CS220: Lab#12B

1. [5 marks] Implement a 16-to-4 encoder that takes a 16-bit value as input, which has exactly one bit set and all other bits reset. The encoder produces the encoded four-bit output corresponding to the 16-bit input i.e., if the n^{th} bit of the input is set, the output should be n where $0 \leq n \leq 15$. If the input has all bits zero or has more than one bit set to one, the output should be -1 in eight-bit two's complement representation. Store the following eight values in the eight rows of a memory with each row being 16-bit wide.

```
0000 0000 0000 0000
1000 1000 0000 0000
0000 0001 0000 0000
1000 0000 0000 0000
0000 0000 0000 0001
0000 1000 0000 0000
1000 0001 0001 0000
0000 0000 1000 0000
```

Use an `initial` block to store these. Maintain a row counter register initialized to zero. On every posedge of clock, you should read the row pointed to by the row counter, encode the value in that row, and increment the row counter. After encoding the eight values, show the sum of all encoded outputs in the LEDs (LED7 is the most significant bit). Also, show the parity of the sum. The parity is defined to be 0 if the number of 1's in the binary representation of the sum is even; otherwise the parity is 1. Use LED0 to display the parity. The display should be changed from the sum value to the parity when a push button is pressed.