75.562 **·** Fundamentos de Computadores **·** 2024-25

Práctica

Apellidos:

Nombre:

**Formato y fecha de entrega**

* Para dudas y aclaraciones sobre el enunciado debéis dirigiros al consultor responsable de vuestra aula.
* Hay que entregar la solución en un fichero PDF utilizando una de las plantillas entregadas conjuntamente con este enunciado.
* Se debe 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).
* **Razonad la respuesta en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.**

**Respuestas**

**PRIMERA PARTE [65 %]**

Dada la máquina de estados siguiente:

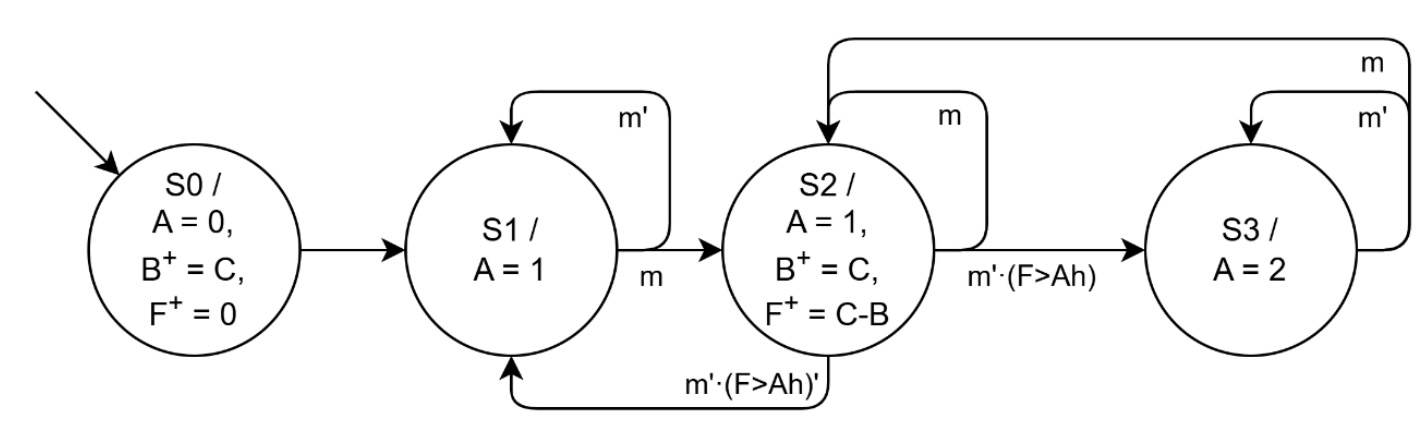
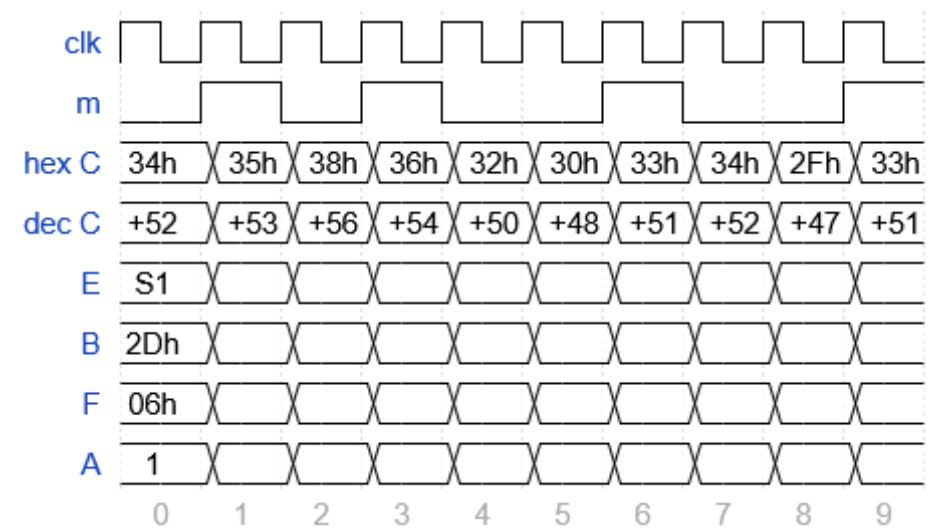


Fig. 1. EFSM de un controlador simplificado de un detector de flujo de personas.

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:

1. [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 en cada ciclo de reloj.

**NOTA**: Tenéis disponible el ejercicio en VerilChart (20242\_PR\_1a\_vch) 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.



1. [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 F10.

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).

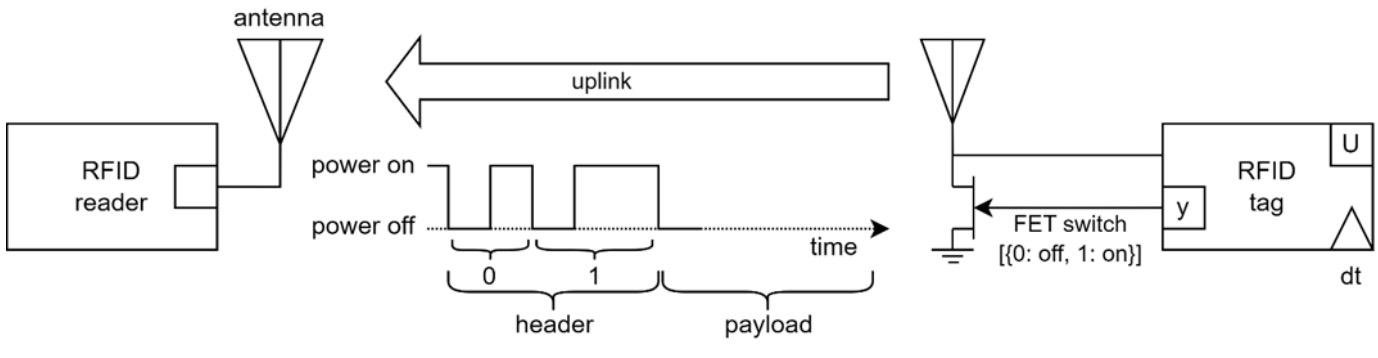


Fig. 2. La etiqueta pasiva (derecha) emite un código U cada vez que recibe suficiente energía a través de la antena.

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.

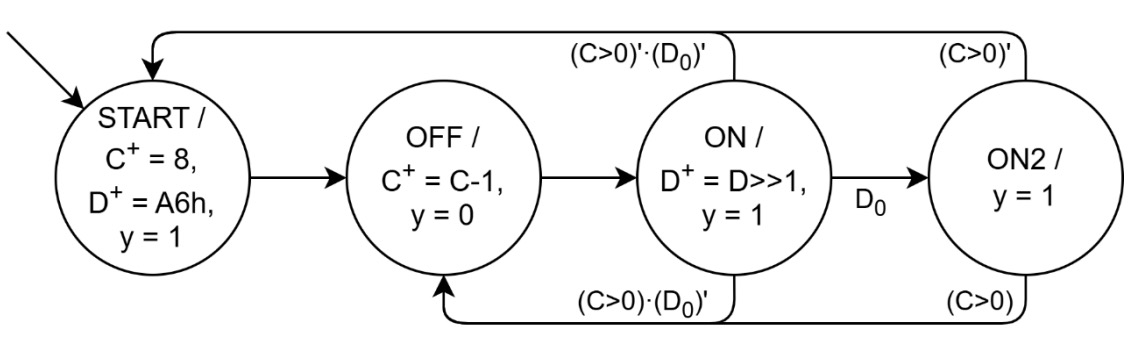
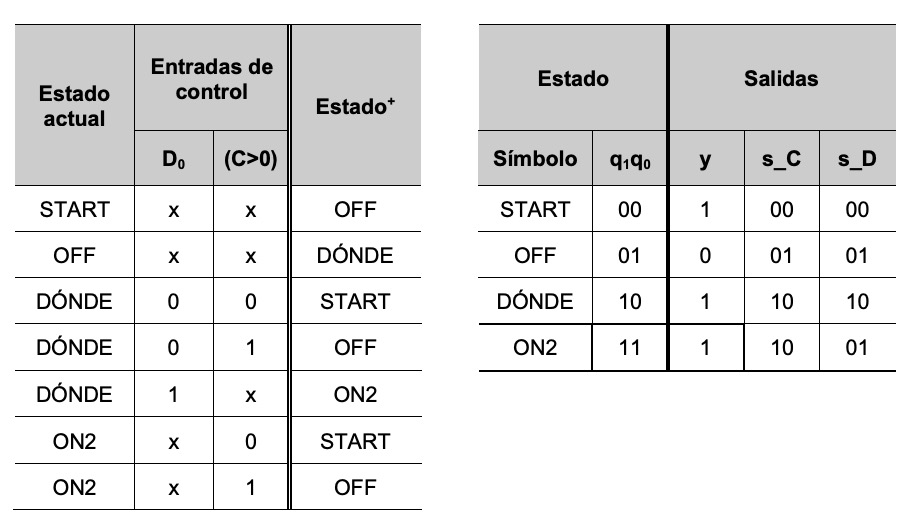


Fig. 3. EFSM de un emisor de la identificación de una etiqueta pasiva.

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

1. [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.



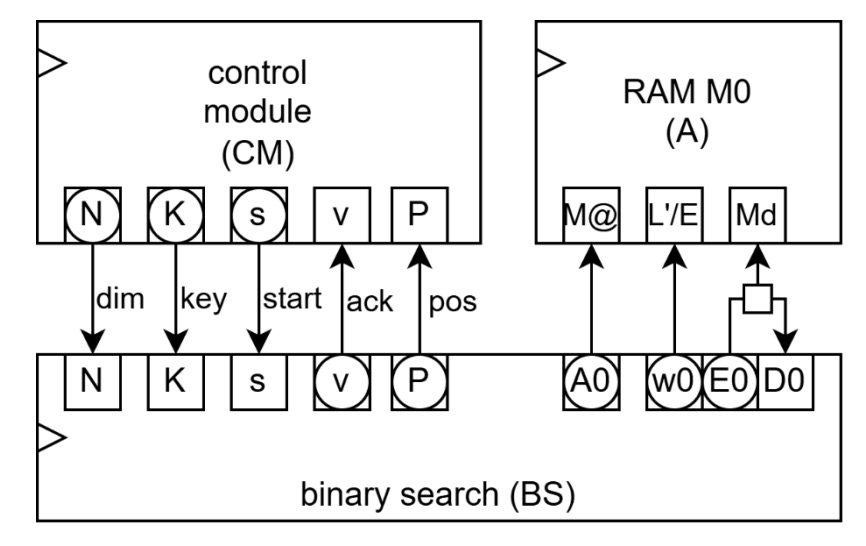
**NOTA**: Tenéis disponible el ejercicio en VerilCirc (20242\_PR\_1c\_circ), y, si queréis comprobar previamente la corrección de una ROM válida con palabras de 7 bits, la tenéis disponible en VerilChart (20242\_PR\_1c\_TB). Tened en cuenta que hay otras opciones igualmente válidas con ROM de diferentes dimensiones. En este entorno, el nombre de la señal de estado es E y el de las entradas D0 y (C>0) son D0 y Cno0, respectivamente.

1. [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 disponible el ejercicio en VerilCirc (20242\_PR\_1d\_circ). 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.

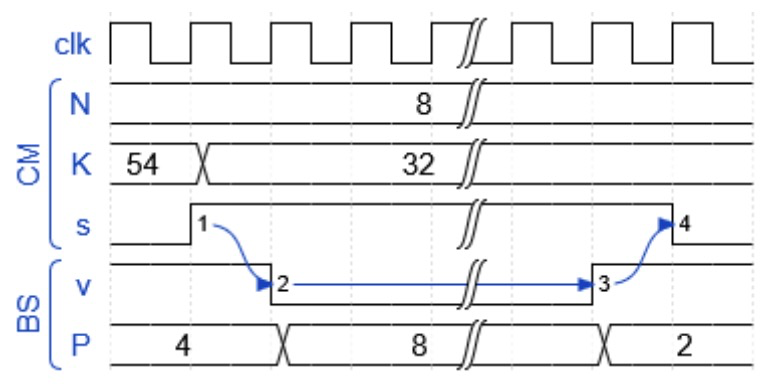
**SEGUNDA PARTE [35 %]**

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 búsqueda (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 el que 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 position  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 (A0), la señal de salida que indica si se hace una lectura o una escritura (w0), la salida que contiene los datos a escribir (E0) y la entrada que proporciona lo que se lee (D0). En este caso, w0 siempre será 0 y E0 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.