Práctica

Presentación

Por fin llegamos al final del curso. Esta práctica servirá para sintetizar todos los conocimientos del curso y ampliarlos con el diseño de circuitos más complejos. En esta práctica se os pedirá el diseño de un circuito a partir de un grafo de estados y el diseño de una máquina de estados en concreto. Para la primera parte, tendréis que aplicar los conocimientos que habéis adquirido en este curso como, por ejemplo, los mapas de Karnaugh o los sistemas de representación de la información. Por lo tanto, se recomienda que repaséis los módulos correspondientes.

Competencias

- Conocer la organización general de un computador como circuito digital.
- Conocer los rasgos distintivos de la arquitectura de Von Neumann.

Objetivos

- Conocer varios modelos de las máquinas de estados y de las arquitecturas de controlador con camino de datos.
- Haber adquirido una experiencia básica en la elección del modelo de máquina de estados más adecuado para la resolución de un problema concreto.
- Ser capaz de diseñar circuitos secuenciales a partir de grafos de transiciones de estados.

Recursos

Los recursos que se recomienda usar por esta práctica son los siguientes:

- **Básicos:** El módulo 5 de los materiales.
- **Prácticas anteriores:** En el apartado de recursos adicionales del aula de CANVAS podéis encontrar prácticas otros semestres con ejemplos de resolución.
- **Complementarios:** VerilCIRC, VerilCHART y el Wiki de la asignatura.

Criterios de valoración

- **La valoración se indica en cada uno de los subapartados.**
- **Razonad la respuesta en todos los ejercicios.**
- **Las respuestas sin justificación no recibirán puntuación.**
- **Los ejercicios realizados con IA generativa no recibirán puntuación.**
- **Los ejercicios que se detecten como plagio mediante la herramienta de plagio de la universidad no recibirán puntuación.**

Uso de herramientas de IA

En esta actividad no está permitido el uso de herramientas de inteligencia artificial. En el plan docente y en la web sobre integridad académica y plagio de la UOC encontraréis información sobre qué se considera conducta irregular en la evaluación y las consecuencias que puede tener.

Formato y fecha de entrega

- Para dudas y aclaraciones sobre el enunciado, dirigíos al consultor responsable de vuestra aula.
- Hay que entregar la solución en un documento PDF, usando una de las plantillas adjuntas a este enunciado.
- Se tiene que entregar a través de la aplicación de **Entrega de la Actividad** correspondiente del apartado **Contenidos** de vuestra aula.
- La fecha límite de entrega es el **28 de mayo,** a las 24 horas.

Enunciado

PRIMERA PARTE [65 %]

Dada la máquina de estados siguiente:

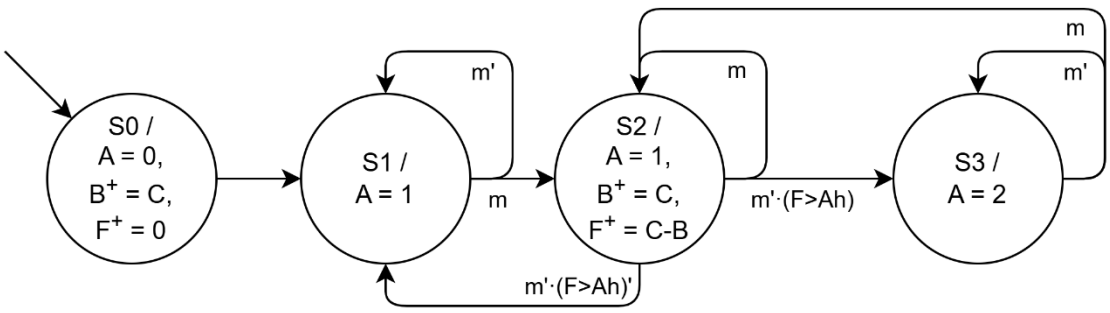
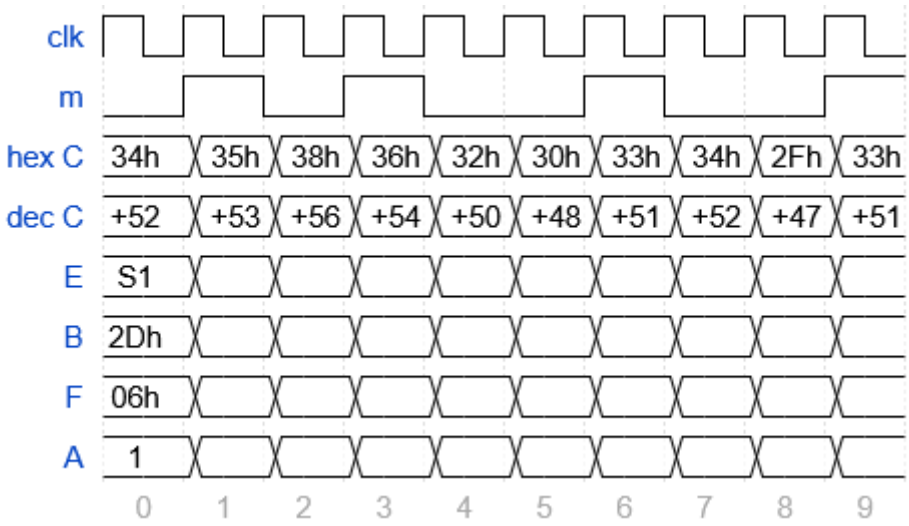


Fig. 1. EFSM de un controlador simplificado de un detector de flujo de personas.

Esta máquina de estados tiene, como entradas, una señal *m* de un bit que indica el paso de un minuto y una señal de datos *C*, que indica cuántas conexiones a la red móvil de datos hay en un momento determinado, y, como salidas, una señal de dos bits (*A*) de aviso de exceso de entrada de personas (se supone que cada conexión es una persona) y el valor calculado del flujo de personas por minuto *F*. Por simplicidad, tanto la señal de entrada *C* como las variables *B* y *F* son números en complemento a 2 de 8 bits. Sabiendo todo esto, se pide lo siguiente:

a) [15%] A partir del grafo del EFSM de la Fig. 1 completad, en el cronograma siguiente, la evolución del estado (*E*) y de las señales *A*, *B* y *F* a cada ciclo de reloj.
Nota: Tenéis el ejercicio disponible en VerilChart, donde el estado es *E* y los valores de las señales numéricas se representan en hexadecimal. De hecho, en VerilChart, la señal de entrada *C* aparece solo en hexadecimal



b) [15 %] Obtened las tablas de transiciones y de salidas de la EFSM de la Fig. 1, así como las codificaciones binarias de estados y salidas correspondientes. Para simplificar la notación, (*F*>0Ah) se denominará *F*10.

Una etiqueta pasiva de RFID envía datos a un dispositivo lector a partir del momento que ha conseguido recibir suficiente energía de la señal que emite el lector. Los datos las emite variando la dispersión de la señal del lector. Esto lo hace alternando la conexión de la antena al suelo (*y*=1) o dejándola desconectada (*y*=0). Para emitir un cero, la antena se desconecta durante un periodo de tiempo (*dt*) y se vuelve a conectar durante el mismo periodo (*dt*). En el caso de la emisión de los unos, el periodo de conexión dura el doble (*dt*).

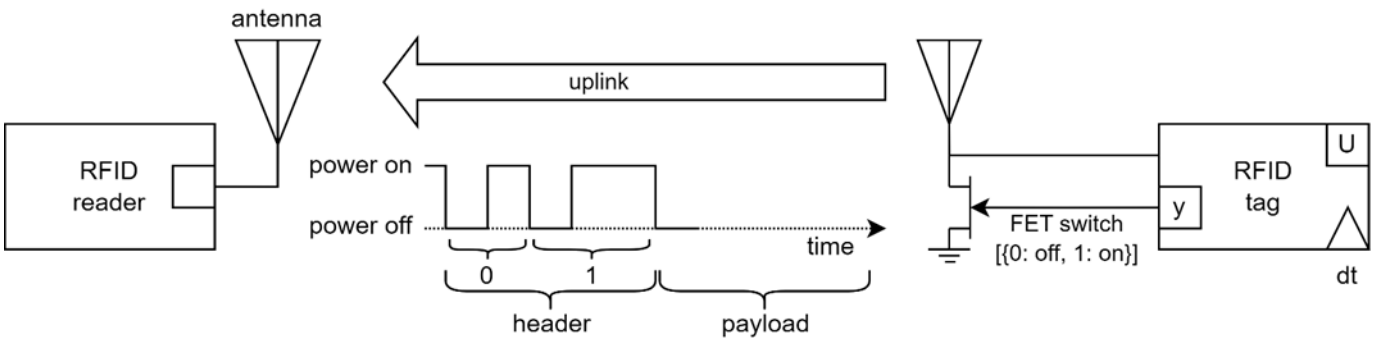


Fig. 2. La etiqueta pasiva (derecha) emite un código *U* cada vez que recibe suficiente energía a través de la antena.

Por simplicidad, se considerará que el código *U* es un código constante de 8 bits y que ya incluye un 01 inicial. En nuestro caso, será A6h.

El comportamiento de esta etiqueta pasiva se representa en el diagrama de estados de la Fig. 3.

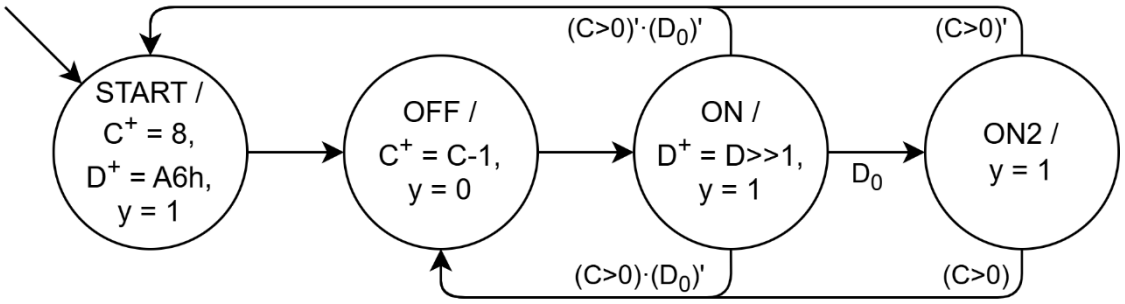


Fig. 3. EFSM de un emisor de la identificación de una etiqueta pasiva.

A la vista de la EFSM correspondiente, se pide que:

c) [15 %] A partir de las tablas de transiciones y de salidas del grafo de la Fig. 3 que se dan a continuación, implementéis la unidad de control usando ROM y los registros, los bloques combinacionales y las puertas lógicas que creáis convenientes. Especificad y justificad las dimensiones de la/las ROM que uséis e indicad su contenido en binario, especificando a qué corresponde cada bit, y en hexadecimal.

Estado actual	Entradas de control		Estado+
	D_0	(C>0)	
START	x	x	OFF
OFF	x	x	DÓNDE
DÓNDE	0	0	START
DÓNDE	0	1	OFF
DÓNDE	1	x	ON2
ON2	x	0	START
ON2	x	1	OFF

Estado		Salidas		
Símbolo	q_1q_0	y	s_C	s_D
START	00	1	00	00
OFF	01	0	01	01
DÓNDE	10	1	10	10
ON2	11	1	10	01

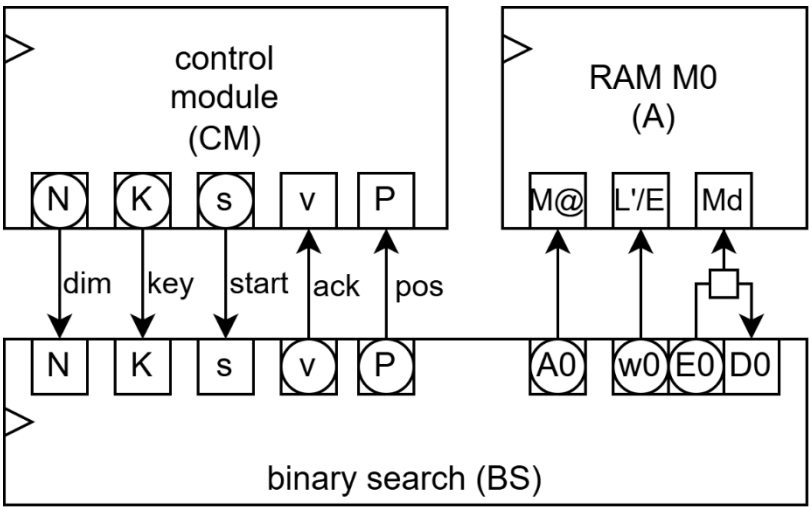
Nota: Tenéis el ejercicio disponible en VerilCIRC y, si queréis comprobar previamente la corrección de una ROM válida con palabras de 7 bits, la tenéis disponible en VerilChart. Tened en cuenta que hay otras opciones igualmente válidas con ROM de diferentes dimensiones. En este entorno, el nombre de la señal de estado es *E* y el de las entradas D_0 y (*C*>0) son *D*0 y *Cno*0, respectivamente.

d) [20 %] Implementad el circuito completo del EFSM de la Fig. 3: Construid el camino de datos usando las puertas lógicas y los bloques necesarios e incorporad la unidad de control obtenida en el apartado anterior. Tened en cuenta que la variable *D* es de 8 bits. Indicad y razonad, para cada bus, su dimensión en bits.

Nota: Tenéis el ejercicio disponible en VerilCIRC. En el circuito, las señales *C* y *D* también son de salida para poder comprobar el funcionamiento más fácilmente. Tened en cuenta que VerilCIRC no puede verificar subcircuitos diseñados por el usuario y, por lo tanto, tendréis que copiar en vuestra solución el circuito de la unidad de control que hayáis diseñado en el apartado anterior.

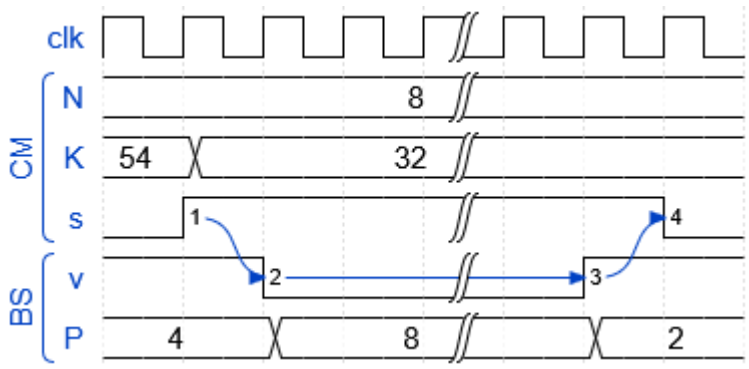
SEGUNDA PARTE [35 %]

Se tiene que diseñar el modelo de un módulo de búsqueda binaria (BS) que usa una memoria (M0) que contiene los datos ordenados (*A*), en las que se tiene que buscar un determinado elemento (*K*) que proporciona un módulo de control (CM).



El módulo BS, además de la señal de entrada *K*, tiene otra (*N*) para saber la longitud del vector de datos *A* y otra para saber en qué ciclo tiene que iniciar la búsquedasca (*s*). Por simplicidad, se supone que cada dato de *A* ocupa una única palabra de memoria.

BS tiene las salidas *P* y *v*, que contienen la dirección del elemento *K* en la memoria M0 y el aviso que este valor ya es válido, respectivamente. La señal *P* será *N* en caso de que no se haya encontrado el elemento (las direcciones válidas son al rango [0, *N*–1]) y *v* se tiene que mantener a 0 desde el ciclo siguiente en qué *s*=1 y hasta el ciclo en que se actualice *P* con el resultado de la búsqueda, tal como se puede ver en el cronograma siguiente.



Las entradas *N*, *K* y *s* se mantienen constantes desde que *s*=1 hasta que *v* vuelva a 1, como mínimo.

El algoritmo que implementa el módulo BS desde que se le ha indicado que empiece la búsqueda (ciclo en qué *s* pasa a 1 desde 0) hasta que lo acaba (ciclo en qué *v* vuelve a 1 desde 0) se describe en el siguiente programa en C.

```
#include <stdio.h>
int main()
{
    // example array
    int A[] = {10, 21, 32, 43, 54, 65, 76, 87};
    int N = 8;
    // variable initialization
    int P = N;         // position of key, N if not found
    int L =  0;        // leftmost position
    int R =  N-1;      // rightmost postition
    int M = (N-1)>>1; // middle position
    int K = 0;         // search key
    // input key from user and store it into variable K
    printf("Search key = "); scanf("%d", &K);
    // while the sublist from L to R is not empty and
    // the middle element is not K (not found)
    while(L<=R && A[M]!=K) {
        if(A[M]<K) {
            L = M + 1;
        } else {
            R = M - 1;
        } // if
        M = (L+R)>>1;
    } // while
    // if sublist is not empty, then
    // the key has been found and its position is stored into P
    if(L<=R) { P = M; }
    // output search outcome
    printf("Position = %d (=%d: not found)\n", P, N);
    return 0;
} // main
```

En el modelo que se tiene que diseñar hay que tener en cuenta la comunicación con la memoria M0, que se hace con la salida que contiene las direcciones de memoria (*A*0), la señal de salida que indica si se hace una lectura o una escritura (*w*0), la salida que contiene los datos a escribir (*E*0) y la entrada que proporciona lo que se lee (*D*0). En este caso, *w*0 siempre será 0 y *E*0 se puede dejar también a 0, puesto que no se harán escrituras.

Diseña la ASM correspondiente al comportamiento del módulo BS que se ha descrito teniendo en cuenta que todas sus salidas son variables de la máquina de estados.