Problem Statement:

**Single Bit Error Correction and Detection using Hamming Code**

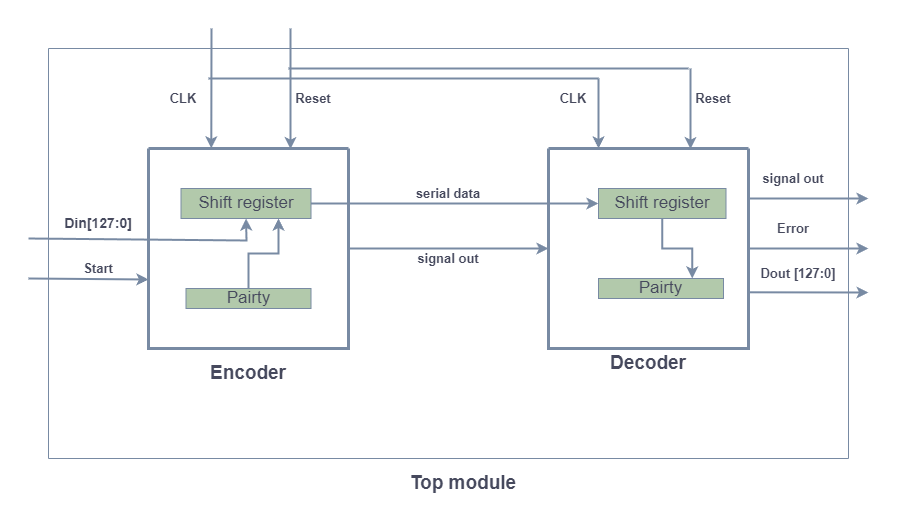
**By,**

**Pavan Myagerimath 01FE20BEC199**

**Harsha Jangamani 01FE20BEC202**

Architecture:

The design consists of two sub-systems namely encoder and decoder.



Encoder:

The encoder takes a 128-bit input data and generates a 136-bit output data with 8 parity bits.

The encoder works as follows:

1.The input data is first checked for parity.If any of the 8 parity bits is incorrect,the encoder will add the necessary parity bits to make the data correct.

2.After the input data has been verified to have correct parity,the encoder will add 8 parity bits to the data to generate the output data.

Decoder:

1.The decoder receives a serial input of data and calculates parity bits .

2.After calculating parity bits,it tries to match bits to value zero.If the equals to zero, it means that there is no error in received data and transmit the same data.

3.If not equal to zero then ,it flips that bit to either from 0 to 1 or from 1 to 0 depending on the value of parity in that location.

4.After flipping the bit,in the data extraction stage original data and parity bits are separated and original data is transmitted to output port.

Theory:

Hamming Code:

Hamming code is a linear error-correcting code named after its inventor, Richard Hamming. Hamming codes can detect up to two simultaneous bit errors, and correct single bit error.

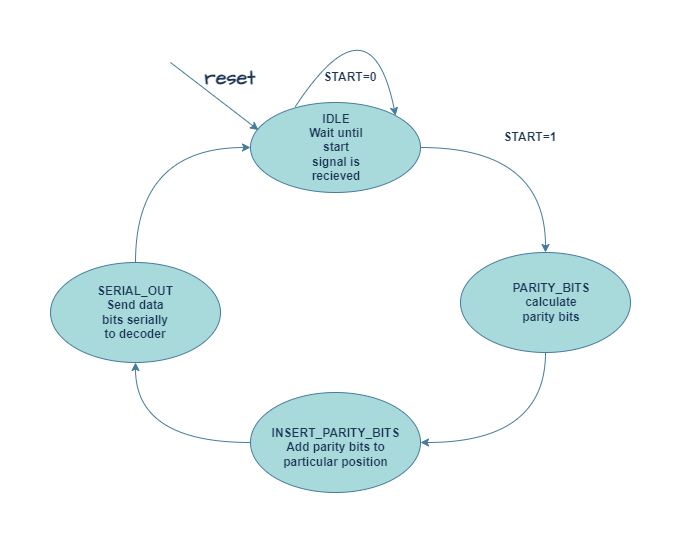
Error Detection and Correction:

For a given practical requirement, detection of errors is simpler than the correction of errors. The decision for applying detection or correction in a given code design depends on the characteristics of the application. When the communication system is able to provide a full duplex transmission (that is, a transmission for which the source and the destination can communicate at the same time, and in a two way mode, as it is in the case of telephone

connection, for instance), codes can be designed for detecting errors, because the correction is performed by requiring a repetition of the transmission.

Finite State Machine:

Encoder:



The module has four states:

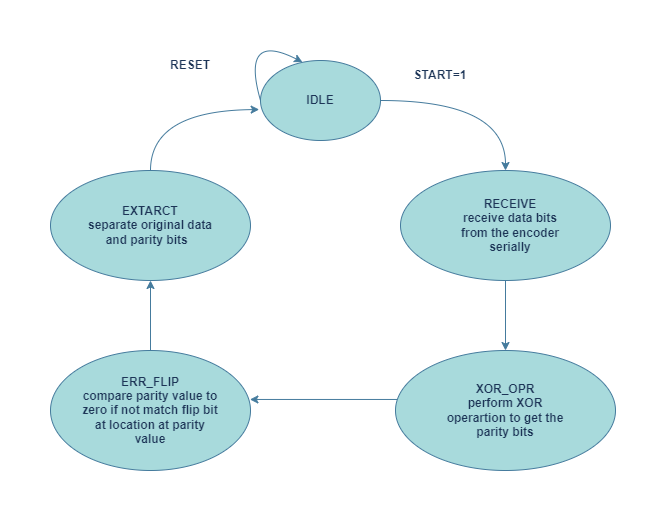
Idle:The initial state, where the encoder waits for the start signal.

Check bits: The state where the encoder checks the input data for parity errors.

Add check bits: The state where the encoder adds the necessary parity bits to the input data.

Serial out: The state where the encoder generates the output data and outputs it.

Decoder:



The decoder module has five states:

1.In the idle state, the decoder is waiting for the start signal to be asserted. When start is asserted, the decoder transitions to the receive state.

2.In the receive state, the decoder receives the input data one bit at a time and stores it in the shift register. The decoder counter register is used to keep track of the number of bits received. When all 136 bits have been received, the decoder transitions to the xor\_opn state.

3.In the xor\_opn state, the decoder calculates the parity bits using a series of XOR operations on the bits stored in the shift register. The calculated parity bits are stored in the parity register, and the decoder transitions to the error\_flip state.

4.In the error\_flip state, the decoder compares the calculated parity bits with the parity bits stored in the shift register. If there is a mismatch, the error signal is set and the decoder transitions to the extract state. Otherwise, the decoder transitions directly to the extract state.

5.In the extract state, the decoder extracts the 128 bits of data from the shift register and stores them in the dout register. The sig\_out signal is set to indicate that the output data is valid, and the decoder transitions back to the idle state.

Results:

**A screenshot of a computer

Description automatically generated**

Conclusion:

In conclusion, 64-bit data strings may be used to construct Hamming code error detection and correction with parity check method in Verilog HDL.

It speed up communication because there is no longer a requirement for data splitting and more data combinations (more information in a single frame) may be communicated with ease. This is made possible by encoding all of the data bits simultaneously and sending them. By employing the same technique at both ends, the complexity of the circuit is also decreased when it comes to recovering the original data from encrypted corrupt received data at the destination end.