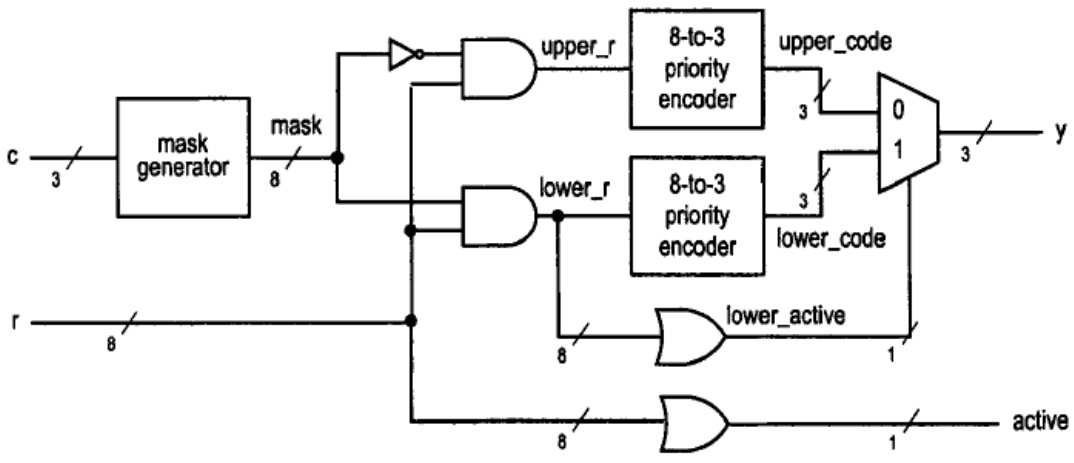
TEMAS PARA PENSAR DE FUNDAMENTOS DE COMPUTADORES
TEMA 4

1. En un codificador de prioridad convencional, la prioridad relativa de cada una de las entradas (la que determina el valor que codifica la salida cuando uno o varias entradas se activan) es fija. Algunas aplicaciones requieren que la prioridad pueda cambiar dinámicamente para dar un acceso a los recursos más equilibrado. Un codificador de prioridad 8 a 3 programable, aparte de las 8 entradas de solicitud convencionales, r_i, tiene una entrada de control, $\underline{c}$, que indica la petición de mayor prioridad. Por ejemplo, si $\underline{c}$ = "011", la prioridad relativa de las entradas será r_3, r_2, r_1, r_0, r_7,..., r_4. En la figura se muestra un esquema de su estructura. El diseño utiliza la señal c para generar 2 máscaras de 8 bits que se usan para eliminar respectivamente la parte baja y alta de la palabra de solicitud. Aplicando ambas máscaras a la palabra de solicitud original y pasando ambos resultados por sendos codificadores de prioridad obtendremos 2 códigos, uno de los cuales deberá ser seleccionado en función de si existe o no una petición en la parte baja de la solicitud. Por ejemplo, si *c* es "011" y *r* es "11010101", la máscara a usar para eliminar los bits menos significativos de la solicitud será "00001111" y la máscara a usar para eliminar los bits más significativos de la solicitud será su inversa, es decir, "11110000". Aplicando ambas máscaras a *r*, tenemos "00000101" y "11010000" que generan respectivamente los códigos "010" y "111". Finalmente "010" es seleccionado ya que la parte baja de la solicitud contiene algún '1'. Diséñese el circuito generador de máscara.



2. Diseñar un codificador de prioridad 8 a 3 dual. El sistema funcionará como un codificador de prioridad convencional con la diferencia de que tiene 2 salidas de 3 bits para códigos y 2 salidas de activación por las que indicará, cuando corresponda, los códigos de las 2 entradas de mayor peso que se activen.

3. Se define la distancia Hamming entre 2 palabras binarias del mismo tamaño como el número de posiciones en las que ambas palabras difieren e indica el número de bits que deben invertirse para convertir una palabra en la otra. Un circuito que calcule la distancia

Hamming debe hacerlo en 2 fases. La primera, debe calcular una palabra que marque mediante '1' los bits en los que difieren las palabras de entrada. La segunda, debe contar el número de '1' resultantes en la anterior fase mediante un circuito conocido como contador de población. Por ejemplo, dadas las palabras "00010011" y "10010010", cuya distancia Hamming es 2 ya que difieren en las posiciones 7 y 0, el resultado de la primera fase será "10000001" y el de la segunda "0010". Diseñar un circuito combinacional que calcule la distancia Hamming entre 2 palabras de 8 bits *(Pista: La primera fase se implementa usando puertas XOR. La segunda fase se implementa mediante un árbol de sumadores completos de 3 etapas en donde los sumadores son respectivamente de 1, 2 y 3 bits y en cada etapa se suman 2 a 2 los resultados obtenidos en la anterior etapa. Téngase en cuenta que en cada etapa el carry-out de la suma debe concatenarse con el resultado de la suma para formar el operando que se sumará en la etapa siguiente).*

4. Diseñe un circuito combinacional que cuente el número de ceros consecutivos que hay a la izquierda de un vector de 16 bits dado. *(Pista: Usar un contador de población que tenga una etapa previa que marque mediante '1' los bits que valen '0' a la izquierda del primer '1' de la palabra de entrada).*

5. La multiplicación por constantes aparece en una multitud de aplicaciones de procesado digital de señal. Para evitar el uso de un multiplicador, que es un recurso muy costoso, en ocasiones se utiliza una implementación basada en ROM y sumadores que se basa en la siguiente idea. Supongamos que deseamos multiplicar por una constante un número entero sin signo codificado en binario con 8 bits, *x*. Dado que dicho número puede representarse como 2 dígitos hexadecimales (los nibbles superior e inferior del byte), su valor en notación polinomial en base 16 es: $10_{16}\times(x_{7..4}) + (x_{3..0})$. Si multiplicamos dicha expresión por la constante el valor nos queda una expresión que permite calcular el valor del producto como: $10_{16}\times Cte\times(x_{7..4}) + Cte\times(x_{3..0})$. Obsérvese que esta expresión requiere 2 ROM (16×12) conteniendo el resultado de multiplicar la constante por los valores del 0 al 15 (es decir, por los 16 dígitos hexadecimales posibles), un sumador de 12 bits (para sumar los productos parciales) y 1 desplazador a derechas de 4 bits (para implementar el producto por 10_{16}). Diseñe un circuito que multiplique un número entero sin signo codificado en binario con 8 bits por 47 ($2F_{16}$).

6. Diseñe un conversor de código genérico que dado un número entero con signo codificado en C2 con n bits, lo represente en MyS con n bits. *(Pista: Usar un sumador binario completo de n bits).*

7. Diseñe un conversor de código genérico que dado número entero con signo codificado en MyS con n bits, lo represente en C2 con n bits. *(Pista: Usar un sumador binario completo de n bits).*

8. Diseñe un módulo genérico que calcule el valor absoluto de un número entero con signo codificado en C2 con n bits. El resultado deberá codificarse en binario de n-1 bits. *(Pista: Usar un sumador binario completo de n bits).*

9. Diseñe un sumador genérico de 2 números enteros con signo codificados en MyS de n bits. (*Pista: El circuito tendrá 2 etapas. La primera ordena los números según su magnitud y la segunda examina los signos para decidir si debe sumar o restar las magnitudes para obtener el resultado*).

10. En algunas aplicaciones de procesado digital de señal es conveniente usar sumadores con saturación para reproducir lo que sucedería en el correspondiente circuito analógico cuando los voltajes de salida de los amplificadores se saturan (es decir, llegan al máximo o mínimo valor posible). Un sumador saturado se comporta como un sumador

convencional excepto cuando se desborda. En ese caso, genera a su salida el mayor o menor valor representable en la codificación escogida según corresponda. Diseñe un sumador saturado genérico de números enteros con signo codificados en C2 con n bits. (*Pista: El circuito tendrá 2 etapas. La primera suma normalmente los números y la segunda deja pasar el resultado o el valor saturado que corresponda*).

11. Diseñe un sumador de 2 números enteros sin signo codificados usando n dígitos BCD. El sistema se realizará componiendo n módulos sumadores de 2 dígitos BCD, cada uno de los cuales tendrá 2 entradas de 4 bits de datos, 1 entrada binaria de acarreo, 1 salida de 4 bits de datos y 1 salida binaria de acarreo. *(Pista: La suma deberá calcularse dígito a dígito usando sumadores binarios completos de 4 bits de manera que si el resultado excede 9 se sume 6, lo que equivale a restar 10, y usar el acarreo resultante en la suma del siguiente dígito).*

12. Diseñe un incrementador de números enteros sin signo codificados usando n dígitos BCD. El sistema tendrá una entrada y una salida de datos ambas de 4·n bits. *(Pista: El sistema deberá realizarse componiendo módulos de 4 bits).*