P1-MULTIPLICADOR SECUENCIAL

Arquitecturas Hardware de Comunicaciones

Multiplicador secuencial. Algoritmo.

- Algoritmo de "lápiz y papel" de un multiplicador secuencial de números sin signos codificados en binario.
  - Ejemplo: 13x10 ; M(m) x Q(q) (siendo m=q=4 bits)

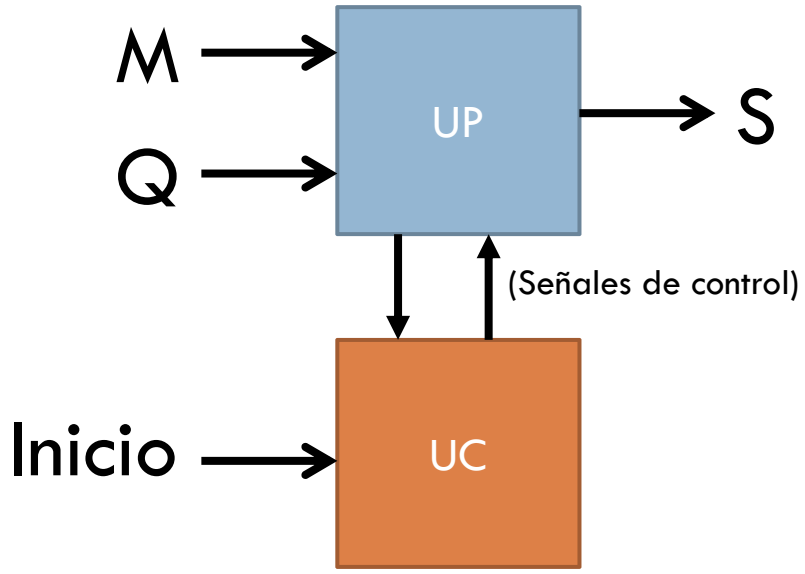
$$
\begin{array}{rl}
1101 & \rightarrow 13 \\
\times\ 1010 & \rightarrow 10 \\
\hline
0000 & \\
1101 & \\
0000 & \\
1101 & \\
\hline
10000010 & \rightarrow 130
\end{array}
$$

M: Multiplicando

Q: Multiplicador

Multiplicador secuencial. Diseño.

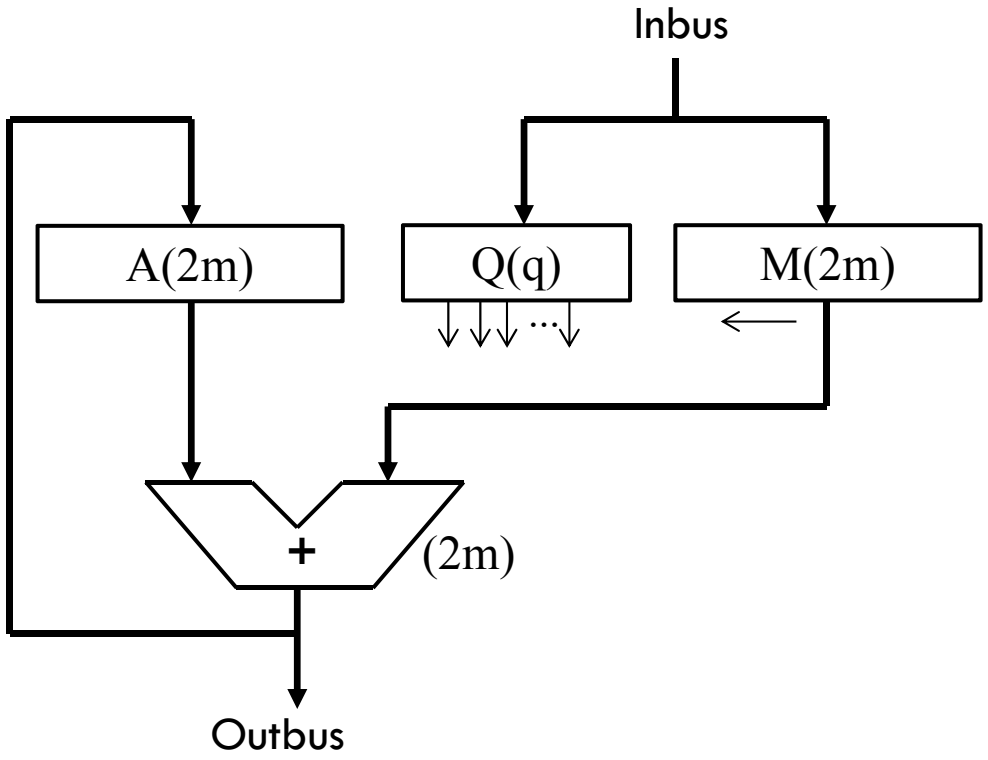
- Estrategia de diseño secuencial formada por UP y UC.
  - UP: Elementos de memoria, recursos de cálculo y recursos de conexión.
  - UC: Determinará el funcionamiento de la UP mediante FSM.



UP: Unidad de proceso
UC: Unidad de control

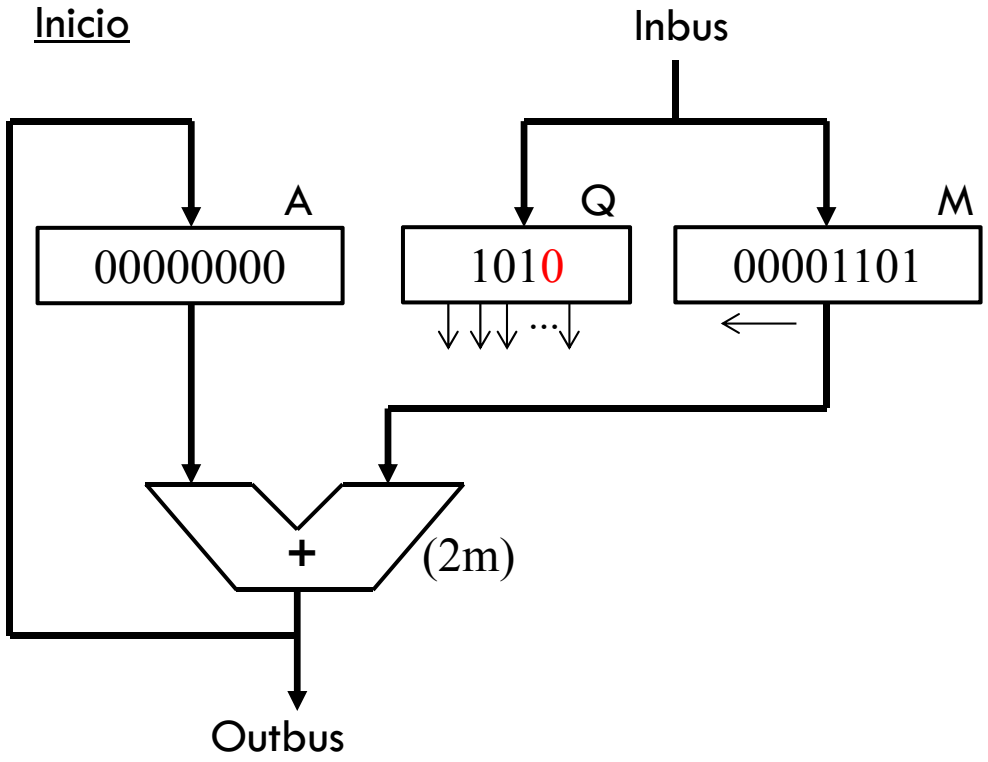
Multiplicador secuencial. Diseño.

- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda



Multiplicador secuencial. Diseño.

- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda



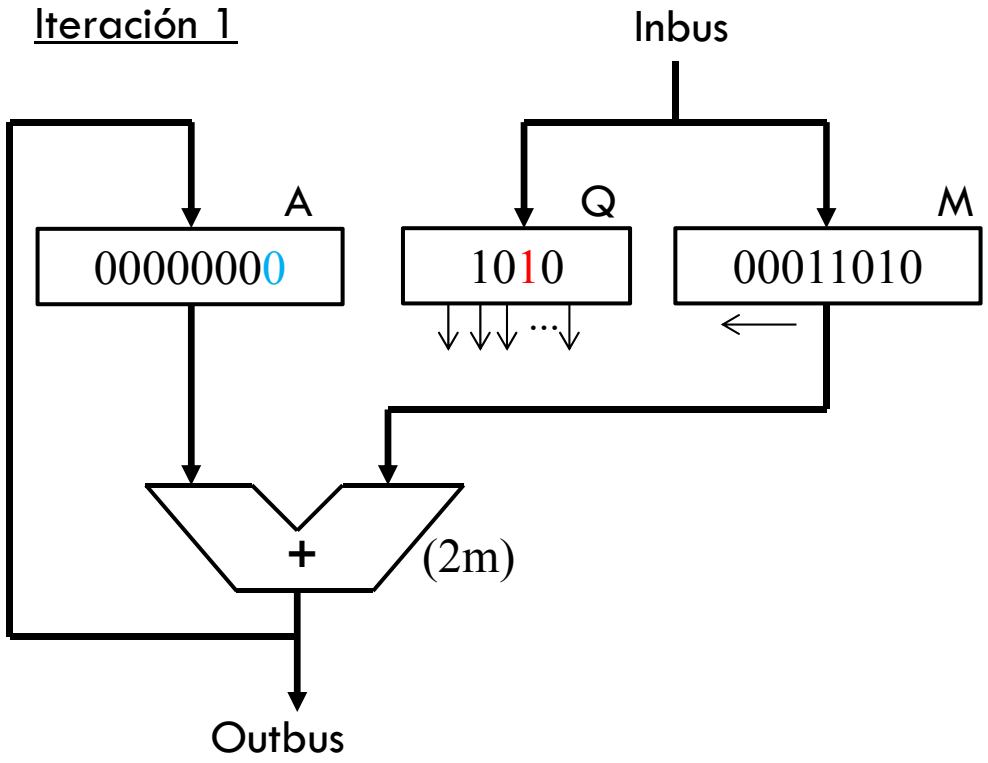
00000000 A

\+ 0000 M (Q(0)=0)

00000000 A

Multiplicador secuencial. Diseño.

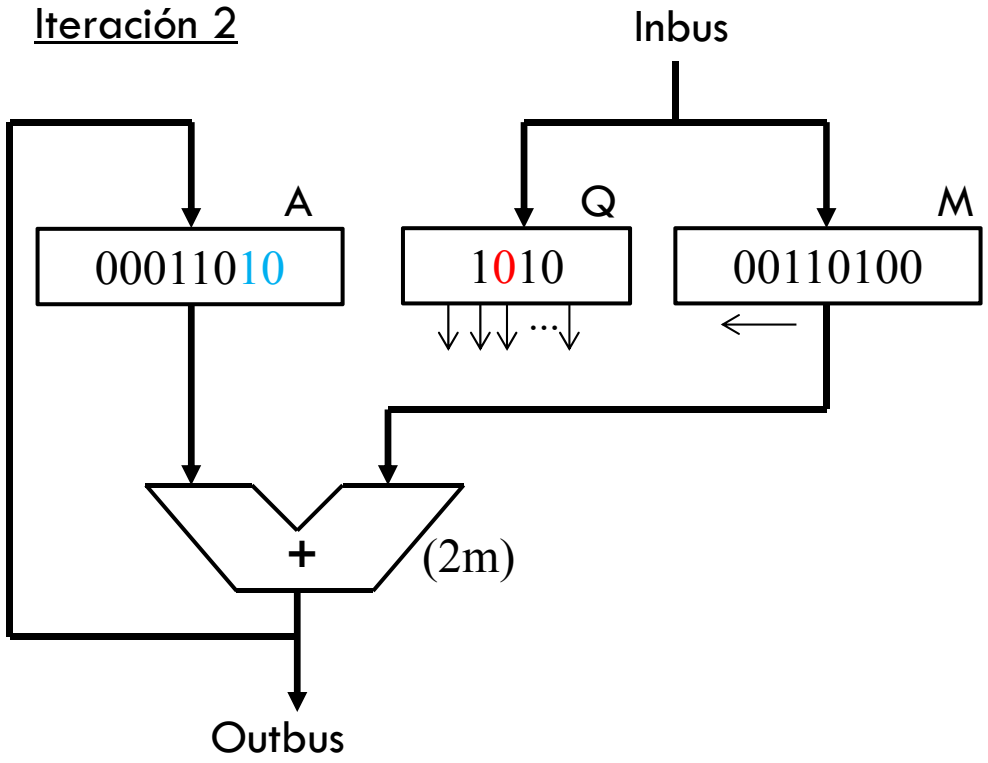
- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda



00000000 A
+ 0000 M (Q(0)=0)
00000000 A
+ 1101 ← M (Q(1)=1)
00011010 A

Multiplicador secuencial. Diseño.

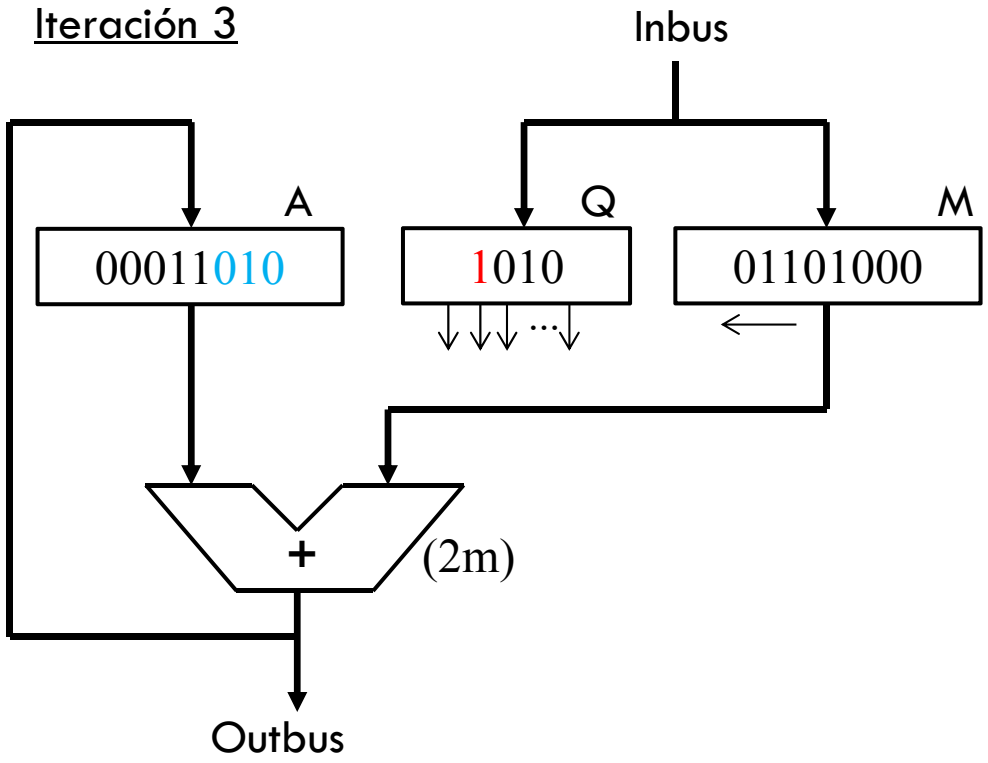
- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda




```
00000000        A
   + 0000       M (Q(0)=0)
--------
00000000        A
 + 1101     ←   M (Q(1)=1)
--------
00011010        A
 + 0000     ←   M (Q(2)=0)
--------
00011010        A
```

Multiplicador secuencial. Diseño.

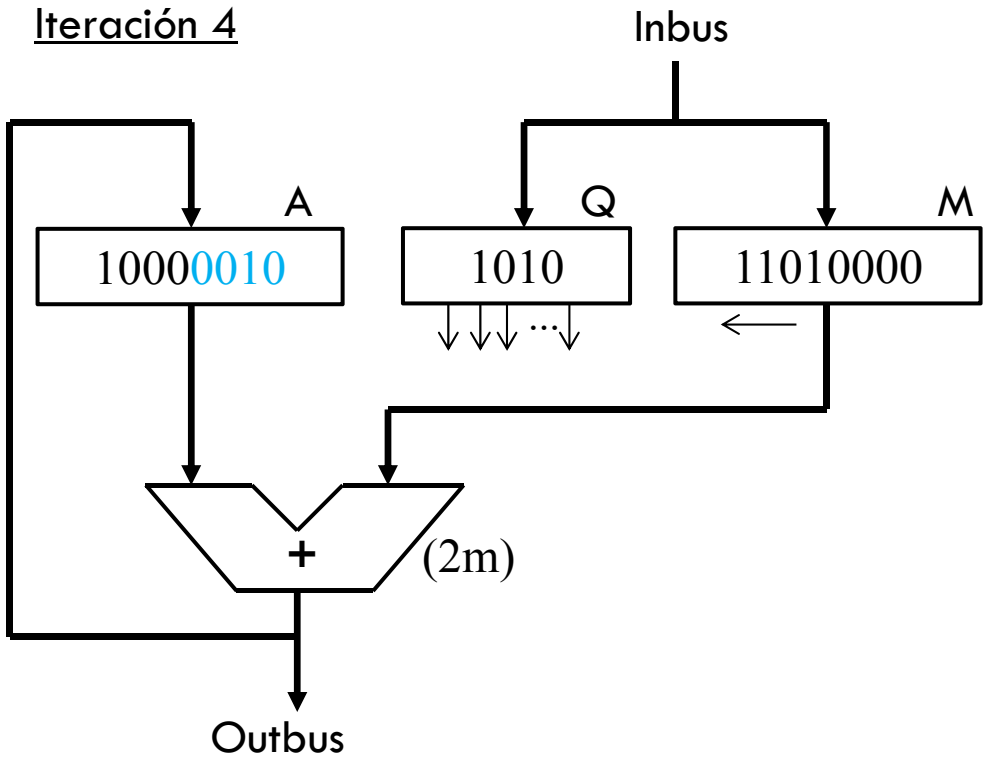
- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda




```
00000000      A
   + 0000     M (Q(0)=0)
--------
00000000      A
  + 1101   ←  M (Q(1)=1)
--------
00011010      A
 + 0000    ←  M (Q(2)=0)
--------
00011010      A
+ 1101     ←  M (Q(3)=1)
--------
10000010
```

Multiplicador secuencial. Diseño.

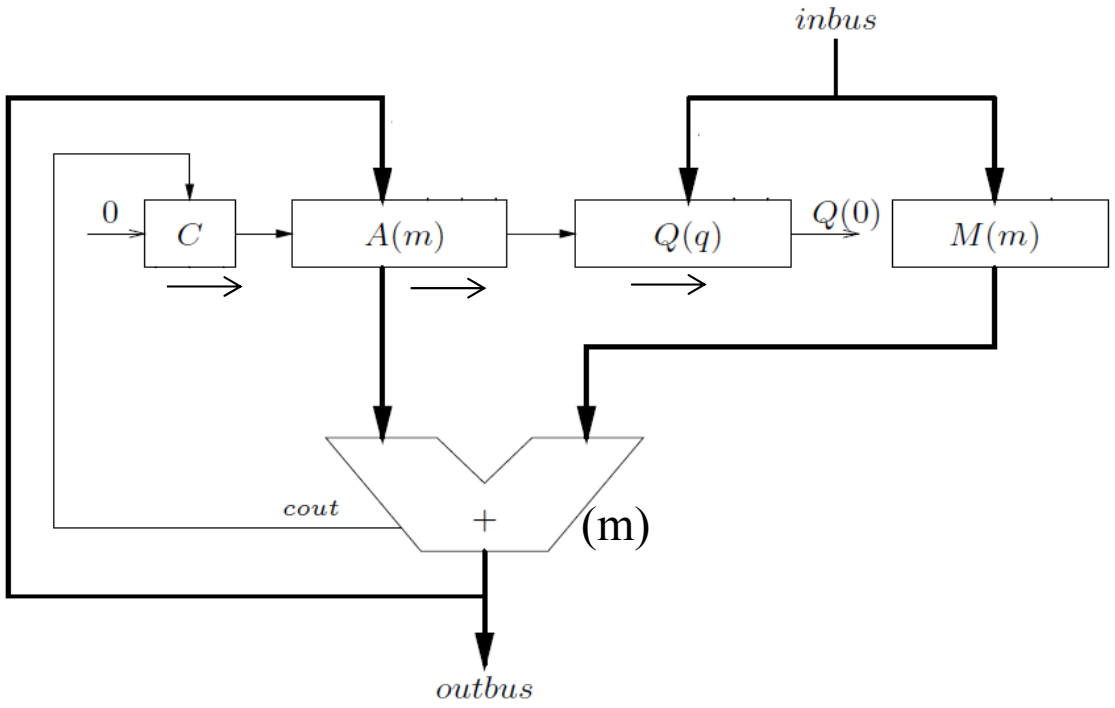
- UP. Primera aproximación.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazamiento del Multiplicando (M) a la izquierda




```
00000000      A
   + 0000     M (Q(0)=0)
--------
00000000      A
 + 1101   ←   M (Q(1)=1)
--------
00011010      A
 + 0000   ←   M (Q(2)=0)
--------
00011010      A
+ 1101    ←   M (Q(3)=1)
--------
10000010
```

Multiplicador secuencial. Diseño.

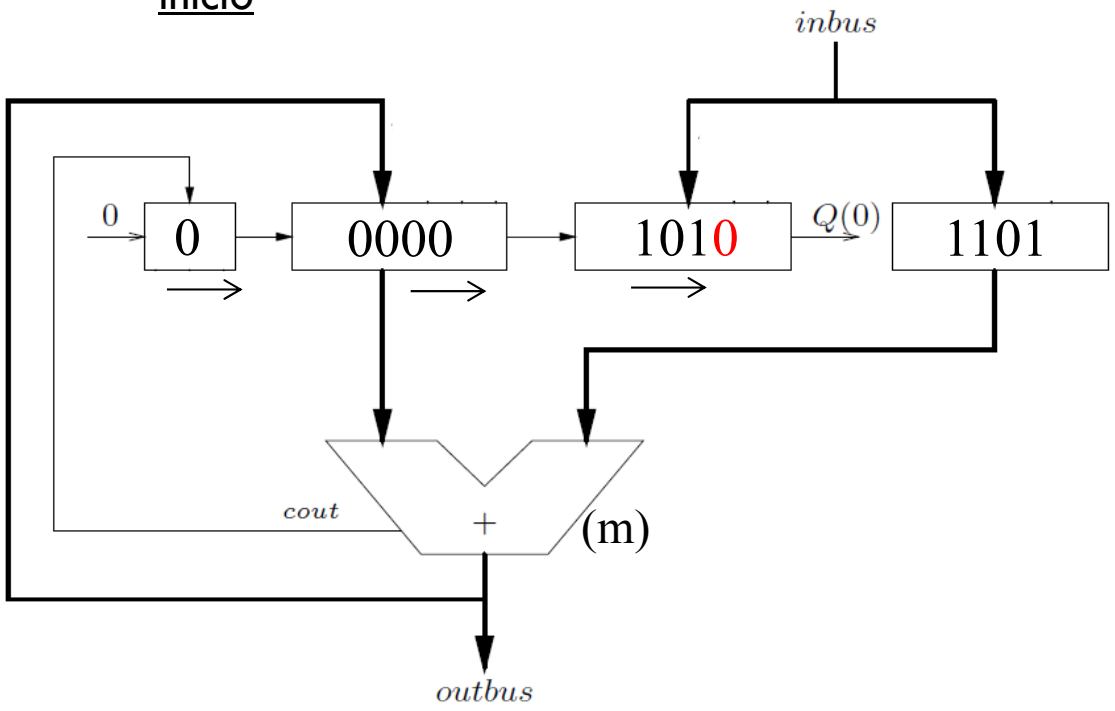
- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

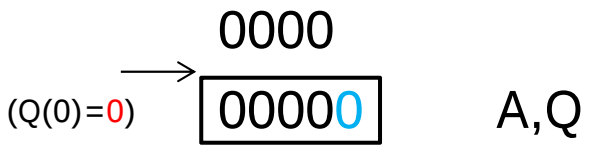


Multiplicador secuencial. Diseño.

- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

Inicio

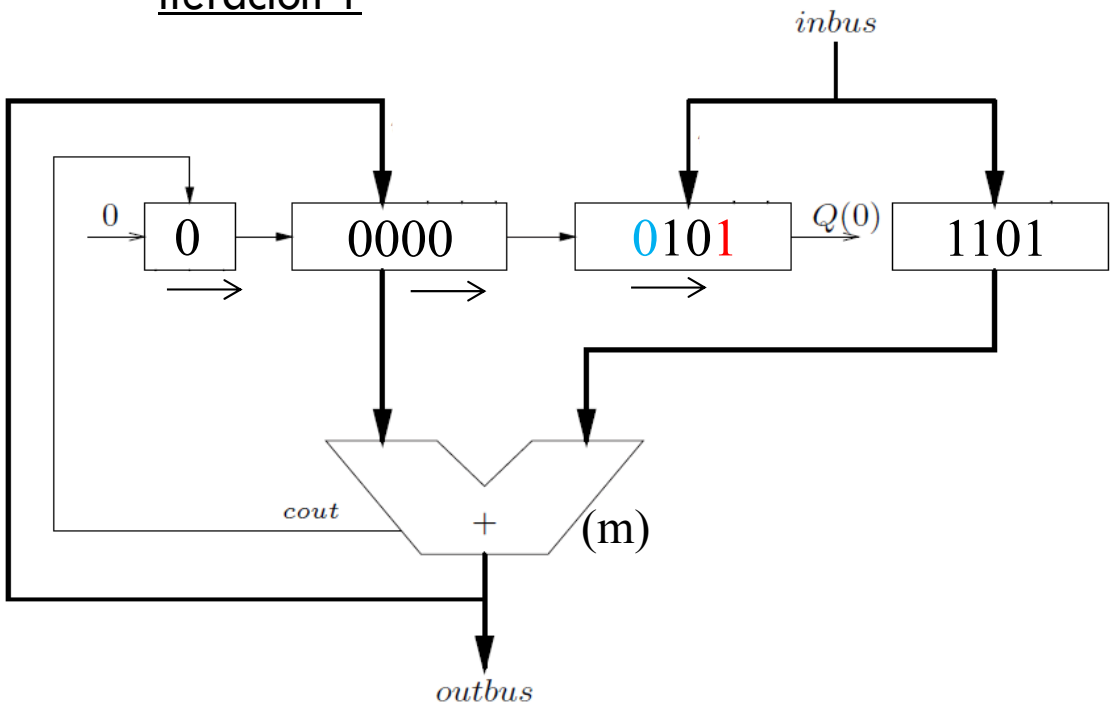




Multiplicador secuencial. Diseño.

- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

Iteración 1



0000

(Q(0)=0) → 00000 A,Q

(Q(0)=1) + 1101

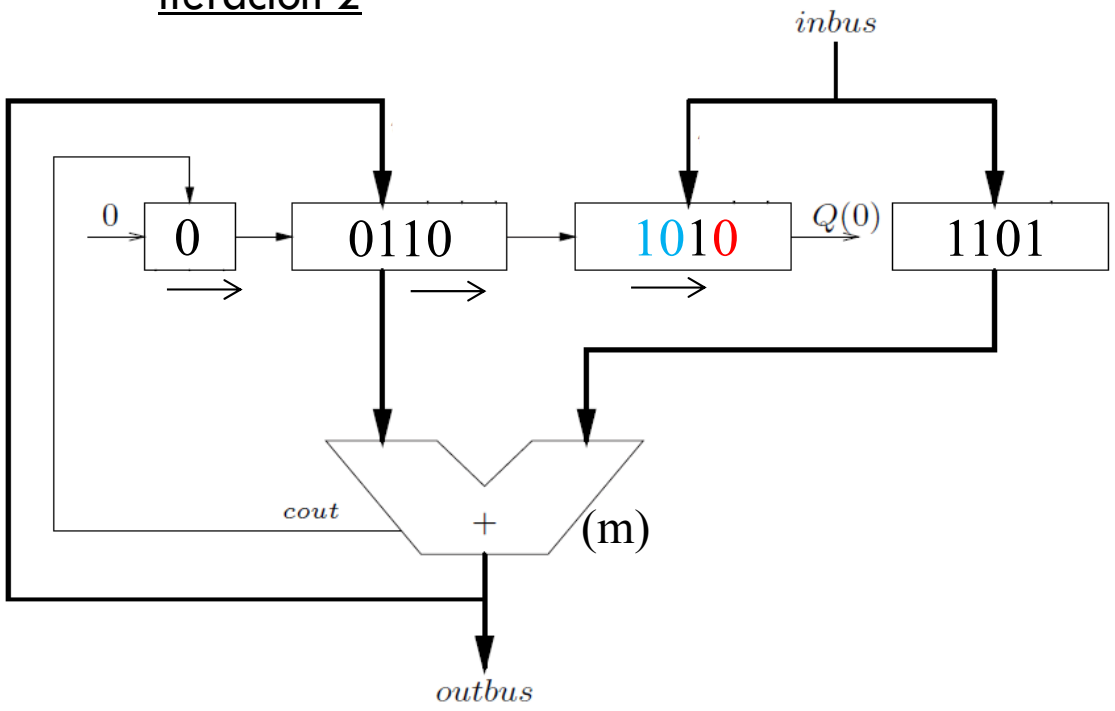
11010

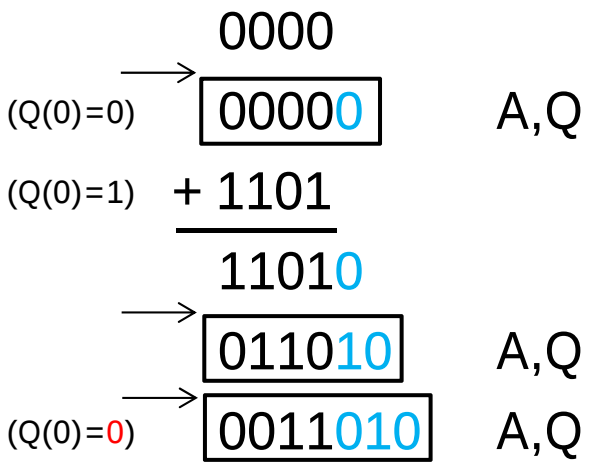
→ 011010 A,Q

Multiplicador secuencial. Diseño.

- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

Iteración 2

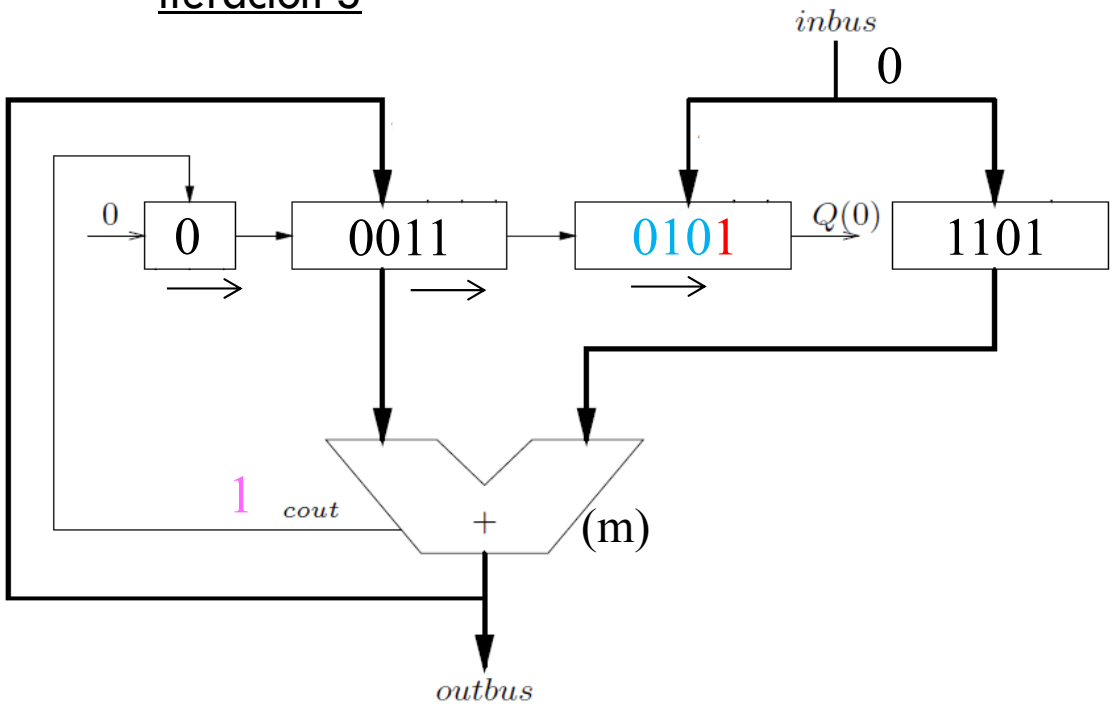




Multiplicador secuencial. Diseño.

- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

Iteración 3

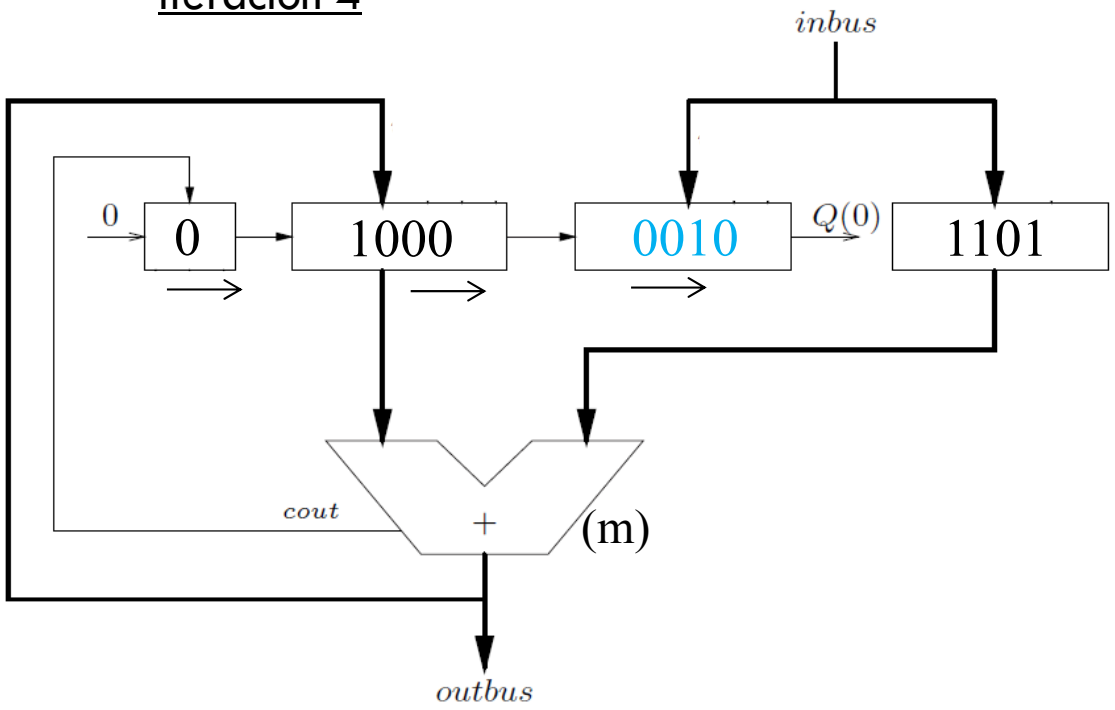



```
                0000
(Q(0)=0)  →   00000        A,Q
(Q(0)=1)      + 1101
              11010
          →   011010       A,Q
(Q(0)=0)  →   0011010      A,Q
(Q(0)=1)      + 1101
              10000010
          →   10000010     A,Q
```

Multiplicador secuencial. Diseño.

- UP. Estructura final.
  - Sumar los resultados parciales 2 a 2 → Un solo sumador de 2 entradas.
  - Desplazar el producto parcial (CA) y Multiplicador (Q) a la derecha.

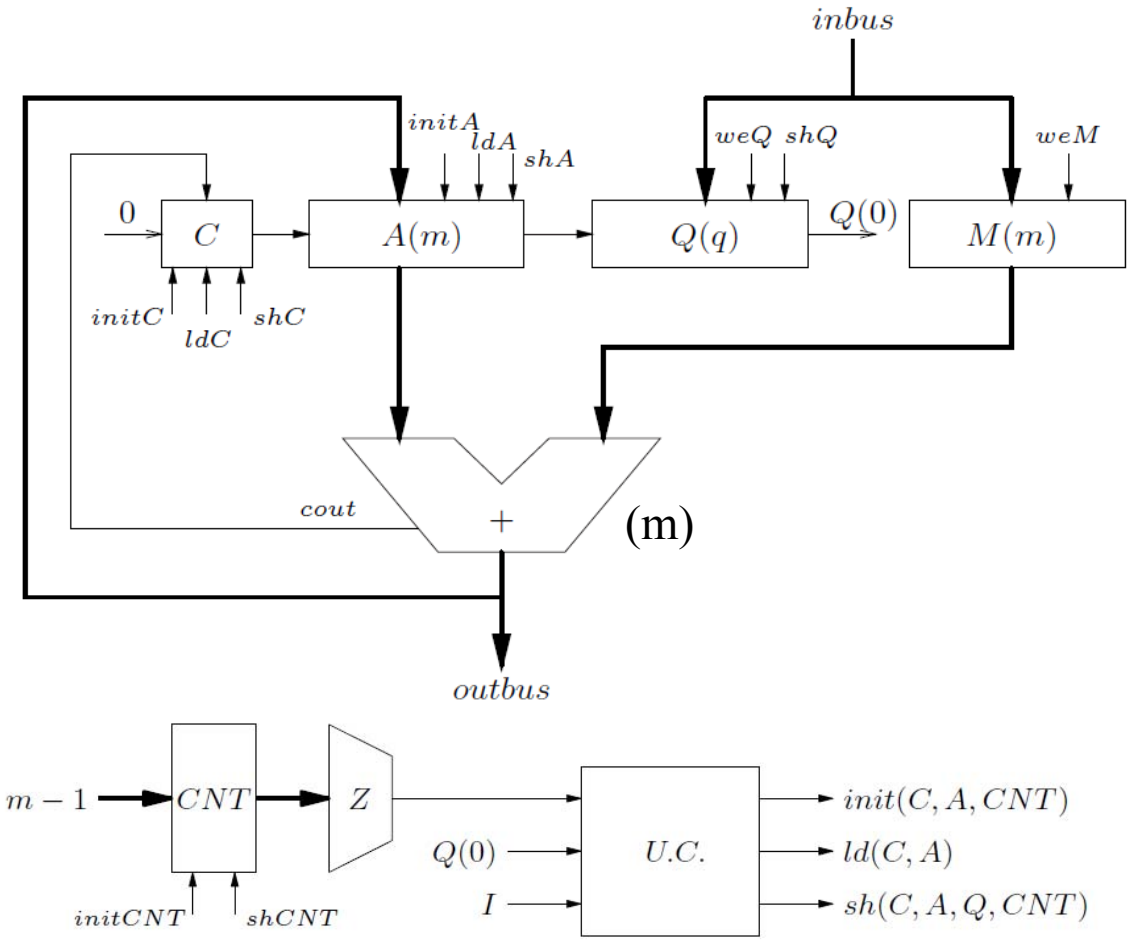
Iteración 4




```
                0000
(Q(0)=0)  →   00000
(Q(0)=1)      + 1101
              11010
          →   011010
(Q(0)=0)  →   0011010
(Q(0)=1)      + 1101
              10000010
          →   10000010
```

Multiplicador secuencial. Diseño.

- UP. Características de los recursos y señales de control.



UP

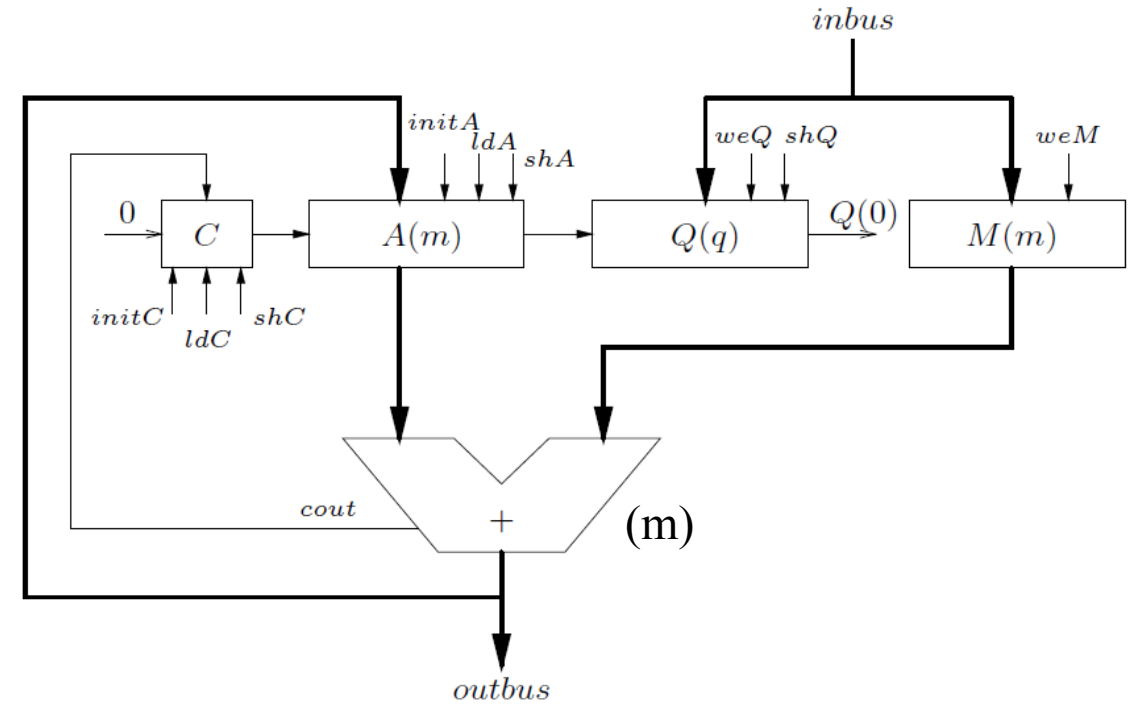
Componente	Operación RTL	Señal de Control
Registro M	$M \leftarrow inbus$	–
Registro Q	$Q \leftarrow inbus$	–
	$RShift(Q)$	shQ
Registro A	$A \leftarrow 0$	$initA$
	$A \leftarrow A + M$	ldA
	$RShift(A)$	shA
Registro C	$C \leftarrow 0$	$initC$
	$C \leftarrow C_{out}$	ldC
	$RShift(C)$	shC
Contador	$CNT \leftarrow n - 1$	$initCNT$
	$CNT \leftarrow CNT - 1$	$shCNT$

UC

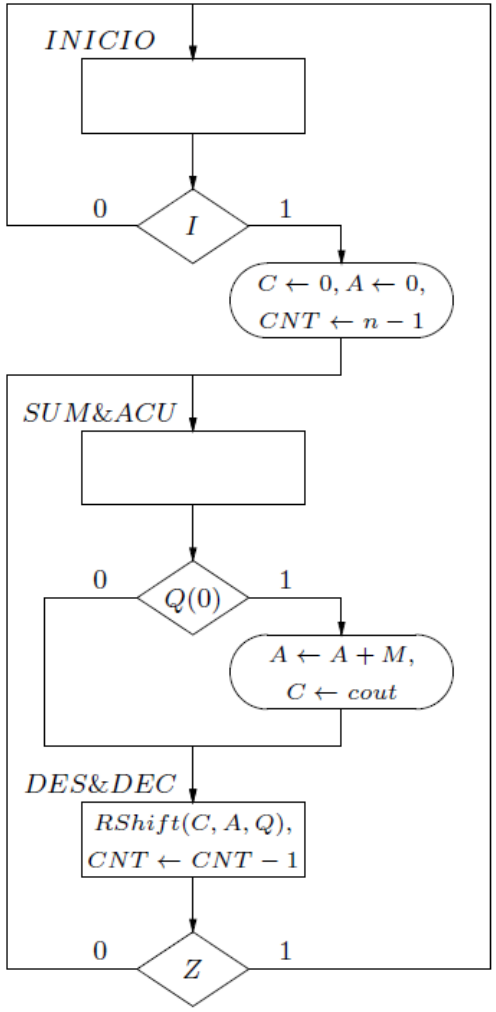
- I, inicio
- Q(0) determinar si des o des y sum.
- Z, Finalización multiplicación.

Multiplicador secuencial. Diseño.

- UC. Diagrama ASM del multiplicador secuencial.

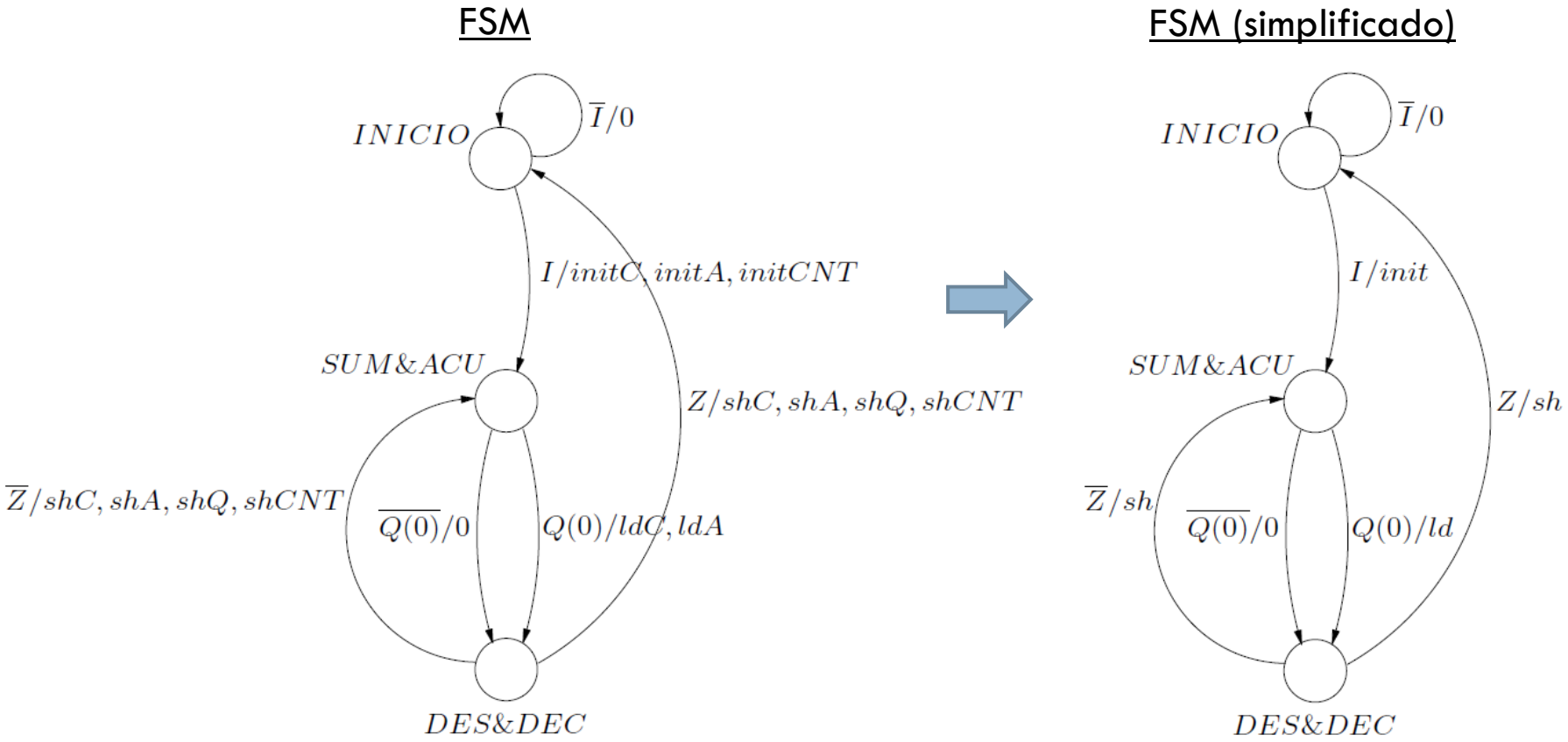


ASM



Multiplicador secuencial. Diseño.

- UC. Diagrama de estados (FSM) y simplificaciones.
  - Reducción de señales de control.



Multiplicador secuencial. Implementación

- VHDL: Entidad y parte declarativa.

```
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.ALL;
 3  use ieee.numeric_std.all;
 4
 5  entity mult_sec is
 6      Port ( inbus  : in std_logic_vector(7 downto 0);
 7             outbus : out std_logic_vector(15 downto 0);
 8             I      : in std_logic;
 9             weQ    : in std_logic;
10             weM    : in std_logic;
11             rst    : in std_logic;
12             clk    : in std_logic);
13  end mult_sec;
14
15  architecture Behavioral of mult_sec is
16
17     type type_state is (INICIO, SUMACU, DESDEC);
18     signal state, nextstate: type_state;
19
20     signal CA, pp: std_logic_vector(8 downto 0);
21     signal Q, M: std_logic_vector(7 downto 0);
22     signal init, ld, sh: std_logic;
23     signal cnt: unsigned (2 downto 0);
24     signal Z: std_logic;
```

Multiplicador secuencial. Implementación

- VHDL: Registros C y A.

```
28    --Registro C y A:
29    process(rst, clk)
30    begin
31       if (rst='1') then
32          CA <= (others=>'0');
33       elsif rising_edge(clk) then
34          if (init='1') then
35             CA <= (others=>'0');
36          elsif (ld='1') then
37             CA <= pp;
38          elsif (sh='1') then
39             CA <= '0' & CA(8 downto 1);
40          end if;
41       end if;
42    end process;
```

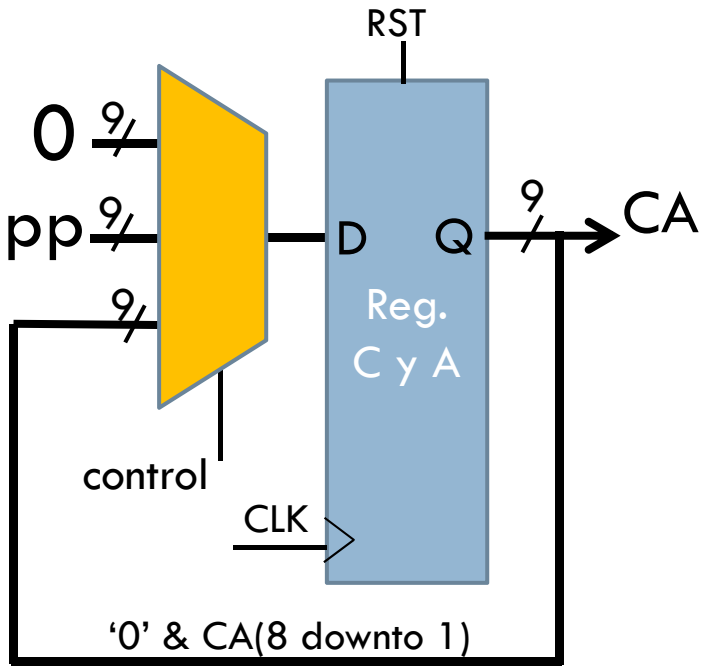
- VHDL: Sumador y salida.

```
70    --Sumador:
71    pp <= std_logic_vector(unsigned('0' & CA(7 downto 0)) + unsigned('0' & M));
72
73    --Salida:
74    outbus <= CA(7 downto 0) & Q;
```

Multiplicador secuencial. Implementación

- VHDL: Registros C y A.

```
28    --Registro C y A:
29    process(rst, clk)
30    begin
31       if (rst='1') then
32          CA <= (others=>'0');
33       elsif rising_edge(clk) then
34          if (init='1') then
35             CA <= (others=>'0');
36          elsif (ld='1') then
37             CA <= pp;
38          elsif (sh='1') then
39             CA <= '0' & CA(8 downto 1);
40          end if;
41       end if;
42    end process;
```



- VHDL: Sumador y salida.

```
70    --Sumador:
71    pp <= std_logic_vector(unsigned('0' & CA(7 downto 0)) + unsigned('0' & M));
72
73    --Salida:
74    outbus <= CA(7 downto 0) & Q;
```

Multiplicador secuencial. Implementación

- VHDL: Detector de cero.

```
90    --Detector de cero:
91    process(cnt)
92    begin
93       if (cnt="000") then
94          Z <= '1';
95       else
96          Z <= '0';
97          end if;
98    end process;
```

- VHDL: Registros Q, M, Contador, UC

¿?