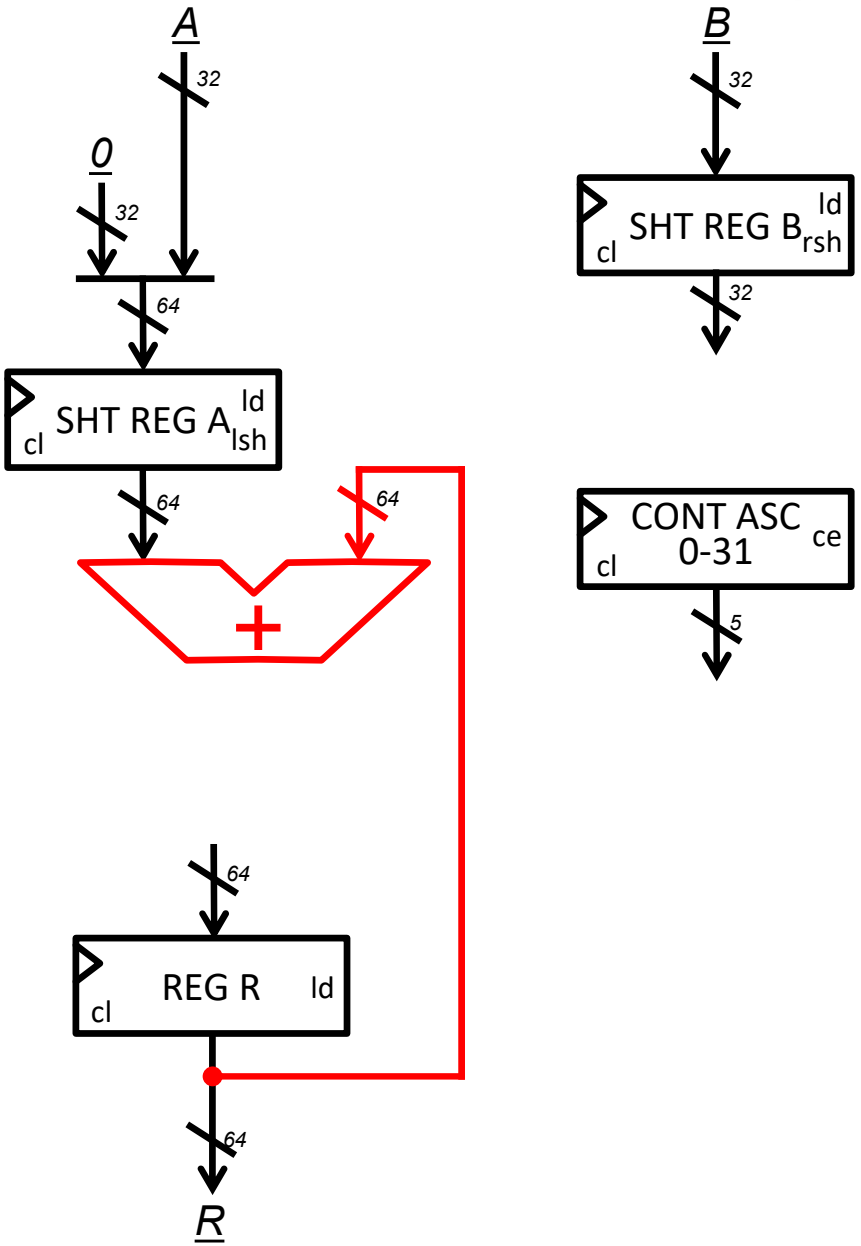

```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

Multiplicador iterativo

Suma de productos parciales



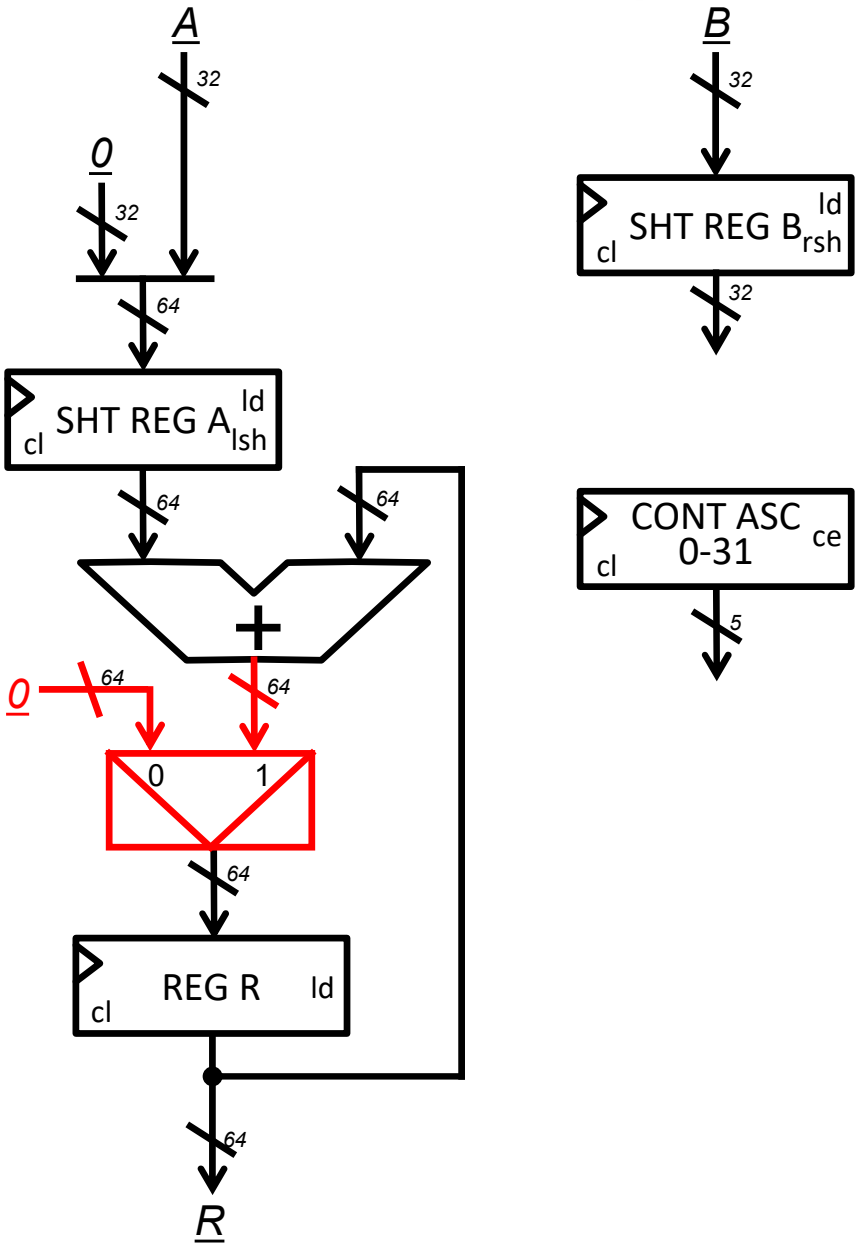

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<32; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

Multiplicador iterativo

Inicialización y almacenaje



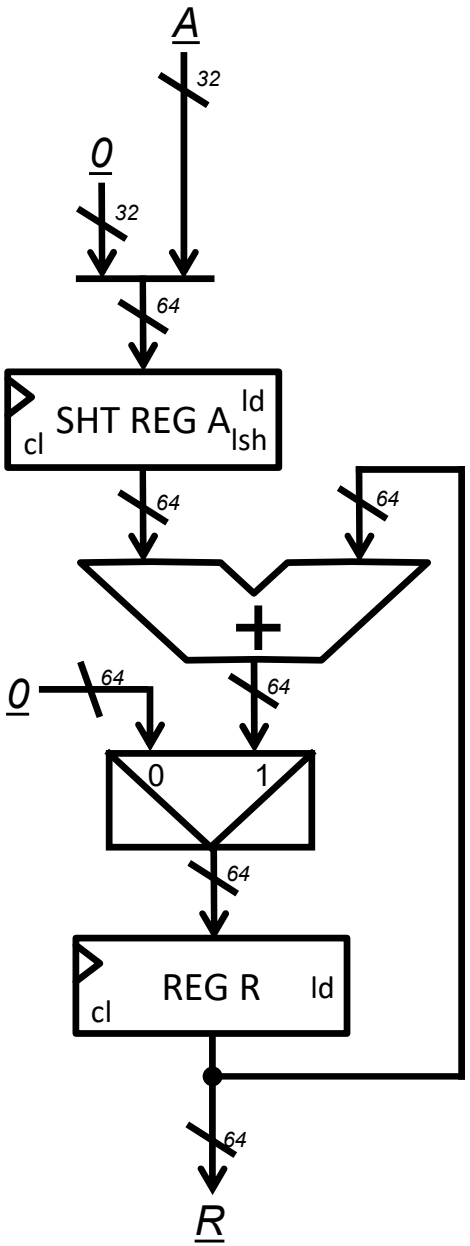

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<32; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

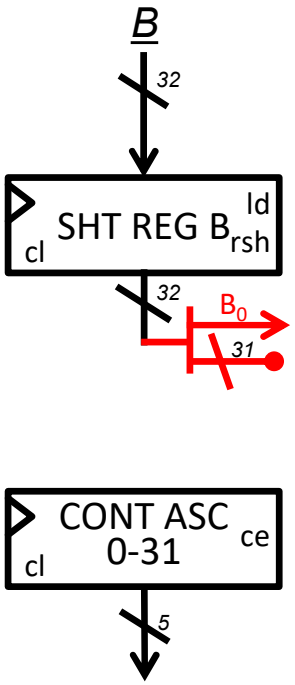
Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

Multiplicador iterativo

Chequeo de condición





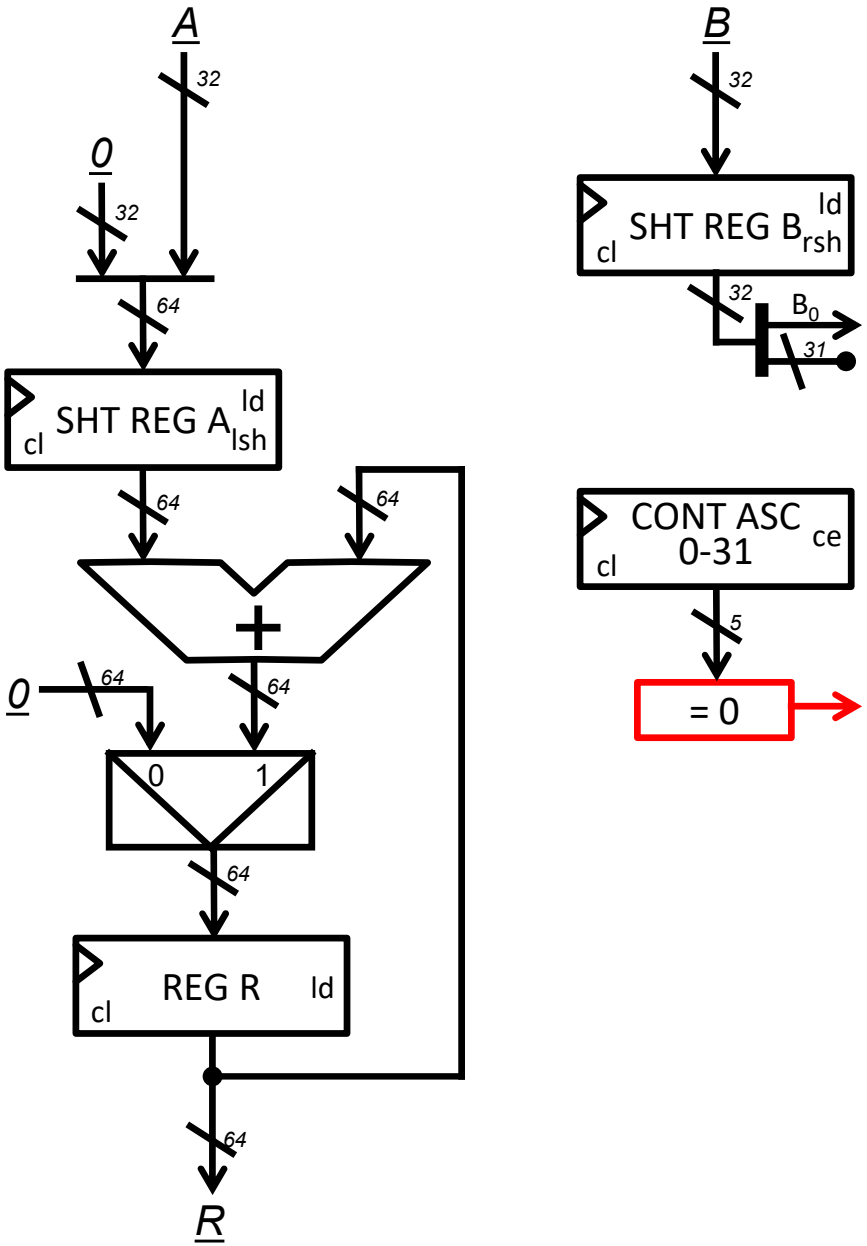

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     for( C=0; C<32; C++ )
     {
4.     if( B0==1 )
         R = R+A;
5.     A = A << 1;
6.     B = B >> 1;
     };
7.   Rout = R;
```

Algoritmo para
operandos de 32 bits

3. Diseño de la ruta de datos

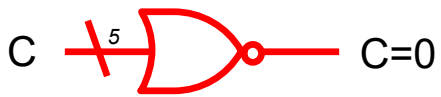
Multiplicador iterativo

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;
```

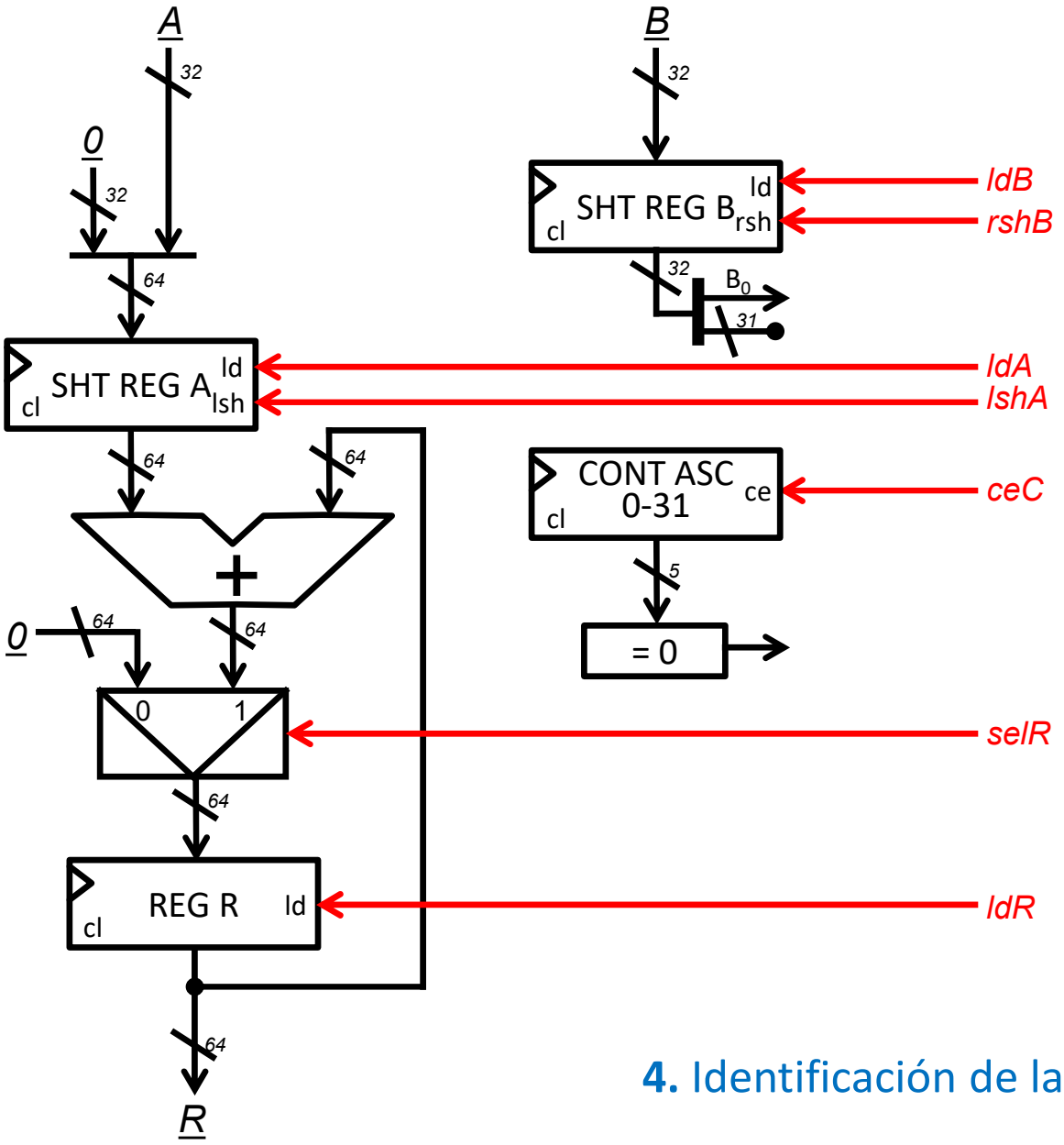
Algoritmo para
operandos de 32 bits



3. Diseño de la ruta de datos

Multiplicador iterativo

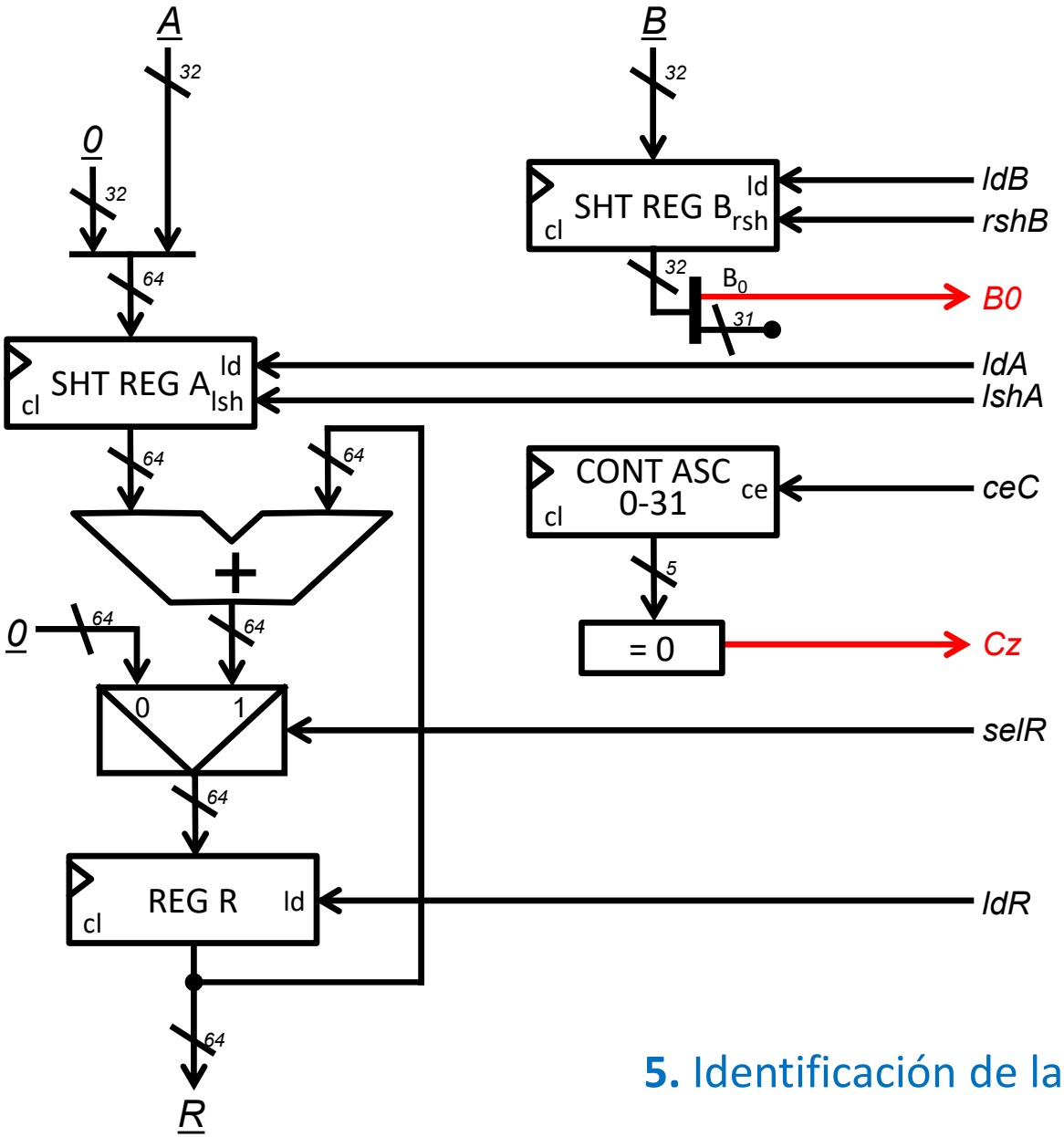
Señales de control



4. Identificación de las señales de control

Multiplicador iterativo

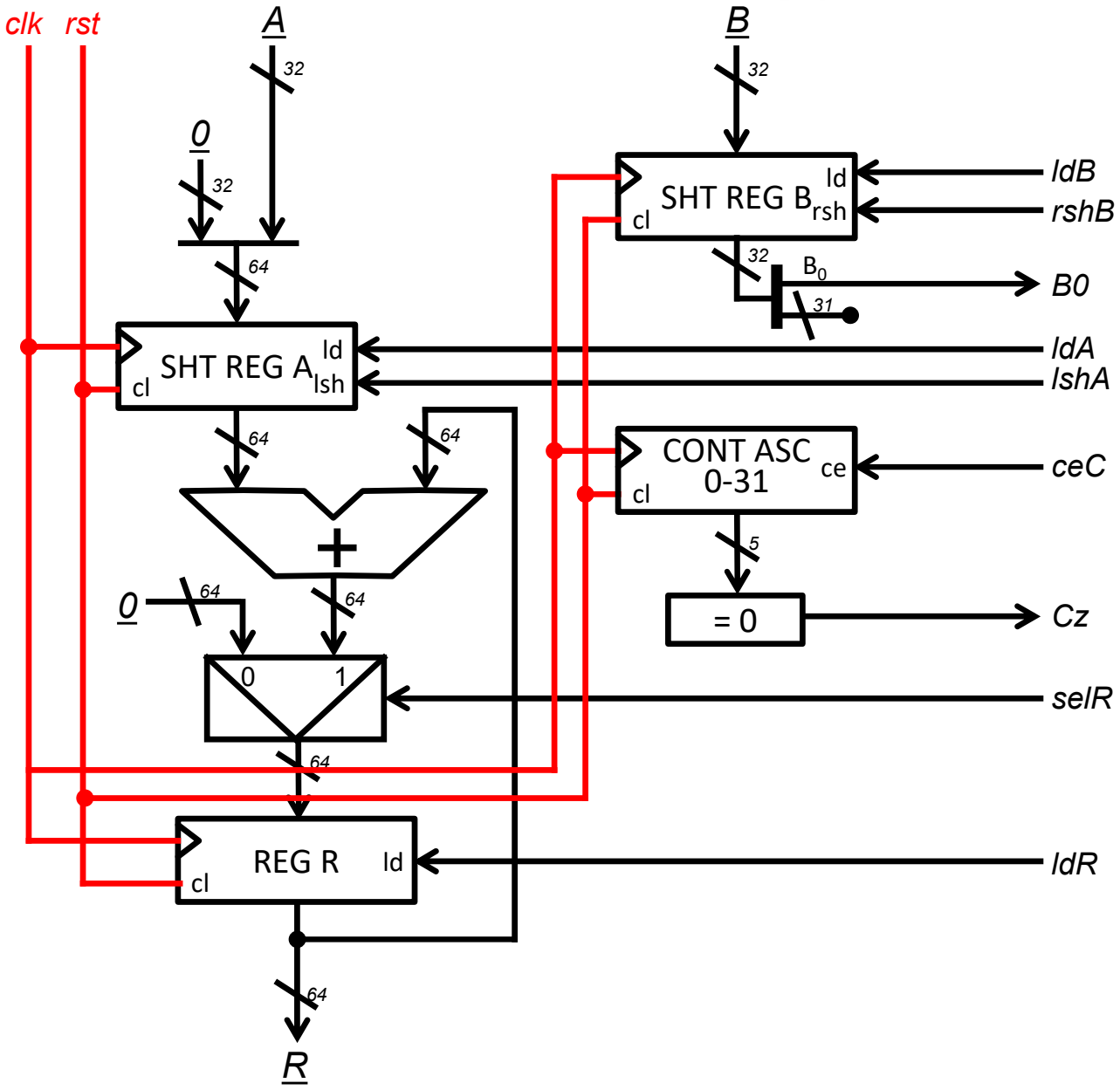
Señales de estado



5. Identificación de las señales de estado

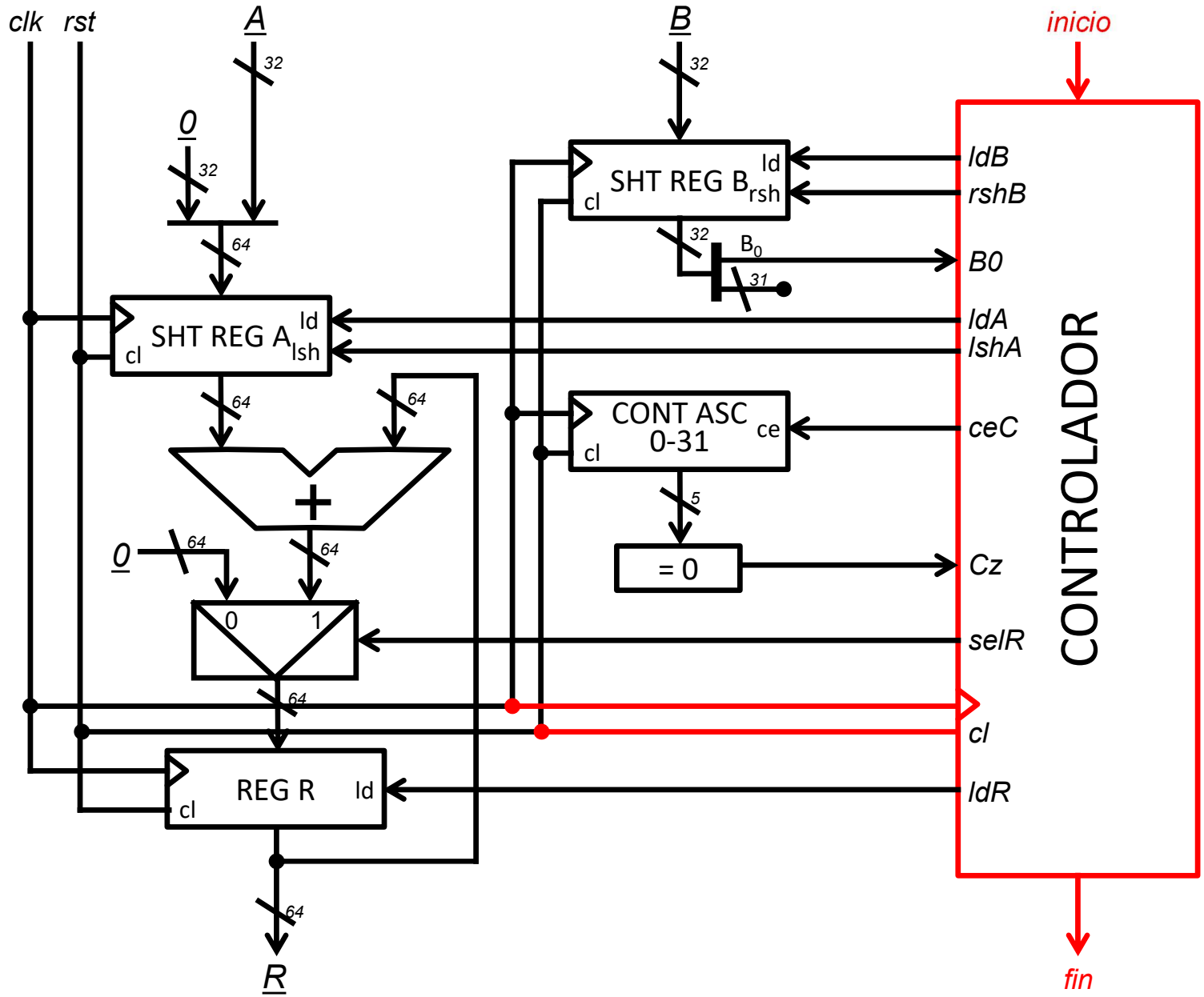
Multiplicador iterativo

Conexión del reloj y reset



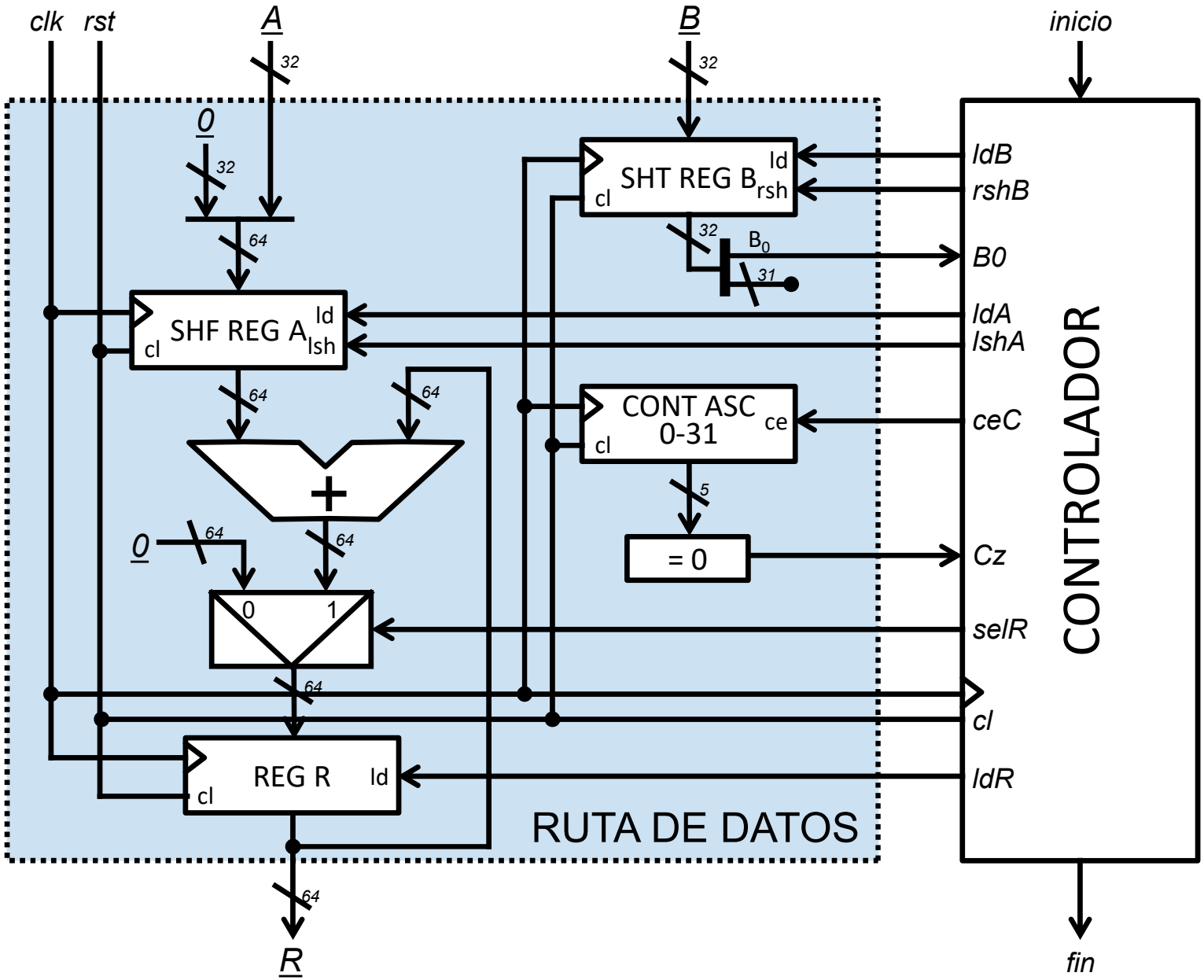
Multiplicador iterativo

Conexión con el controlador



Multiplicador iterativo

Estructura del sistema completo

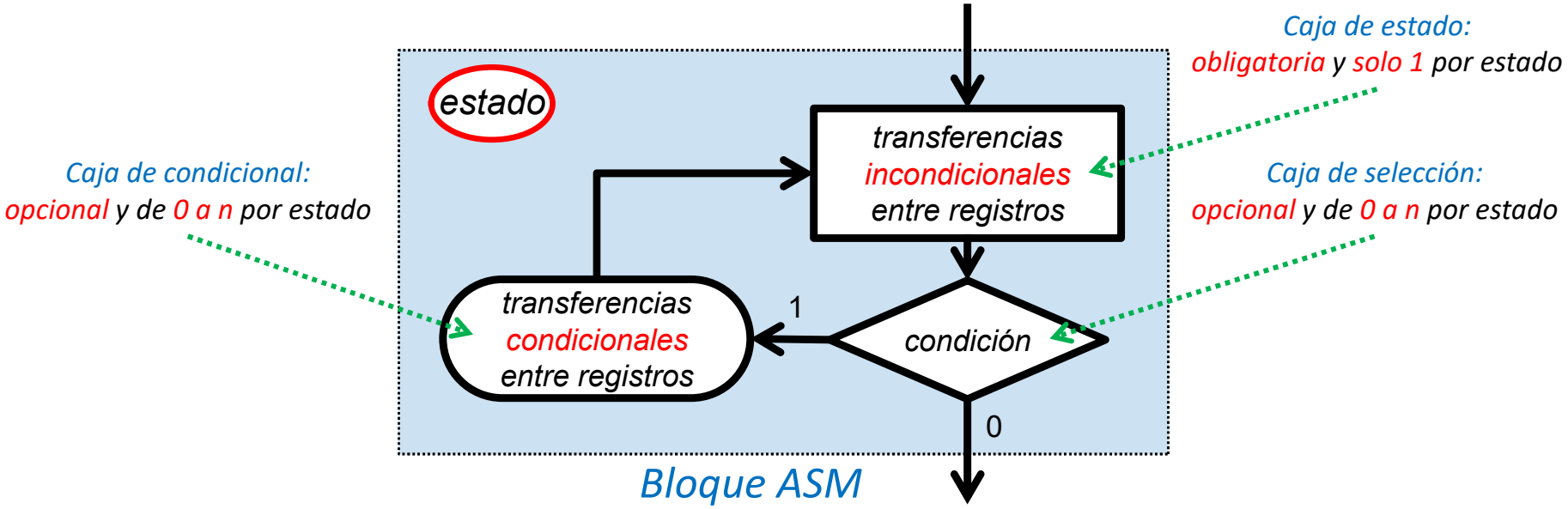


Multiplicador iterativo

Formalismo ASM (Algorithmic State Maquine)

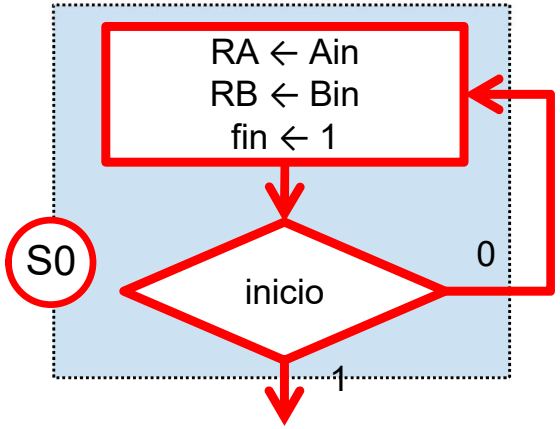
- Para facilitar la expresión del algoritmo como una FSM usaremos un diagrama ASM como formalismo intermedio.

- Un diagrama ASM es un grupo de bloques interconectados.
  - Cada bloque se corresponde con un estado de la FSM y usando cajas indica todas las acciones que se realizan en paralelo en dicho estado y todos los estados a los que puede saltar.



Multiplicador iterativo

Algoritmo como ASM

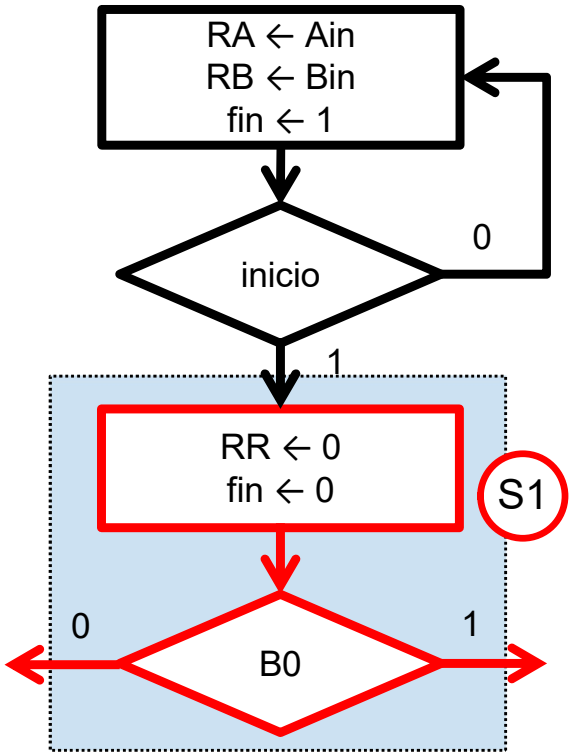



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM

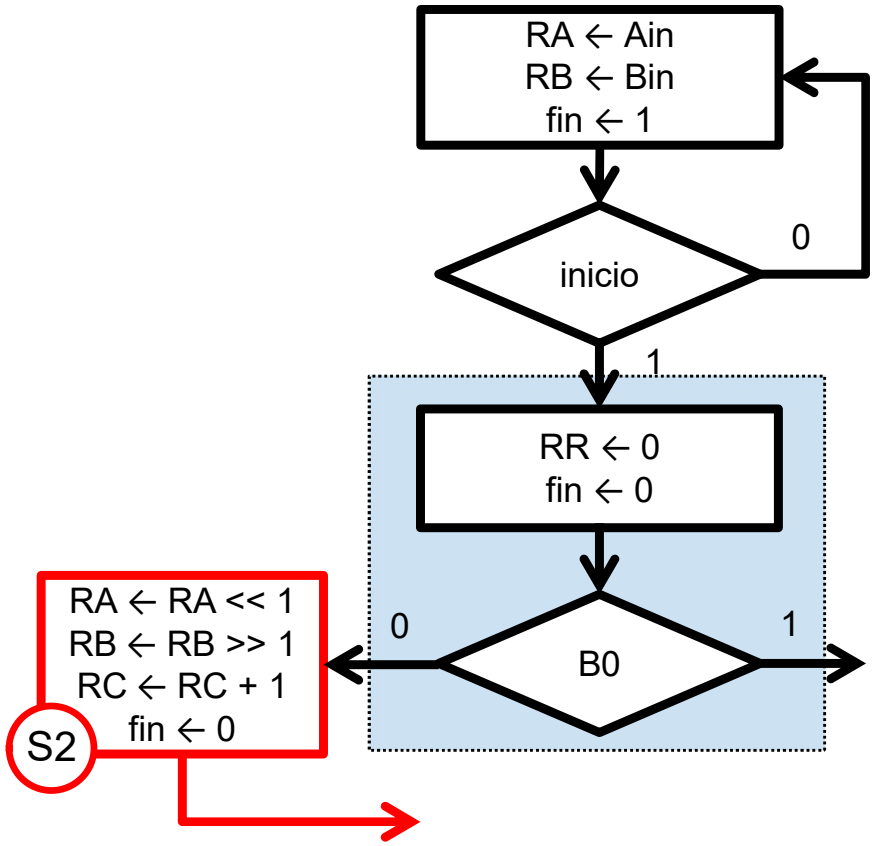



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM

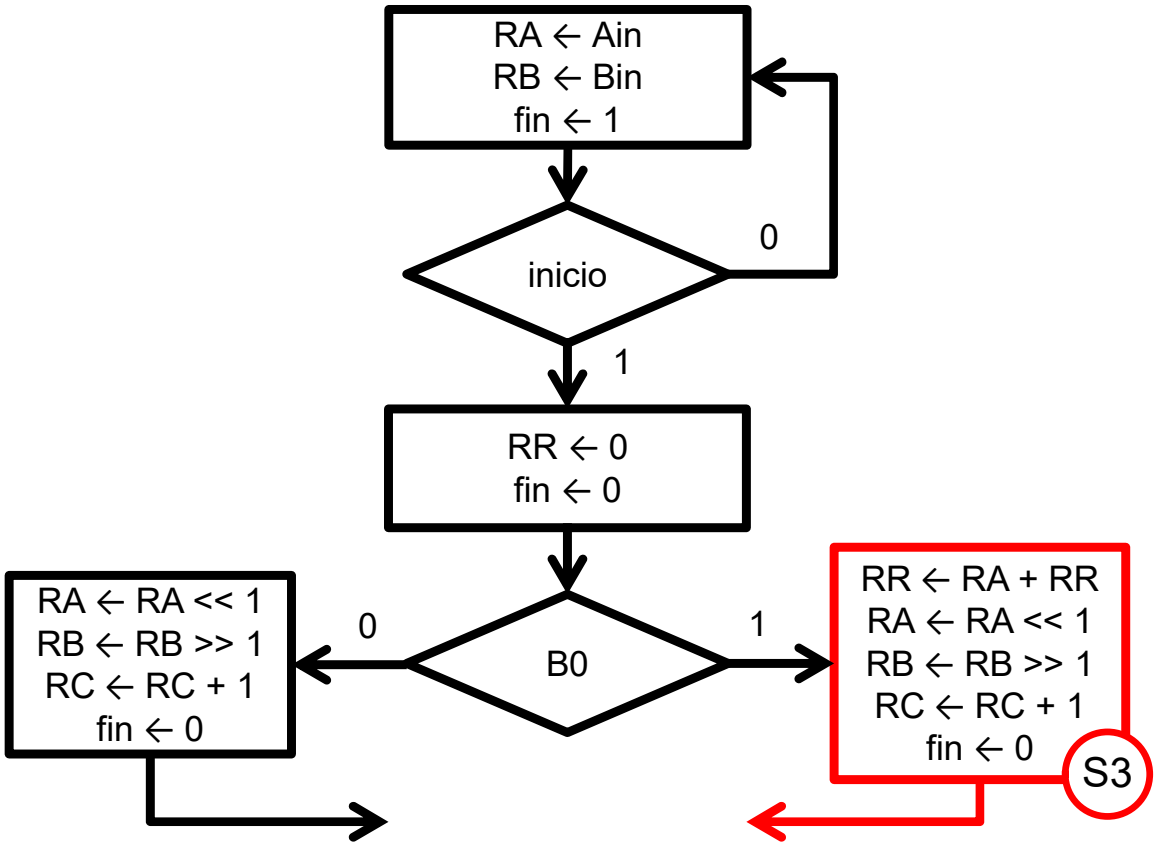



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM

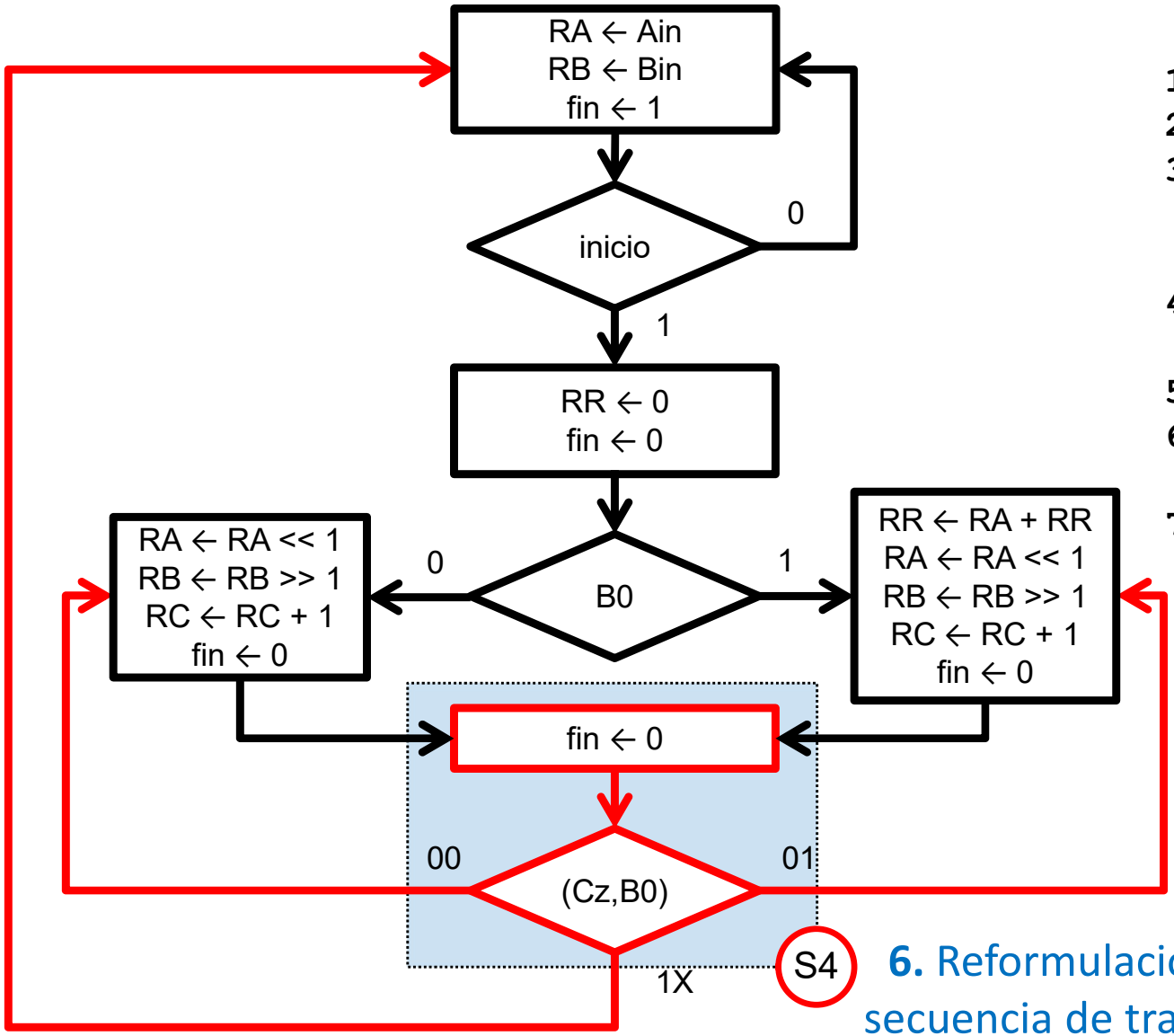



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM

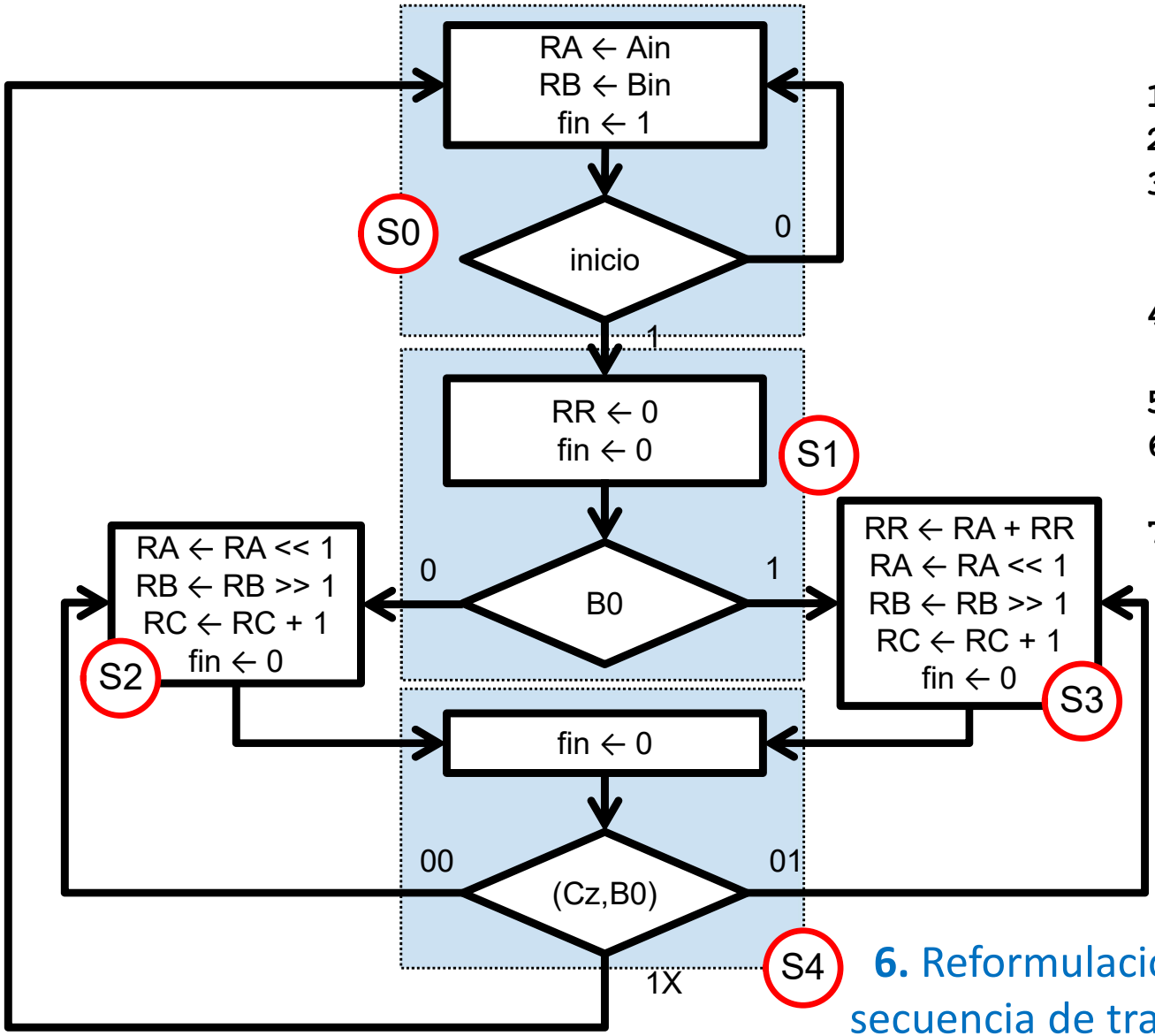



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM

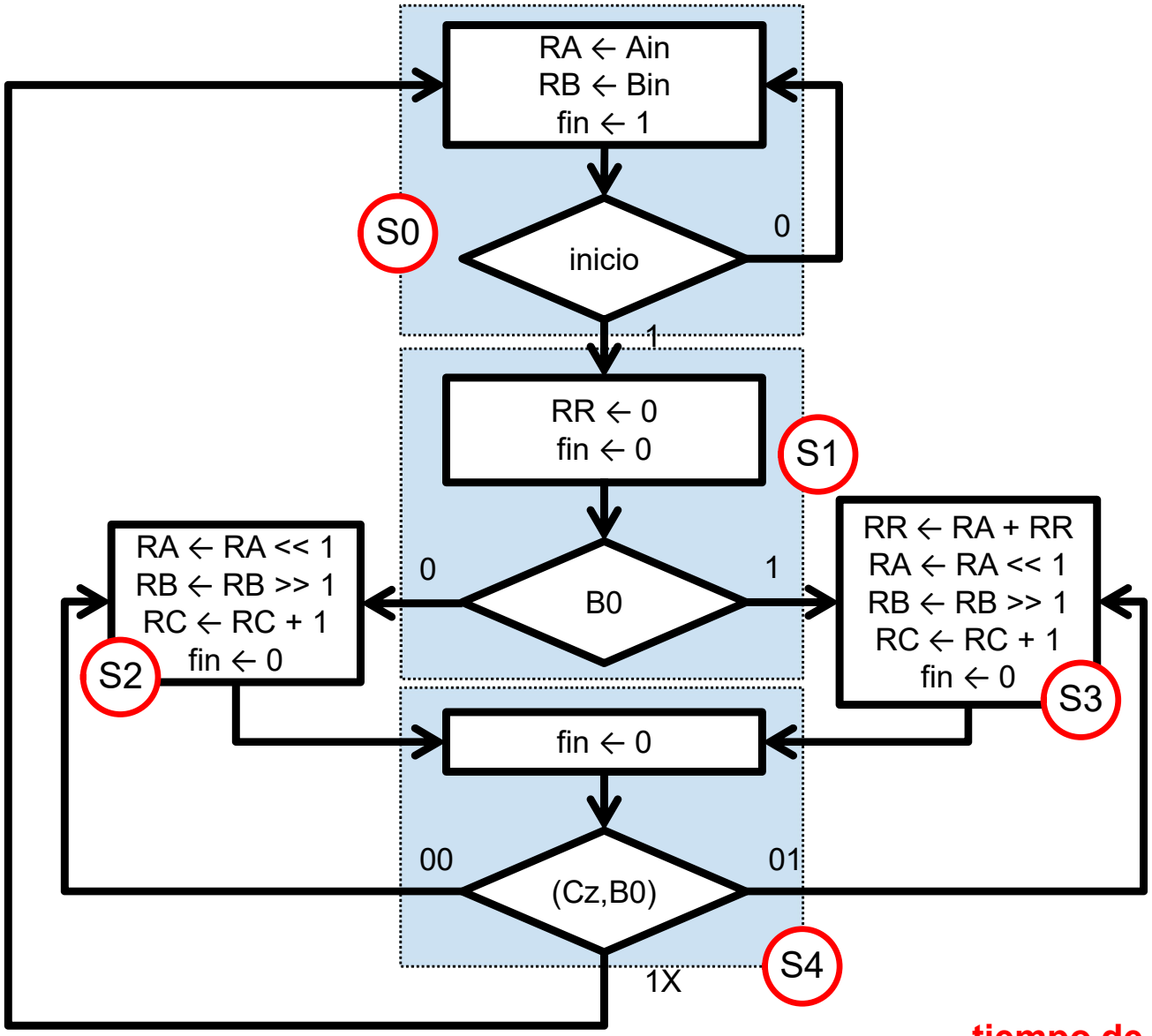



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    for( C=0; C<32; C++ )
    {
4.    if( B0==1 )
        R = R+A;
5.    A = A << 1;
6.    B = B >> 1;
    };
7.  Rout = R;
```

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

Multiplicador iterativo

Algoritmo como ASM



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;

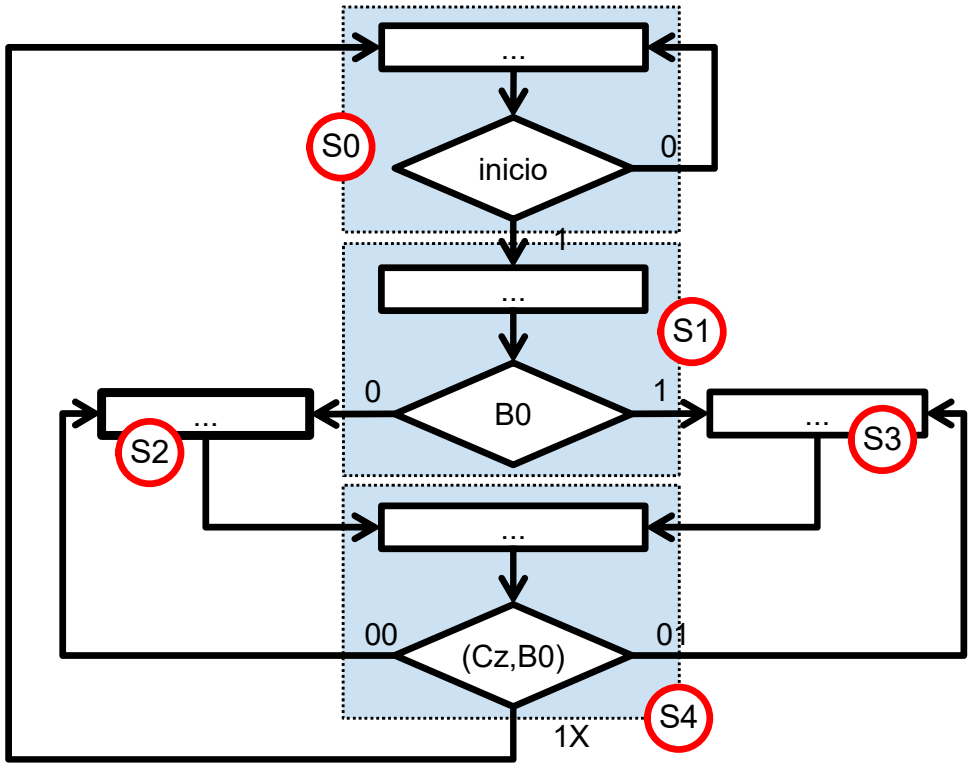
S4 fin ← 0;
si (Cz=1) ir a S0;
si (Cz=0 y B0=0) ir a S2;
si (Cz=0 y B0=1) ir a S3;

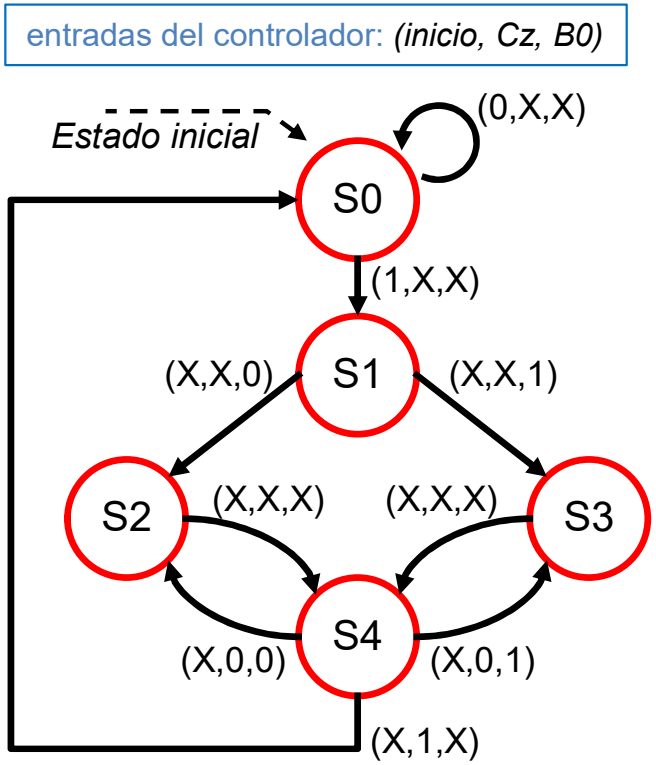
tiempo de cálculo: 2+32×2 = **66 ciclos**

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.



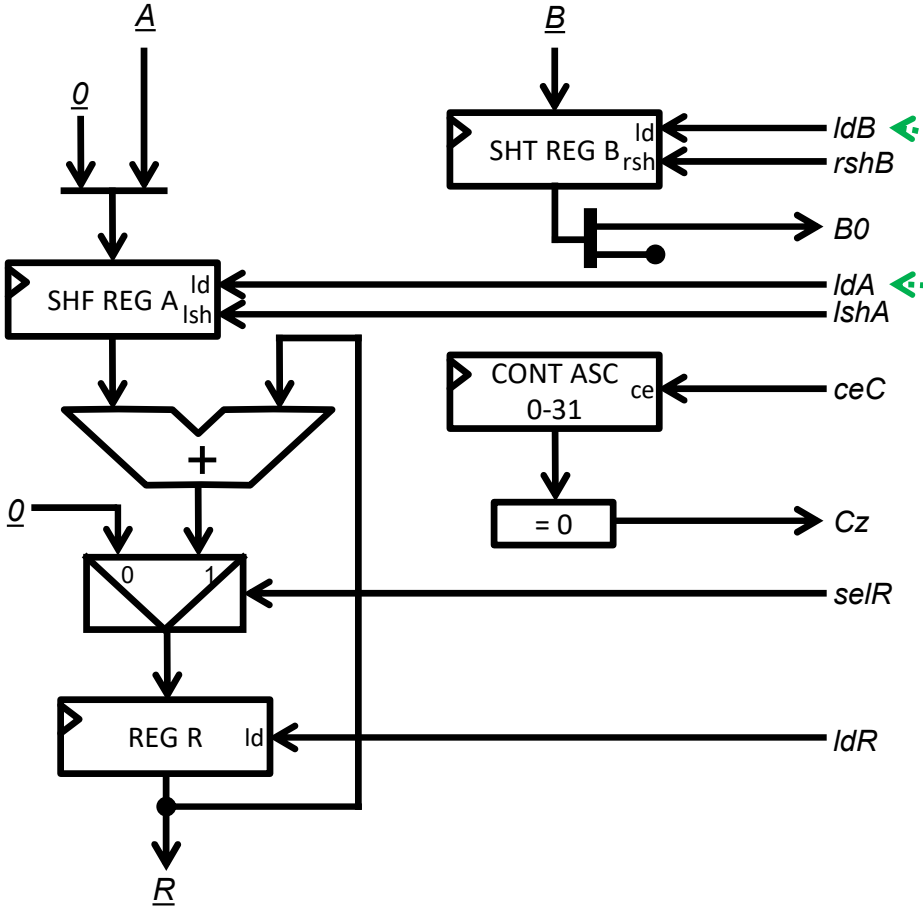


7. Especificación del controlador como FSM

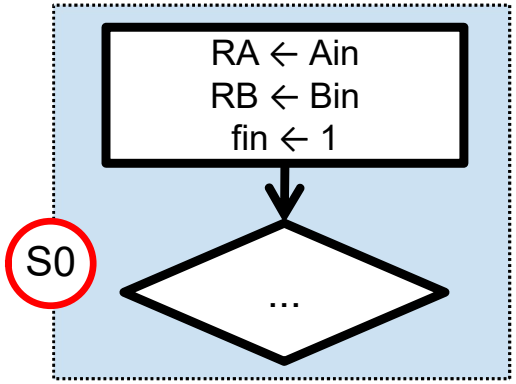
Multiplicador iterativo

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



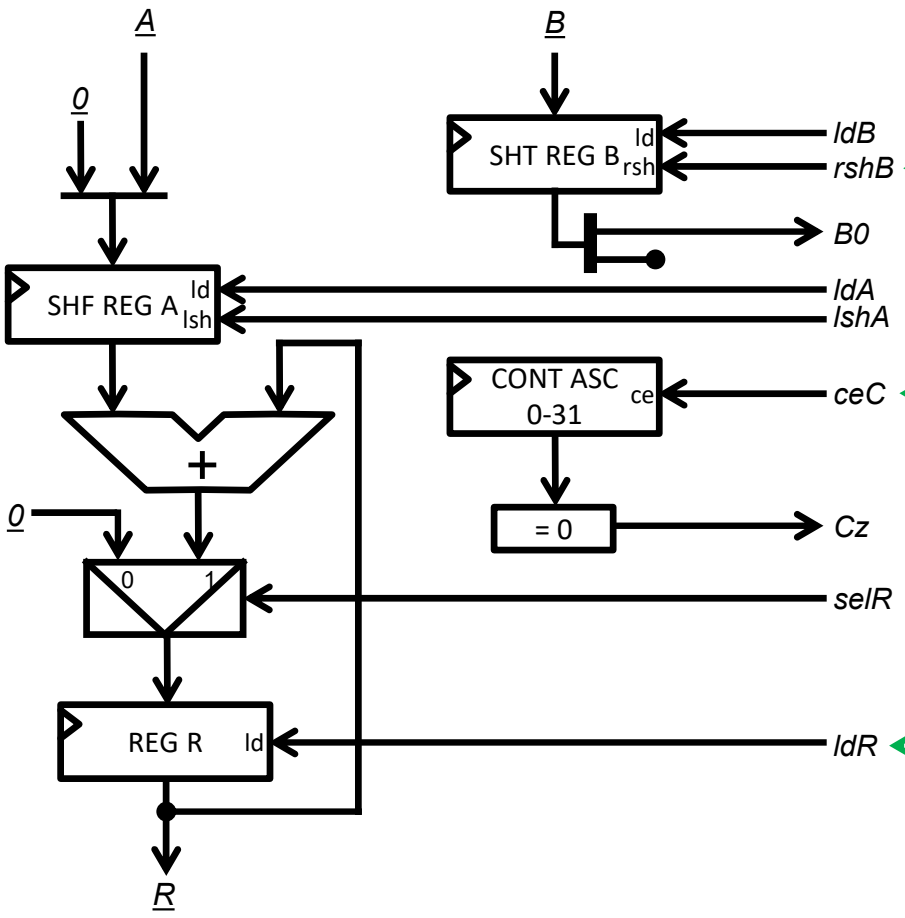
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1		1					1

7. Especificación del controlador como FSM

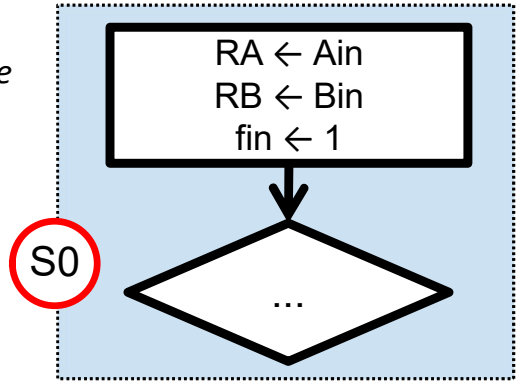
Multiplicador iterativo

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.



Resto de señales de control de elementos secuenciales a 0



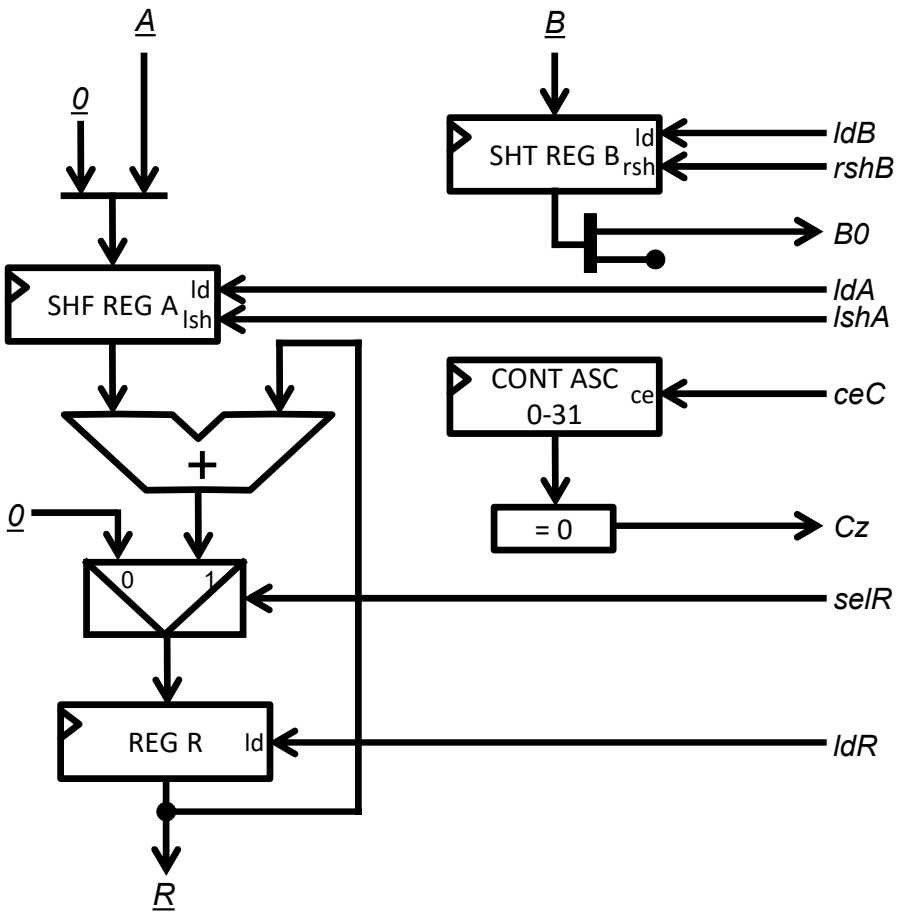
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0		0	1

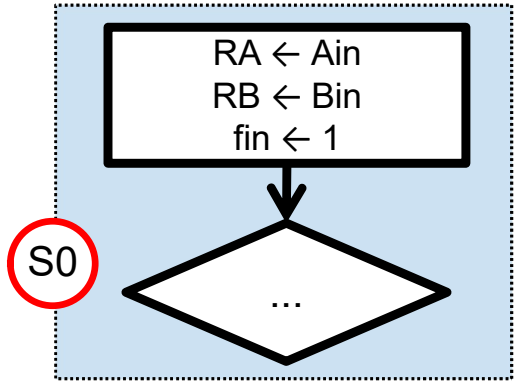
7. Especificación del controlador como FSM

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.





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

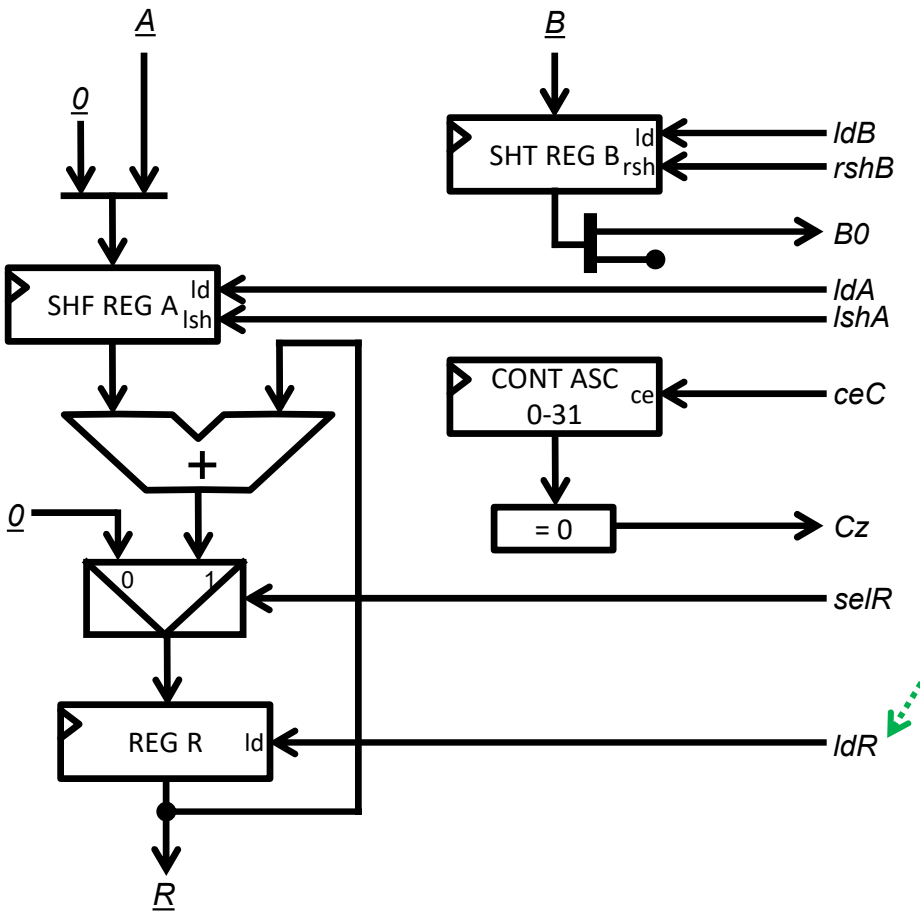
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1

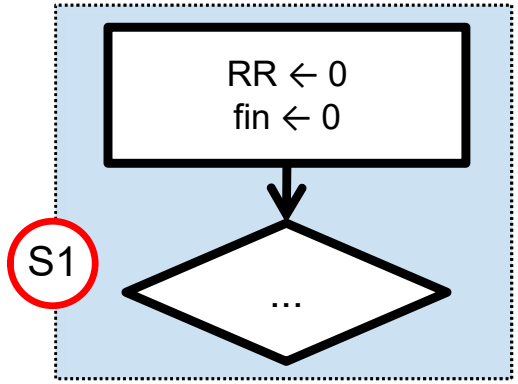
7. Especificación del controlador como FSM

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

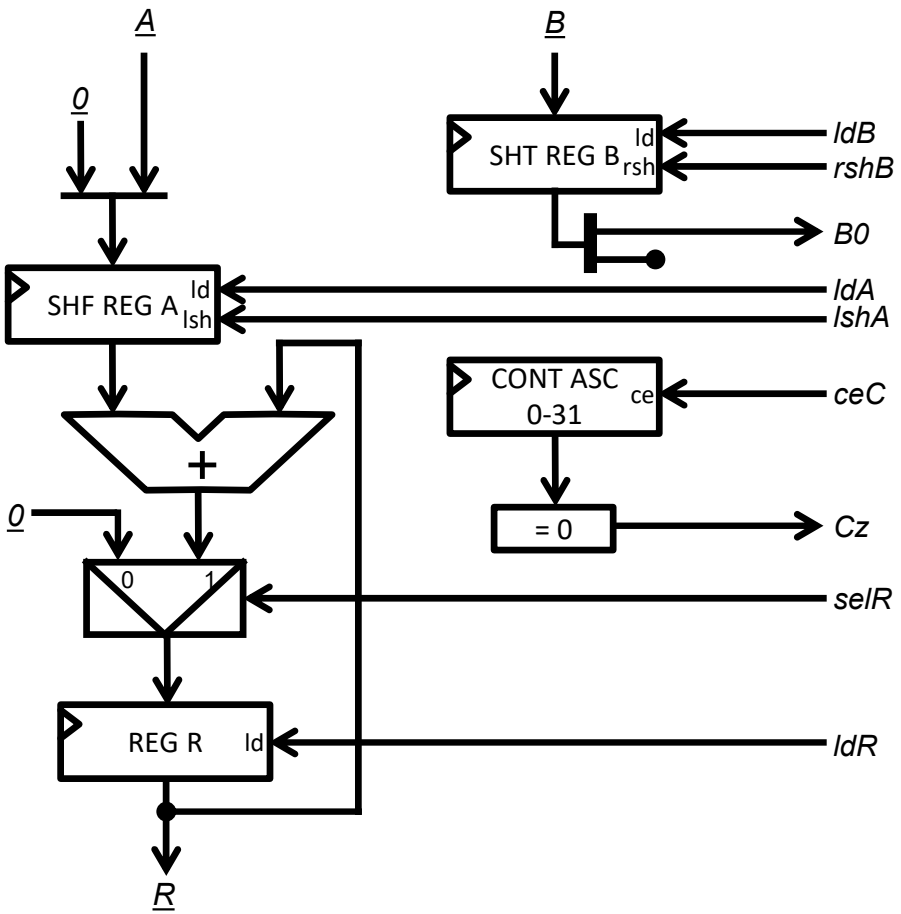
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1							1	0

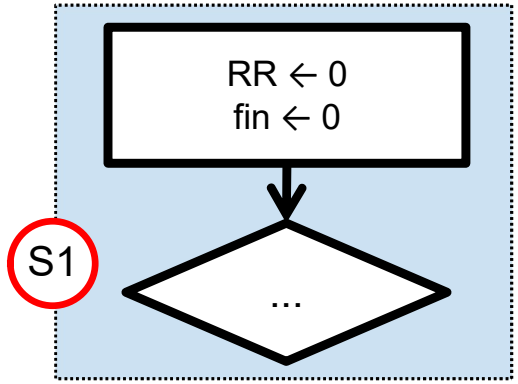
7. Especificación del controlador como FSM

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

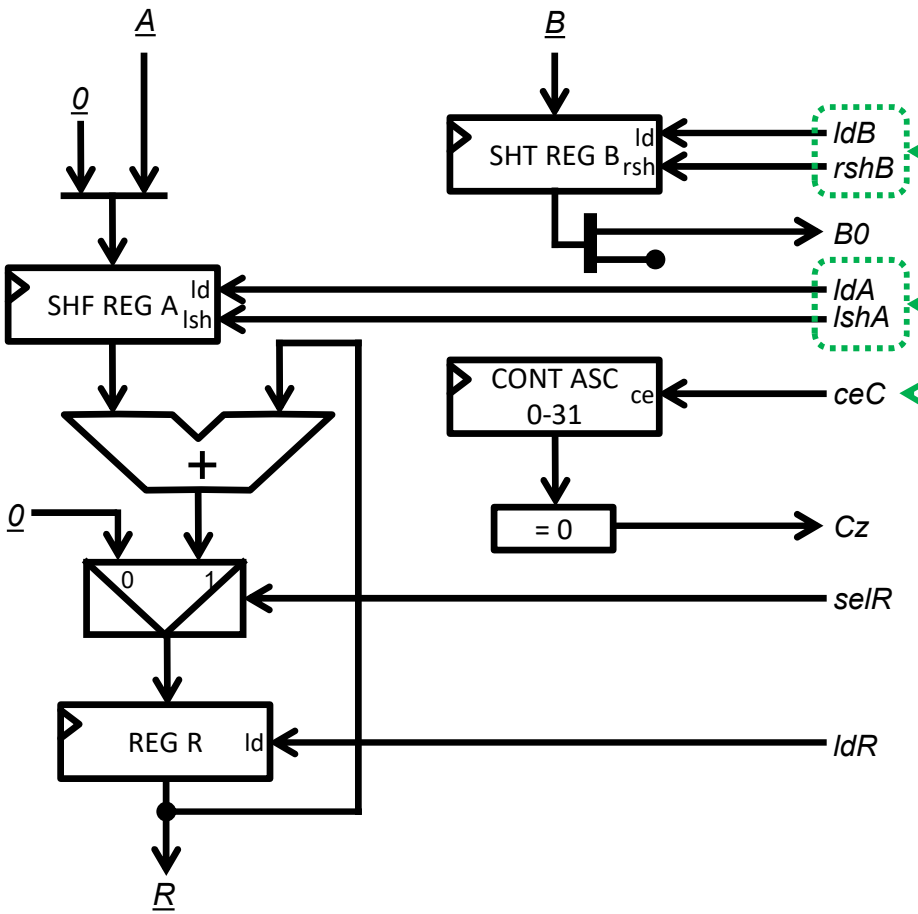
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1						0	1	0

7. Especificación del controlador como FSM

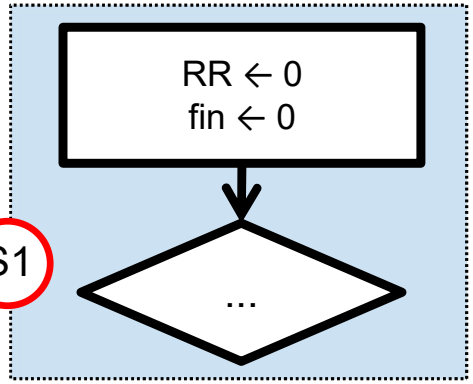
Multiplicador iterativo

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.



Resto de señales de control de elementos secuenciales a 0



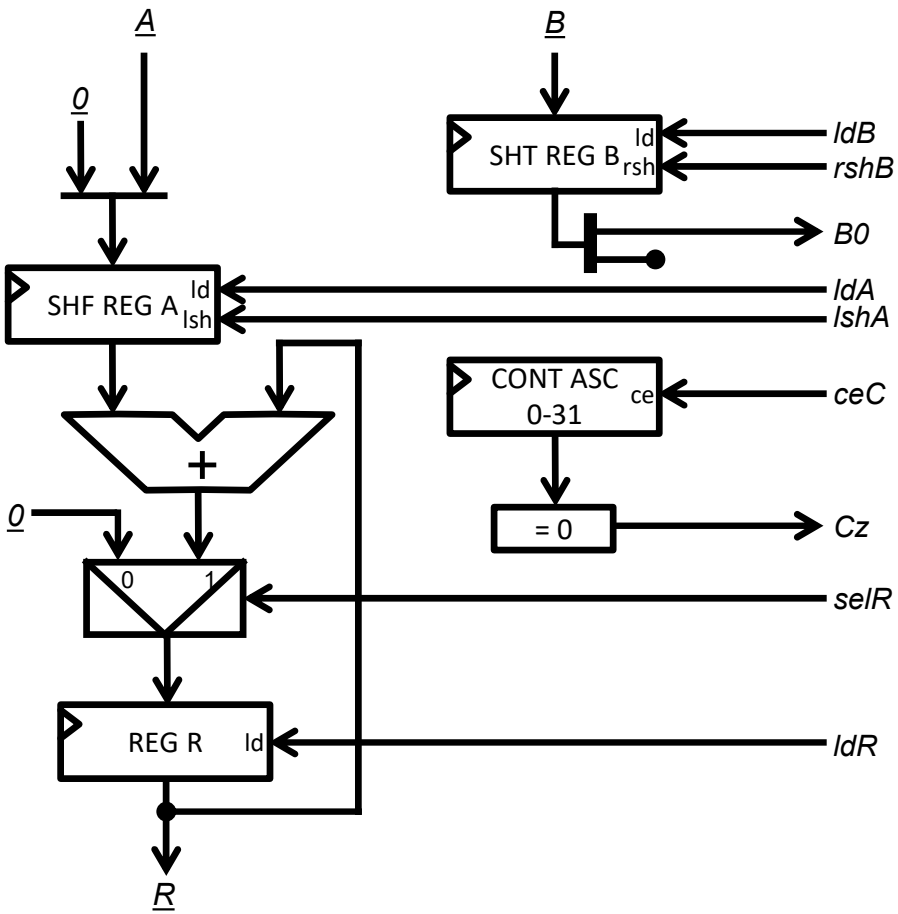
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0

7. Especificación del controlador como FSM

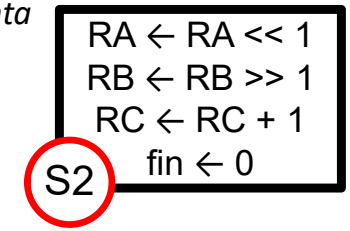
Multiplicador iterativo

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 desplazamiento / cuenta de registros destino a 1



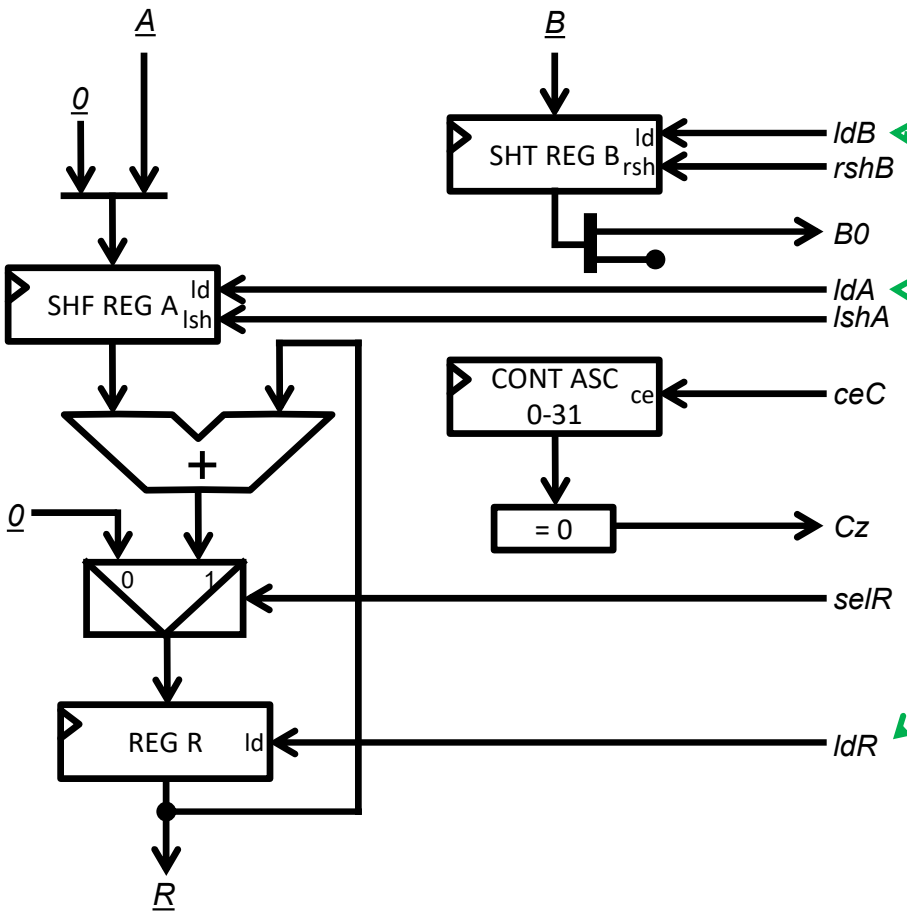
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2		1		1	1			0

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.



Resto de señales de carga de registros a 0

RA ← RA << 1
RB ← RB >> 1
RC ← RC + 1
fin ← 0

S2

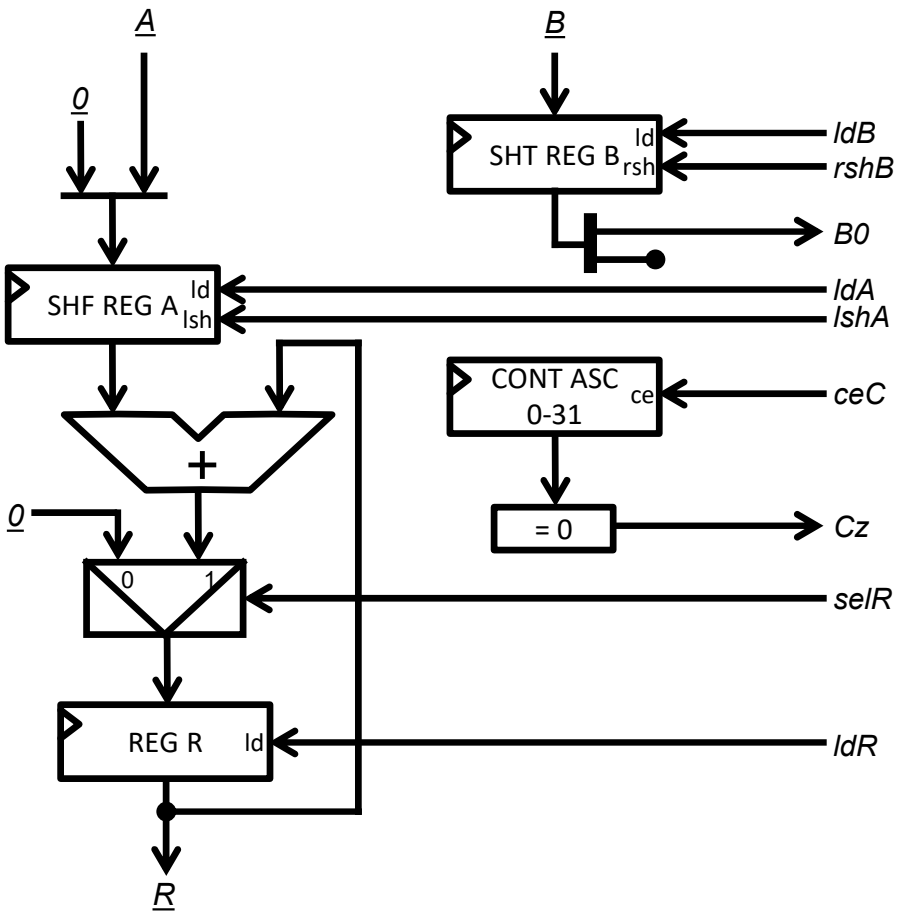
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1		0	0

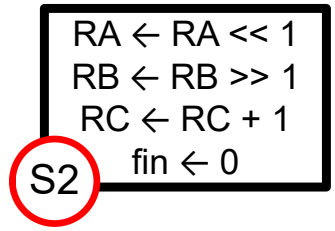
7. Especificación del controlador como FSM

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.





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

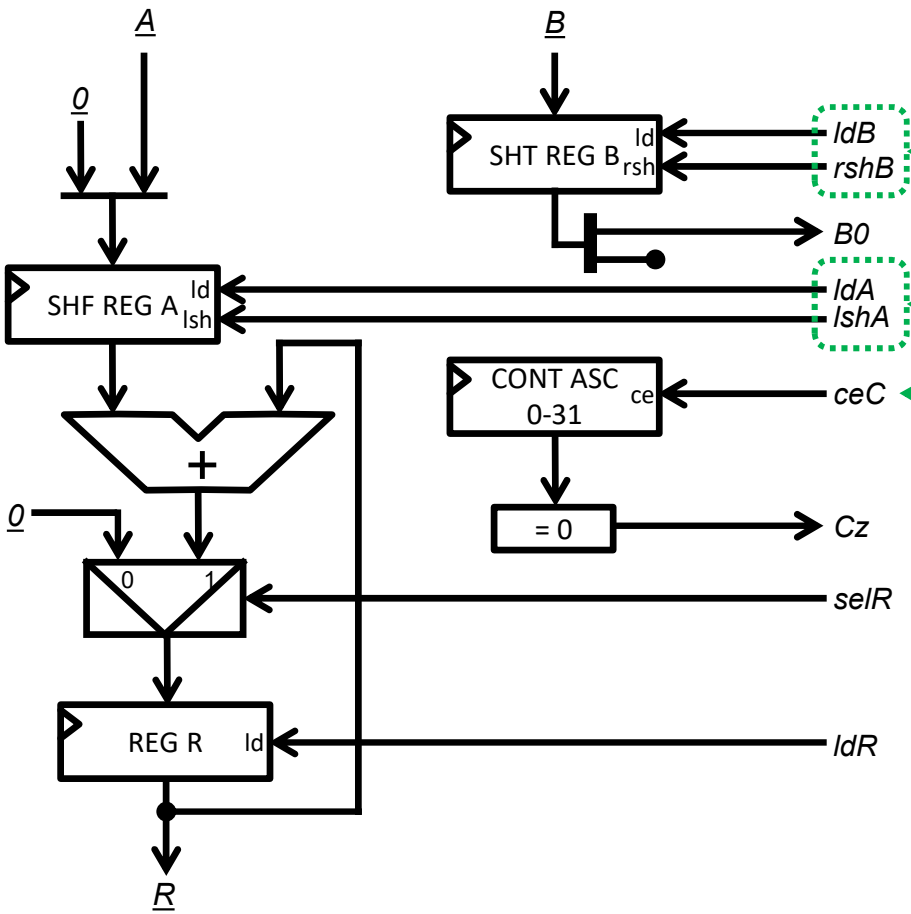
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	–	0	0

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.



Idénticas transferencias, idénticos valores

RR ← RA + RR
RA ← RA << 1
RB ← RB >> 1
RC ← RC + 1
fin ← 0

S3

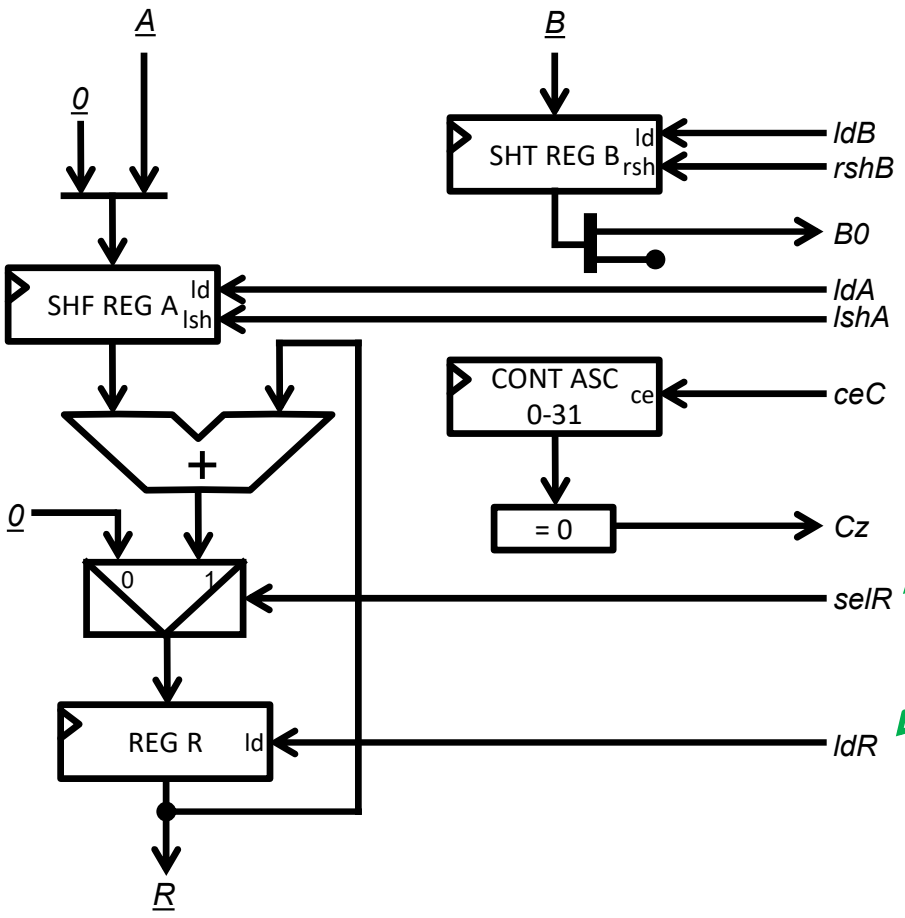
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	–	0	0
S3	0	1	0	1	1			0

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.



RR ← RA + RR
RA ← RA << 1
RB ← RB >> 1
RC ← RC + 1
fin ← 0
S3

R carga el valor calculado por el sumador

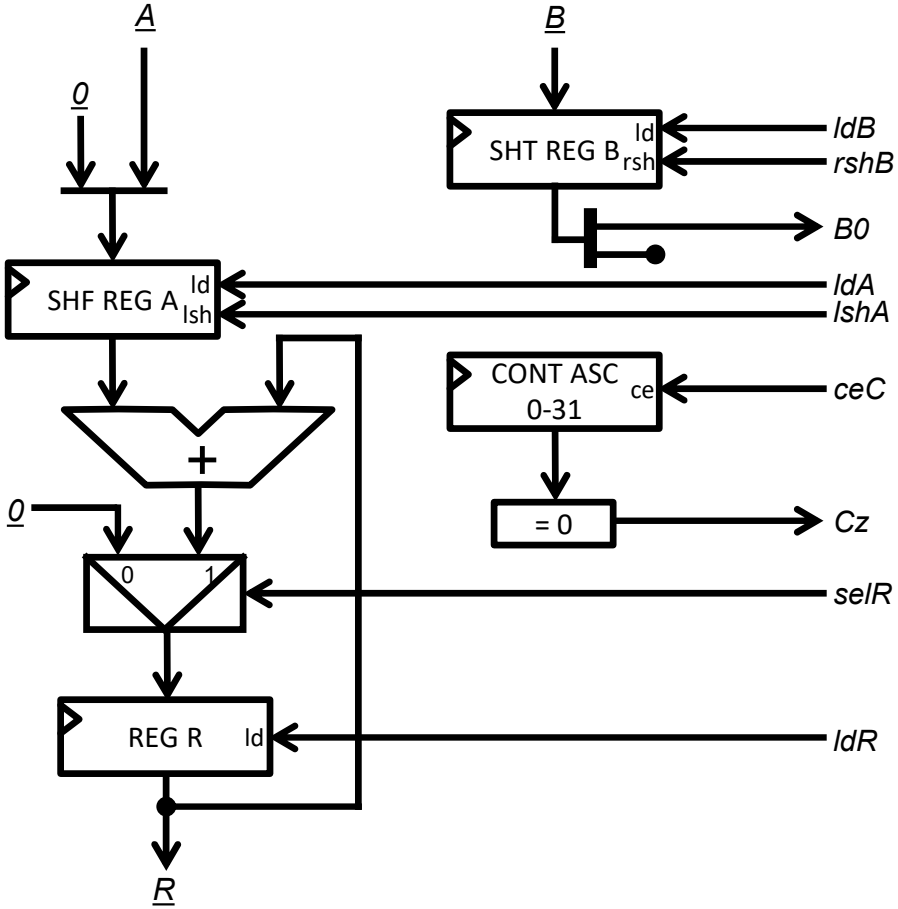
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	–	0	0
S3	0	1	0	1	1	1	1	0

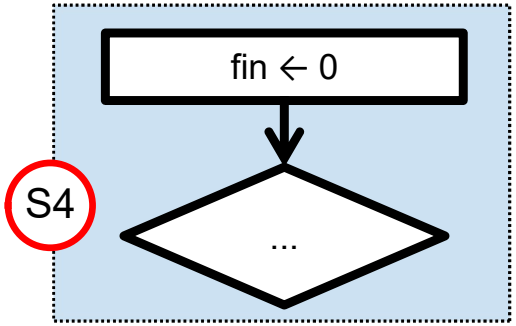
7. Especificación del controlador como FSM

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.





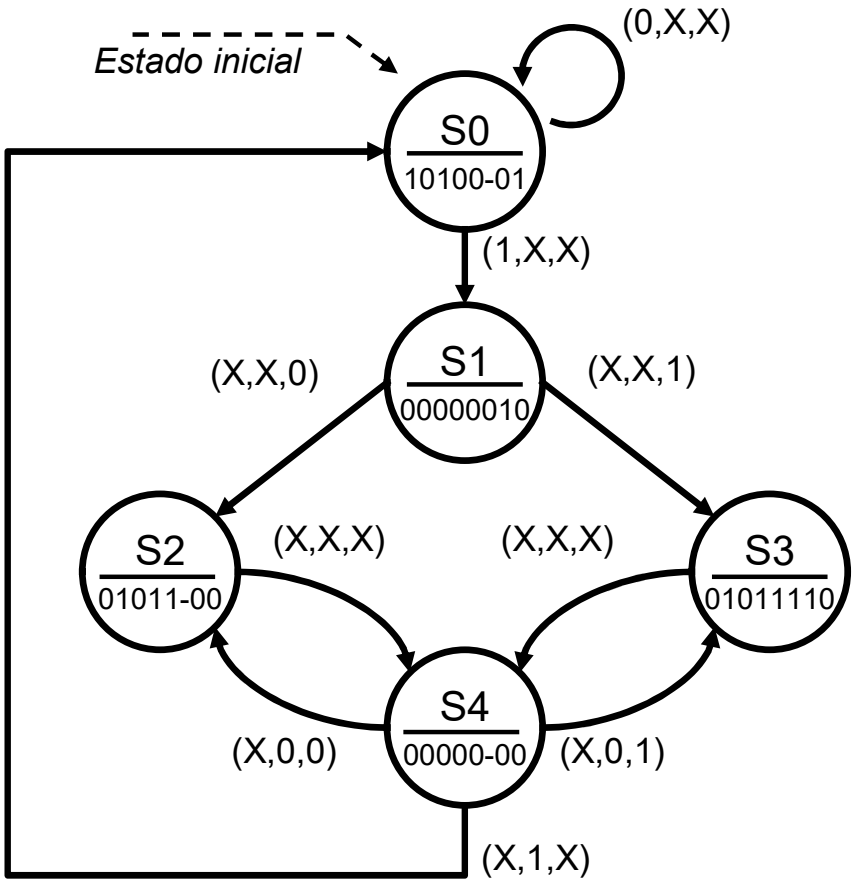
estado	ldB	rshB	ldA	lshA	ceC	selR	ldR	fin
S0	1	0	1	0	0	–	0	1
S1	0	0	0	0	0	0	1	0
S2	0	1	0	1	1	–	0	0
S3	0	1	0	1	1	1	1	0
S4	0	0	0	0	0	–	0	0

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

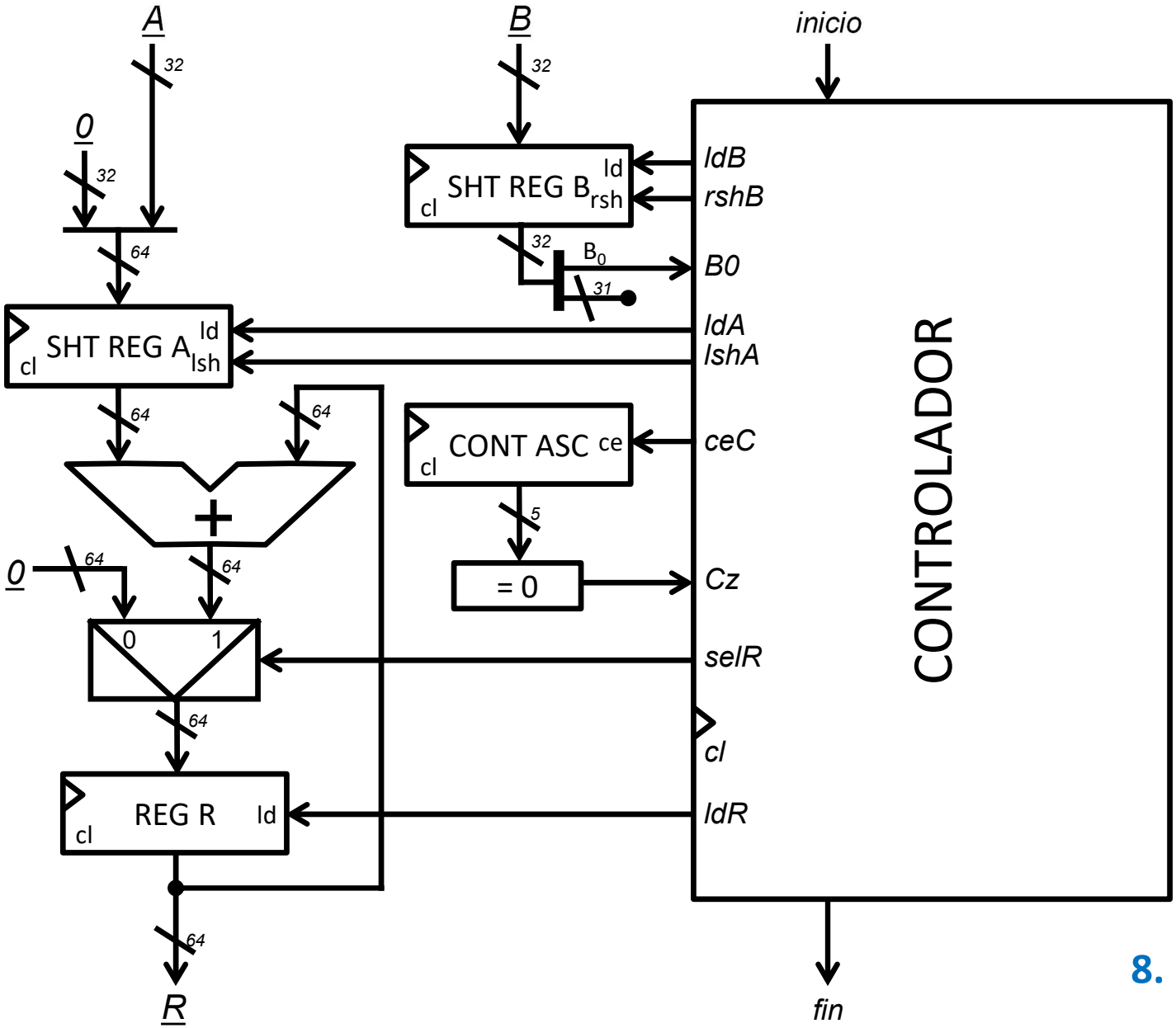
Función de salida

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

7. Especificación del controlador como FSM

Multiplicador iterativo

Diseño del controlador



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;

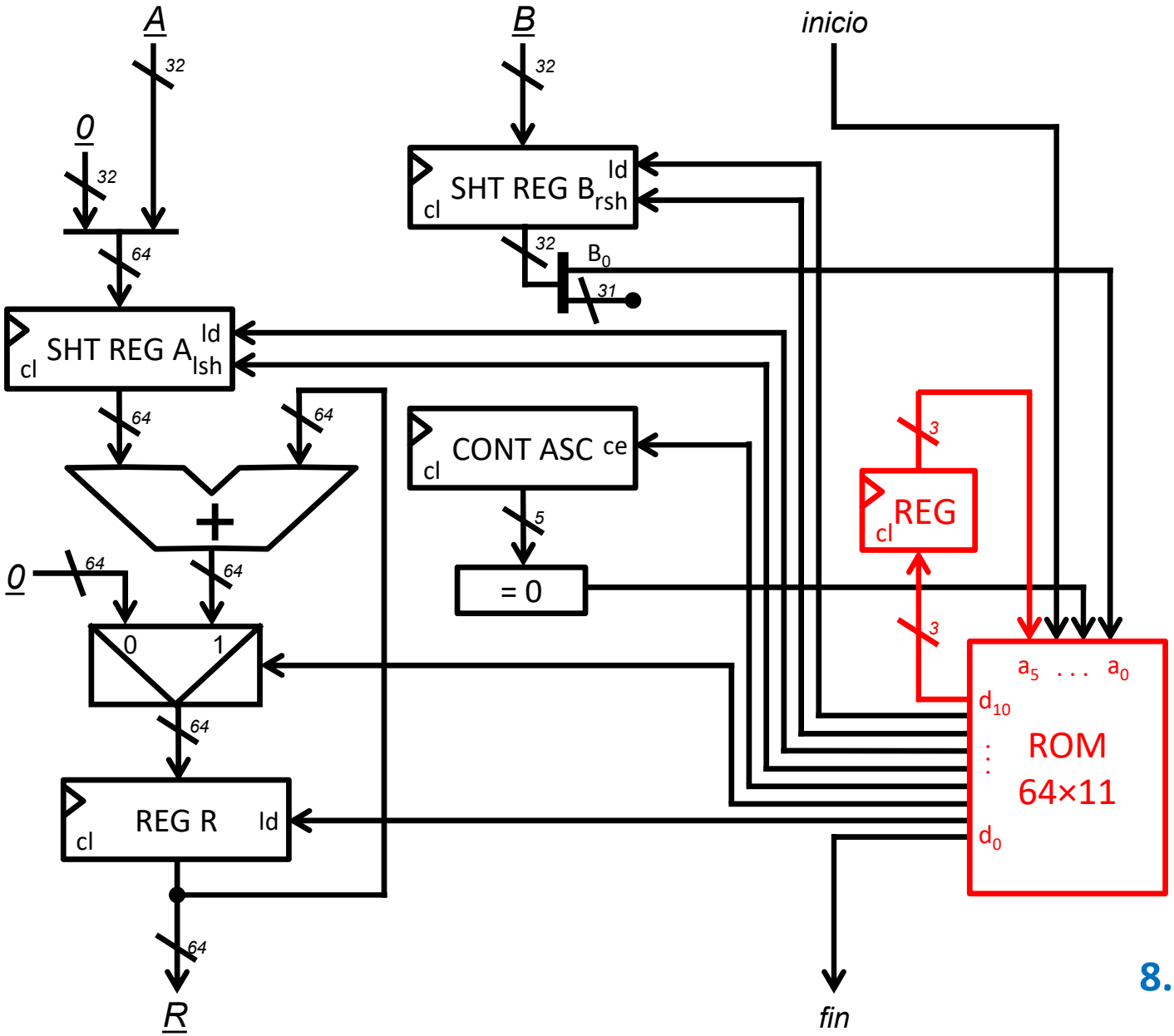
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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

ROM + registro de estado



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;

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											
5											
6											
7											
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

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

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

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	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											

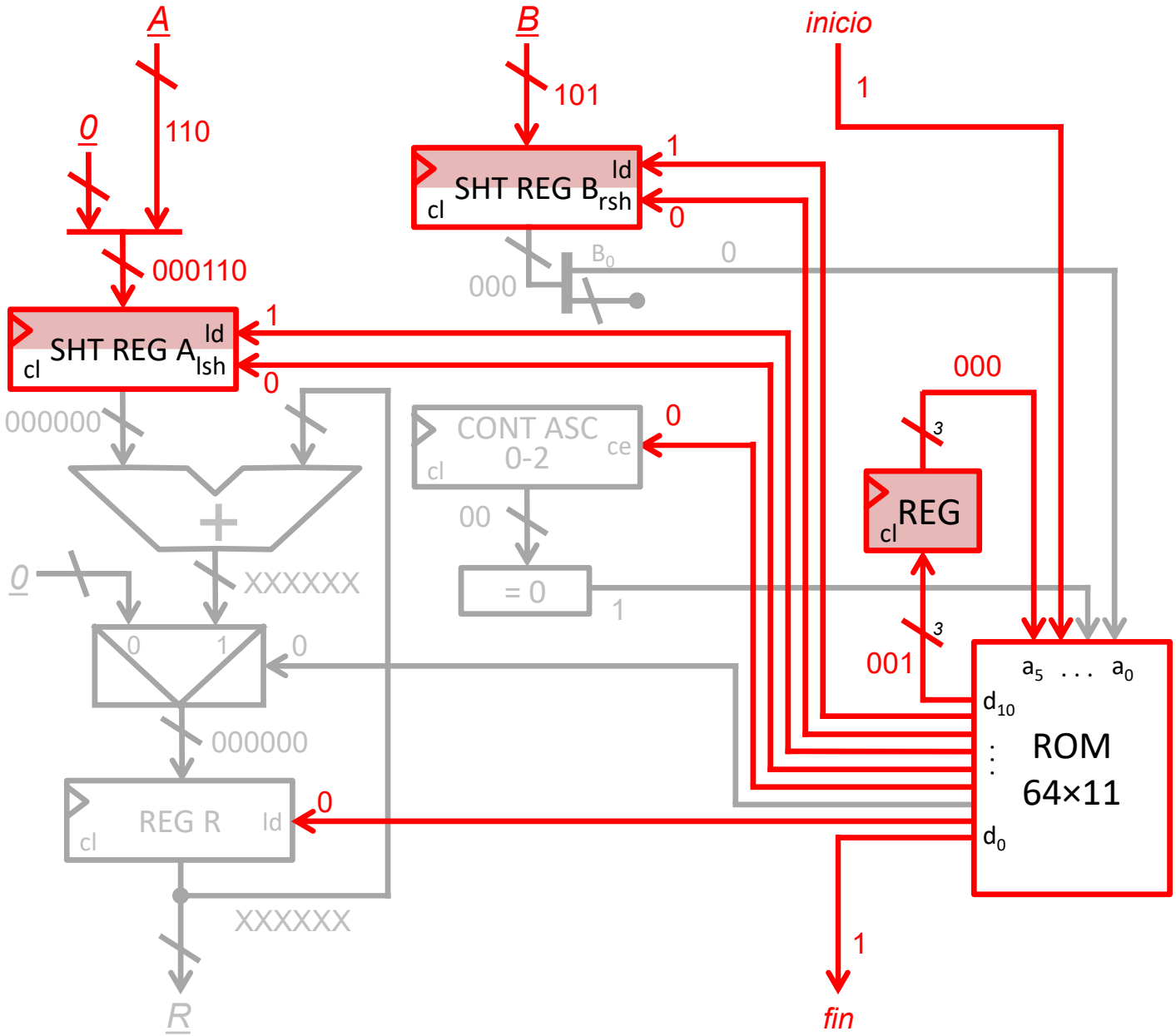
ROM 64×11

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

Multiplicador iterativo

Simulación del sistema (operandos de 3 bits)



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;

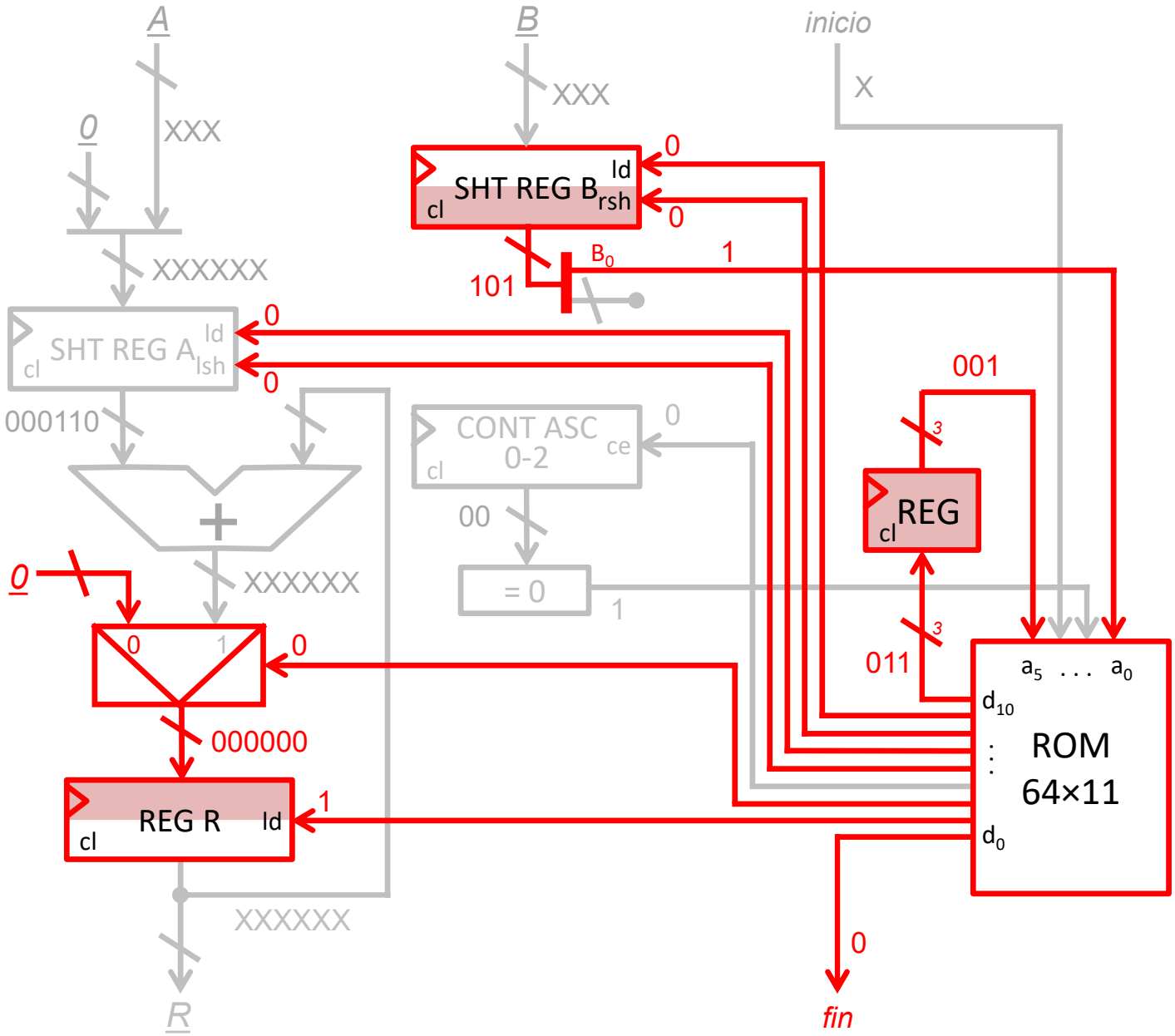
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

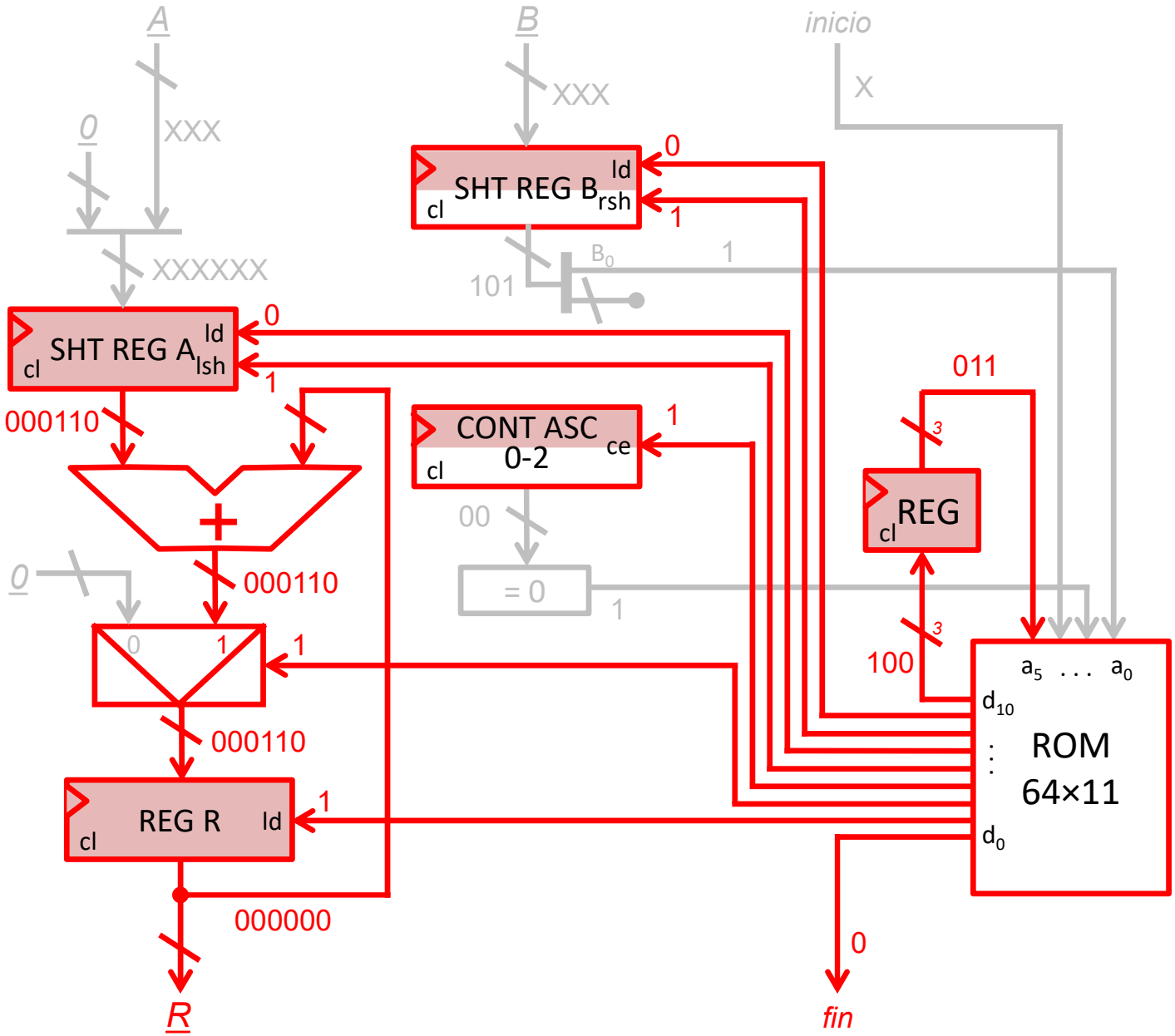
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

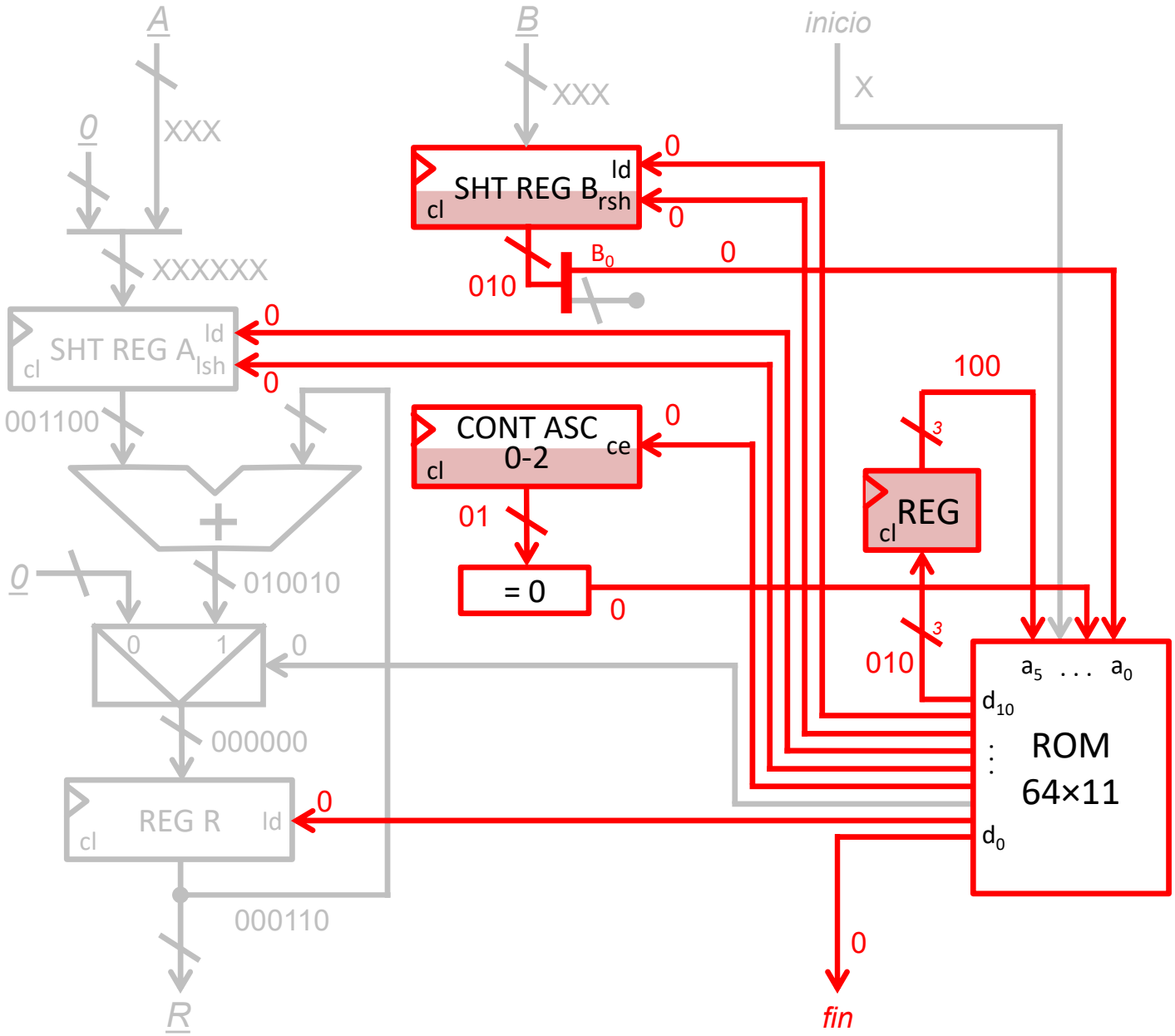
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

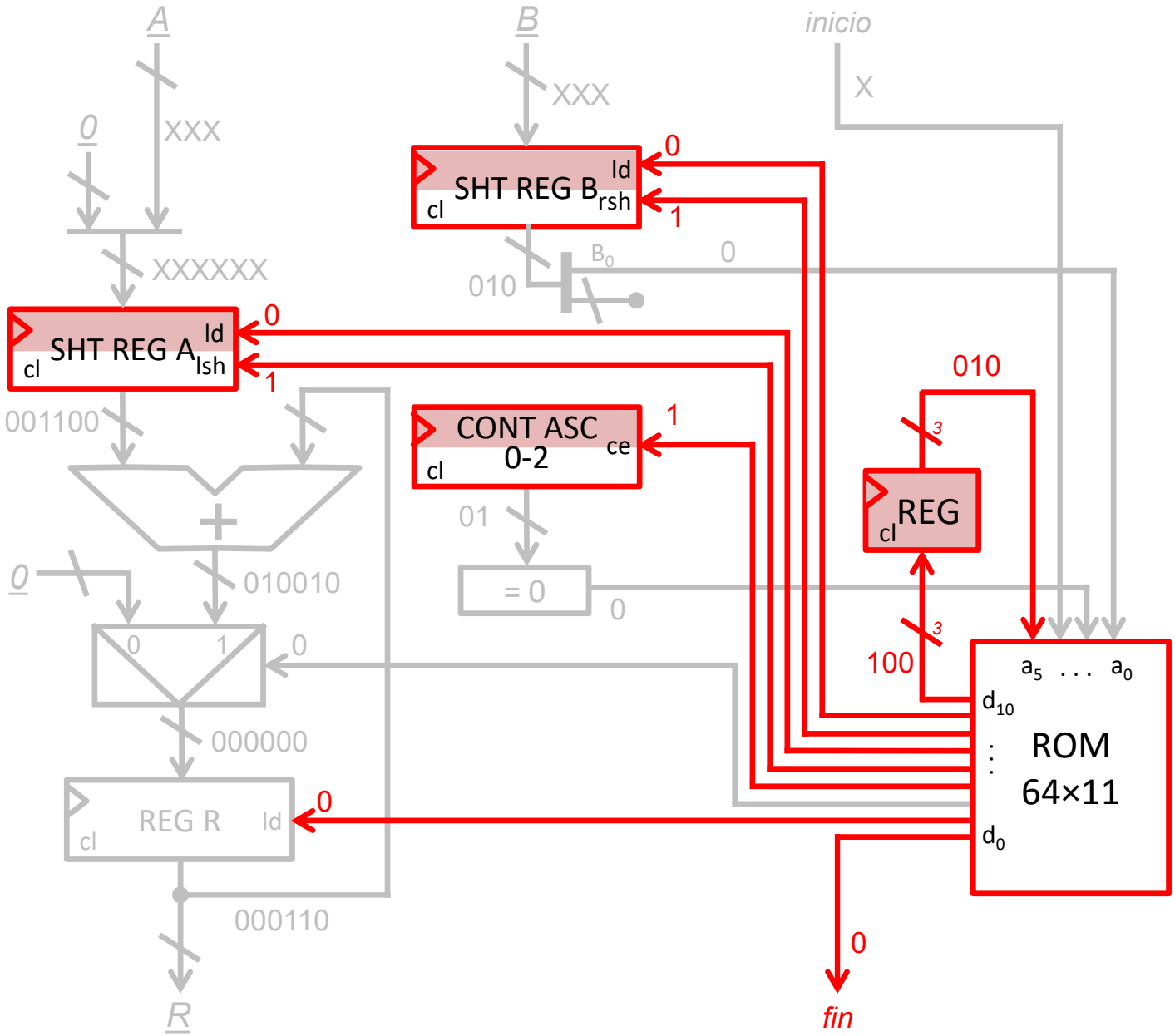
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

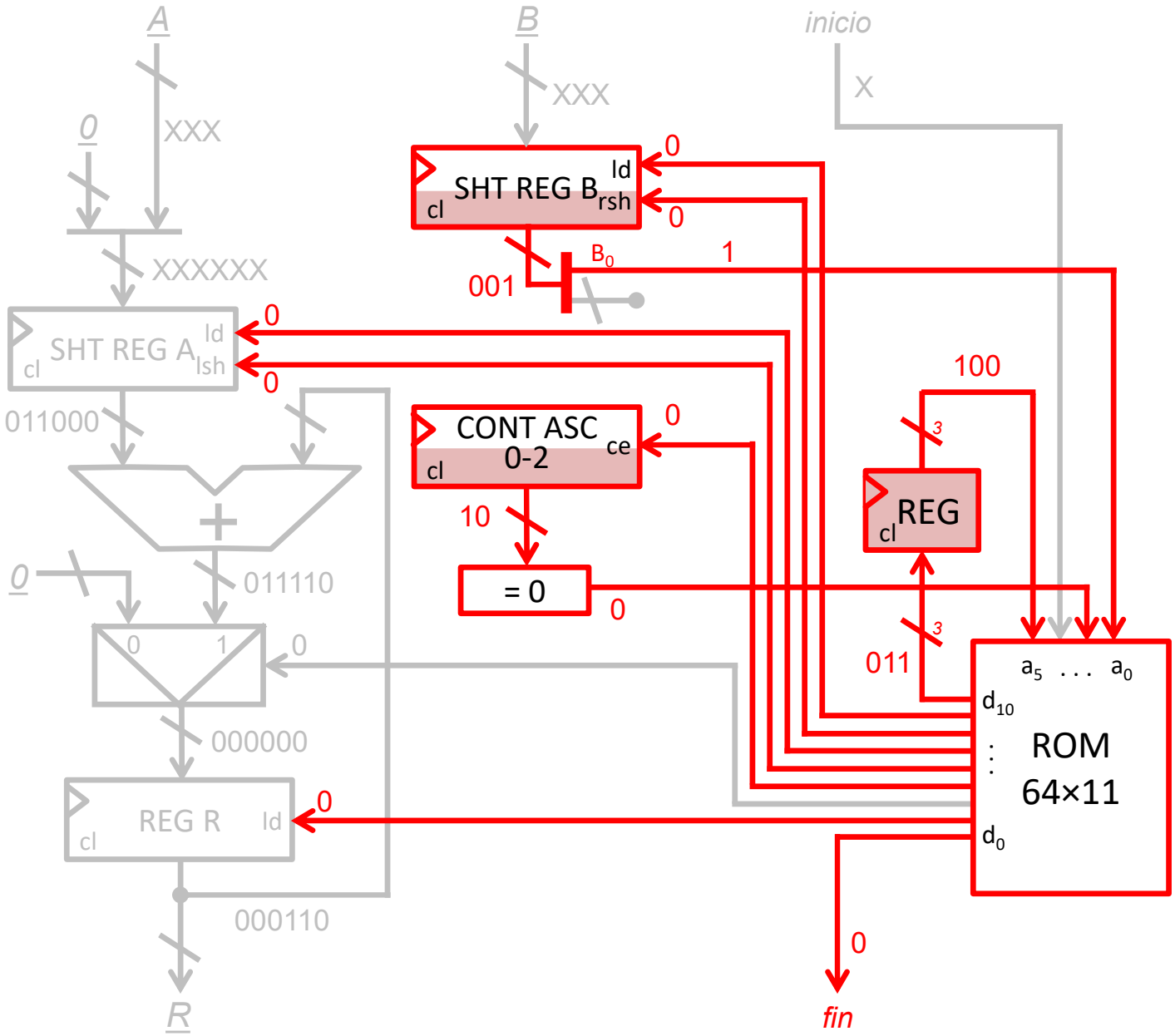
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

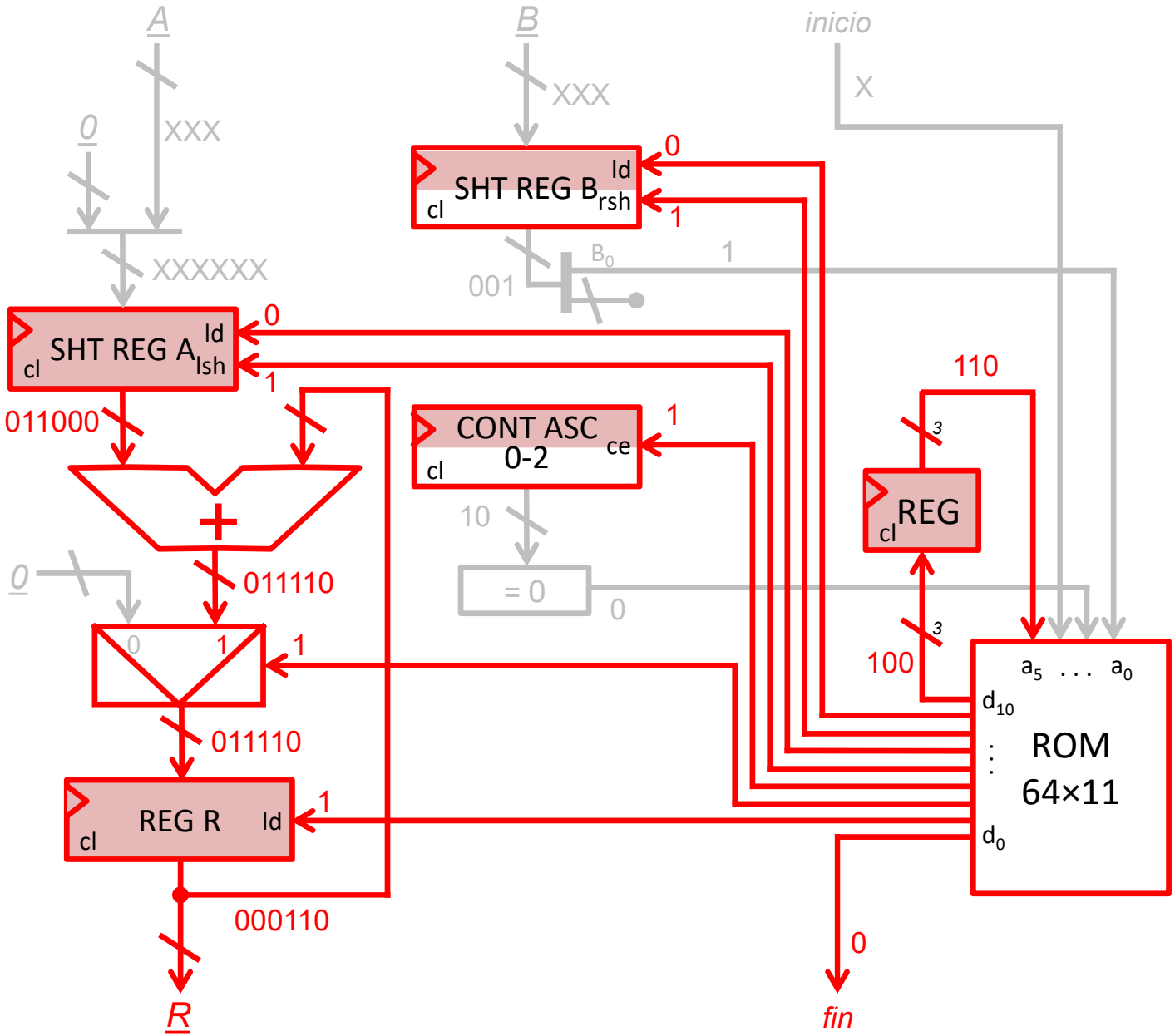
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

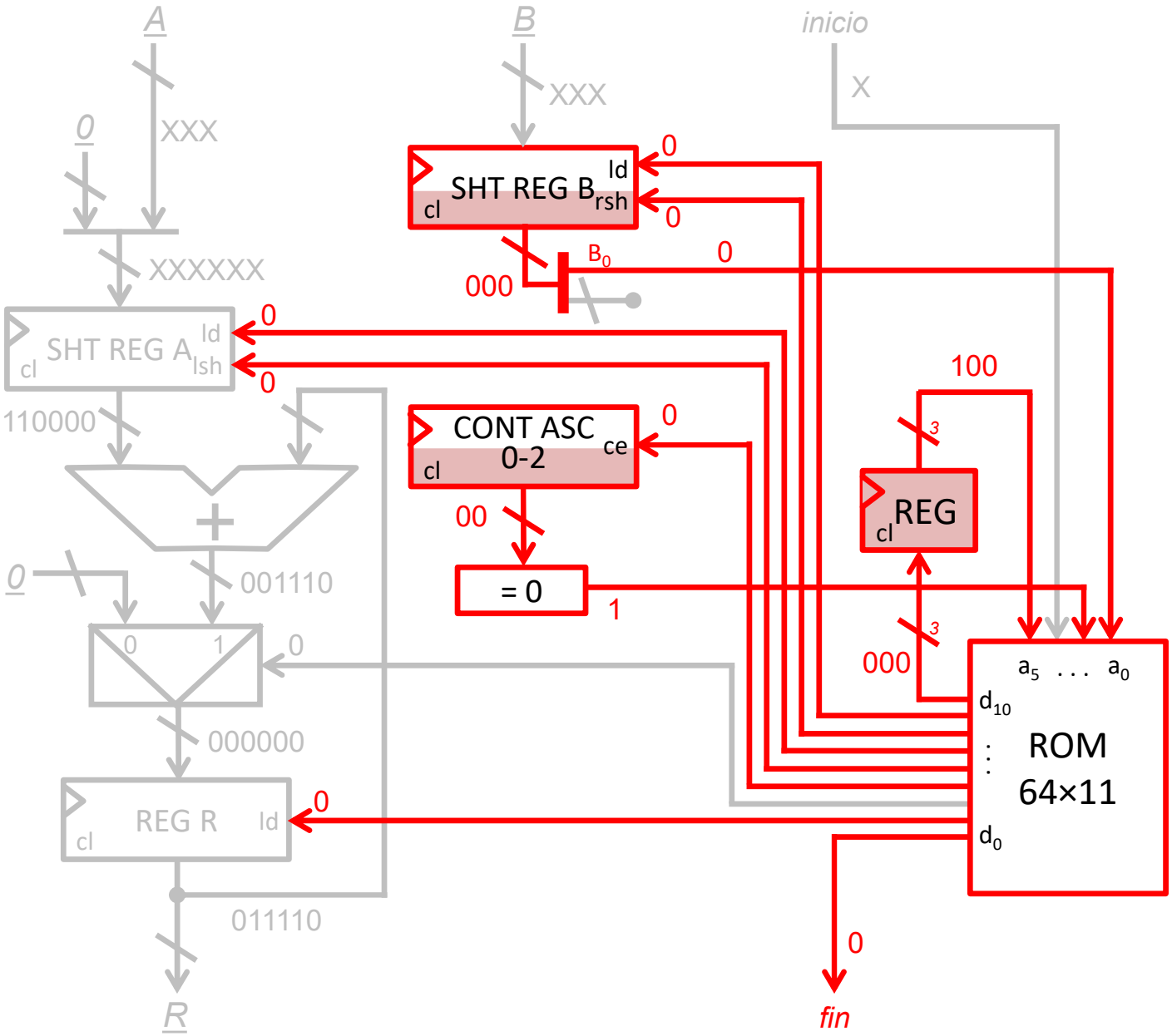
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

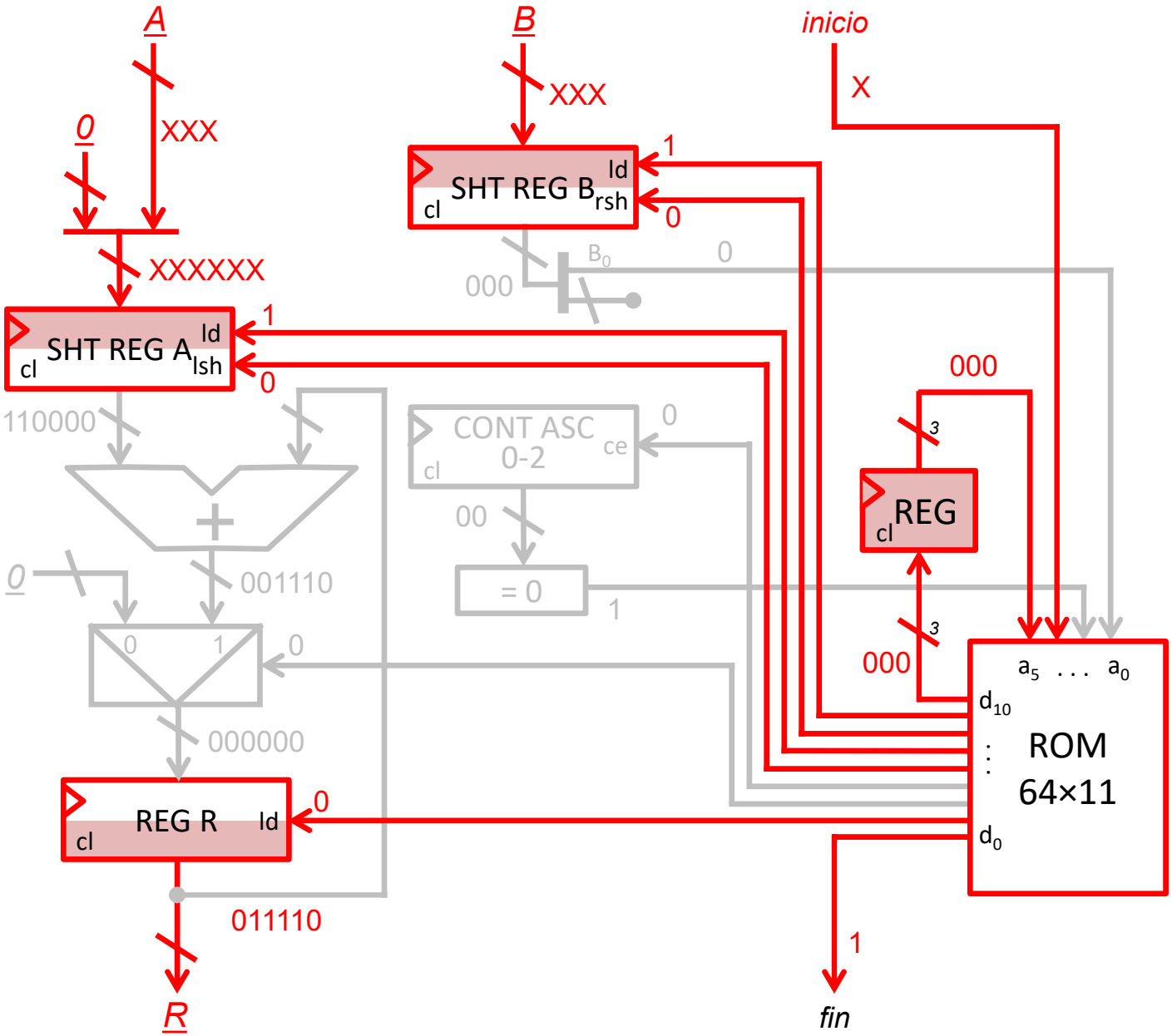
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

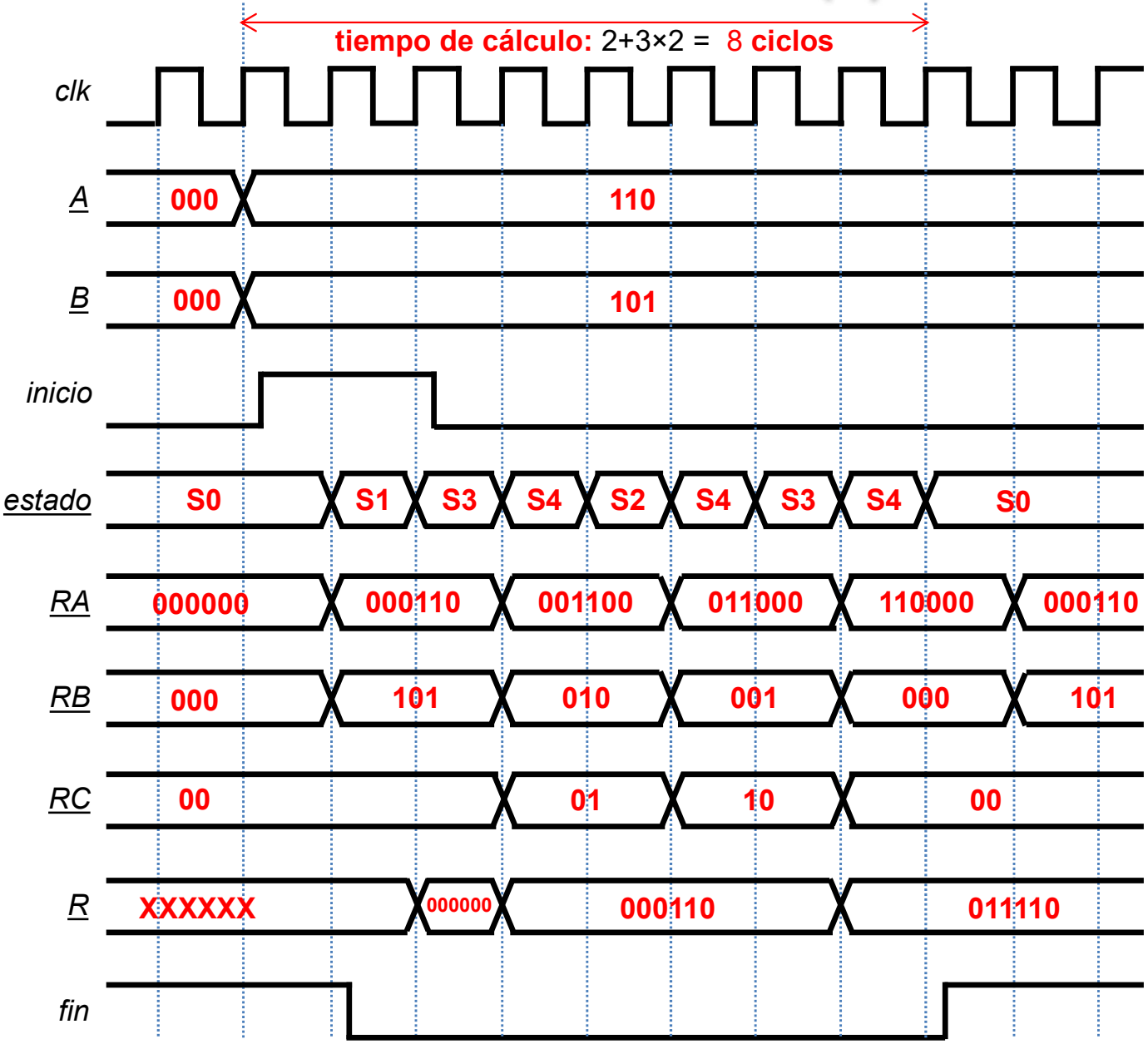
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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 ← 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;

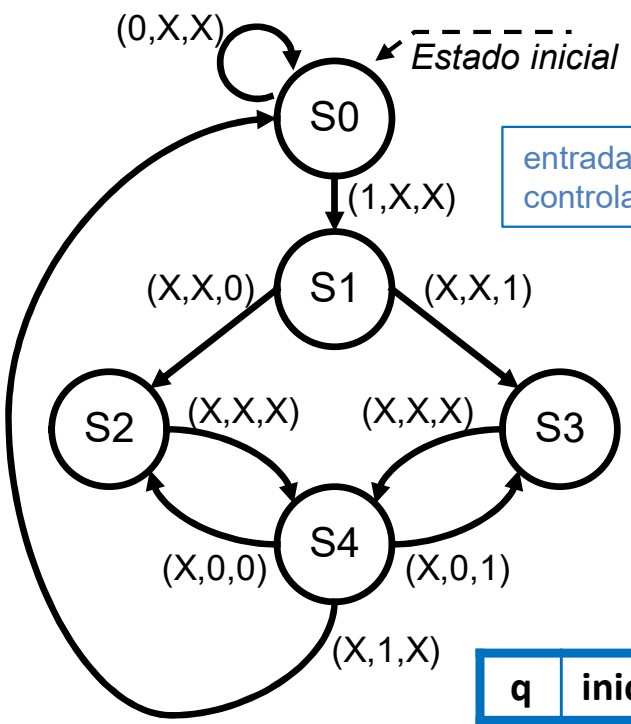
S2 RA ← RA << 1;
RB ← RB >> 1;
RC ← RC + 1; fin ← 0;
ir a S4;

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;

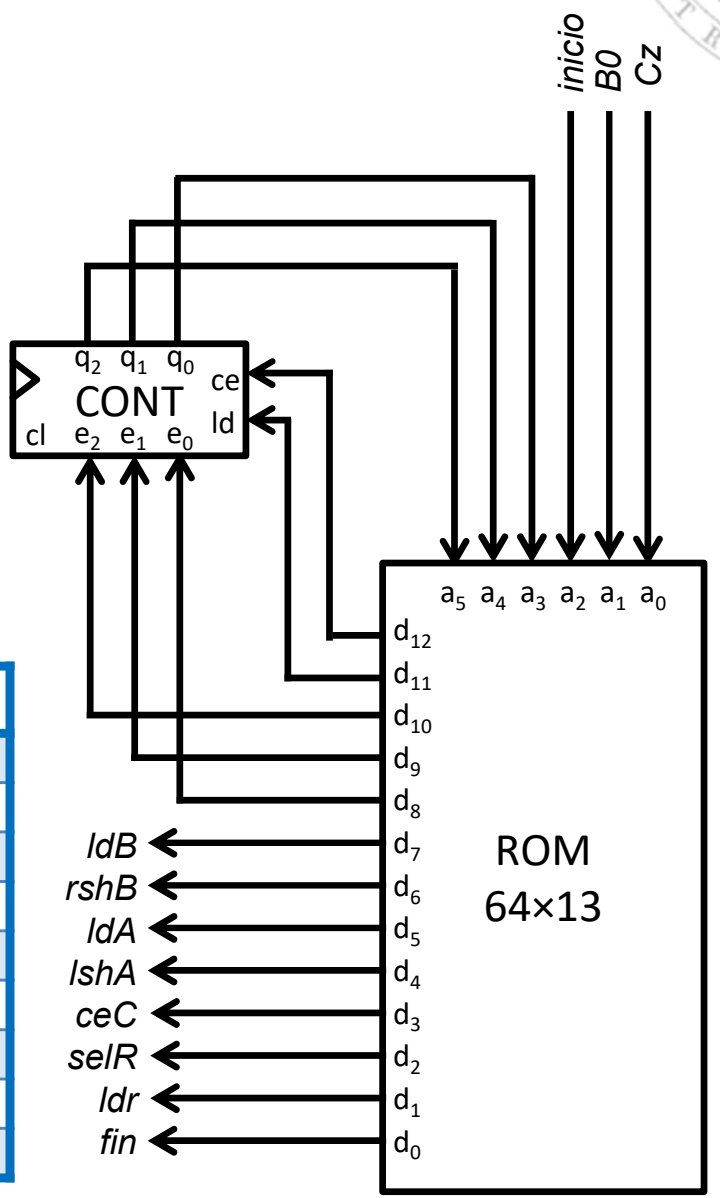
Multiplicador iterativo

Otro posible controlador: ROM + contador

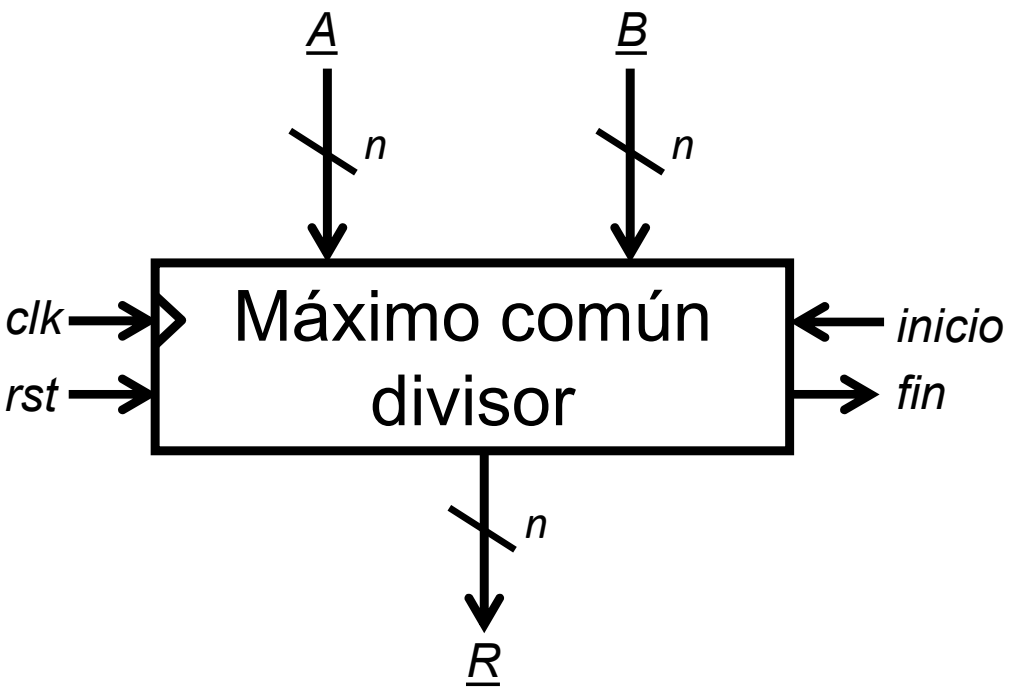


entradas del controlador: *(inicio, Cz, B0)*

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

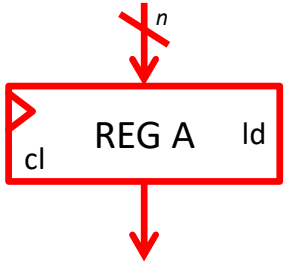


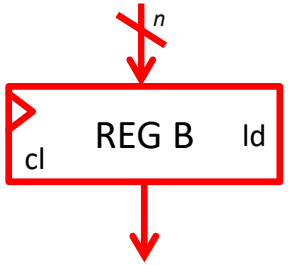
	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	R_3	R_2	R_1	R_0
1.	1	0	0	1	-	-	-	-	-	-	-	-
2.	1	0	0	1	0	1	1	0	-	-	-	-
3.	1	0	0	1	0	1	1	0	0	0	0	0
4.	0	0	1	1	0	1	1	0	0	0	0	0
5.	0	0	1	1	0	0	1	1	0	0	0	0
6.	0	0	1	1	0	0	1	1	0	0	1	1

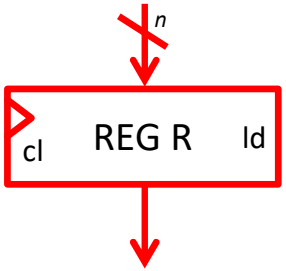
```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

algoritmo de Euclides

Máximo común divisor

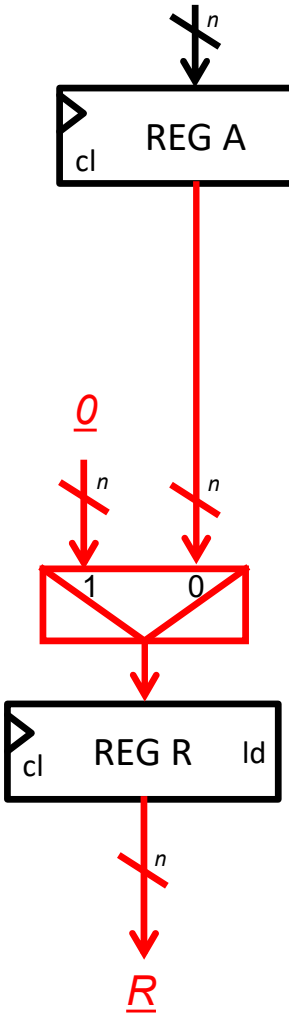


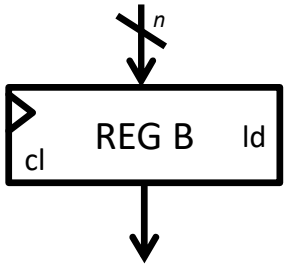





```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

Máximo común divisor

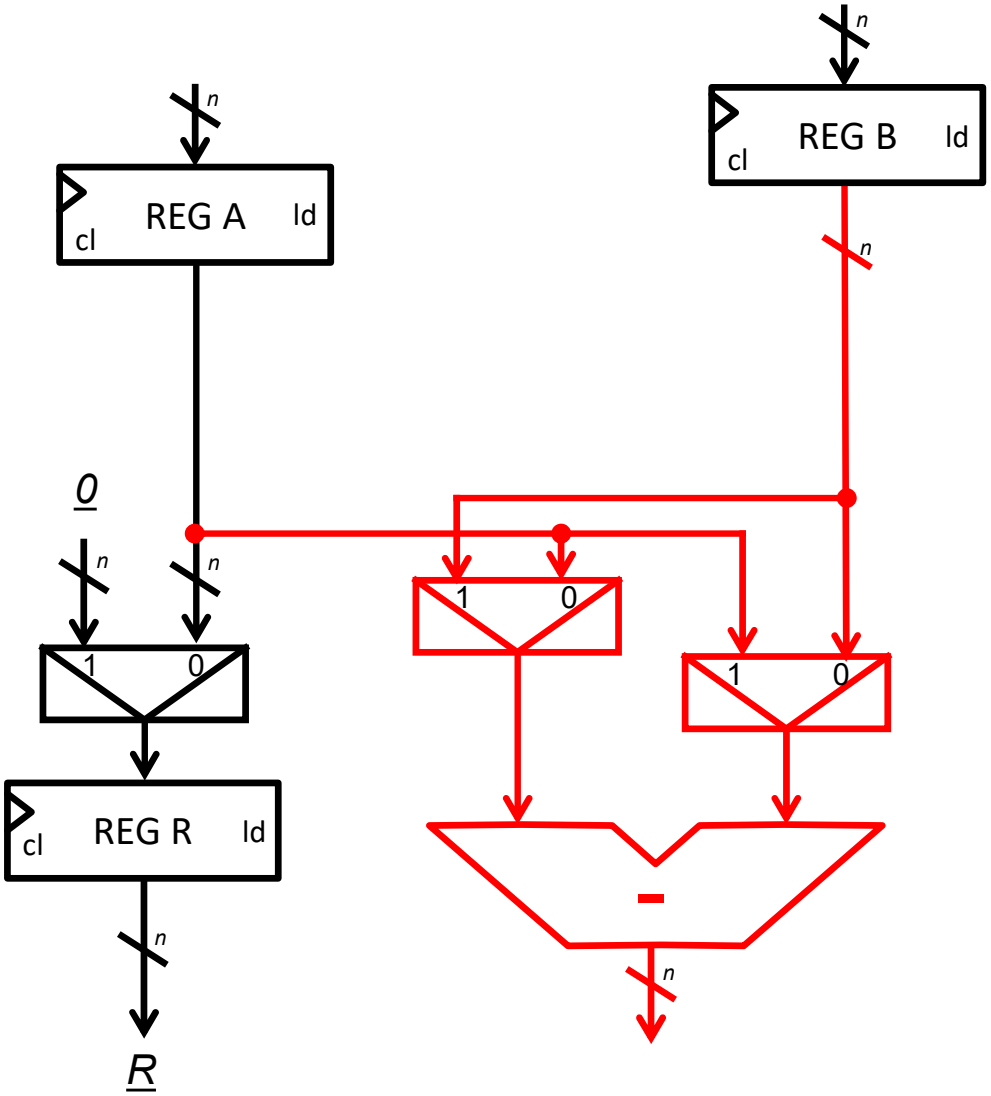





```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

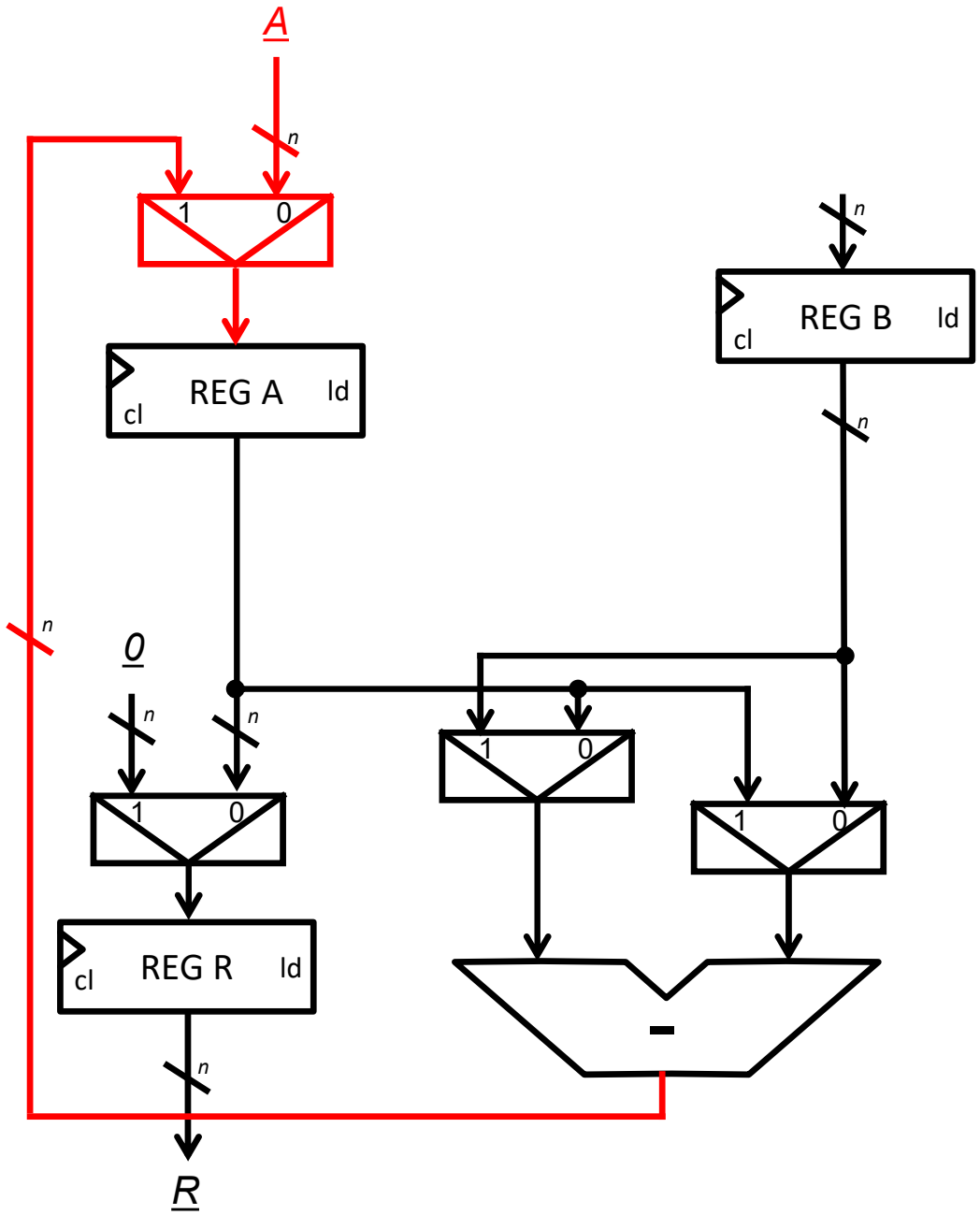
Máximo común divisor

Suma/resta parcial



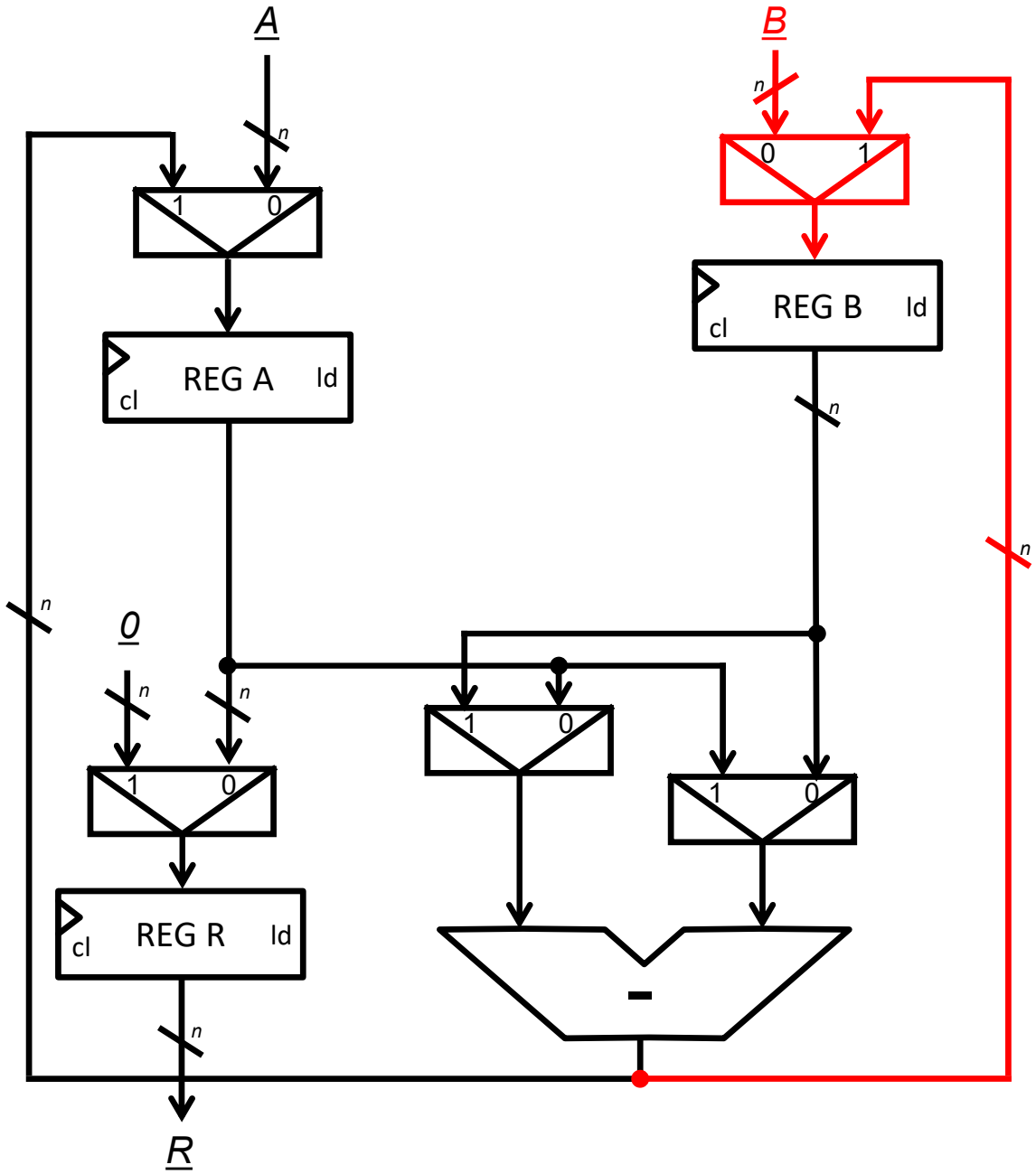

```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

Máximo común divisor



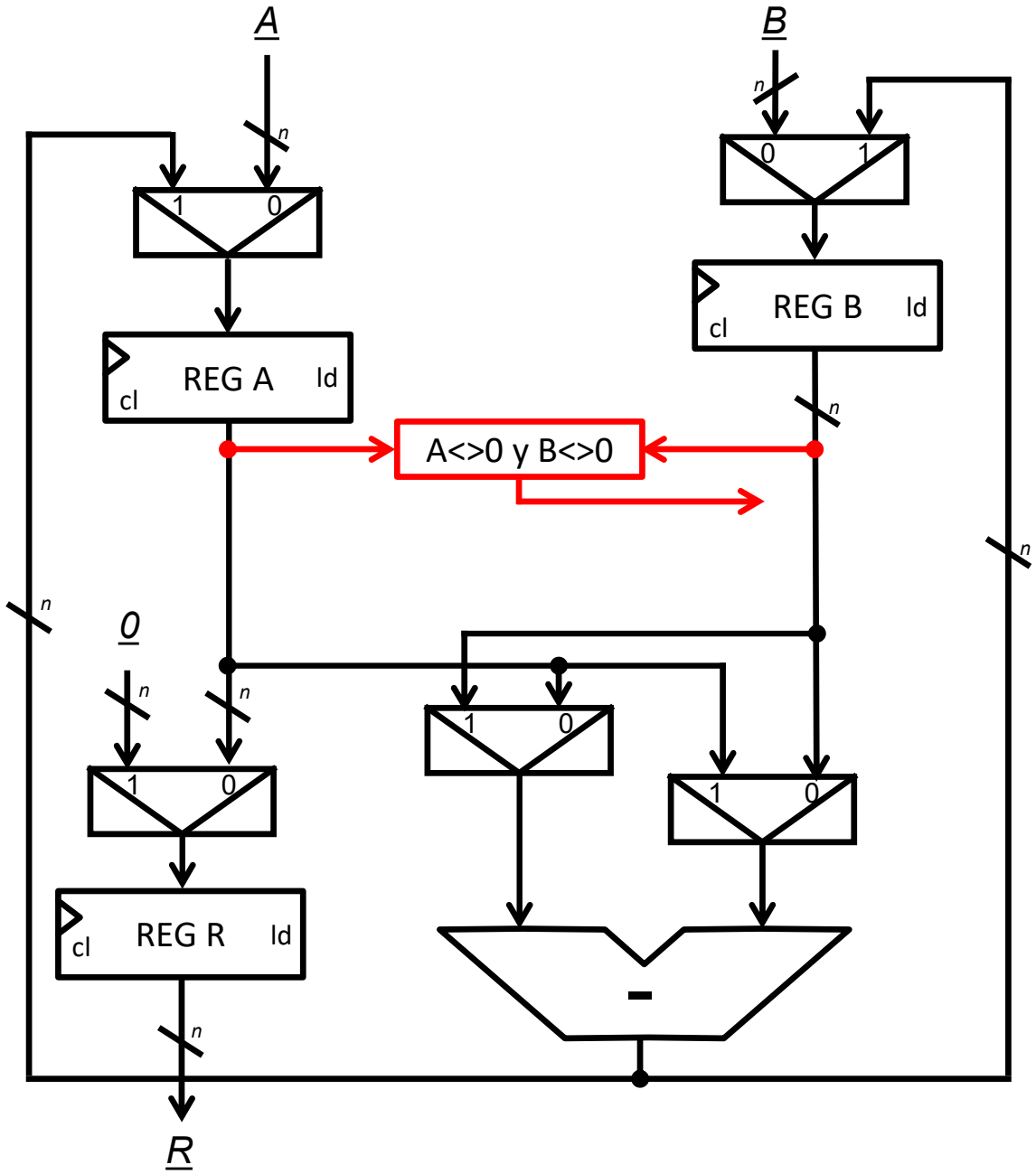

```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

Máximo común divisor

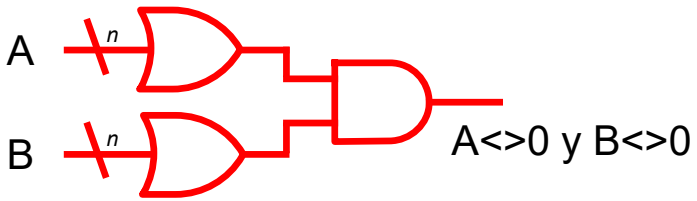



```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

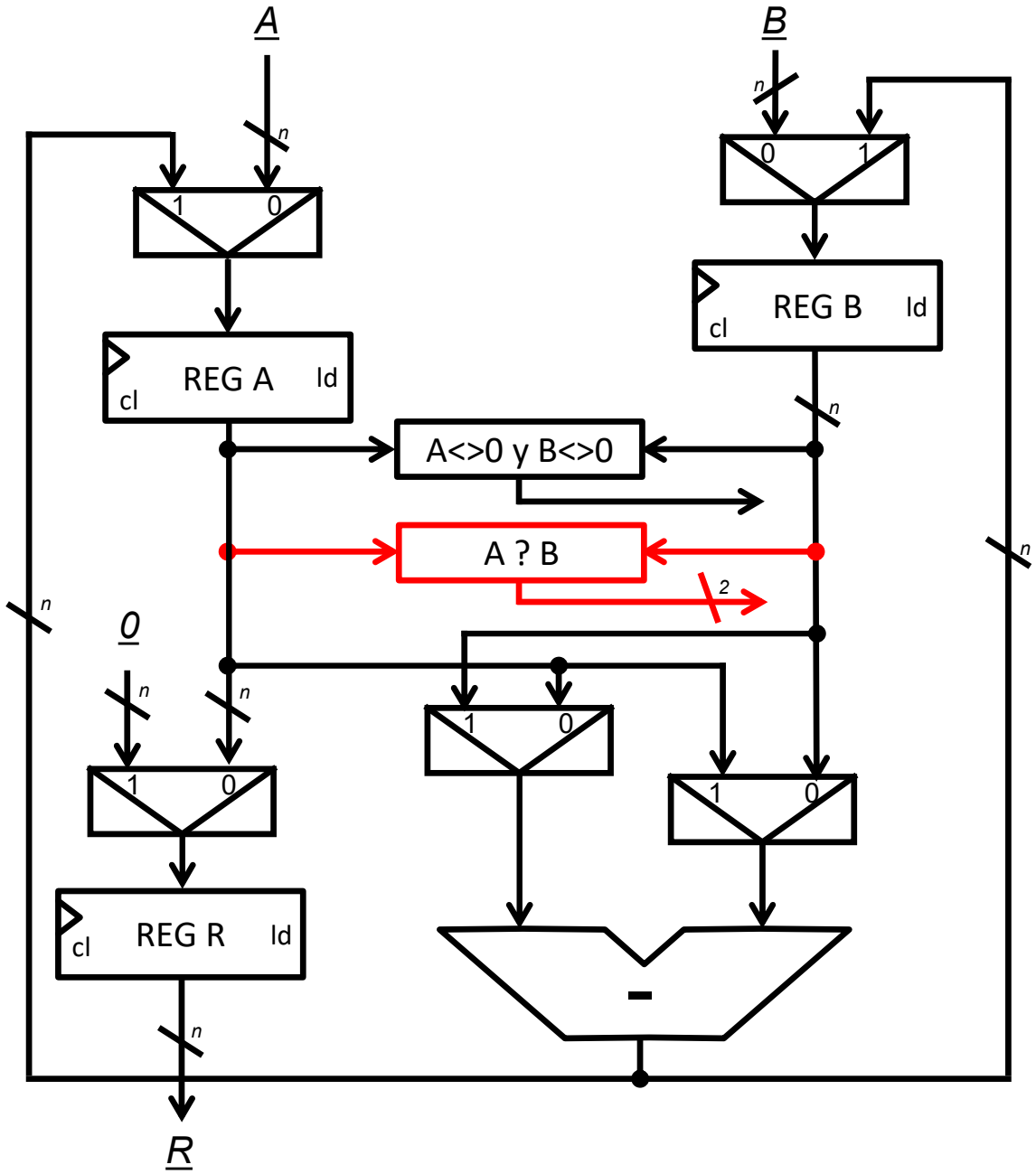
Máximo común divisor




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



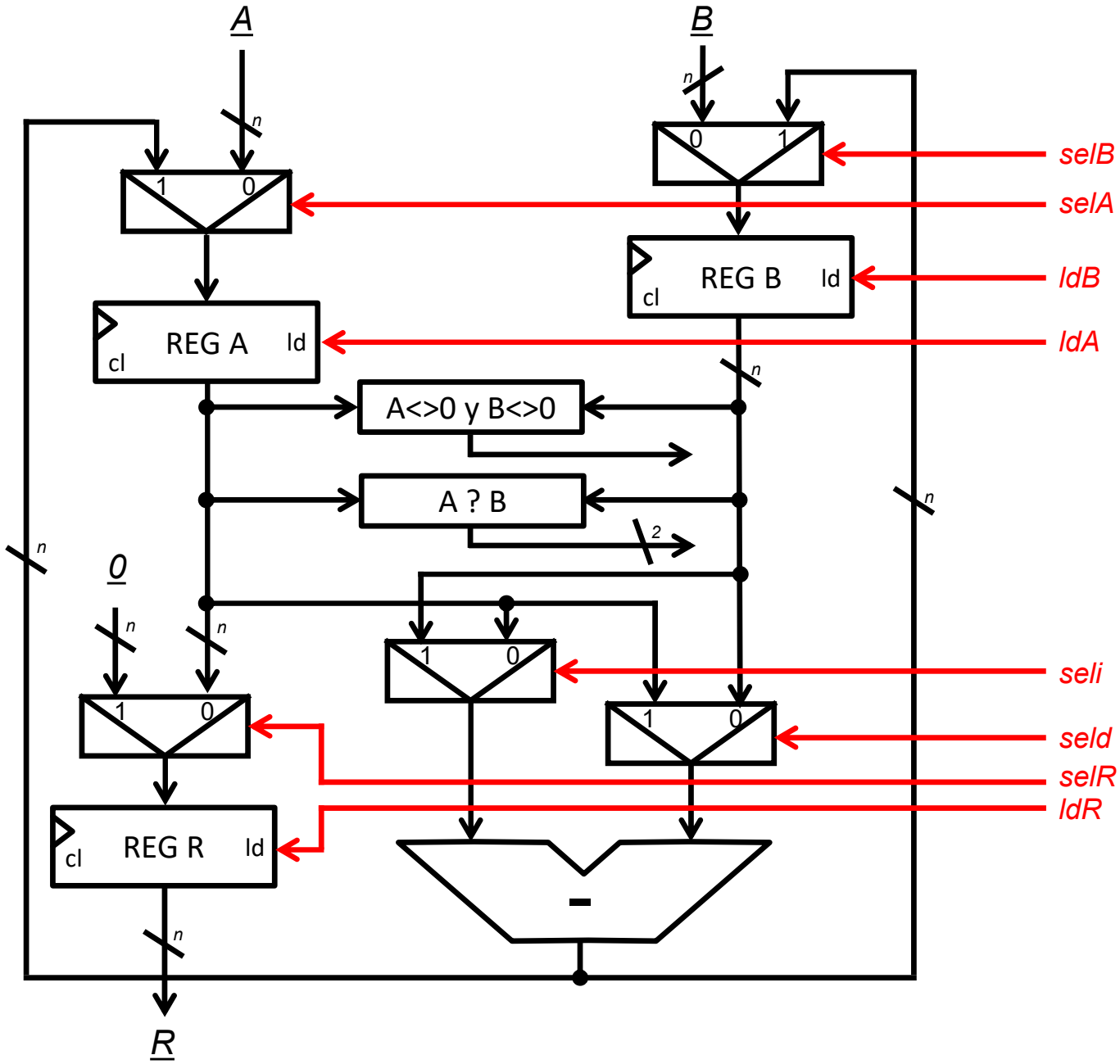
Máximo común divisor



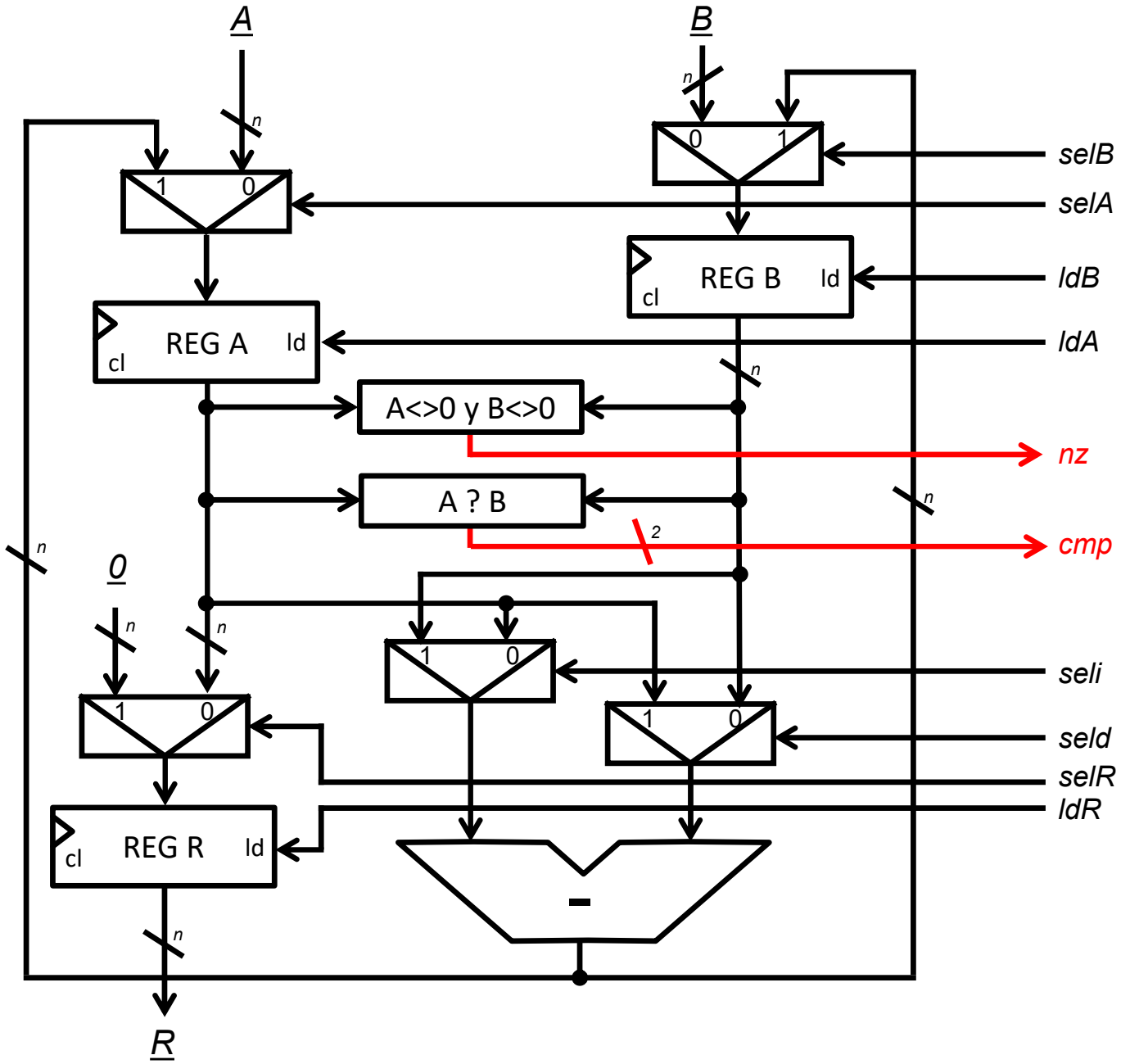

```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

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

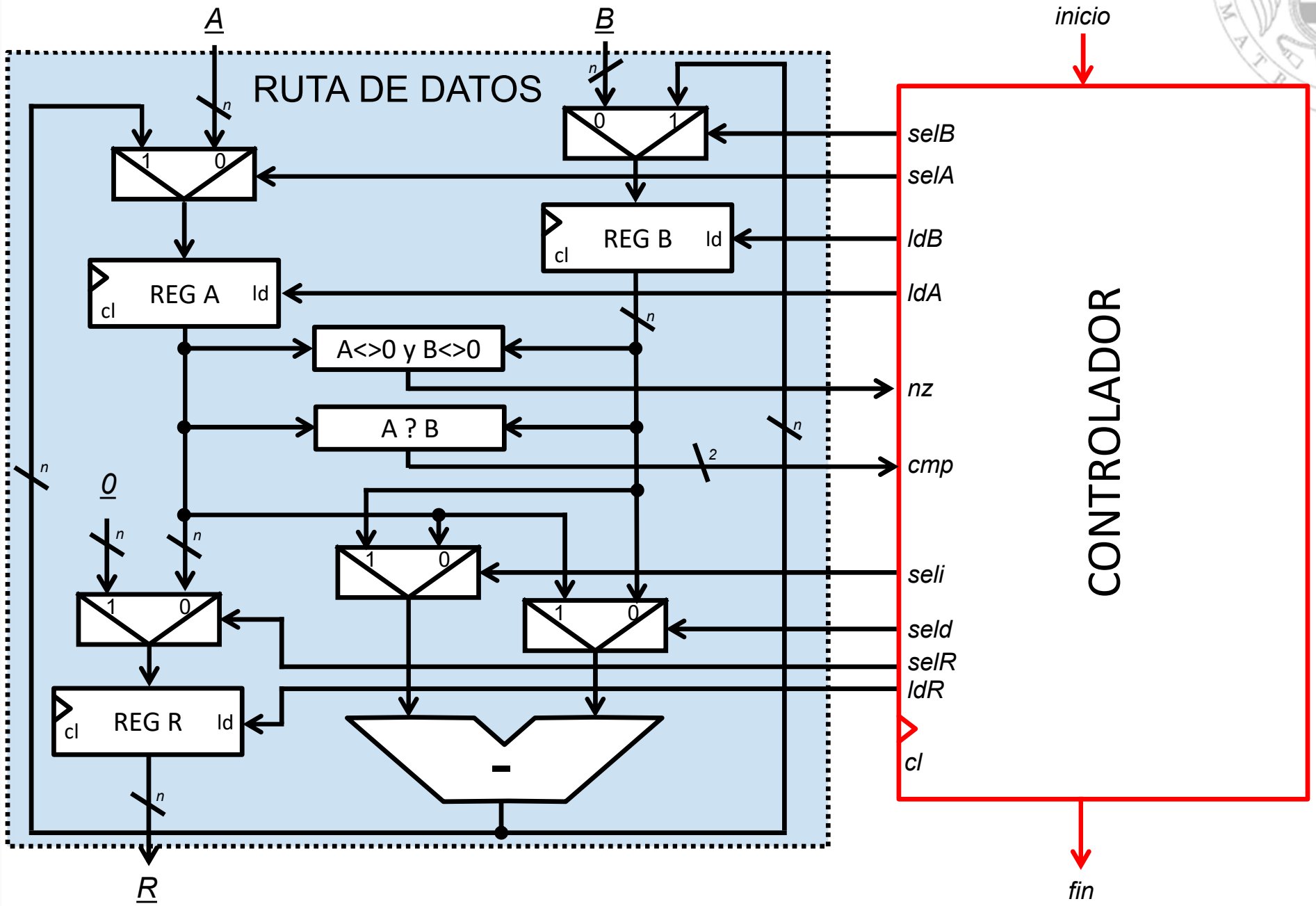
Máximo común divisor



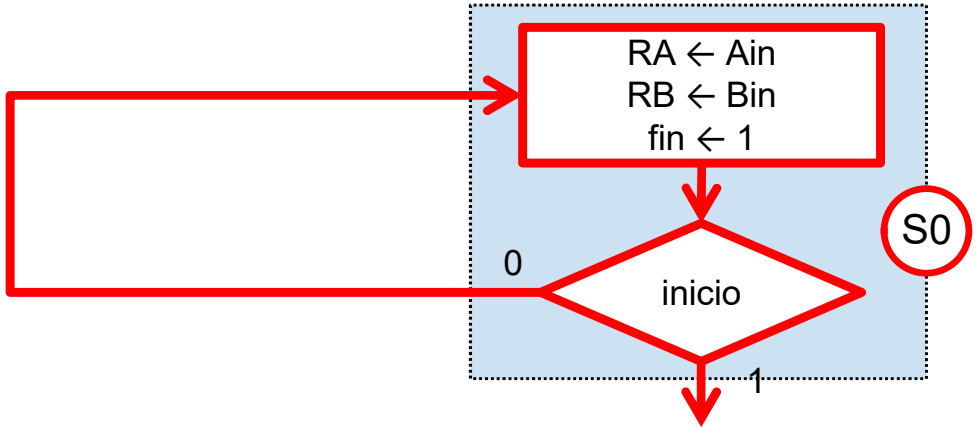
Máximo común divisor



Máximo común divisor

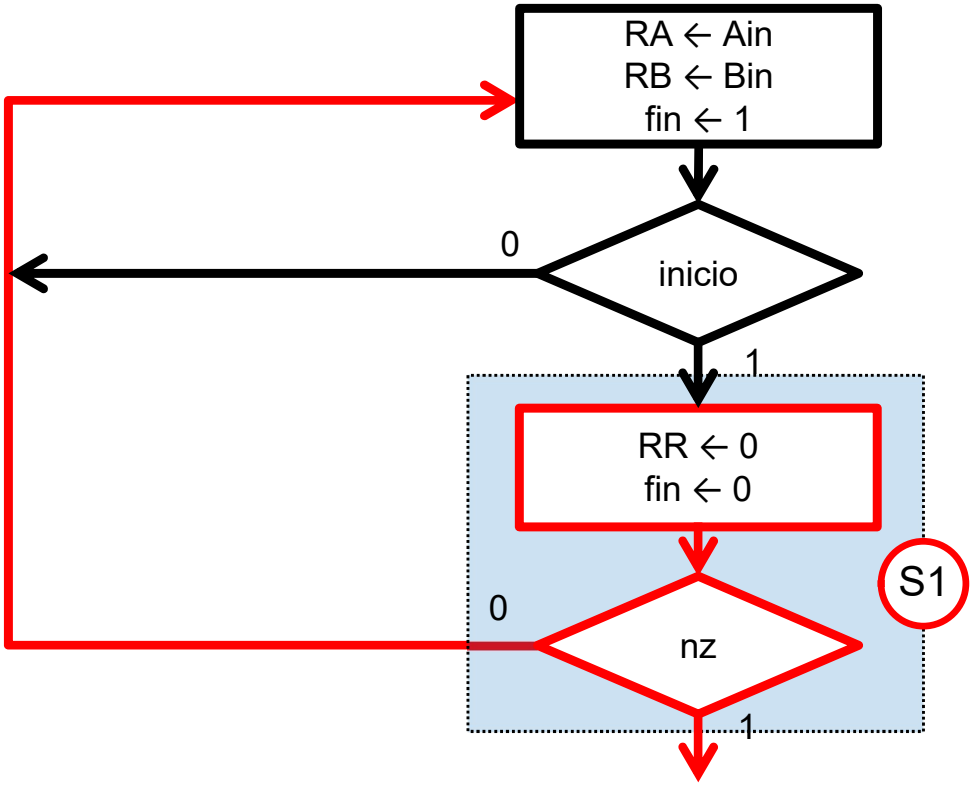


Máximo común divisor



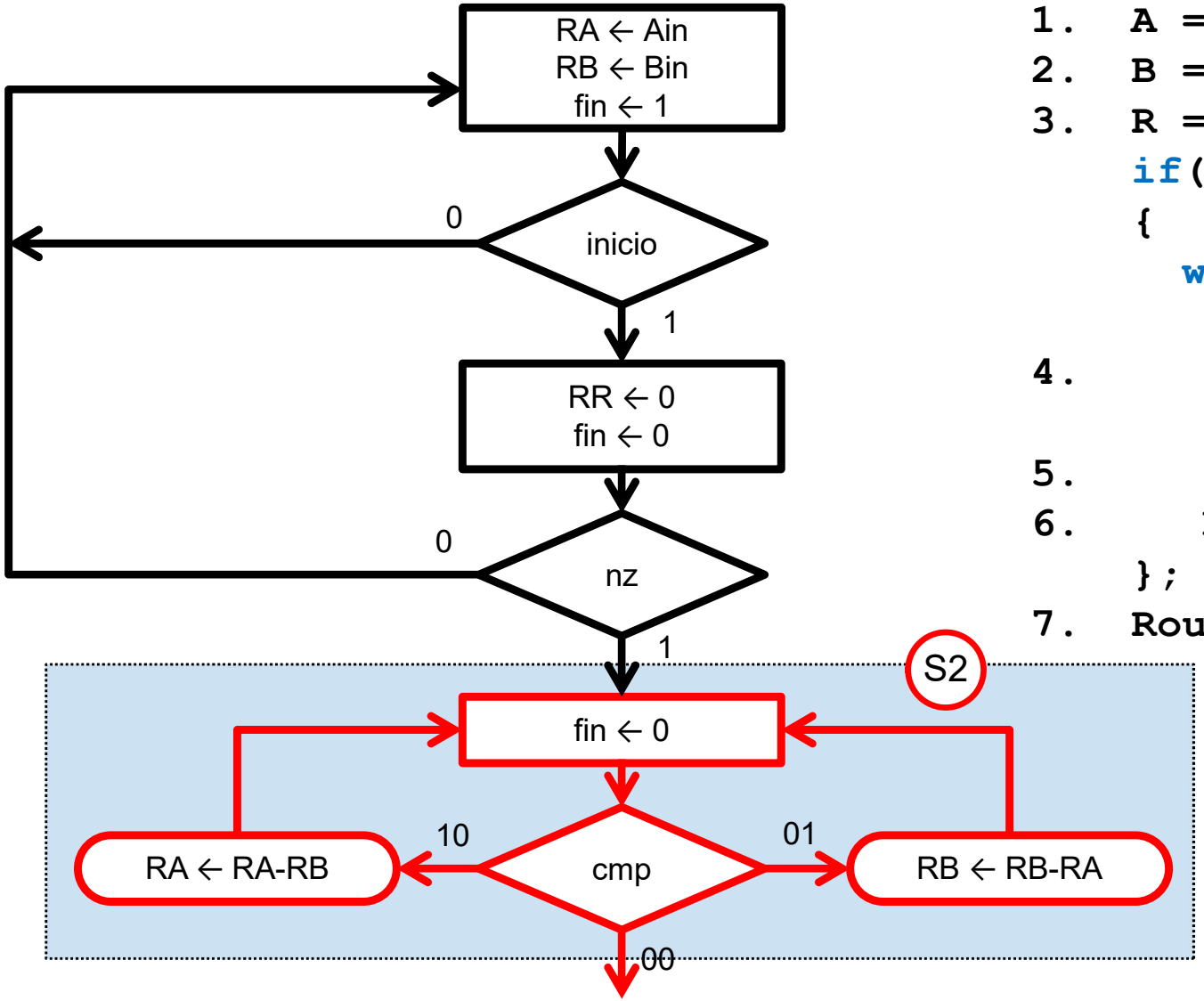

```
1.  A = Ain;
2.  B = Bin;
3.  R = 0;
    if( A!=0 && B!=0 )
    {
      while( A!=B )
        if( A>B )
4.        A = A-B;
        else
5.        B = B-A;
6.    R = A;
    };
7.  Rout = R;
```

Máximo común divisor



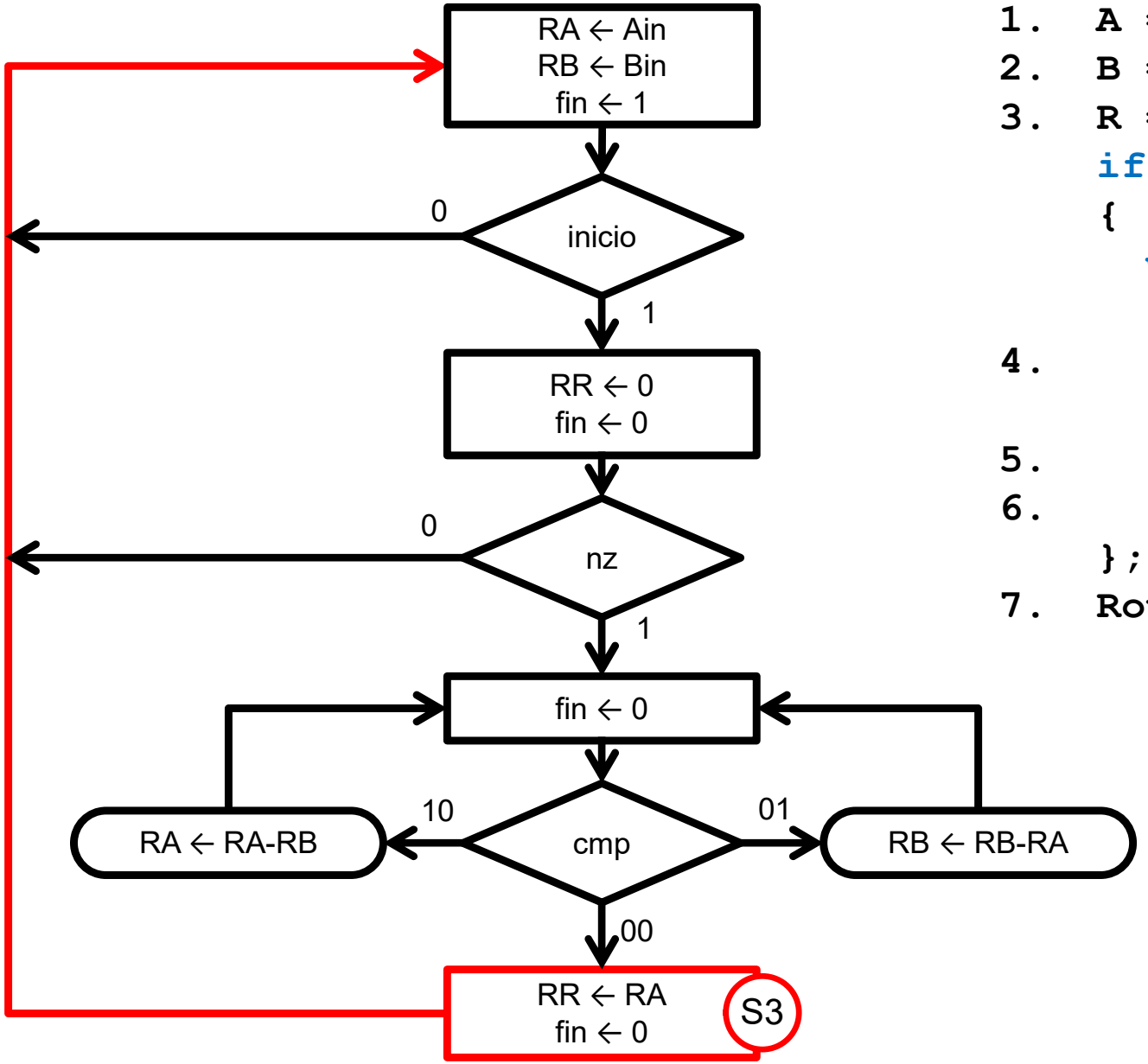

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     if( A!=0 && B!=0 )
     {
       while( A!=B )
         if( A>B )
4.         A = A-B;
         else
5.         B = B-A;
6.     R = A;
     };
7.   Rout = R;
```

Máximo común divisor



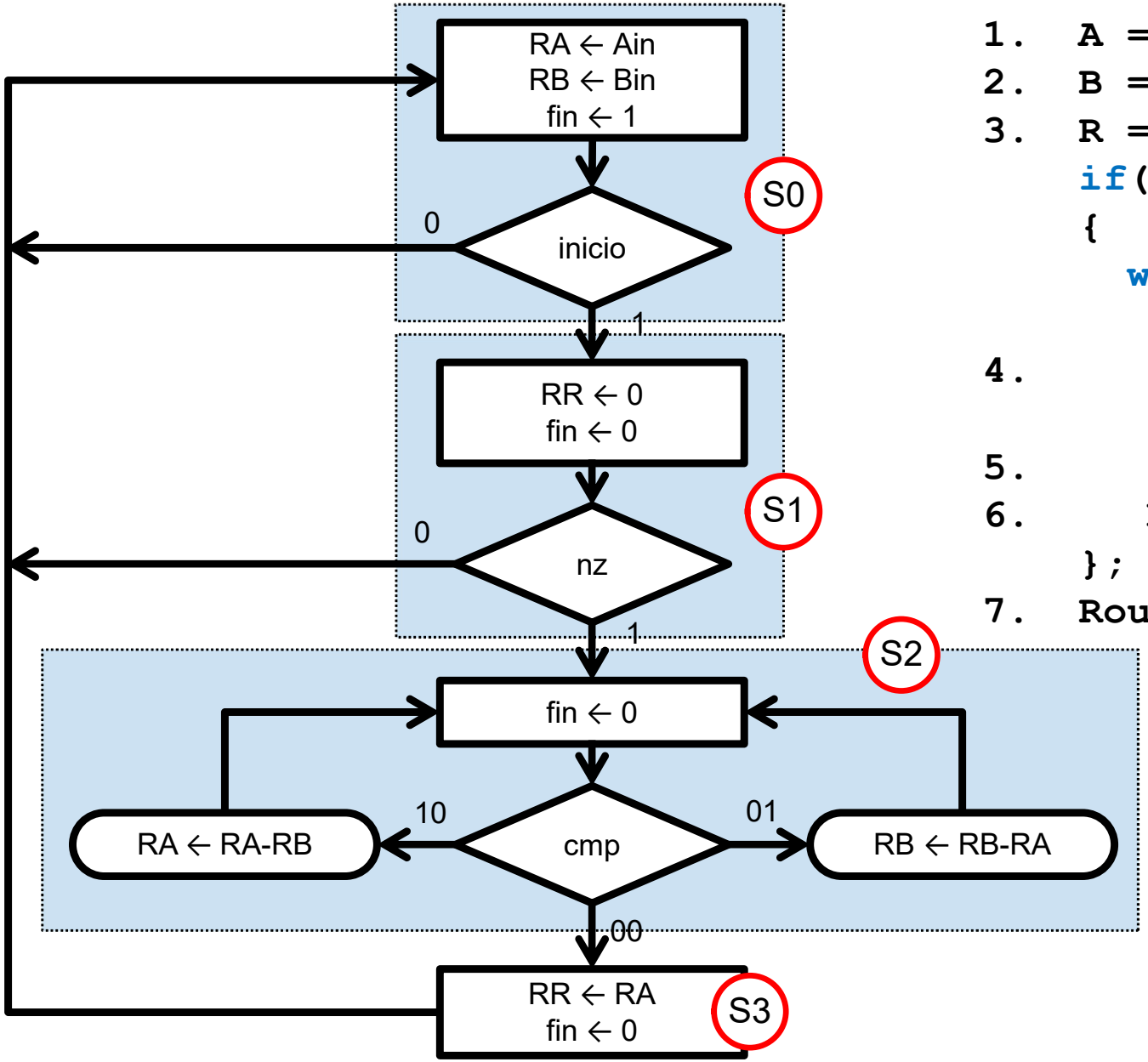

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     if( A!=0 && B!=0 )
     {
       while( A!=B )
         if( A>B )
4.         A = A-B;
         else
5.         B = B-A;
6.     R = A;
     };
7.   Rout = R;
```

Máximo común divisor



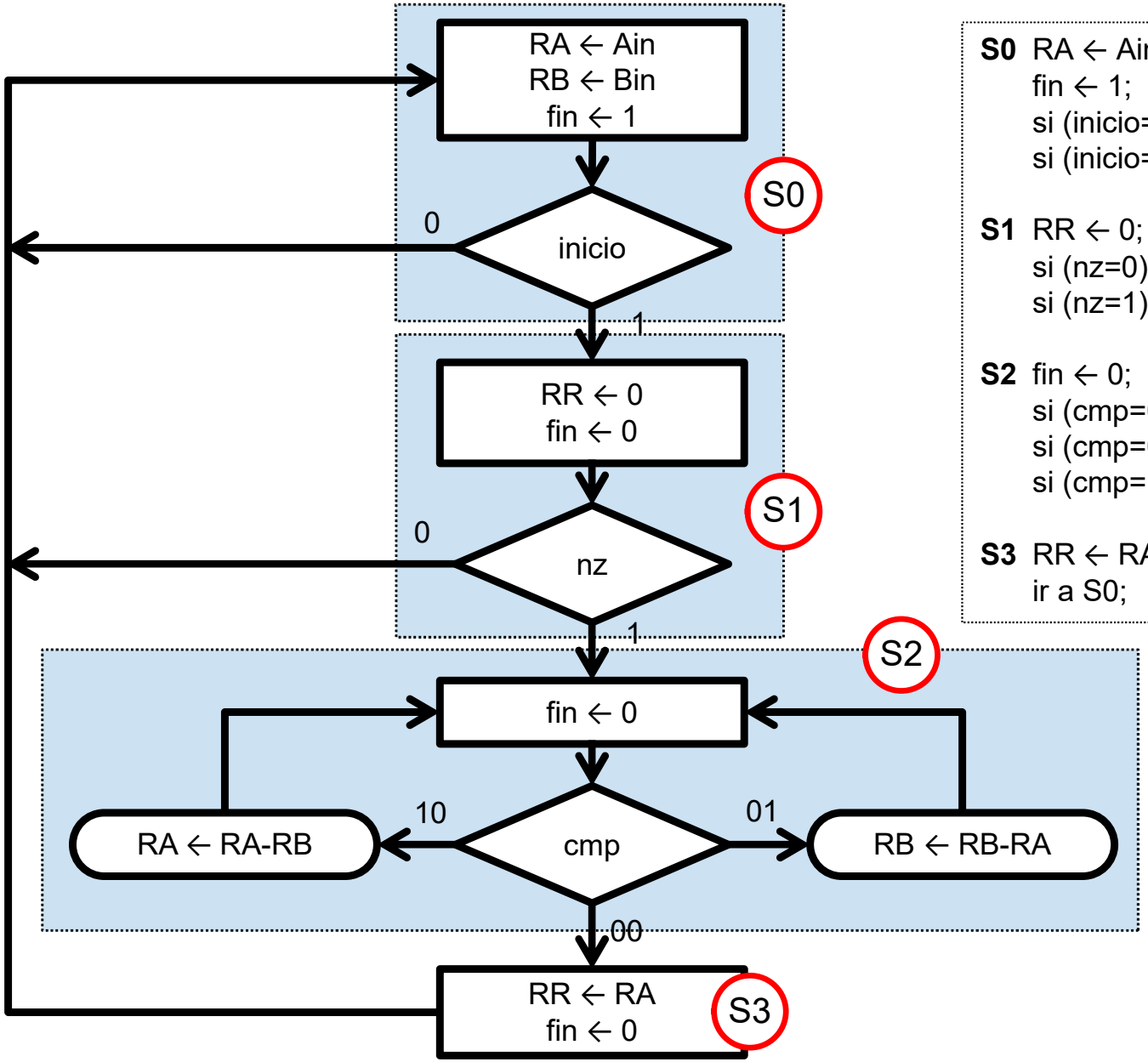

```
1.   A = Ain;
2.   B = Bin;
3.   R = 0;
     if( A!=0 && B!=0 )
     {
       while( A!=B )
         if( A>B )
4.         A = A-B;
         else
5.         B = B-A;
6.     R = A;
     };
7.   Rout = R;
```

Máximo común divisor




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

Máximo común divisor



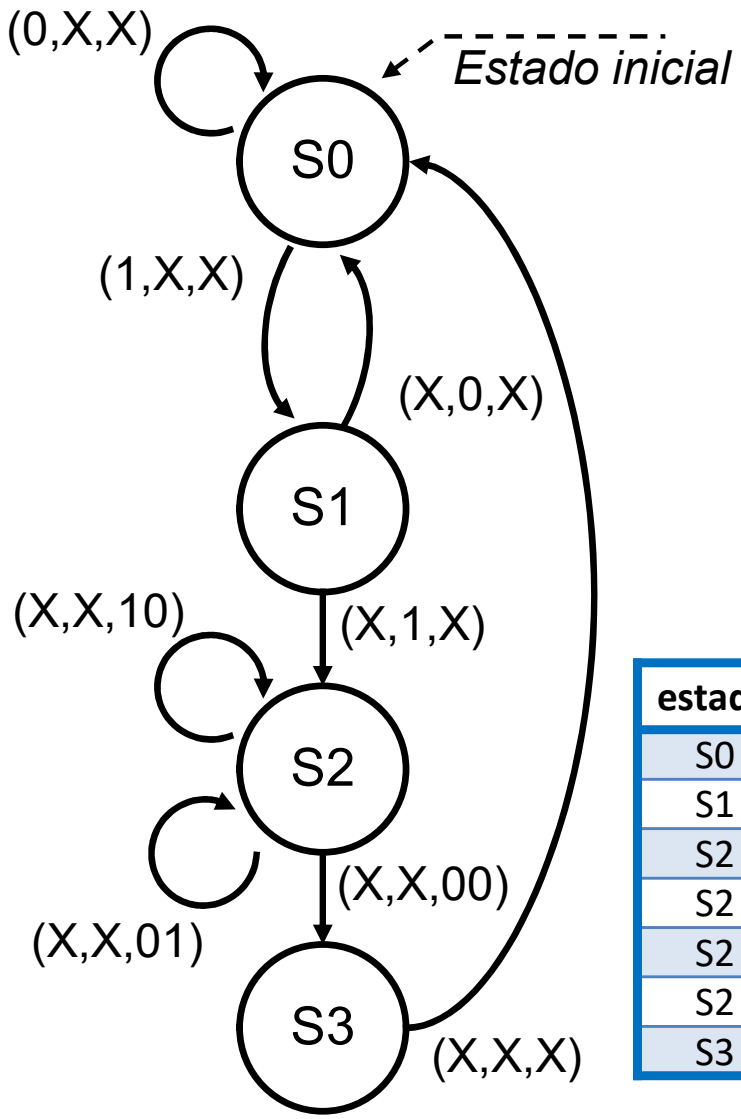
S0 RA ← Ain; RB ← Bin;
fin ← 1;
si (inicio=0) ir a S0;
si (inicio=1) ir a S1;

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

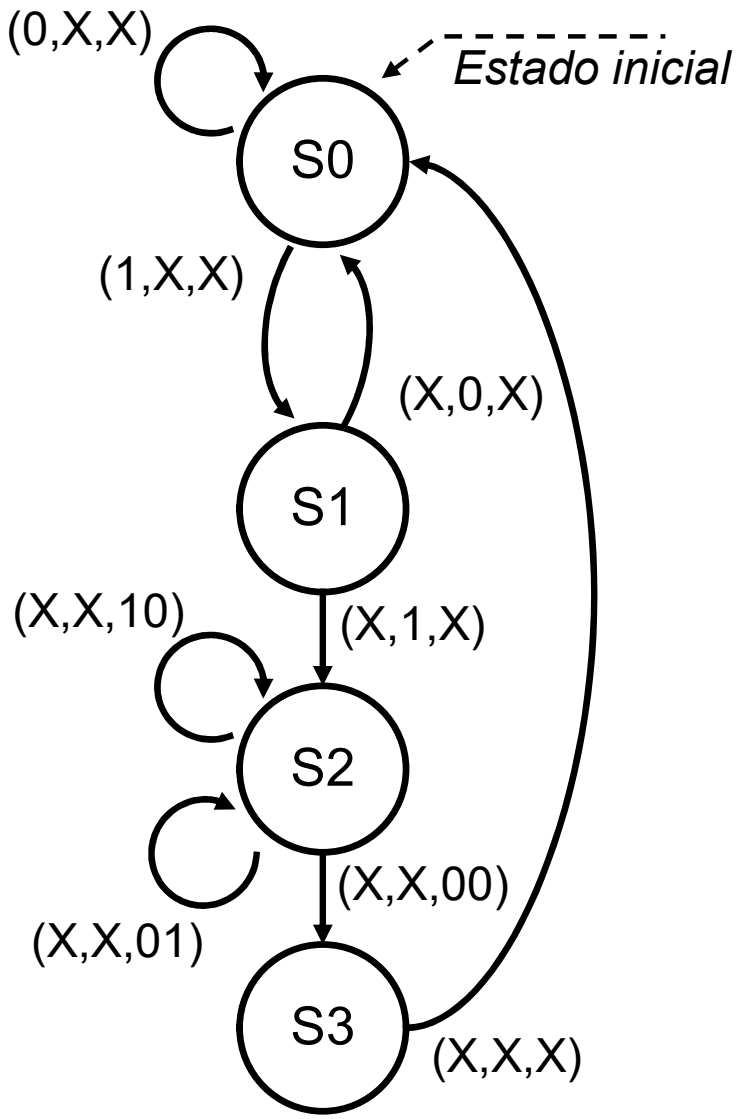
estado	cmp_1	cmp_0	selA	selB	ldA	ldB	seli	seld	selR	ldR	fin
S0	X	X	0	0	1	1	-	-	-	0	1
S1	X	X	-	-	0	0	-	-	1	1	0
S2	0	0	-	-	0	0	-	-	-	0	0
S2	0	1	-	1	0	1	1	1	-	0	0
S2	1	0	1	-	1	0	0	0	-	0	0
S2	1	1	-	-	-	-	-	-	-	-	-
S3	X	X	0	0	0	0	-	-	0	1	0

compatibles

compatibles

entradas del controlador: *(inicio, nz, cmp)*

Máximo común divisor

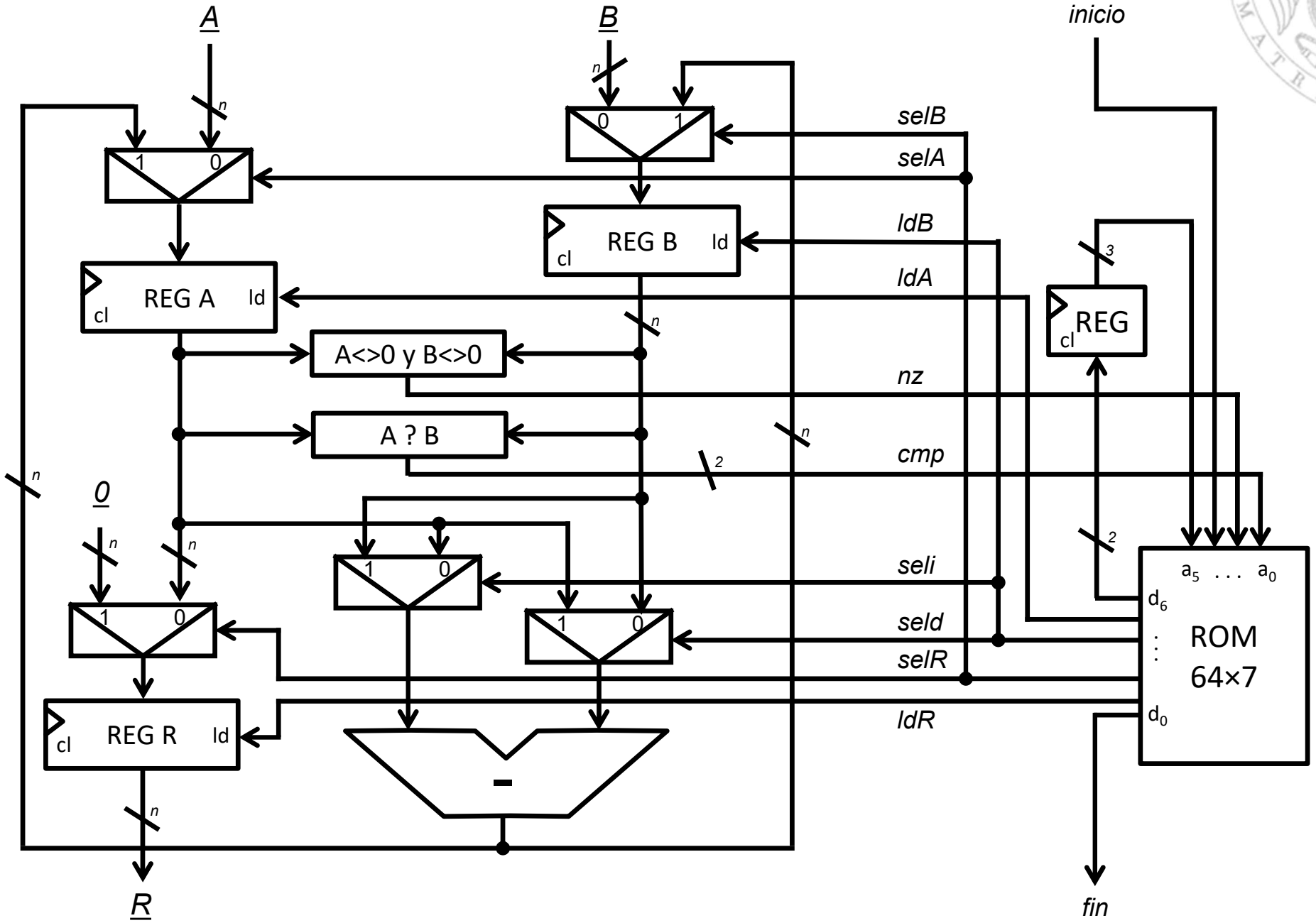


entradas del controlador: *(inicio, nz, cmp)*

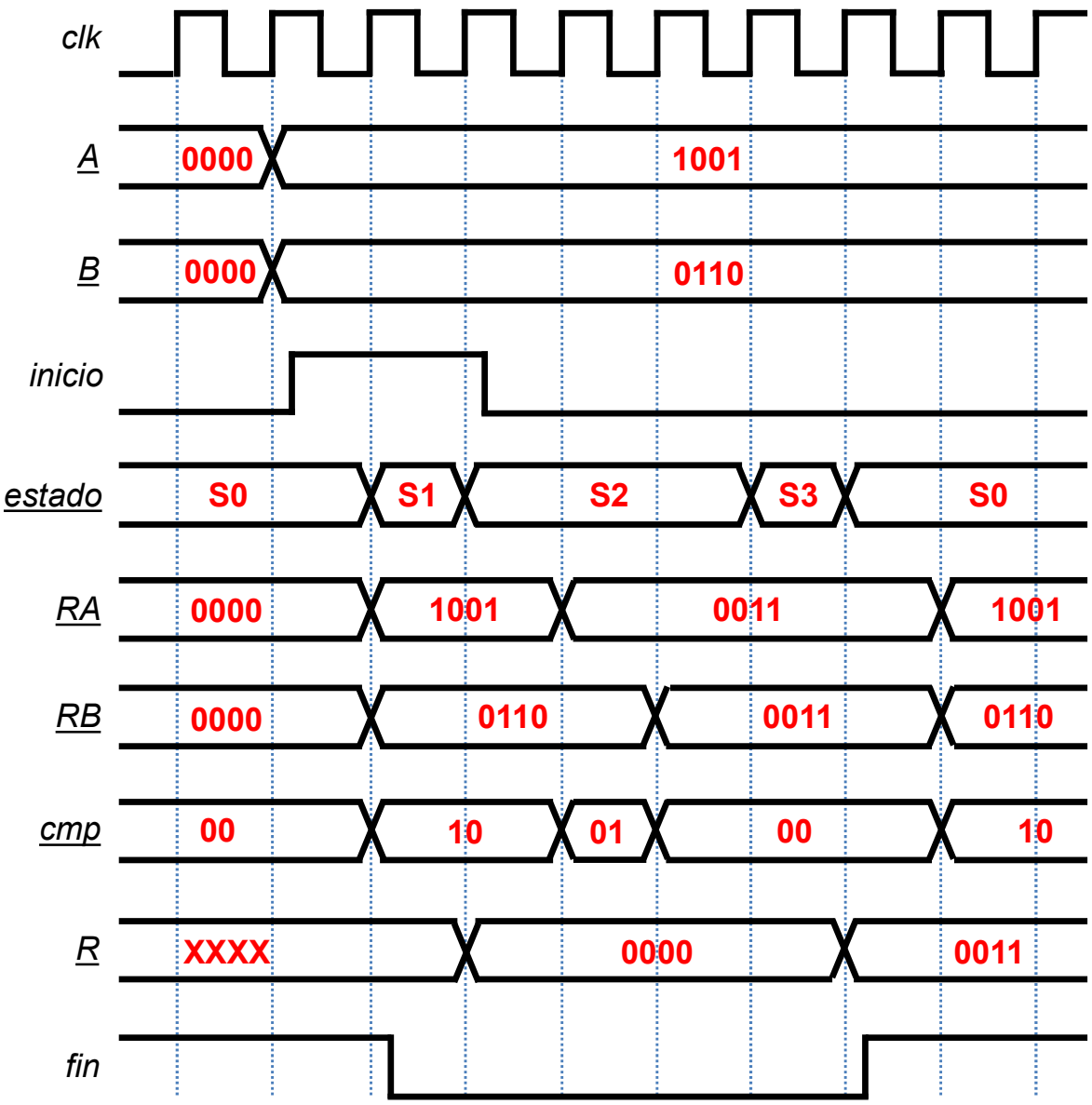
estado	inicio	nz	cmp_1	cmp_0	estado'
S0	0	X	X	X	S0
S0	1	X	X	X	S1
S1	X	0	X	X	S0
S1	X	1	X	X	S2
S2	X	X	0	0	S3
S2	X	X	0	1	S2
S2	X	X	1	0	S2
S2	X	X	1	1	-
S3	X	X	X	X	S0

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



Máximo común divisor



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

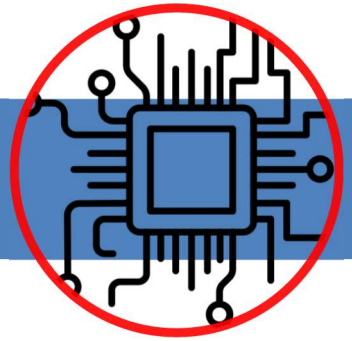
S1 RR ← 0; fin ← 0;
si (nz=0) ir a S0;
si (nz=1) ir a S2;

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;

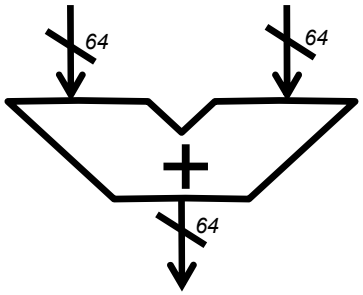
- Cálculo del coste.
- Cálculo del tiempo de ciclo.

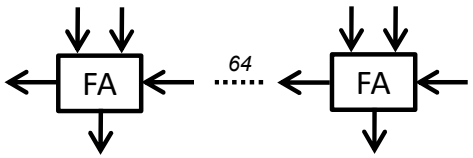
Apéndice tecnológico



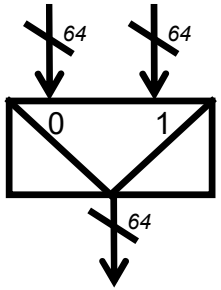
Multiplicador iterativo

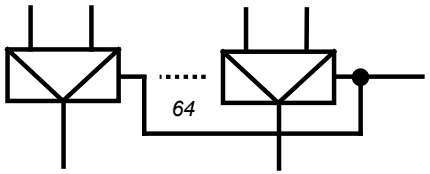
Cálculo del coste y tiempo de ciclo



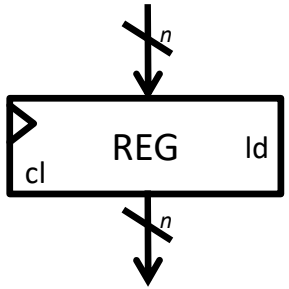


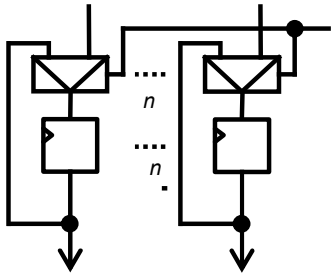
área: 64×29.49 = **1887 μm²**
retardo: 64×226 = **14464 ps**



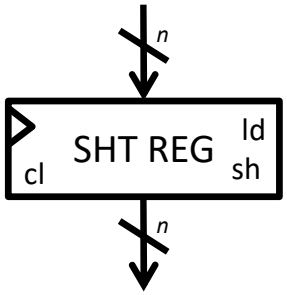


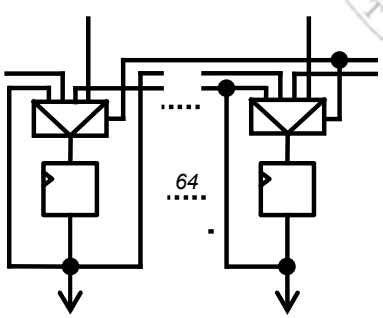
área: 64×11.05 = **707.2 μm²**
retardo: 1×223 = **223 ps**



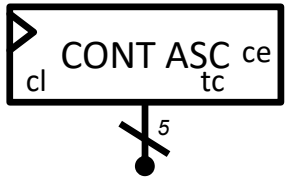


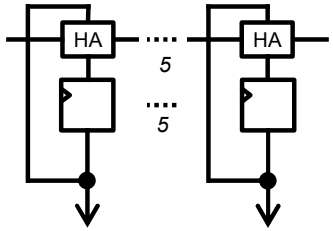
área: n×11.05 + n×32.26 = **n×43.31 μm²**
retardo CLK→Q: 1×167 = **167 ps**
retardo in→D: 1×223 = **223 ps**



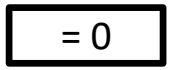


área: n×23.04 + n×32.26 = **n×55.3 μm²**
retardo CLK→Q: 1×167 = **167 ps**
retardo in→D: 1×250 = **250 ps**

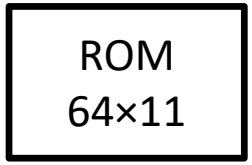




área: 5×15.77 + 5×32.26 = **240.6 μm²**
retardo CLK→Q: 1×167 = **167 ps**
retardo in→D: 5×114 = **570 ps**



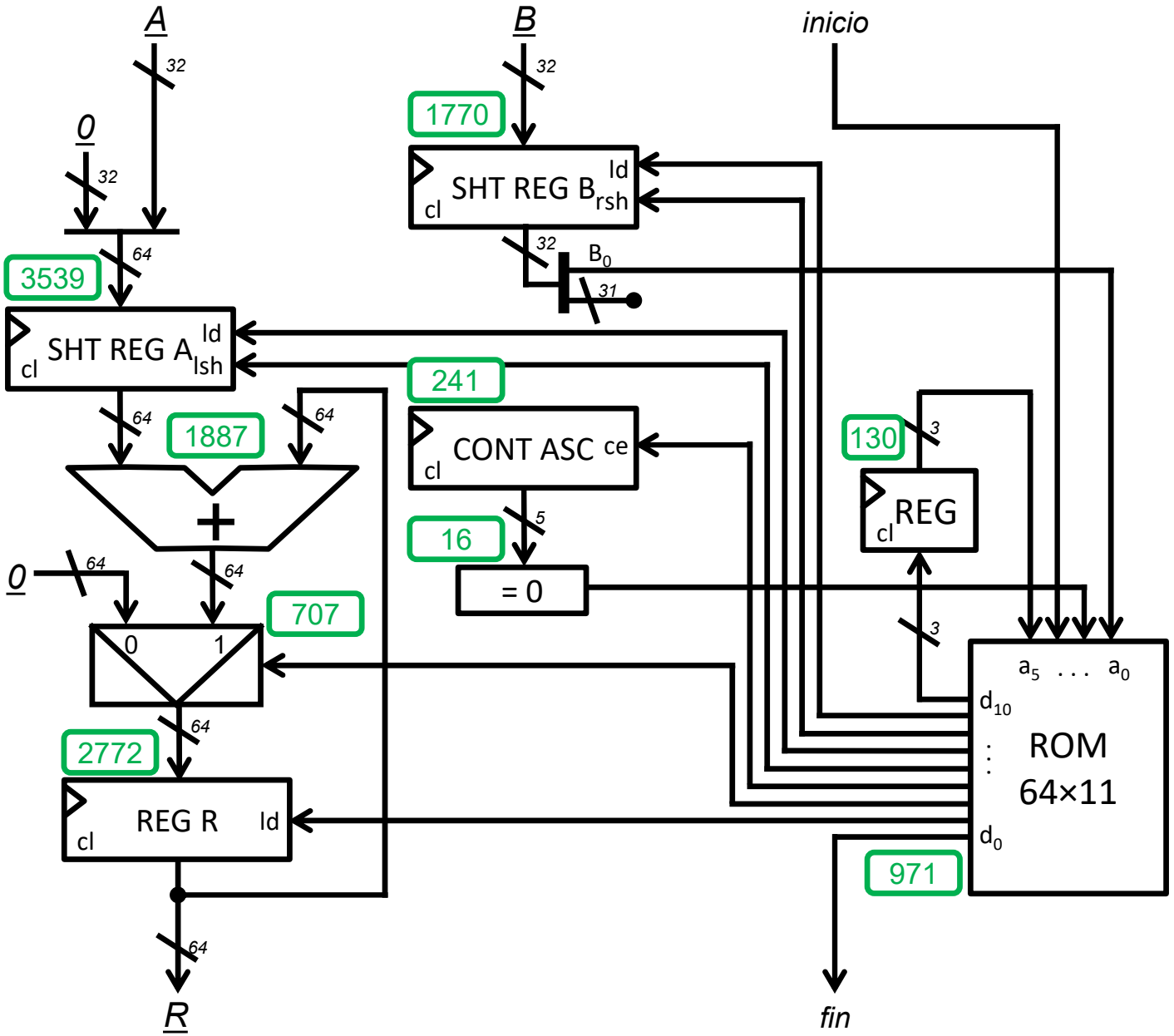
área: 15,67 μm²
retardo: 126 ps



área: 971 μm²
retardo: 573 ps

Multiplicador iterativo

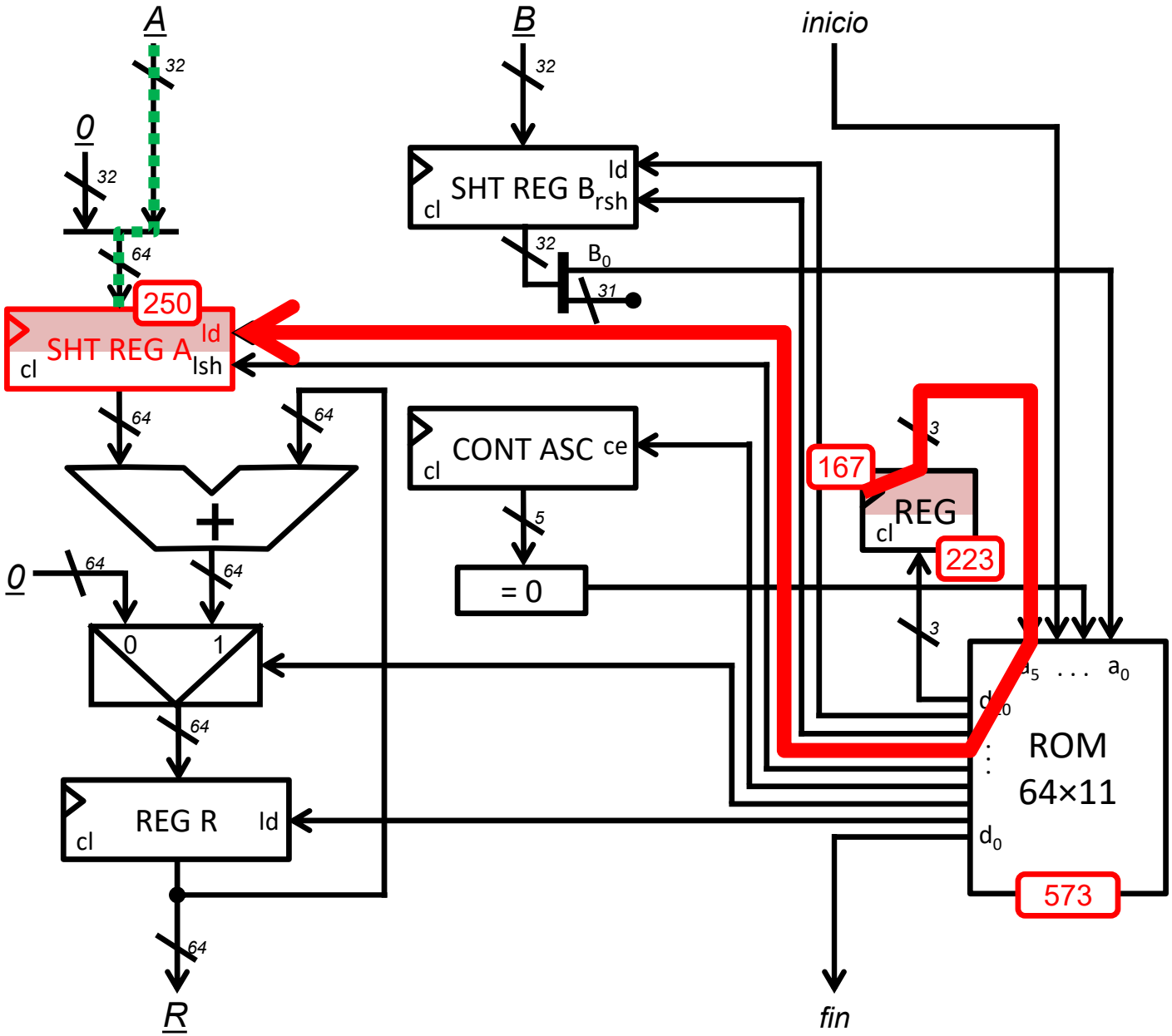
Cálculo del coste (CMOS 90 nm)



área: 12033 μm²

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

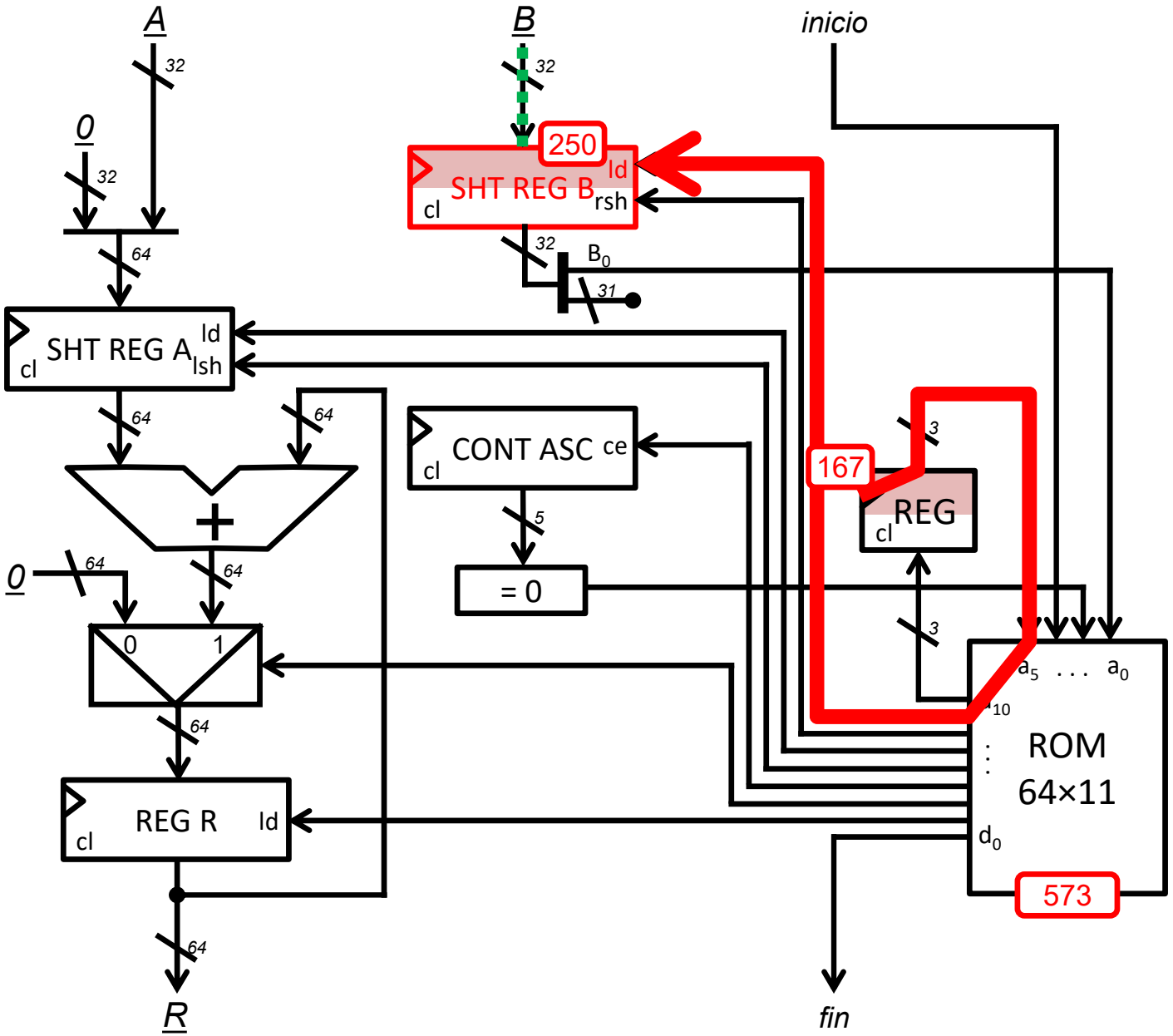


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

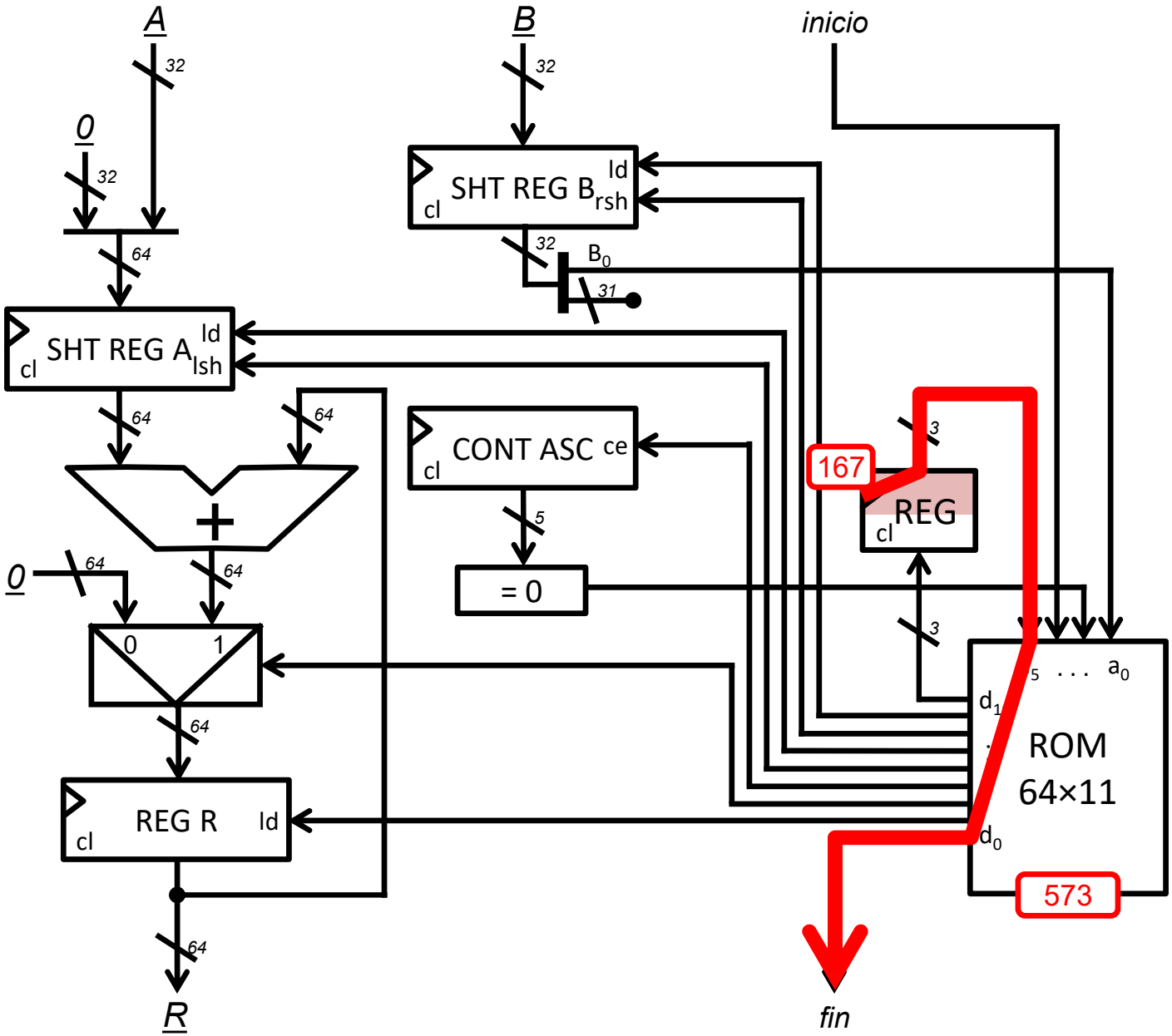


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

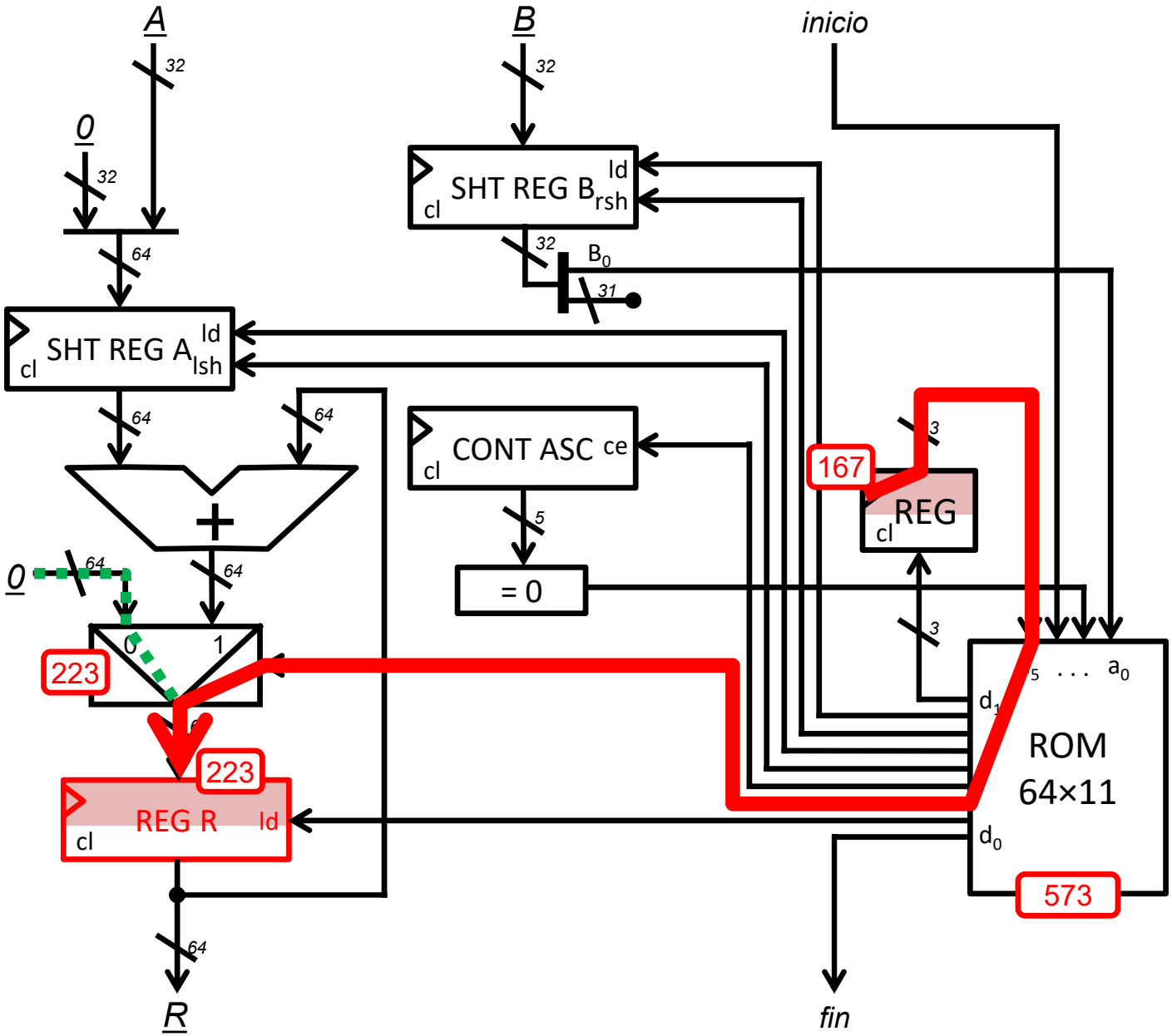


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

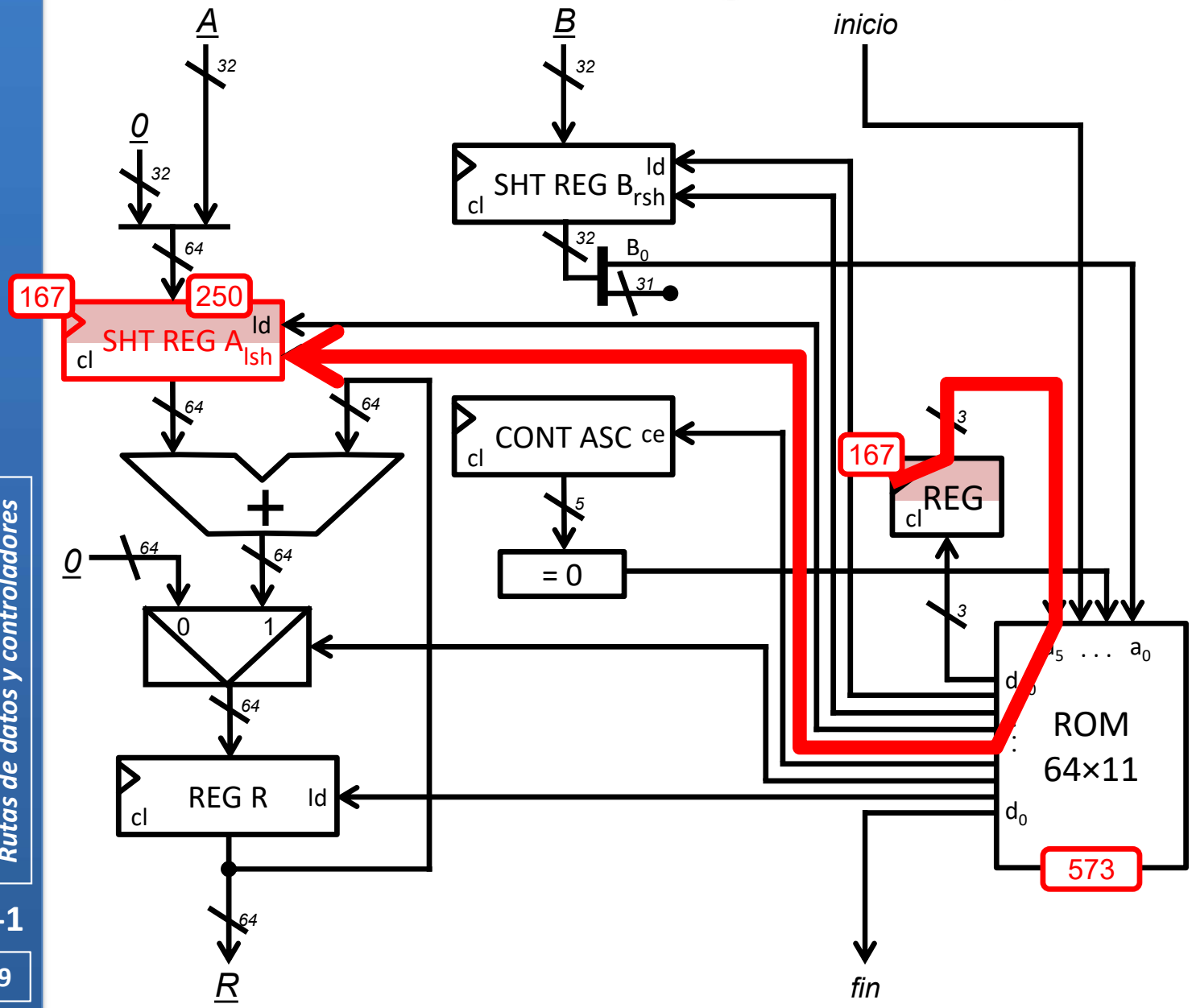


área: 12033 μm²

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

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

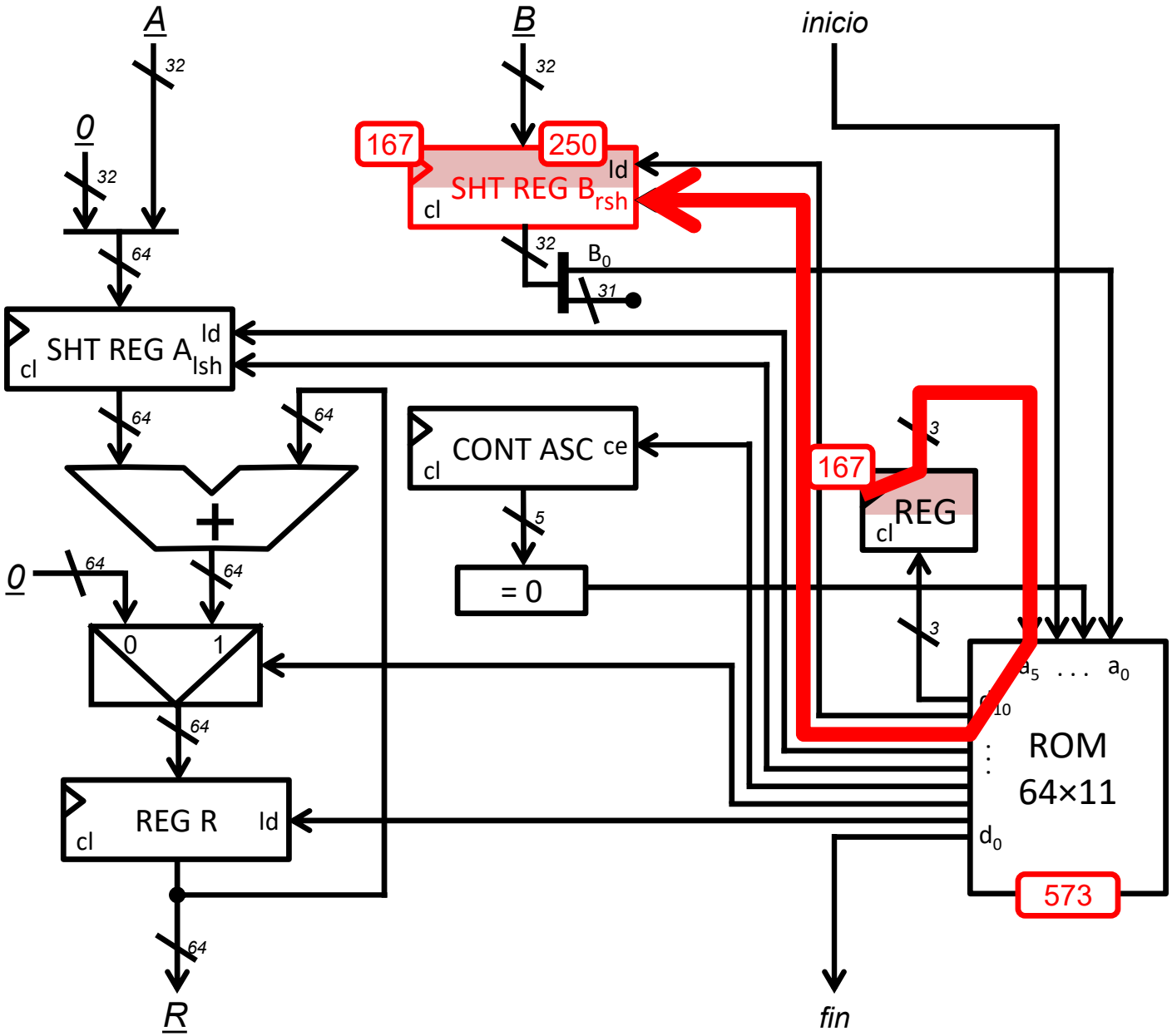


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

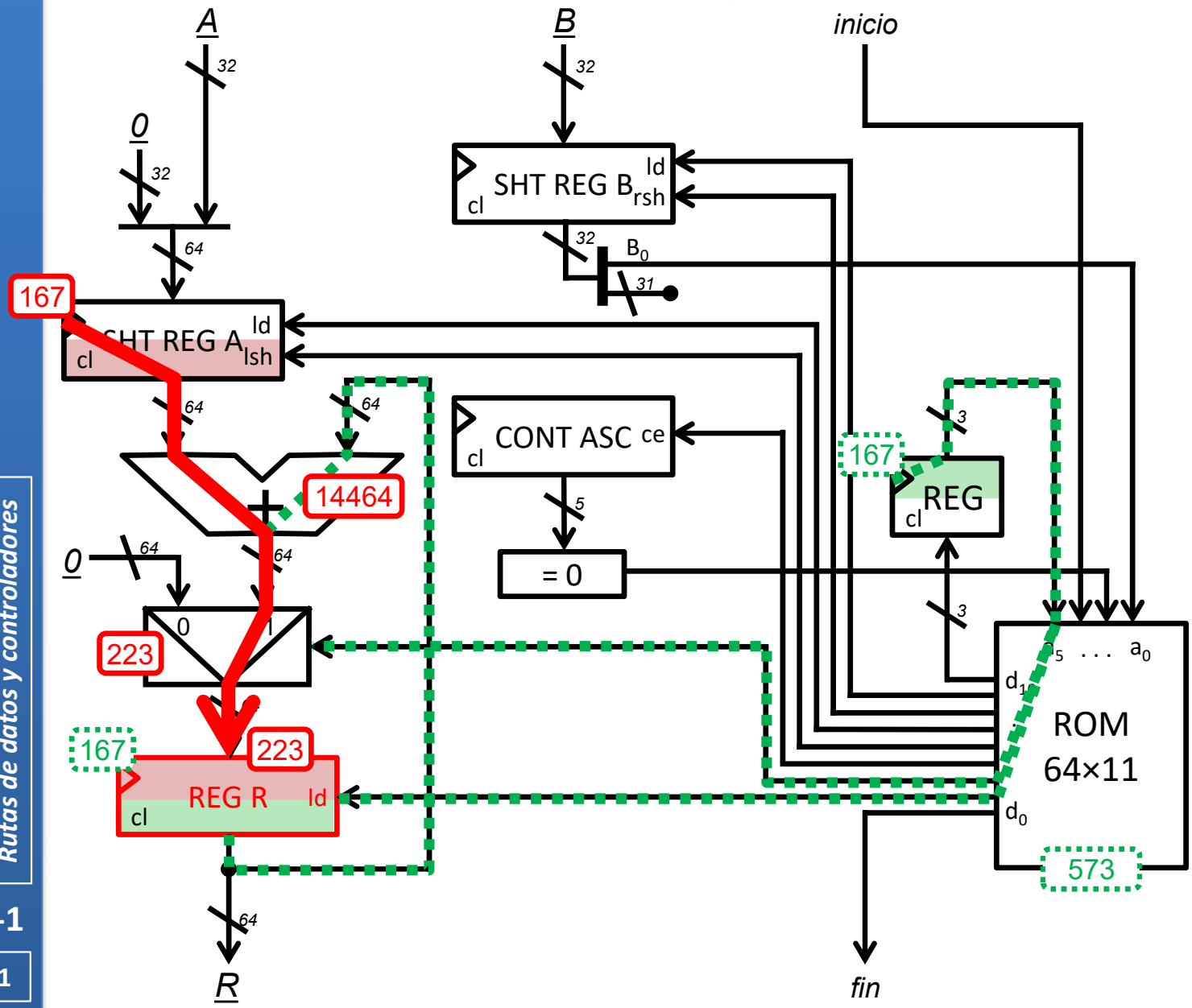


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps
RB ← RB >> 1	990 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

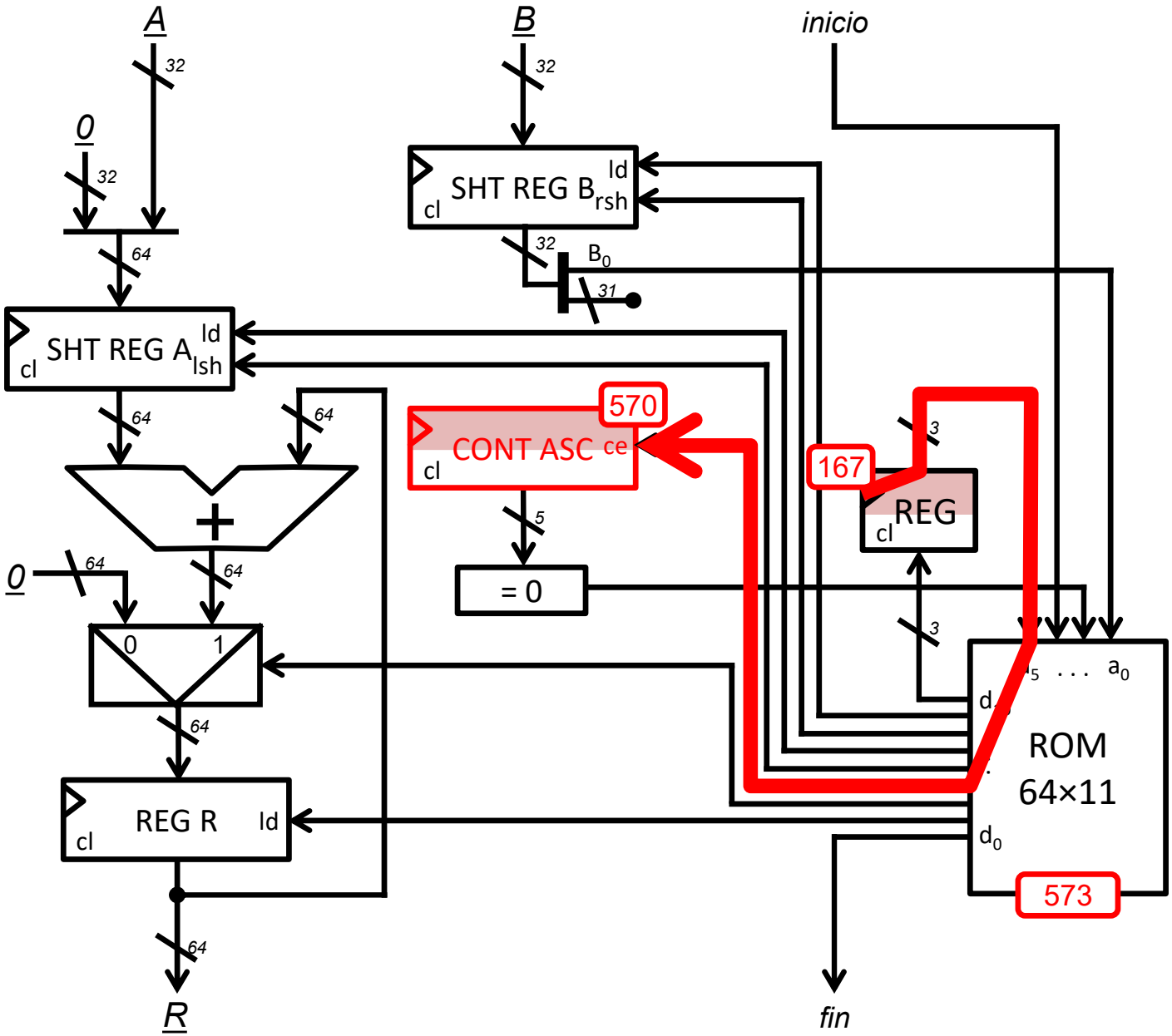


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps
RB ← RB >> 1	990 ps
RR ← RA + RR	15077ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

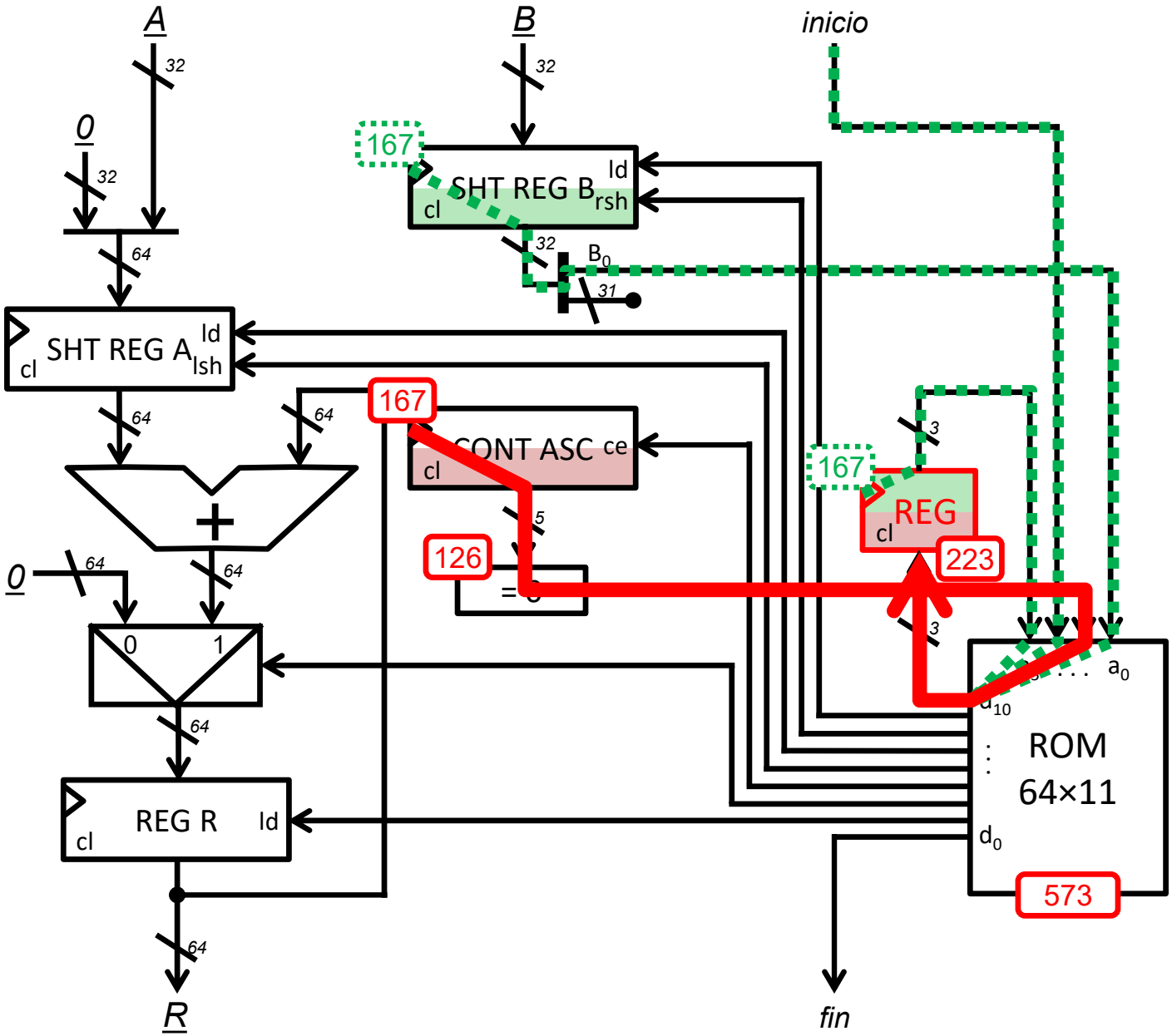


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps
RB ← RB >> 1	990 ps
RR ← RA + RR	15077ps
RC ← RC + 1	1350 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)

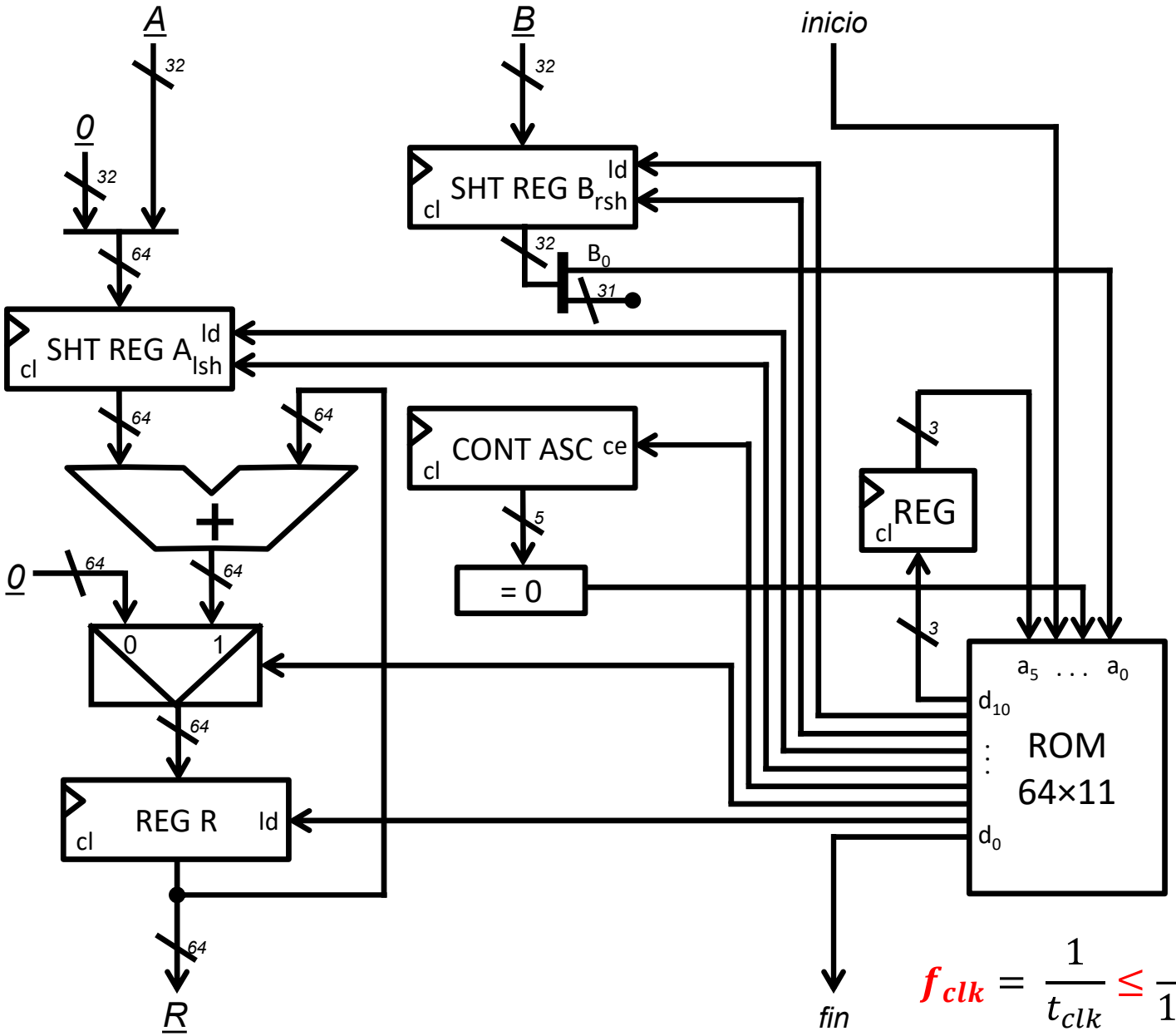


área: 12033 μm²

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps
RB ← RB >> 1	990 ps
RR ← RA + RR	15077ps
RC ← RC + 1	1350 ps
cálculo de estado	1089 ps

Multiplicador iterativo

Cálculo del tiempo de ciclo (CMOS 90 nm)



área: 12033 μm²

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

transferencia	retardo
RA ← Ain	990 ps
RB ← Bin	990 ps
fin ← X	740 ps
RR ← 0	1186 ps
RA ← RA << 1	990 ps
RB ← RB >> 1	990 ps
RR ← RA + RR	**15077ps**
RC ← RC + 1	1350 ps
cálculo de estado	1089 ps
máximo	**15077ps**

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

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/