PART -1

```
module bintoascii (
  input [111:0] binary_input,
  output reg [6:0] part_7790_0, part_7790_1, part_7790_2, part_7790_3, part_7790_4, part_7790_5,
                   part_7790_6, part_7790_7, part_7790_8, part_7790_9, part_7790_10,
                   part_7790_11, part_7790_12, part_7790_13, part_7790_14, part_7790_15
);

  always @* begin
    part_7790_0  <= binary_input[6:0];
    part_7790_1  <= binary_input[13:7];
    part_7790_2  <= binary_input[20:14];
    part_7790_3  <= binary_input[27:21];
    part_7790_4  <= binary_input[34:28];
    part_7790_5  <= binary_input[41:35];
    part_7790_6  <= binary_input[48:42];
    part_7790_7  <= binary_input[55:49];
    part_7790_8  <= binary_input[62:56];
    part_7790_9  <= binary_input[69:63];
    part_7790_10 <= binary_input[76:70];
    part_7790_11 <= binary_input[83:77];
    part_7790_12 <= binary_input[90:84];
    part_7790_13 <= binary_input[97:91];
    part_7790_14 <= binary_input[104:98];
    part_7790_15 <= binary_input[111:105];
  end

endmodule
```

The "bintoascii" Verilog module transforms a binary input into various ASCII elements. The module is described as follows:

1. Inputs:
    - binary_input (112-bit input): This is a binary input that contains the ASCII characters that need to be divided.
2. Outputs:
    - part0 to part15 (7-bit outputs): These show the 16 ASCII character parts that were derived from the binary input. 7 bits are assigned to each component**.**
3. Behavior:
    - The always @* block specifies a combinational logic that runs each time the inputs (binary_input) change.
    - The individual parts (part0 to part15) are assigned values based on specific ranges of bits from the binary_input.
    - For example, part0 is assigned the value of binary_input[6:0], which extracts the first 7 bits from binary_input.
    - Similarly, the other parts are assigned values based on different ranges of bits from binary_input.

The purpose of this module is to divide a 112-bit binary input into 16 smaller parts, each representing a specific section of the ASCII characters being encoded.

I tried to do with arrays, but I couldn't make it because I was getting some errors and after searching about that errors that I knowledge that Verilog doesn’t support arrays.

OUTPUT

```
# part_7790_0  = H
# part_7790_1  = e
# part_7790_2  = l
# part_7790_3  = l
# part_7790_4  = o
# part_7790_5  =
# part_7790_6  = C
# part_7790_7  = O
# part_7790_8  = M
# part_7790_9  = P
# part_7790_10 = 3
# part_7790_11 = l
# part_7790_12 = l
# part_7790_13 = -
# part_7790_14 = 2
# part_7790_15 = !
# ** Note: $finish    : C:/intelFPGA_lite/18.1/navaneet/bintoascii/bintoascii_tb.v(168
#    Time: 234 ns  Iteration: 0  Instance: /bintoascii_tb
```

PART -2

I implemented the same logic in my coding also.

7 BIT -BINARY	11BIT HAMMING CODE	HEXA DECIMAL
1. 1001000	00110010000	190
2. 1100101	10111000101	1C5
3. 1101100	00101011100	15C
4. 1101100	00101011100	15C
5. 1101111	10101011111	55F
6. 0100000	10011000000	4C0
7. 1000011	01100000011	303
8. 1001111	00110011111	19F
9. 1001101	01110010101	395F
10. 1010000	10110100000	5A0
11. 0110011	01001100011	263
12. 0110001	00001101001	69
13. 0110001	00001101001	69
14. 0101101	00001010101	55
15. 0110010	10001101010	46A
16. 0100001	01011001001	2C9

PART-2

* I got 16 outputs from part 1 each output Contain 7bit binary number
* I am Converting all the 16 Outputs into hamming Code
* The obtained hamming codeword is Converted into hexa-dec

Conversion :-

we know that $2^p - 1 \geq m+1$

where p is no. of parity bits

m is no of message bits

In our Case p satisfies '4'

* I am taking Hamming Code length is 11 ~~bits~~

1	2	3	4	5	6	7	8	9	10	11
P_1	P_2	m_6	P_3	m_5	m_4	m_3	P_4	m_2	m_1	m_0

Our first output is 1001000 So

P_1	P_2	1	P_3	0	0	1	P_4	0	0	0

To obtain P_1

we need to XOR the places

$P_1 = 3 \oplus 5 \oplus 7 \oplus 9 \oplus 11$

$P_1 = 0$

For P_2

$P_2 = 3 \oplus 6 \oplus 7 \oplus 10 \oplus 11 = \boxed{P_2 = 0}$

For P_3 we need to XOR

$P_3 = 5 \oplus 6 \oplus 7$

$= 0 \oplus 0 \oplus 1$

$\boxed{P_3 = 1}$

For P_4 we need to XOR

$P_4 = 9 \oplus 10 \oplus 11$

$= 0 \oplus 0 \oplus 0$ $\boxed{P_4 = 0}$

The required 11 bit hamming Code is 00110010000

OUTPUT

```
out_77990_0  = 190
out_77990_1  = 1c5
out_77990_2  = 15c
out_77990_3  = 15c
out_77990_4  = 55f
out_77990_5  = 4c0
out_77990_6  = 303
out_77990_7  = 19f
out_77990_8  = 395
out_77990_9  = 5a0
out_77990_10 = 263
out_77990_11 = 069
out_77990_12 = 069
out_77990_13 = 055
out_77990_14 = 46a
out_77990_15 = 2c9
```

QOR

```
****************************************
Report : qor
Design : hamming
Version: S-2021.06-SP4
Date   : Tue May 23 12:37:22 2023
****************************************


  Timing Path Group (none)
  -----------------------------------
  Levels of Logic:               2.00
  Critical Path Length:          0.43
  Critical Path Slack:         uninit
  Critical Path Clk Period:       n/a
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------


  Cell Count
  -----------------------------------
  Hierarchical Cell Count:         16
  Hierarchical Port Count:        288
  Leaf Cell Count:                112
  Buf/Inv Cell Count:               0
  Buf Cell Count:                   0
  Inv Cell Count:                   0
  CT Buf/Inv Cell Count:            0
  Combinational Cell Count:       112
  Sequential Cell Count:            0
  Macro Count:                      0
  -----------------------------------


  Area
  -----------------------------------
  Combinational Area:      585.547768
  Noncombinational Area:     0.000000
  Buf/Inv Area:              0.000000
  Total Buffer Area:             0.00
  Total Inverter Area:           0.00
  Macro/Black Box Area:      0.000000
  Net Area:                 67.630114
  -----------------------------------
  Cell Area:               585.547768
  Design Area:             653.177881


  Design Rules
  -----------------------------------
  Total Number of Nets:           224
  Nets With Violations:             0
  Max Trans Violations:             0
  Max Cap Violations:               0
  -----------------------------------
```