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 el de los sistemas de representación. Por lo tanto, se recomienda que hagáis un repaso de los mismos.

Competencias

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

Objetivos

— Conocer varios modelos de máquinas de estados y 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 para esta práctica son los siguientes:

Básicos: El módulo 5 de los materiales.

Complementarios: VerilCIRC, VerilCHART y el Wiki de la asignatura.

Criterios de valoración

— Razonad la respuesta en todos los ejercicios. Las que no estén justificadas no recibirán puntuación.
— La valoración de cada apartado está indicada en los enunciados correspondientes.

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 fichero PDF usando una de las plantillas proporcionadas conjuntamente con este enunciado.
— Se tiene que entregar a través de **Entrega de la Actividad** correspondiente de **Contenidos** de vuestra aula.
— La fecha límite de entrega es el **20 de diciembre** (a las 24 horas).

PRIMERA PARTE [65 %]

Una de las cuestiones más importantes en las redes de comunicación es la seguridad, que se consigue, entre otras cosas, *encriptando* los mensajes que se transmiten. Esto dificulta que nadie que no tenga la clave para descifrarlos pueda acceder al contenido y que, además, se pueda determinar la autoría.

En un sistema criptográfico sencillo, los mensajes pueden ser textos que se codifican con una determinada clave de forma que resulten ilegibles para nadie que no tenga la clave. Para hacerlo, se pueden transformar los mensajes de texto en secuencias de códigos que solo representen a las letras del alfabeto y, después, cifrar estas secuencias con una clave para esconder el mensaje a quien no la tenga.

La EFSM siguiente se corresponde con el de una máquina que recibe caracteres codificados en ASCII de 8 bits y los transforma en códigos de 5 bits de forma que solo se codifican las letras del alfabeto inglés. (Los arcos no etiquetados indican transiciones incondicionales, que se efectúan siempre.)

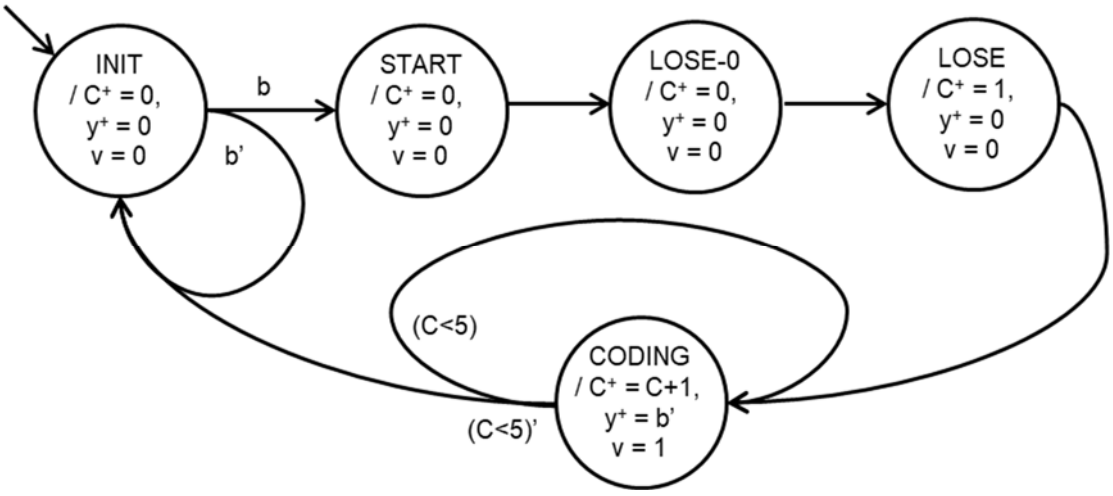
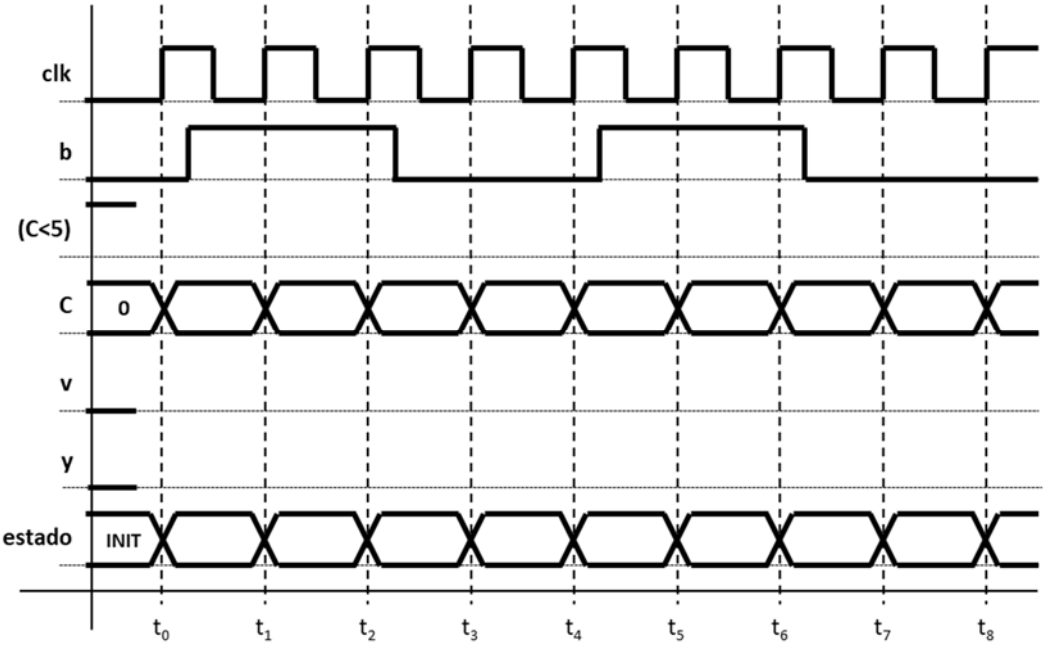


Fig. 1. EFSM del módulo de reducción de bits de la codificación ASCII.

Se pide que:

a) [15 %] A partir del grafo de la EFSM de la fig. 1 completad, en el cronograma siguiente, la evolución de las señales *C*, *v* e *y*, y del estado en que se encuentra la máquina en cada ciclo de reloj.

Nota: Tenéis el ejercicio disponible en VerilChart donde los valores de la señal *C* se representan en hexadecimal y las señales (*C*<5) y *estado* están representadas por *C*5 y *E*.



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.

La EFSM siguiente se corresponde con el funcionamiento de un encriptador sencillo que usa una clave entera de 5 bits, *C*, para codificar la secuencia de bits de entrada, *y*, siempre que sea válida, cosa que viene indicada por la señal de entrada *v*. La salida del módulo es una secuencia de bits *z* que cifra la secuencia de entrada *y*. Los bits de la secuencia *z* solo serán correctos si la salida *k* es 1. Si no, indicará que no se está cifrando ningún mensaje.

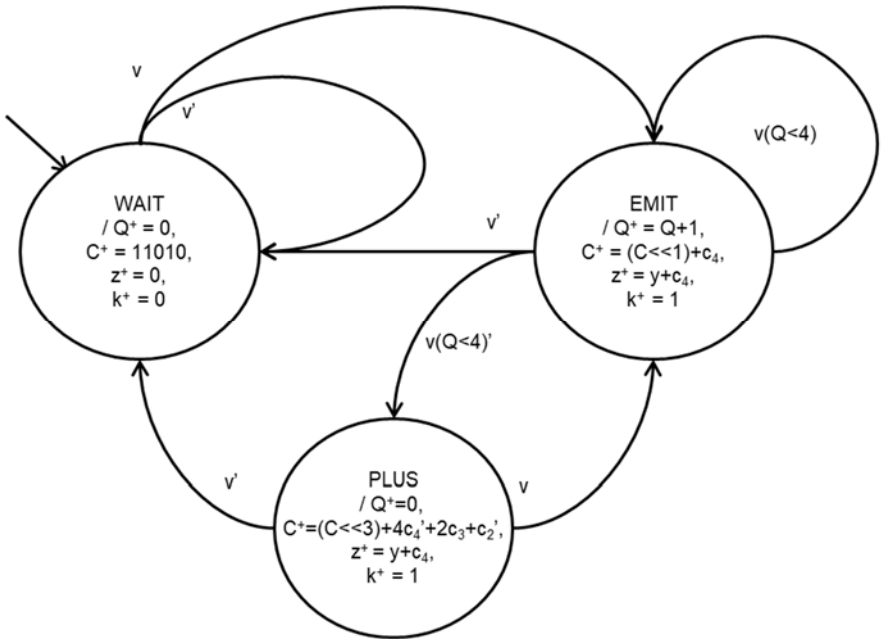


Fig. 2. EFSM de un módulo de encriptación con clave (*C*) de 5 bits.

En el grafo anterior, las sumas para el cálculo de C^+ son aritméticas.

Se pide que:

c) [15 %] A partir de las tablas de transiciones y de salidas del grafo de la Fig. 2, que se dan a continuación, implementad 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 el contenido.

Estado actual	Entradas		**Estado⁺**
	v	**(Q<4)**	
WAIT	0	x	WAIT
WAIT	1	x	EMIT
EMIT	0	x	WAIT
EMIT	1	0	PLUS
EMIT	1	1	EMIT
PLUS	0	x	WAIT
PLUS	1	x	EMIT

Estado		Salidas			
Símbolo	**q_1q_0**	**s_Q**	**s_C**	**s_z**	**s_k**
WAIT	00	0	00	0	0
EMIT	01	1	01	1	1
PLUS	10	0	10	1	1

Hay que tener presente que las señales *s_Q*, *s_C*, *s_z* y *s_k* se usan como selectores para elegir cuál de los resultados se asignan a las variables *Q*, *C*, *z* y *k*, respectivamente.

Nota: Tenéis el ejercicio disponible en VerilUOC y, si queréis comprobar previamente la validez de la ROM, la tenéis disponible en VerilChart. En este entorno, las señales (*Q*<4) y *Estado* son *Q*4 y *E*, respectivamente.

d) [20 %] Implementad el circuito completo de la EFSM de la Fig. 2: Construid el camino de datos usando las puertas lógicas y los bloques necesarios e incorporad la unidad de control obtenida en el apartado anterior. Indicad y razonad, para cada bus, su dimensión en bits teniendo en cuenta que las señales de entrada ocupan el mínimo número de bits posible según su rango.

Nota: Tenéis el ejercicio disponible en VerilUOC. En el circuito, se han introducido las señales *Q*, *C*, *C*4 y *E* del apartado anterior como señales de salida para poder comprobar más fácilmente el correcto funcionamiento. Tened en cuenta que VerilUOC no soporta la verificación de subcircuitos diseñados por el usuario. Por lo tanto, tendréis que copiar en vuestra solución el circuito de la unidad de control diseñada en el ejercicio anterior.

SEGUNDA PARTE [35%]

Un método para encriptar textos se basa en codificar cada carácter en otro según una operación con un carácter de una palabra clave. Una opción sencilla para hacerlo es que el código correspondiente sea el carácter que esté en la posición que resulte de restar las posiciones del carácter original y la posición del carácter de la palabra clave (podríamos elegir cualquier otra transformación con sentido).

Por ejemplo, si el carácter a codificar es 'C' y el carácter de la palabra clave que se usa es 'K', según la tabla de caracteres de más abajo, el código resultante es 'T' porque ocupa la posición 21, que es el resultado de restar la posición de la 'C' (2) y la posición de 'K' (2-11 mod 30 = 21).

Tabla 1. Ejemplo de tabla de codificación de caracteres.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	Ç	D	E	F	G	H	I	J	K	L	M	N
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	.	-

Si el código resultante es más pequeño que 0, entonces se hace la suma descrita anteriormente en módulo 30.

Para cada nuevo carácter a codificar se usa un nuevo carácter de la palabra clave y, al llegar al final de la palabra clave, se vuelve al principio.

Un módulo encriptador que use este método tendría que recibir el texto codificado tal como se muestra en la tabla 1 y emitir el texto cifrado con una palabra clave prefijada. El módulo empezaría a funcionar cada vez que recibiera la secuencia "---" y volvería al modo de espera cuando recibiera un punto o, si se quiere, cuando recibiera un 28. La salida sería la secuencia de caracteres cifrados o bien '.' (28).

Por ejemplo, el mensaje "UN-EJEMPLO." se vería codificado con la tabla como <29, 29, 29, 22, 14, 29, 5, 10, 5, 13, 17, 12, 16, 28>, de usar "CODIGO", es decir <2, 16, 4, 9, 7, 16>, como palabra clave, se obtendría el mensaje cifrado que se ve a la última fila de la tabla siguiente.

Tabla 2. Ejemplo de cifrado.

-	-	-	U	N	-	E	J	E	M	P	L	O	.	.
29	29	29	22	14	29	5	10	5	13	17	12	16	28	28
			2	16	4	9	7	16	2	16	4	9	7	
29	29	29	20	28	25	26	3	19	11	1	8	7	21	28
-	-	-	S	.	X	Y	Ç	R	K	B	H	G	T	.

Fijaos que también se codifica el punto final. Hay que tener en cuenta que, en la tabla anterior se muestra el cifrado en la misma columna, pero que, en la realidad, habría un decalaje temporal entre la entrada y la salida equivalente.

Se pide que diseñéis una máquina de estados algorítmica que implemente este método de encriptación. Para lo cual, tened presente que:

- Recibe una entrada *D* de 5 bits que contiene el valor de la posición en la tabla 1 del carácter que representa;
- Tiene una memoria ROM de 8 x 5 bits con la palabra clave, que siempre es de ocho caracteres;
- Dispone de un registro de direcciones de memoria *A* y de una entrada de los datos de la memoria *K*, y
- Genera una salida *W* de 5 bits que representa cada uno de los caracteres cifrados o bien es '.' (posición 28) en caso contrario.

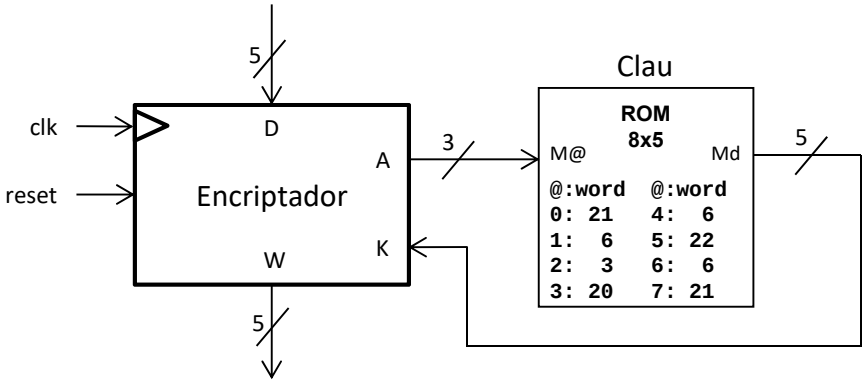


Fig. 3. Esquema de bloques de un encriptador de mensajes de texto.