Actividad 11.2

1) El codificador 8b10b se implementó mediante ROMs con tablas pre cargadas, donde el dato funciona como dirección de la misma.

Como parámetro de entrada se indica si el dato es D o K y la polaridad de RD. Además, se tienen en cuenta los casos especiales en donde cambia los valores generados por el bloque 3b4b del dato tipo D.

Hay datos que no permiten cambio de polaridad, tales como: D.03, D.05, D.06, D.09, D.10, D.11, D.12, D.13, D.14, D.17, D.18, D.19, D.20, D.21, D.22, D.25, D.26, D.28, D.x.1, D.x.2, D.x.5 y D.x.6.

Se agregó una señal de Error indicando que la codificación recibió un parámetro equivocado (direccionando desde la memoria a un resultado nulo, lo cual no corresponde con ninguna codificación correcta).

A continuación se muestra el VHDL del mismo

```
begin
    Ins_DCh : DCh
      PORT MAP(a => Dato, spo => Out_DatoD);

    Ins_Dplus : Dplus
      PORT MAP(a => Plus, spo => Out_PlusD);

    Ins_KCh : KCh
      PORT MAP(a => Dato, spo => Out_DatoK);

    Ins_Kplus : Kplus
      PORT MAP(a => Plus, spo => Out_PlusK);

    Toggle6 <= "111111" when nRD = '1' else "000000";
    Toggle4 <= "1111" when nRD = '1' else "0000";

    D17_7_Case    <= '1' when Dato = "10001" and DnK = '1' else '0';
    Unique_DatoD  <= '1' when (Dato = "00011" or
                               Dato = "00101" or
                               Dato = "00110" or
                               Dato = "00111" or
                               Dato = "01001" or
                               Dato = "01010" or
                               Dato = "01011" or
                               Dato = "01100" or
                               Dato = "01101" or
                               Dato = "01110" or
                               Dato = "10001" or
                               Dato = "10010" or
                               Dato = "10011" or
                               Dato = "10100" or
                               Dato = "10101" or
                               Dato = "10110" or
                               Dato = "11001" or
                               Dato = "11010" or
                               Dato = "11100") and DnK = '1' else '0';

    Unique_PlusD  <= '1' when (Plus = "001" or
                               Plus = "010" or
                               Plus = "101" or
                               Plus = "110" and DnK = '1' else '0';

    Dato <= Dato_In(4 downto 0);
    Plus <= D17_7_Case & Dato_In(7 downto 5); -- Con D17_7_Case se salvan los casos
                                              -- tales como D17.7

    Out_DatoD_Aux <= Out_DatoD XOR Toggle6 when Unique_DatoD = '0' else Out_DatoD;
    Out_PlusD_Aux <= Out_PlusD XOR Toggle4 when Unique_PlusD = '0' else Out_PlusD;

    Out_DatoK_Aux <= Out_DatoK XOR Toggle6;
    Out_PlusK_Aux <= Out_PlusK XOR Toggle4;

    Dato_Out <= Out_DatoD_Aux & Out_PlusD_Aux when DnK = '1' else
                Out_DatoK_Aux & Out_PlusK_Aux;

    Error <= '1' when Out_DatoD = "00000" or Out_PlusD = "000" or
             Out_DatoK = "00000" or Out_PlusK = "000" else '0';
end Arq_Cod_8b10b;
```

2) En el caso del decodificador se pensó de la misma manera, solo que ahora el dato recibido será la dirección de la ROM previamente cargada. En caso de recibir un dato erróneo, en la memoria se dispondrán en esas posiciones un valor que ayudará a detectar que no hay valor acorde a lo recibido. En caso de haberse recibido mal pero así y todo corresponderse con un valor, se analizará posteriormente por CRC por el sistema.

Data_Name	Data	RD	RD+_Number
N00	100011	0	0
N01	100011	1	1
N02	100011	10	2
N03	100011	11	3
N04	100011	100	4
nD23	110111	000101	5
nD08	101000	000110	6
nD07	100111	000111	7
N8	100011	1000	8
nD27	111011	001001	9
nD04	100100	001010	10
D20	010100	001011	11
nD24	111000	001100	12
D12	001100	001101	13
D28	011100	001110	14
K28	011100	001111	15
N16	100011	10000	16
nD29	111101	010001	17
nD02	100010	010010	18
D18	010010	010011	19
nD31	111111	010100	20
D10	001010	010101	21
D26	011010	010110	22
D15	001111	010111	23
nD00	100000	011000	24
D06	000110	011001	25
D22	010110	011010	26
D16	010000	011011	27
D14	001110	011100	28
D01	000001	011101	29
D30	011110	011110	30
N31	100011	11111	31
N32	100011	100000	32
nD30	111110	100001	33
nD01	100001	100010	34
D17	010001	100011	35
nD16	110000	100100	36
D09	001001	100101	37
D25	011001	100110	38
D00	000000	100111	39
nD15	101111	101000	40
D05	000101	101001	41
D21	010101	101010	42
D31	011111	101011	43
D13	001101	101100	44
D02	000010	101101	45
D29	011101	101110	46
N47	100011	101111	47
nK28	111100	110000	48
D03	000011	110001	49
D19	010011	110010	50
D24	011000	110011	51
D11	001011	110100	52
D04	000100	110101	53
D27	011011	110110	54
N55	100011	110111	55
D07	000111	111000	56
D08	001000	111001	57
D23	010111	111010	58
N59	100011	111011	59
N60	100011	111100	60
N61	100011	111101	61
N62	100011	111110	62
N63	100011	111111	63

Esta tabla muestra la totalidad de la memoria de datos implementada (tener en cuenta que se cambiaron muchos valores respecto a la tabla de la presentación del módulo).

RD_Number corresponde a la dirección de memoria en la cual se cargará cada valor en el COE File.

Aquí se pueden observarse 4 colores diferentes:

- Valor sin polaridad en RD
- Valor con RD+
- Valor con RD-
- Valor inexistente

En el caso de los datos con RD- se le agregó un ‘1’ al inicio para poder discriminarlos. En caso de dato inexistente, se le asignó la “versión negada” de un dato sin polaridad (por lo que no puede estar en ‘1’ el primer bit).

Para saber si fue un D o un K, se realizará el siguiente procedimiento:

1. Buscar valor en tabla para ver si es analizable
2. Analizar si es uno de los 12 K-Characters, en dicho caso procede a decir que es un dato K
3. Caso contrario, se busca un dato D, para lo cual no puede ser la dirección 15 (caso prohibido, solo hay dato K con codificación RD+ = “001111”)

Las siguientes tablas corresponden a las tablas de los extensores. La columna RD* significa que se modificaron las posiciones respecto a lo que llega, ya que al compartir valores, la polaridad de RD la define la tabla de búsqueda de datos anteriores (la de 6 bits).

Data_Name	Data	RD*	Num_RD
N00	1000	0000	0
N01	1000	0001	1
N02	1000	0010	2
N03	1000	0011	3
N04	1000	0100	4
D02	0010	0101	5
D06	0110	0110	6
N07	1000	0111	7
N08	1000	1000	8
D01	0001	1001	9
D05	0101	1010	10
D00	0000	1011	11
D03	0011	1100	12
D04	0100	1101	13
D07	0111	1110	14
N15	1000	1111	15

Data_Name	Data	RD*	Num_RD
N00	1000	0000	0
N01	1000	0001	1
N02	1000	0010	2
N03	1000	0011	3
N04	1000	0100	4
K05	0101	00101	5
K01	0001	00110	6
K07	0111	00111	7
N08	1000	1000	8
K06	0110	01001	9
K02	0010	01010	10
K00	0000	01011	11
K03	0011	01100	12
K04	0100	01101	13
N14	1000	1110	14
N15	1000	1111	15

A todo esto, y a modo de pre-código, debe tenerse en cuenta que no todas las combinaciones son posibles, y esto será parte de la detección primaria de errores del decodificador.

A continuación se muestra el VHDL del mismo

```
begin

    Dato  <= Dato_In(9 downto 4);

    Ins_DCh : Inv_Ch
      PORT MAP(a => Dato, spo => Out_DatoD);

    nRD_Aux <= Out_DatoD(Out_DatoD'high);

    Toggle4 <= "1111" when nRD_Aux = '1' else "0000";

    Plus  <= Dato_In(3 downto 0) XOR Toggle4;

    Ins_Dplus : Inv_Dplus
      PORT MAP(a => Plus, spo => Out_PlusD);

    Ins_Kplus : Inv_Kplus
      PORT MAP(a => Plus, spo => Out_PlusK);

    Dato_Out_Aux_D <= Out_PlusD(2 downto 0) & Out_DatoD(Out_DatoD'high-1 downto 0);
    Dato_Out_Aux_K <= Out_PlusK(2 downto 0) & Out_DatoD(Out_DatoD'high-1 downto 0);

    DnK_Aux <= '0' when Dato_Out_Aux_K = x"38" or -- K.28.0
                    Dato_Out_Aux_K = x"3C" or -- K.28.1
                    Dato_Out_Aux_K = x"3A" or -- K.28.2
                    Dato_Out_Aux_K = x"3E" or -- K.28.3
                    Dato_Out_Aux_K = x"39" or -- K.28.4
                    Dato_Out_Aux_K = x"3D" or -- K.28.5
                    Dato_Out_Aux_K = x"3B" or -- K.28.6
                    Dato_Out_Aux_K = x"3F" or -- K.28.7
                    Dato_Out_Aux_K = x"EF" or -- K.23.7
                    Dato_Out_Aux_K = x"DF" or -- K.27.7
                    Dato_Out_Aux_K = x"BF" or -- K.29.7
                    Dato_Out_Aux_K = x"7F"    -- K.30.7
               else '1';

    Dato_Out <= Dato_Out_Aux_K when DnK_Aux = '0' else "111" & Out_DatoD(Out_DatoD'high-1 downto 0) when Dato_Out_Aux_D = x"F1" else Dato_Out_Aux_D;

    DnK <= DnK_Aux;
    nRD <= nRD_Aux;
    Error <= '1' when Out_DatoD = "100011" or Out_PlusD = "1000" or Out_PlusK = "1000" or Out_DatoD(Out_DatoD'high-1 downto 0) = "11100" and DnK = '1' else '0';
end Arq_Dec_8b10b;
```

3) La ventaja es que se aumenta mucho la tasa de datos últiles, bajando la relación de los bits agregados respecto a los originales. La desventaja que puedo ver al realizar los módulos es que se necesitarían memorias muy grandes (muy demasiadamente inmensas) para manejarlo de esta manera, lo cual hace imposible la solución por tabla y debería de buscarse otra alternativa.