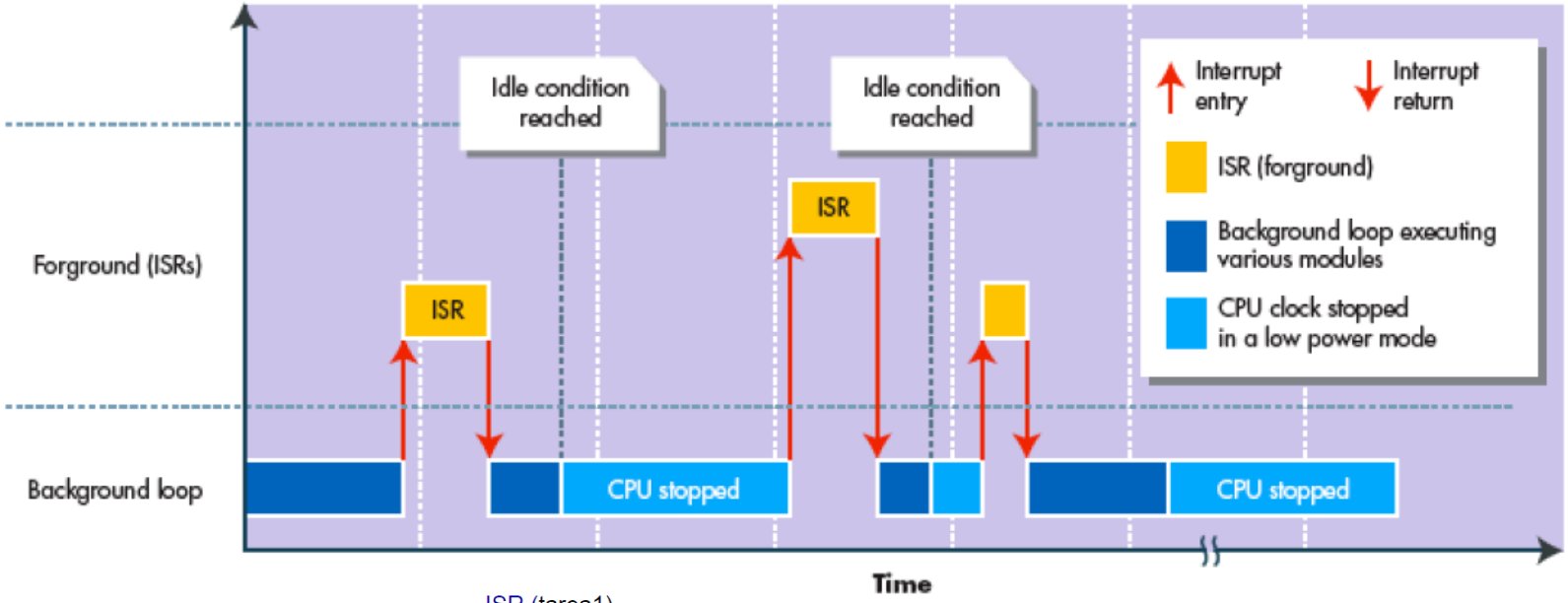
Esquemas de Software

Foreground-Background




```
main (void)
{
 tarea1_Init();
 tarea2_Init();
 tarea3_Init();

 sei();

 while(1)
 {
   if(evento_tarea1)
     tarea1();
   if(evento_tarea2)
     tarea2();
   if(evento_tarea3)
     tarea3();
   }
}
```

Tareas en Background

Comunicación por flags globales y buffers

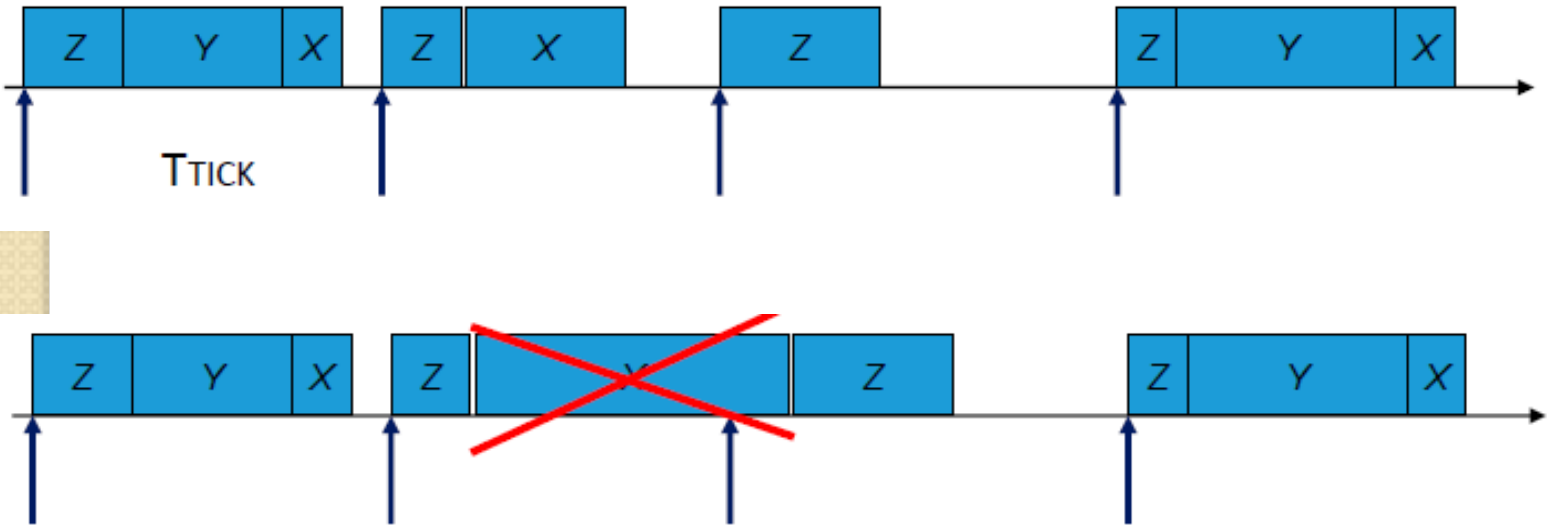
```
ISR (tarea1)
{
…
}

ISR (tarea2)
{
…
}

ISR (tarea3)
{
…
}
```

Tareas en Foreground

Time-Triggering

Temporización básica con Ticks

Scheduling básico



Temporización: Planificación-Despacho de Tareas

```
Volatile unsigned char Flag_X=0,
Volatile unsigned char Flag_Y=0;
Volatile unsigned char Flag_Z=0;

static unsigned char contX=0,
static unsigned char contY=0;
static unsigned char contZ=0;

/* ------------------------------------------------------------ */
---ISR TIMER : ocurre cada 1 ms
/* ------------------------------------------------------------ */
_interrupt void ISRrtc (void)
{
 SEOS_SCHTasks();
 RTCSC_RTIF=0;
}
```

```
/* ------------------------------------------------------------ */
Variables públicas del planificador
/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */
Variables privadas del planificador
/* ------------------------------------------------------------ */

void SEOS_SCHTasks (void)
{
  if (++contX==200) {
     Flag_X=1;  //Tarea programada  cada  200 ms
     contX=0;
  }
  if (++contY==50) {
     Flag_Y=1;  //Tarea programada  cada  50 ms
     contY=0;
  }
  if (++contZ==10) {
     Flag_Z=1;  //Tarea programada  cada  10 ms
     contZ=0;
  }
}
```

Planificador

Temporización de Tareas

Despachador

Prioridad

```
/*------------------------------------------------------------
Main.c
------------------------------------------------------------*/
#include "device.h"
#include "seos.h"

/* ------------------------------------------------------------ */
void main(void)
{
     // Inicializar MCU, RTC, y tareas

     for(;;) {   // Super Loop

        SEOS_Dispatch_Tasks();
        SEOS_Go_To_Sleep();
      }
} /* ------------------------------------------------------------ */
```

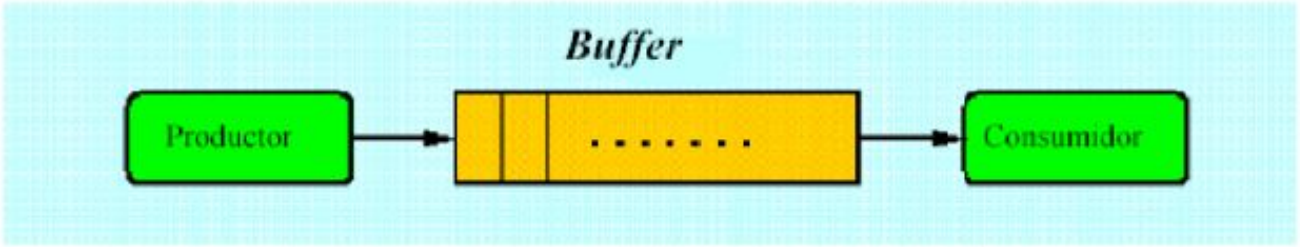
```
void SEOS_Dispatch_Tasks(void) {
    if (Flag_Z) {
        Z();  //Tarea programada  cada  10 ms
        Flag_Z =0;
    }
    if (Flag_Y) {
        Y();  //Tarea programada  cada  50 ms
        Flag_Y =0;
    }
    if (Flag_X) {
        X();  //Tarea programada  cada  200 ms
        Flag_X =0;
    }
}
```

Interrupción Periódica

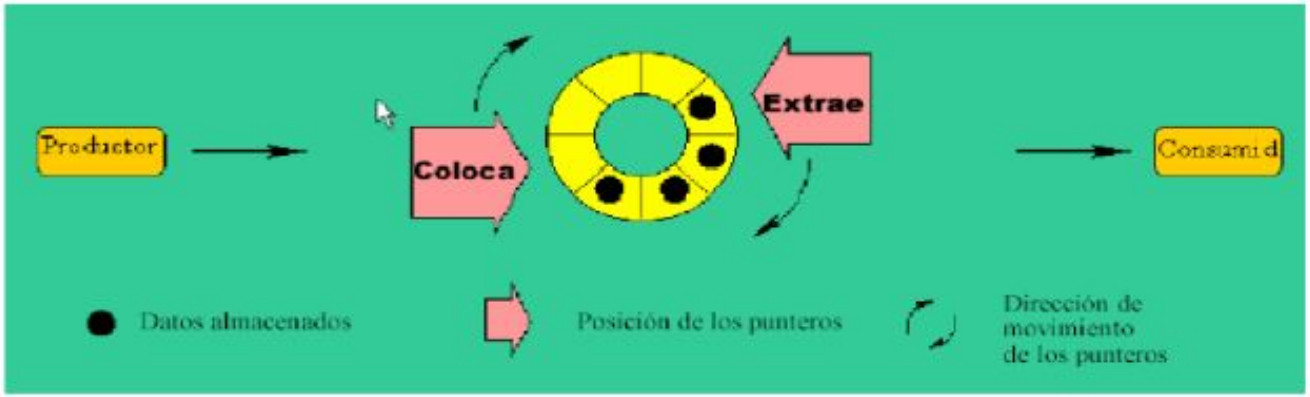
Productor-Consumidor

Problema: El productor envía un dato cada vez, y el consumidor consume un dato cada vez. Si uno de los dos procesos no está listo, el otro debe esperar.

Solución: Es necesario introducir un buffer en el proceso de transmisión de datos

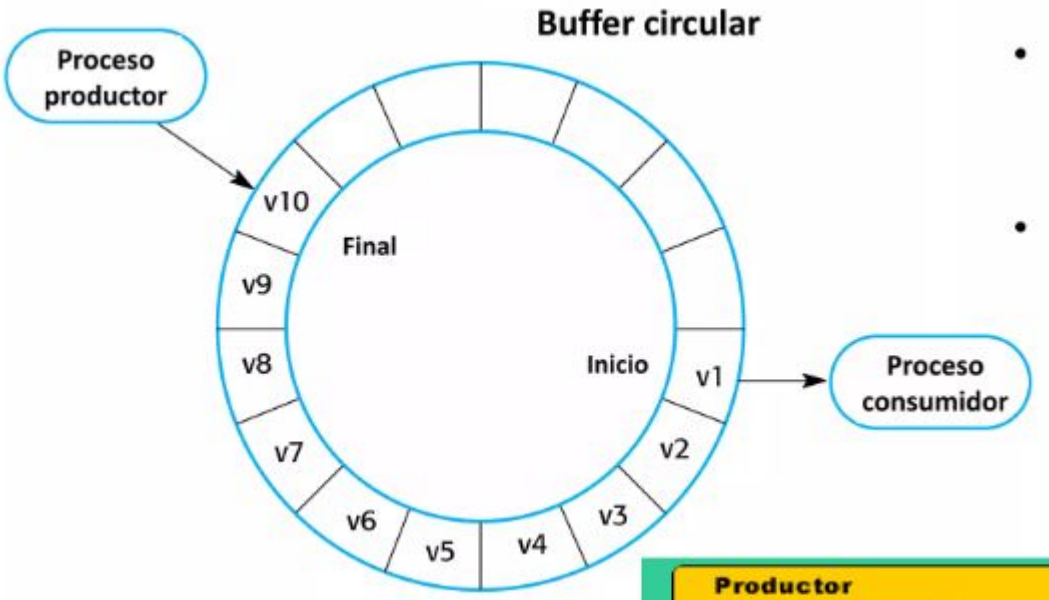


El buffer puede ser infinito. No obstante esto no es realista

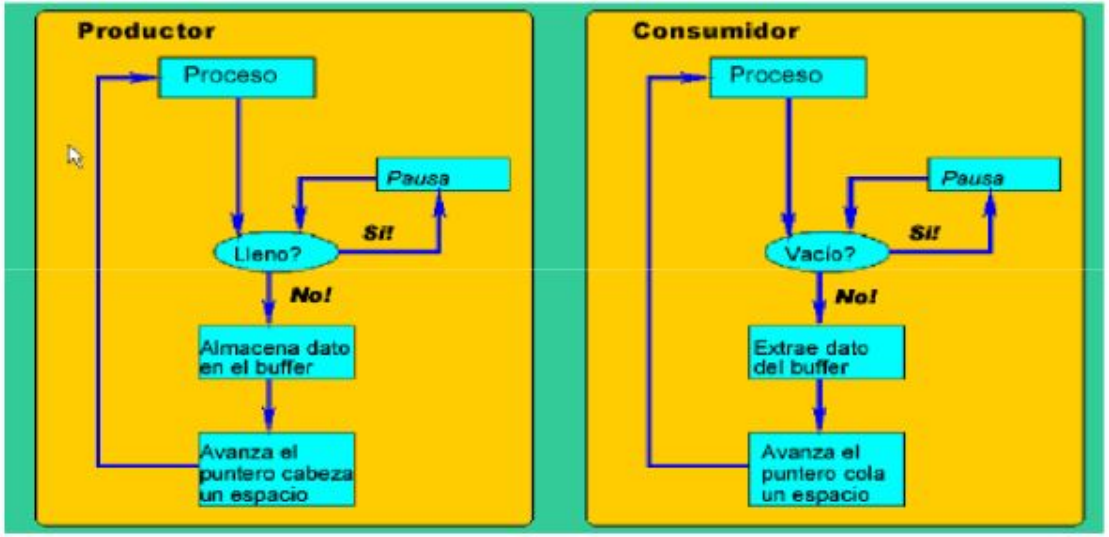


Alternativa: Buffer acotado en cola circular

Productor-Consumidor



- Función Productor
  Almacena en Buffer
  e incrementa puntero de Escritura
- Función Consumidor
  Lee del buffer
  e incrementa puntero de Lectura



Productor-Consumidor

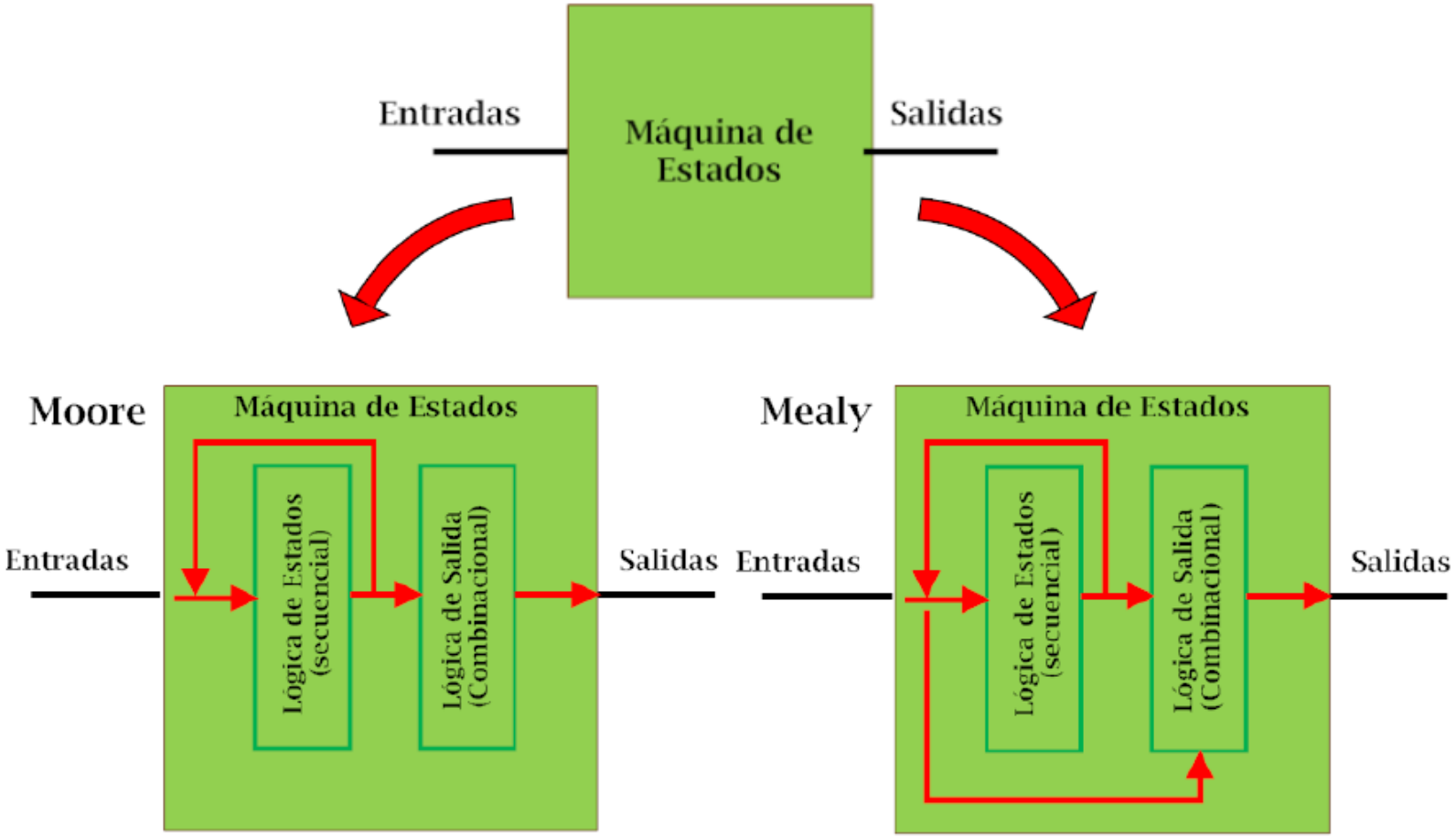
- En el contexto de una arquitectura Background/Foreground una tarea productora puede ser un handler de interrupción y la consumidora una tarea de segundo plano o viceversa.

- En el contexto de la planificación Time-Triggered las tareas que producen y consumen datos a diferentes ritmos se implementan como tareas multietapas (no bloqueantes).

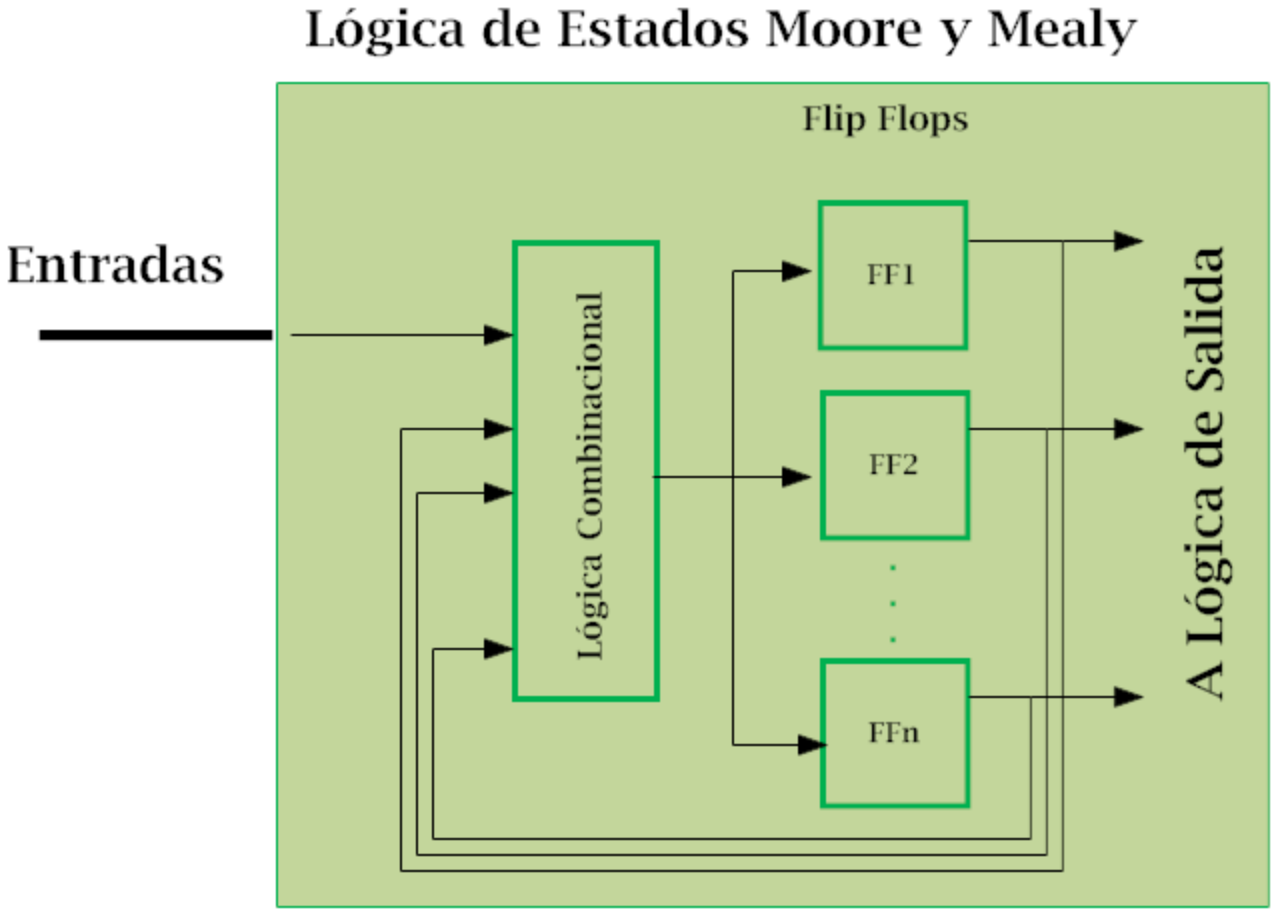
Ejemplos Típicos a los que se pueden aplicar dichos modelos:

Productor	**Consumidor**
•Teclado matricial	tarea que procesa texto
•Recepción UART	tarea que procesa comandos
•Mensajes de salida	tarea que transmite o imprime en display LCD
•Muestreo analógico	tarea que procesa muestras
•Reconstrucción analógica	tarea que controla el DAC

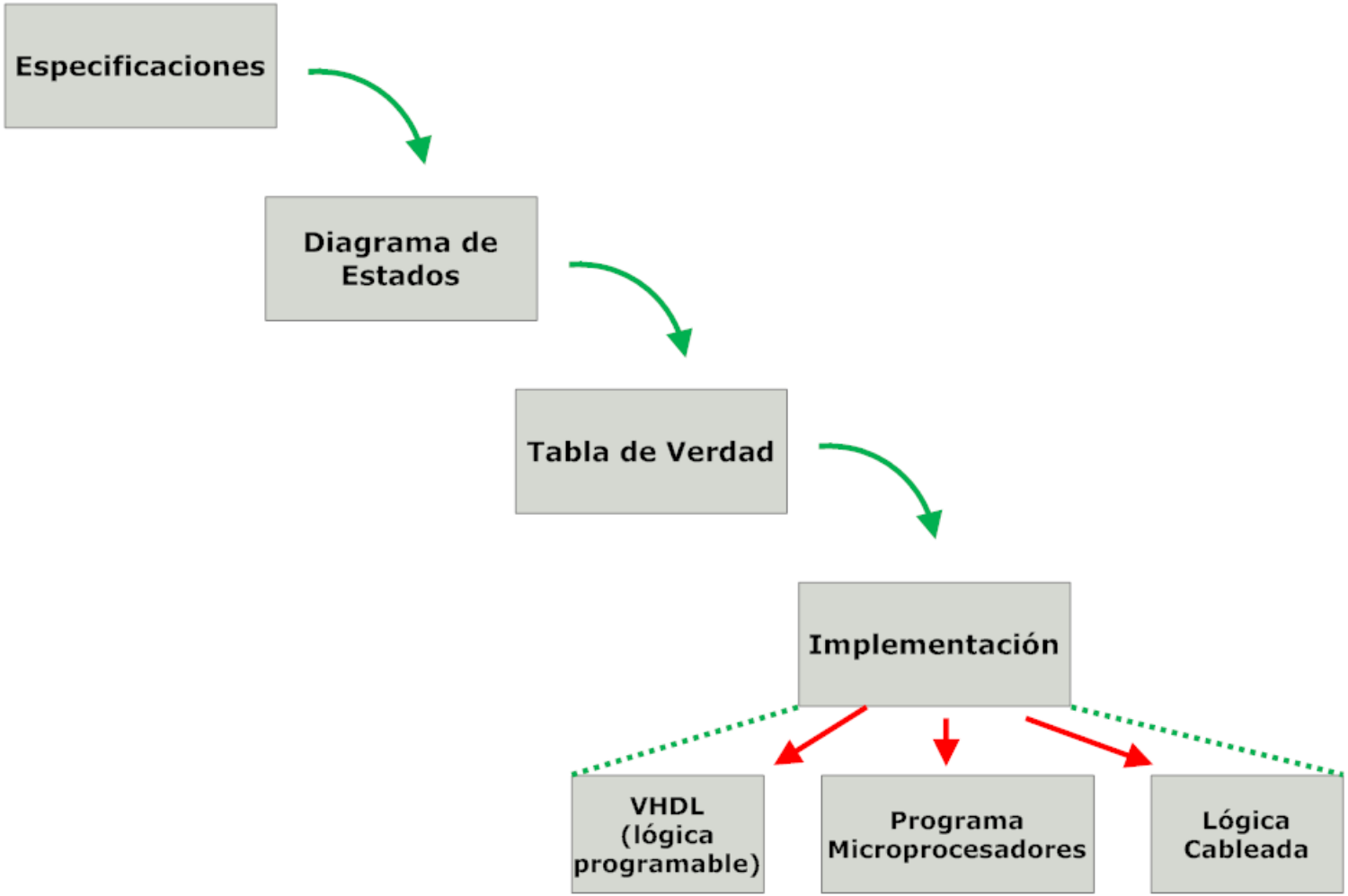
Máquinas de Estados Finitos



Máquinas de Moore y Mealy



Flujo de Diseño



Ejemplo práctico: Detector de Secuencia

Dado un canal de datos serie, detectar una secuencia determinada.

Especificaciones:

Detectar la secuencia 1011 sin solapamiento en la recepción de un canal de datos serie.

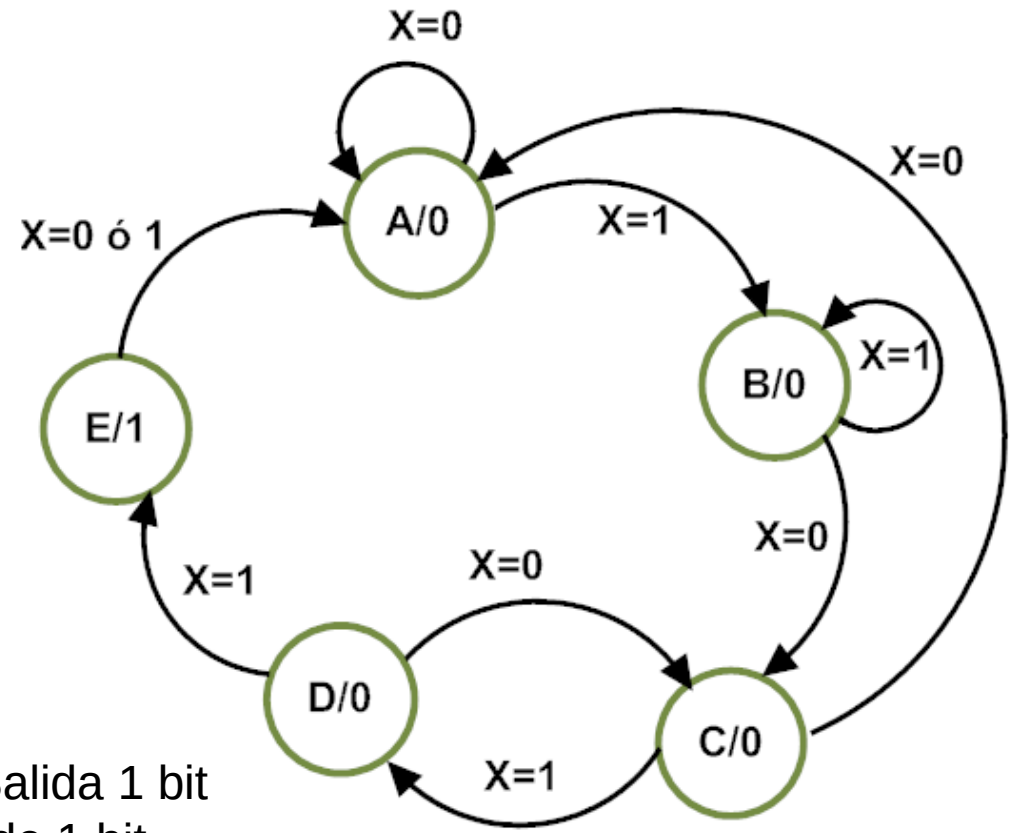
Posible utilización práctica:

-Detección de identidad en un protocolo de comunicación (ethernet, I2C, etc.)

-Cerradura con combinación numérica.

Ejemplo práctico: Detector de Secuencia

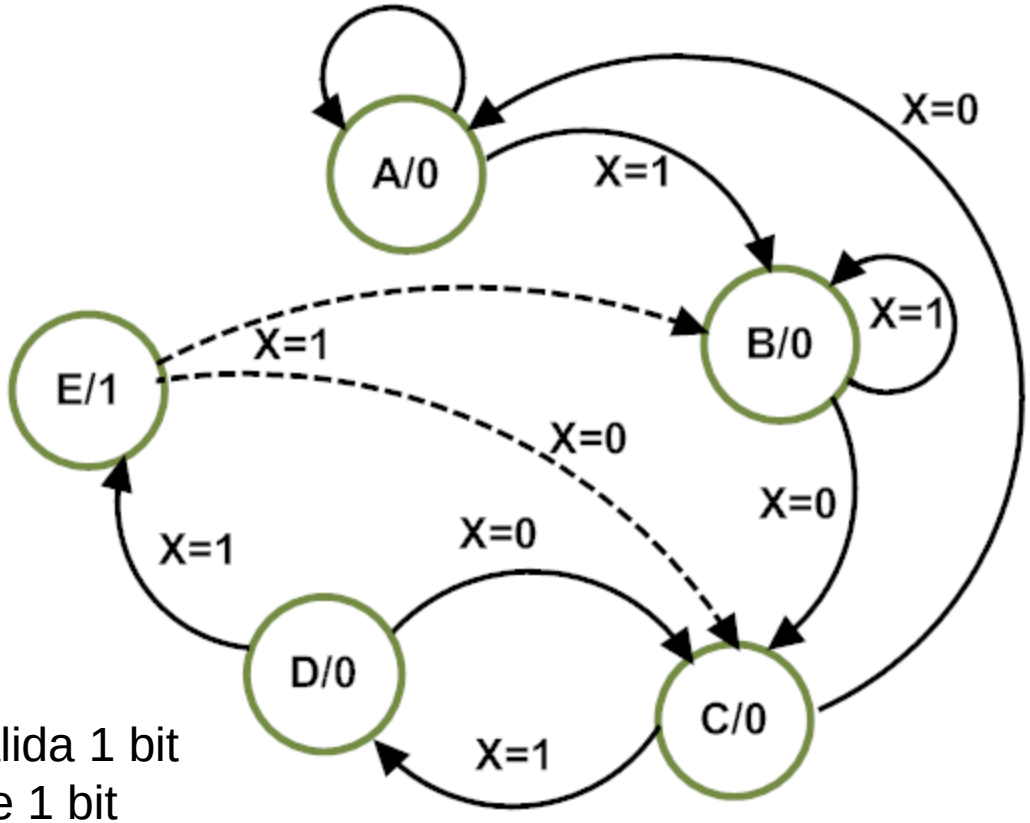
Diagrama de Estados – Moore (sin solapamiento)



A/0 => Estado/Salida 1 bit
X => Entrada de 1 bit

Ejemplo práctico: Detector de Secuencia

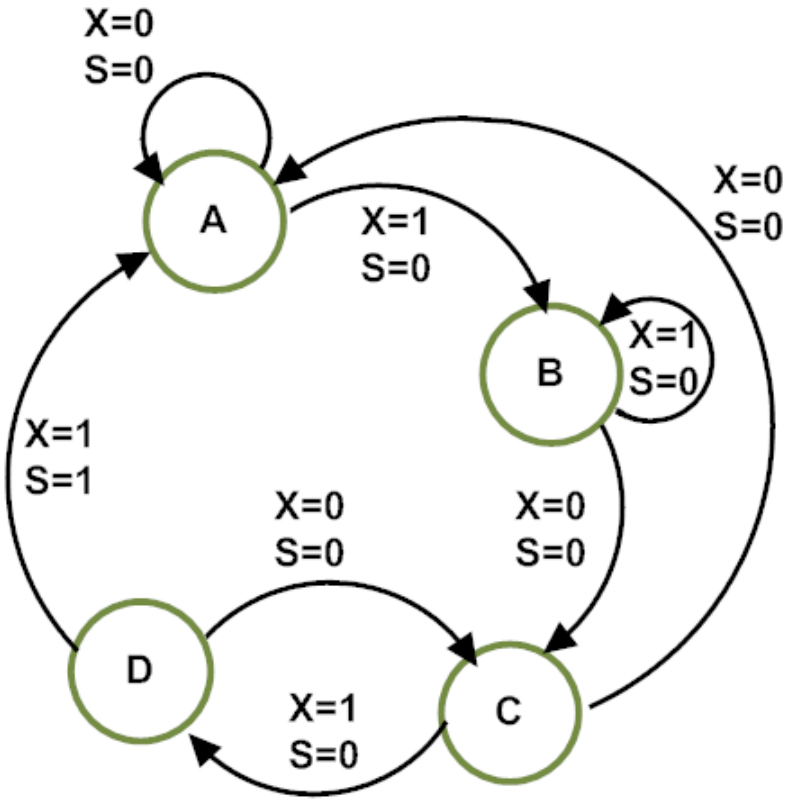
Diagrama de Estados – Otras opciones: Moore (con solapamiento)



A/0 => Estado/Salida 1 bit
X => Entrada de 1 bit

Ejemplo práctico: Detector de Secuencia

Diagrama de Estados – Otras opciones: Mealy (sin solapamiento)



A => Estado
X => Entrada de 1 bit
S => Salida 1 bit (no asociada al estado)

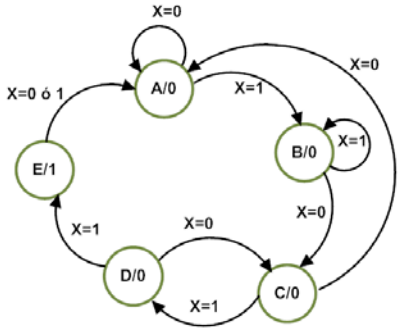
Ejemplo práctico: Detector de Secuencia

Tablas de Verdad

EA	ES	
	X=0	X=1
A	A	B
B	C	B
C	A	D
D	C	E
E	A	B
F	A	A
G	A	A
H	A	A

EA = Estado Actual
ES = Estado Siguiente

Estados no contemplados



5 estados = 3 ff
3 estados no contemplados

Codificación de Estados

EA $Q_2Q_1Q_0$	ES $D_2D_1D_0 = Q'_2Q'_1Q'_0$	
	X = 0	X=1
A = 000	000	001
B = 001	010	001
C = 010	000	011
D = 011	010	100
E = 100	000	000
F = 101	000	000
G = 110	000	000
H = 111	000	000

Ejemplo práctico: Detector de Secuencia

Tablas de Verdad

X	EA $Q_2Q_1Q_0$	ES $D_2D_1D_0 = Q'_2Q'_1Q'_0$
0	A = 000	000
0	B = 001	010
0	C = 010	000
0	D = 011	010
0	E = 100	000
0	F = 101	000
0	G = 110	000
0	H = 111	000
1	A = 000	001
1	B = 001	001
1	C = 010	011
1	D = 011	100
1	E = 100	000
1	F = 101	000
1	G = 110	000
1	H = 111	000

Otra manera
de expresar
la tabla de verdad

Ejemplo práctico: Detector de Secuencia

Tablas de Verdad

X	EA $Q_2Q_1Q_0$	ES $D_2D_1D_0 = Q'_2Q'_1Q'_0$
0	A = 000	000
0	B = 001	010
0	C = 010	000
0	D = 011	010
0	E = 100	000
0	F = 101	000
0	G = 110	000
0	H = 111	000
1	A = 000	001
1	B = 001	001
1	C = 010	011
1	D = 011	100
1	E = 100	000
1	F = 101	000
1	G = 110	000
1	H = 111	000

Flip flop tipo D

D	Q	Q + 1
0	0	0
0	1	0
1	0	1
1	1	1

Flip flop tipo JK

J	K	Q	Q + 1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Ejemplo práctico:
Detector de Secuencia

Karnaugh

X	EA $Q_2Q_1Q_0$	ES $D_2D_1D_0 = Q'_2Q'_1Q'_0$	S
0	A = 000	000	0
0	B = 001	010	0
0	C = 010	000	0
0	D = 011	010	0
0	E = 100	000	1
0	F = 101	000	0
0	G = 110	000	0
0	H = 111	000	0
1	A = 000	001	0
1	B = 001	001	0
1	C = 010	011	0
1	D = 011	100	0
1	E = 100	000	1
1	F = 101	000	0
1	G = 110	000	0
1	H = 111	000	0

D_2

XQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10			1	

$D_2 = XQ'_2Q_1Q_0$

D_1

XQ_2 \ Q_1Q_0	00	01	11	10
00		1	1	
01				
11				
10				1

$D_1 = X'Q'_2Q_0 + XQ'_2Q_1Q'_0$

D_0

XQ_2 \ Q_1Q_0	00	01	11	10
00				
01				
11				
10	1	1		1

$D_0 = XQ'_2Q'_1 + XQ'_2Q'_0$

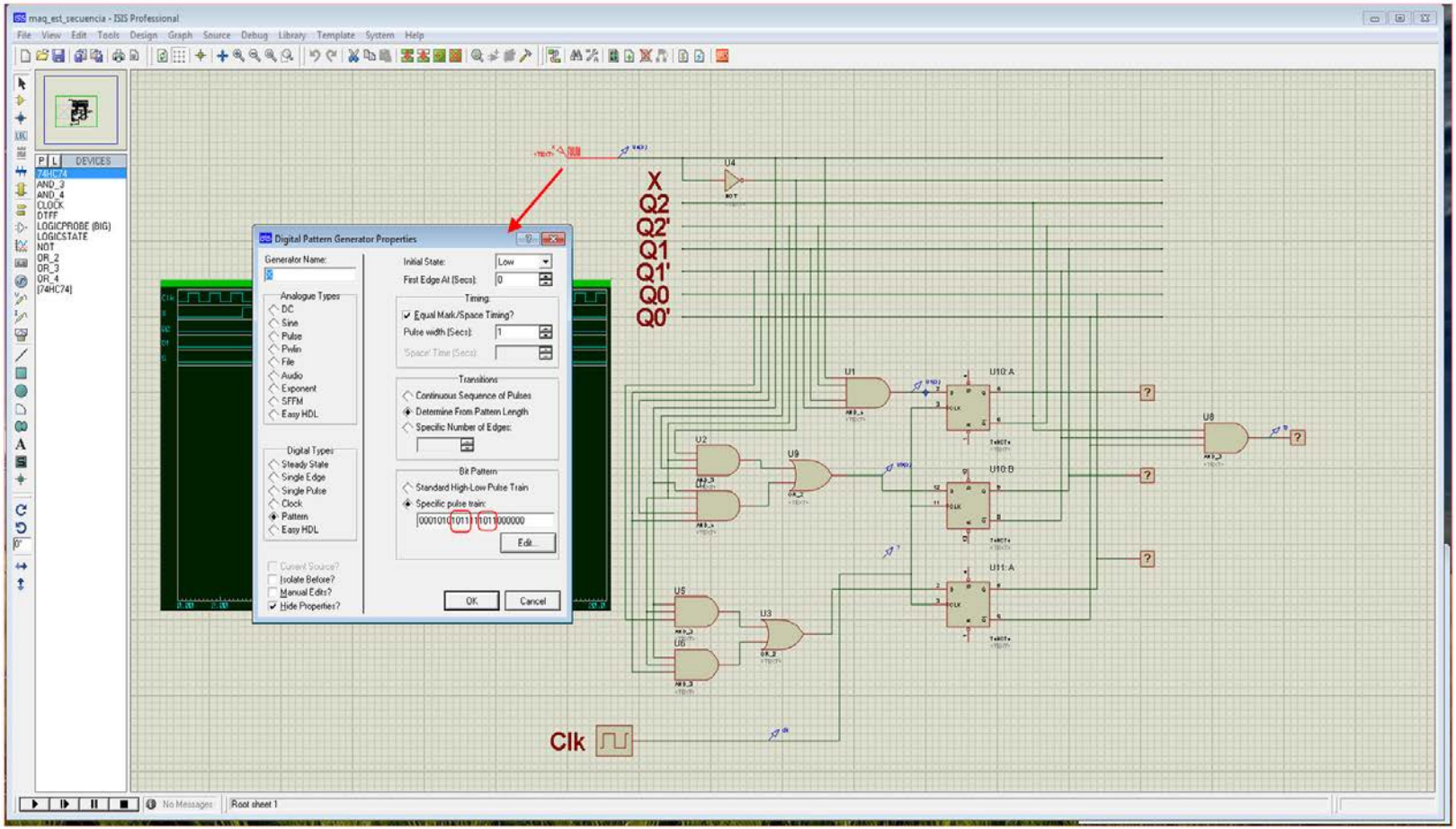
S

Q_2 \ Q_1Q_0	00	01	11	10
0				
1	1			

$S = Q_2Q'_1Q'_0$

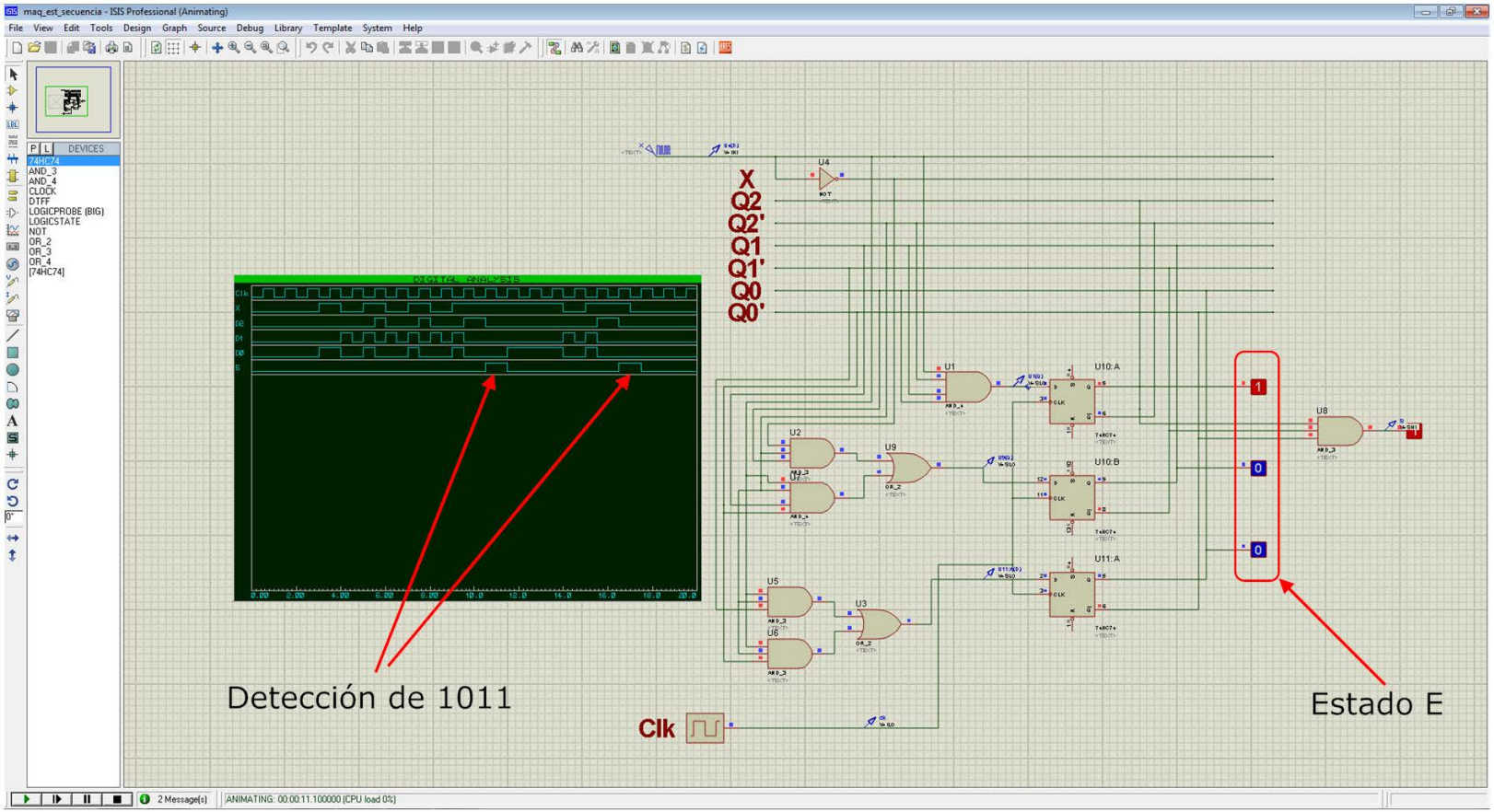
Ejemplo práctico:
Detector de Secuencia

Implementación



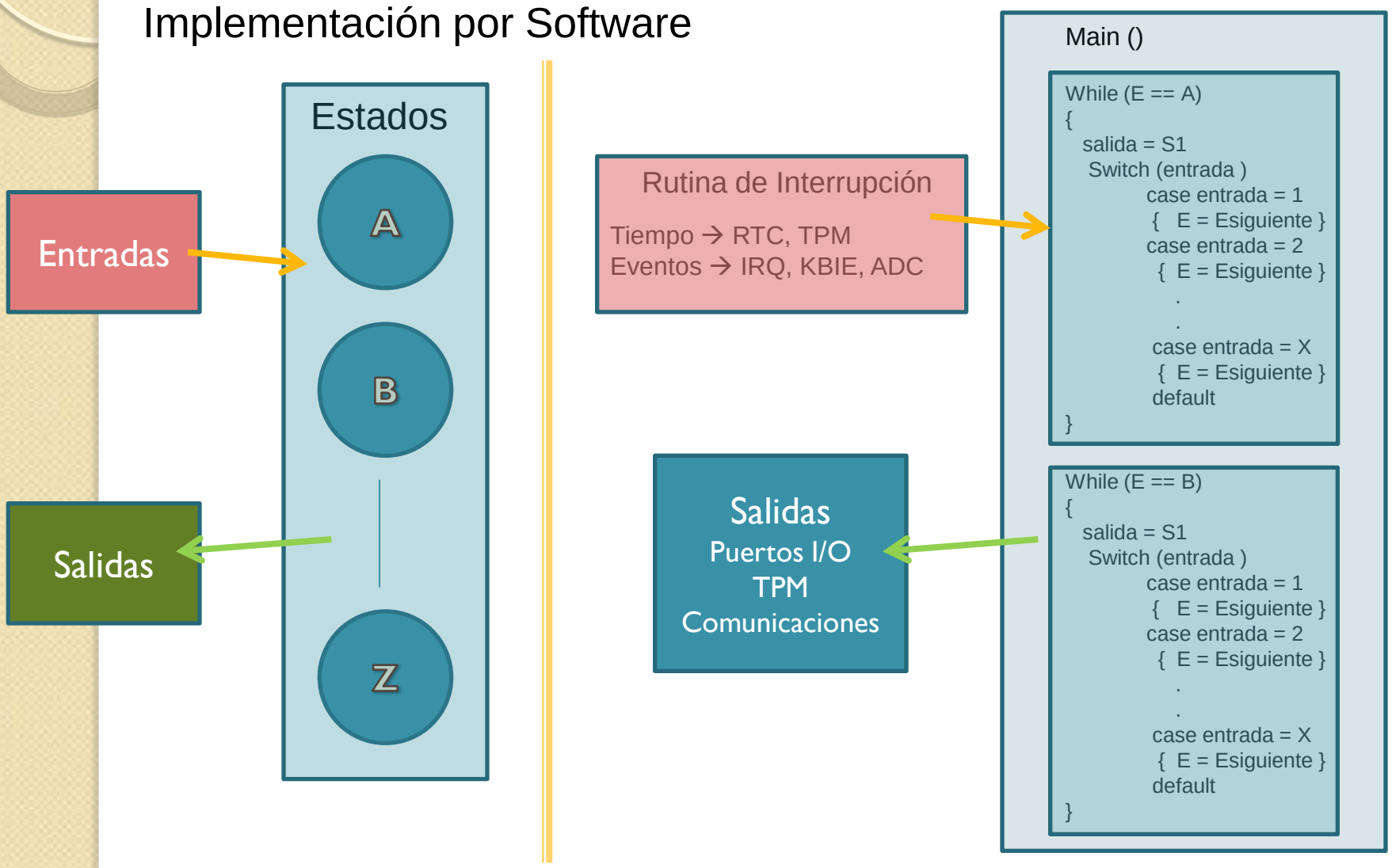
Ejemplo práctico:
Detector de Secuencia

Implementación



Ejemplo práctico: Detector de Secuencia

Implementación por Software



Máquinas de Estados Finitos

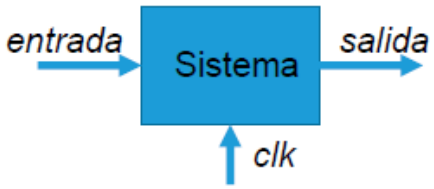
MEF temporizadas

```
void main(void){
   Iniciar_MEF();

   while(1){
      if(FLAG_TIMER){
            Actualizar_MEF();
            FLAG_TIMER=0;
      }
      sleep();
}
```

Modificado x ISR() cada T seg.

Debe ser
No-bloqueante

Similar a la implementación en Hardware



Consumo de corriente Icc [mA]