Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica

IEE2713 — Sistemas Digitales

# Tarea 5 - entrega 10 de octubre 2019

---

**Instrucciones**: Por favor entregar respuestas en documento en PDF con nombre de todos los integrantes. El documento puede ser elaborado con word, LaTeX, fotos digitalizadas, etc., pero el archivo es un pdf. No recibiremos tareas en otro formato y tampoco recibiremos tareas atrasadas.

**Las tareas son en grupos de 2 o 3 estudiantes**. Se recomienda enfáticamente que realicen la tarea en grupo a conciencia, discutiendo todas las preguntas entre todos los integrantes y sin repartirlas. No olviden respetar el código de honor de la UC.

1. Determine el valor decimal de los siguientes números binarios expresados en complemento de 2 (1 punto):
   *a*) 0111011110
   *b*) 1011100111
   *c*) 1111111110
2. Convierta los siguientes números decimales en números binarios de 12 bits expresados en complemento de 2 (1 punto):
   *a*) 73
   *b*) 1906
   *c*) -95
   *d*) -1630
3. Implemente la función $f(w_1, w_2, w_3) = \Sigma m(1, 2, 3, 5, 6)$ usando un decodificador binario 3:8, y una compuerta OR. Entregue un esquemático (1 punto).
4. Considere la función $f = \bar{w_1}\bar{w_3} + w_2\bar{w_3} + \bar{w_1}w_2$. Diseñe un circuito con compuertas lógicas que implemente la función $f$ usando un multiplexor 2:1. Entregue un esquemático (1 punto).
5. Utilice la expansión de Shannon sobre la función $f(w_1, w_2, w_3) = \Sigma m(0, 2, 3, 6)$ para derivar una implementación que utilice un multiplexor 2:1 y otras compuertas lógicas. Entregue un esquemático (1 punto).

6. Cuando enviamos mensajes digitales a través de canales de comunicación muy ruidosos, los mensajes pueden llegar al receptor con bits alterados. Para mejorar la calidad de la comunicación y reducir la probabilidad de recibir mensajes erróneos utilizamos **códigos correctores de errores**. Estos códigos agregan redundancia a palabras digitales y las transmiten en más bits de los necesarios, permitiendo corregir bits incorrectos. Por ejemplo, existe la codificación Hamming(7,4) `https://en.wikipedia.org/wiki/Hamming(7,4)` que codifica nibbles (4 bits) en palabras de 7 bits, agregando 3 bits de paridad. Esta redundancia permite **corregir** errores de un bit en el mensaje recibido. Supongamos que queremos transmitir el nibble $M = m_1m_2m_3m_4$. Entonces generamos tres bits de paridad: $p_1 = m_1 \oplus m_2 \oplus m_4$; $p_2 = m_1 \oplus m_3 \oplus m_4$; y $p_3 = m_2 \oplus m_3 \oplus m_4$; y transmitimos la palabra completa $T = p_1p_2m_1p_3m_2m_3m_4$. Esa palabra pasa por un canal de comunicaciones imperfecto que podría alterar algún bit. A partir de la palabra de 7 bits recibida, generamos un código de chequeo de paridad $C = c_3c_2c_1$, donde $c_1 = p_1 \oplus m_1 \oplus m_2 \oplus m_4$, $c_2 = p_2 \oplus m_1 \oplus m_3 \oplus m_4$ y $c_3 = p_3 \oplus m_2 \oplus m_3 \oplus m_4$. Lo interesante de esta codificación Hamming(7,4) es que si el canal de comunicaciones altera un solo bit, entonces el valor de $C$ corresponde a la posición del bit erróneo en $T$, donde $C = 0$ indica que no hay error, $C = 1$ indica que el error ocurrió en $p_1$, y así sucesivamente hasta $C = 7$, que indica error en $m_4$. Describa en Verilog un módulo **PGen** que tome como entrada un nibble $M = m_1m_2m_3m_4$ y genere como salida los bits de paridad del código Hamming(7,4) $P = p_1p_2p_3$ (1 punto).

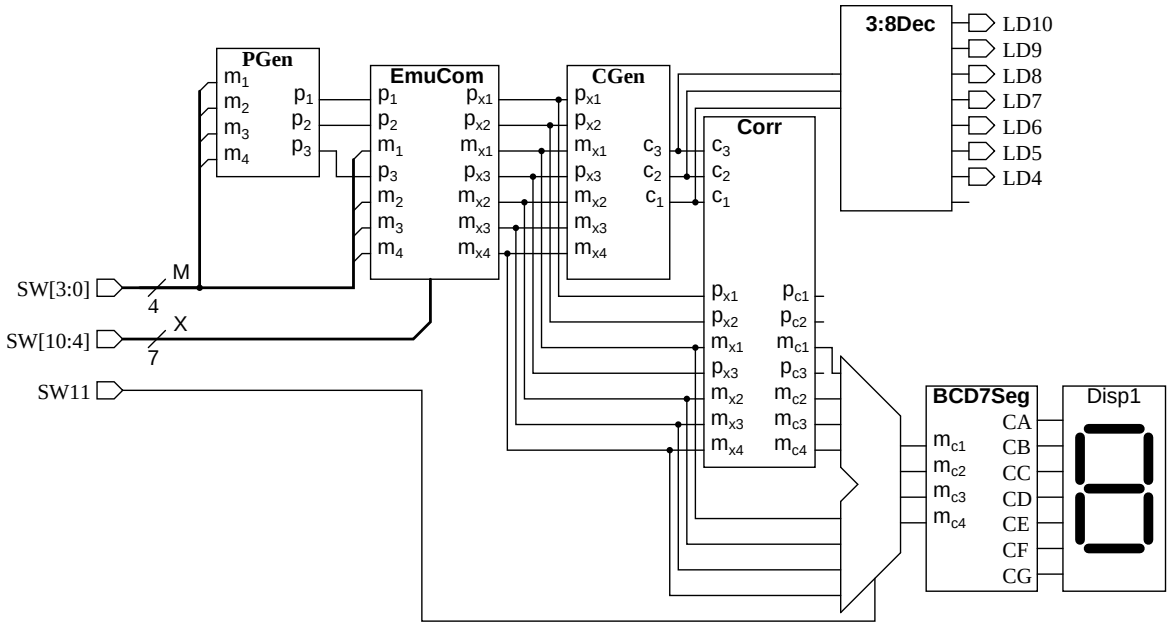
7. Describa en Verilog un módulo **CGen** que tome como entrada una palabra de 7 bits $T = p_{x1}p_{x2}m_{x1}p_{x3}m_{x2}m_{x3}m_{x4}$ y genere como salida los bits de chequeo de paridad del código Hamming(7,4) $C = c_3c_2c_1$ (1 punto).

8. Describa en Verilog un módulo **Corr** que tome como entradas una palabra de 7 bits (posiblemente con un bit erróneo) $Terr = p_{x1}p_{x2}m_{x1}p_{x3}m_{x2}m_{x3}m_{x4}$ y una palabra de 3 bits de chequeo de paridad $C = c_3c_2c_1$ que indica la posición del posible error, y produzca como salida la palabra corregida $Tc = p_{c1}p_{c2}m_{c1}p_{c3}m_{c2}m_{c3}m_{c4}$ (1 punto).

9. Describa en Verilog un módulo **EmuCom** que emule un canal de comunicaciones imperfecto. El módulo debe tomar dos entradas de 7 bits, una correspondiente al vector $T = p_1p_2m_1p_3m_2m_3m_4$ a ser transmitido, y otra $X$ correspondiente a los bits alterados por el canal de comunicaciones imperfecto. La salida del bloque $Terr = p_{x1}p_{x2}m_{x1}p_{x3}m_{x2}m_{x3}m_{x4}$ corresponde a los bits obtenidos por el receptor (1 punto).

10. Implemente en la tarjeta Basys3 un circuito como el del diagrama de bloques de la figura en la página siguiente. Las entradas son una palabra de 4 bits a ser transmitida $M = m_1m_2m_3m_4$, una palabra de 7 bits $X$ correspondiente a los bits alterados por el canal de comunicaciones imperfecto, y un bit que elige si la salida sale directamente de los bits recibidos por el receptor o de los bits corregidos. Las salidas son la palabra recibida por el receptor $Mx$ o la palabra corregida $Mc$ (dependiendo del selector del MUX2:1), desplegada en un display de 7 segmentos (use el módulo implementado en la tarea anterior), y 7 bits one-hot desplegados en 7 leds diferentes, que indican la posición del posible error. Entregue el código completo de todos los módulos (3 puntos).

11. Entregue el link de un video en Youtube, de elaboración propia y muy simple, que muestre el funcionamiento del circuito. El video no debe durar más de 30 segundos (3 puntos).


PGen
EmuCom
CGen
Corr
3:8Dec
BCD7Seg
Disp1
SW[3:0]
SW[10:4]
SW11
M
X
4
7
LD10
LD9
LD8
LD7
LD6
LD5
LD4
CA
CB
CC
CD
CE
CF
CG