EDCHC: Error Detection and Correction based on Hamming Code

Raja Aadhithan, {taadhithan_mtech21@thapar.edu}
Department of Electronics and Communication,
Thapar Institute of Engineering and Technology, Patiala, Punjab, India

Introduction on Hamming code:

Hamming code makes use of the concept of parity and parity bits, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission. Using more than one parity bit, an error-correction code can not only identify a single bit error in the data unit, but also its location in the data unit.

In this paper we will be first designing a circuit to generate hamming code, then one more design to detect and correct a single bit error.

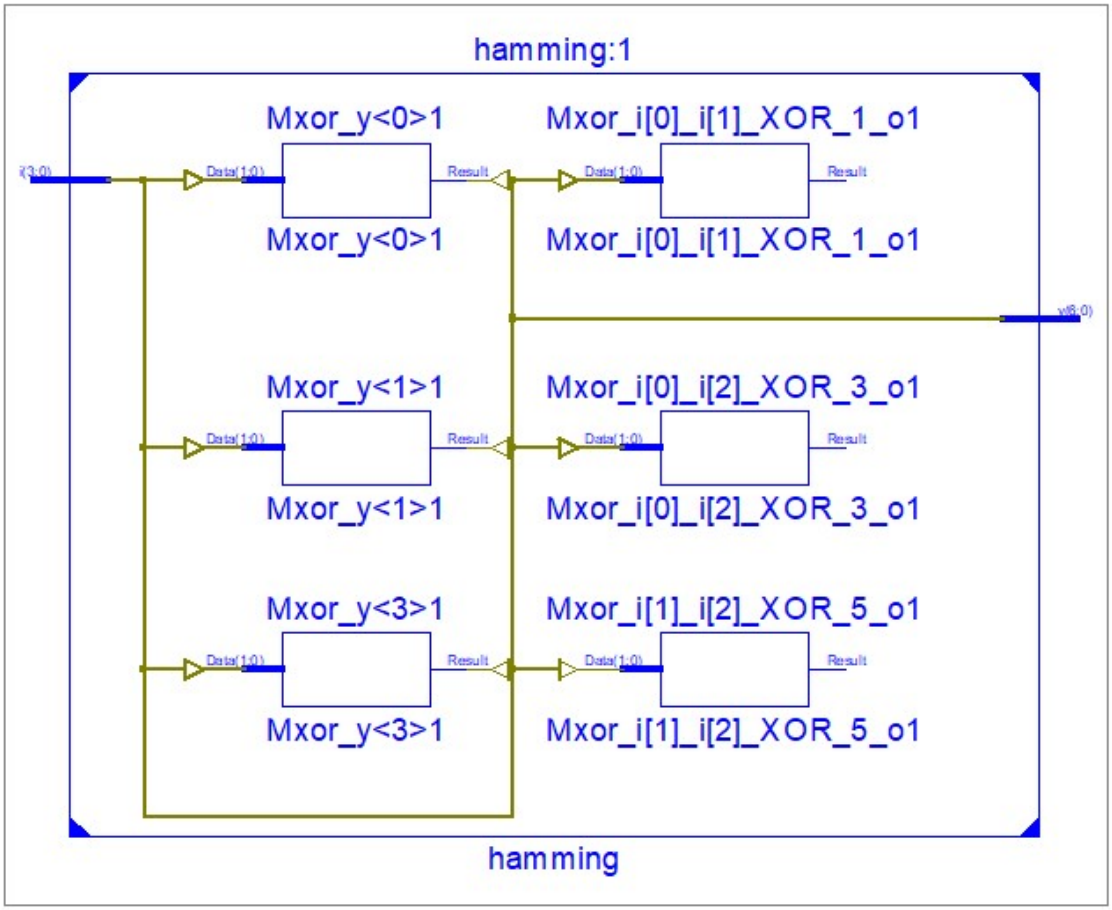
Code Generator:

First we implement a module which can generate a 7 bit hamming code output for a 4 bit hamming code input. The source codes of the DUTs and Test benches could be found in https://github.com/raja-aadhithan/M.Tech_Verilog/tree/main/Hamming

The output of the generator is:

```
VCD info: dumpfile dump.vcd opened for output.
input is 0100- output is 0101010
input is 0001- output is 0000111
input is 1001- output is 1001100
input is 0011- output is 0011110
input is 1101- output is 1100110
input is 0101- output is 0101101
input is 0010- output is 0011001
input is 0001- output is 0000111
input is 1101- output is 1100110
0
Finding VCD file...
./dump.vcd
[2021-11-23 03:54:50 EST] Opening EPWave...
Done
```

RTL of the DUT:



Error Detection and Correction:

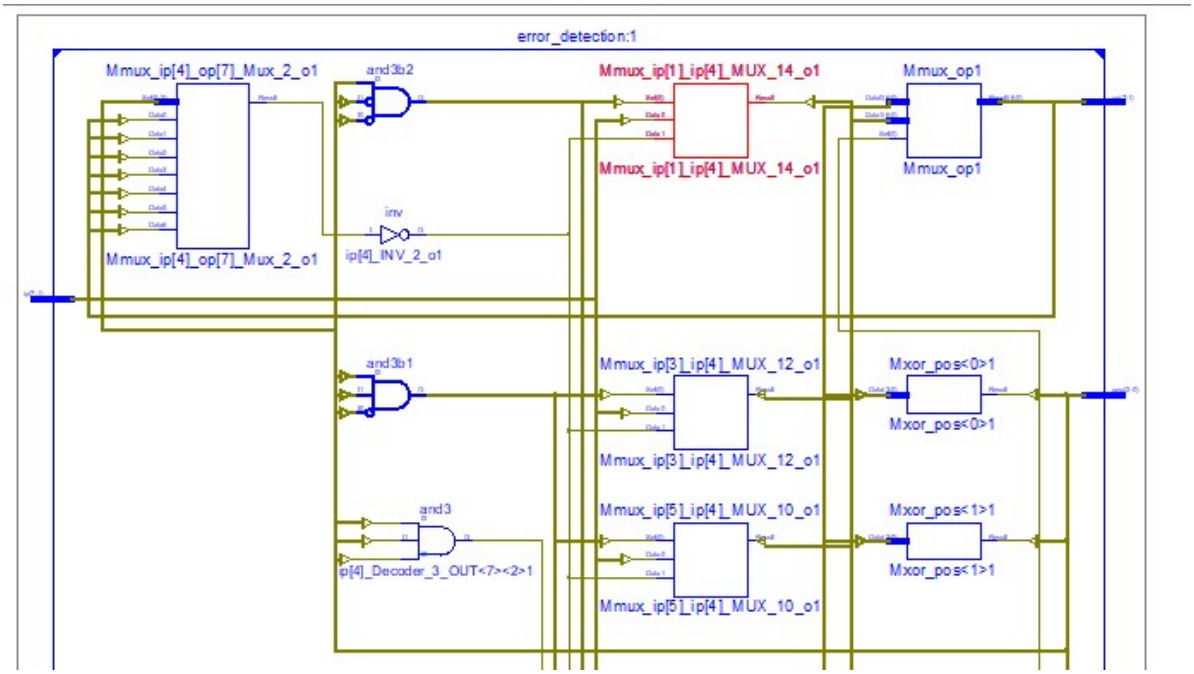
The error detection DUT could detect if one bit is manipulated and can restore its value, but if more than 1 value are changed, the output will still show a valid hamming code but not the original one.

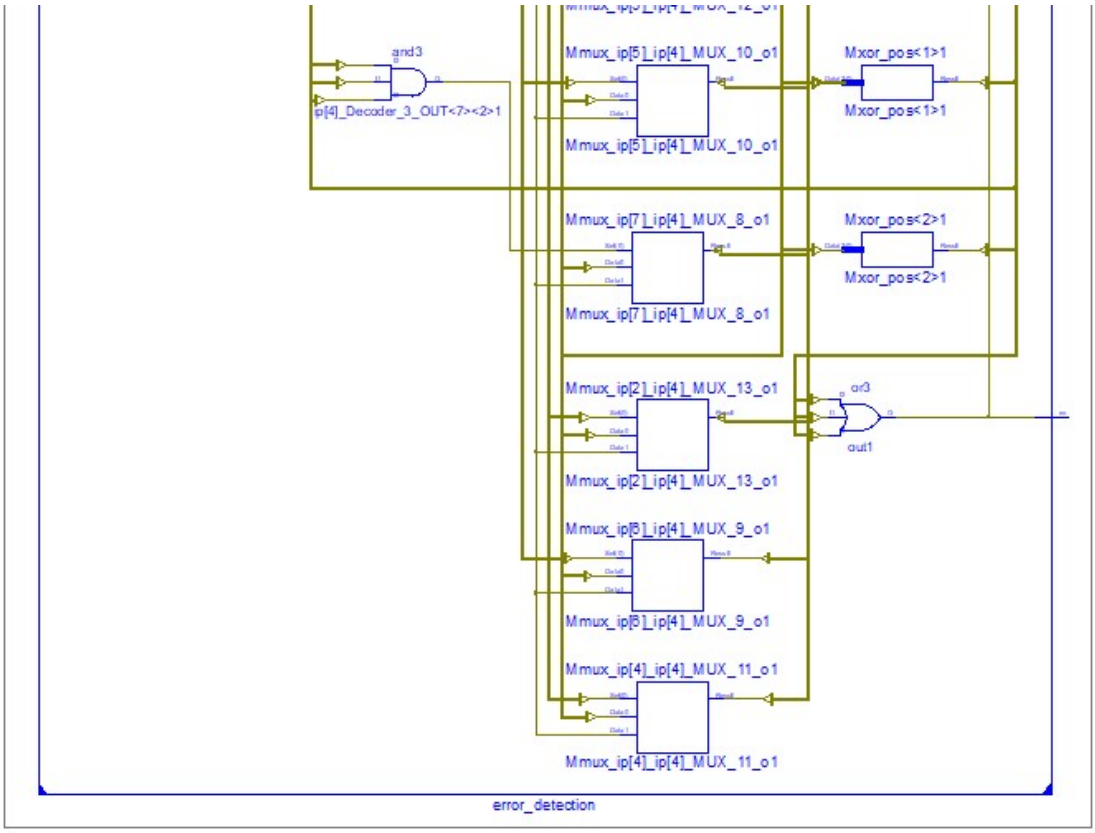
The output also shows the position at which the bit was wrong.

Simulation output:

```
[2021-11-23 04:00:55 EST] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
24,34,1,101,x
24,34,1,101,0
01,00,1,001,0
09,19,1,101,0
63,61,1,010,0
0d,2d,1,110,0
65,61,1,011,0
12,52,1,111,0
01,00,1,001,0
0d,2d,1,110,0
Done
```

RTL of the error corrector:




and3
ip[4]_Decoder_3_OUT<7><2>1
Mmux_ip[5]_ip[4]_MUX_10_o1
Mmux_ip[5]_ip[4]_MUX_10_o1
Mxor_pos<1>1
Mxor_pos<1>1
Mmux_ip[7]_ip[4]_MUX_8_o1
Mmux_ip[7]_ip[4]_MUX_8_o1
Mxor_pos<2>1
Mxor_pos<2>1
Mmux_ip[2]_ip[4]_MUX_13_o1
Mmux_ip[2]_ip[4]_MUX_13_o1
or3
out1
Mmux_ip[6]_ip[4]_MUX_9_o1
Mmux_ip[6]_ip[4]_MUX_9_o1
Mmux_ip[4]_ip[4]_MUX_11_o1
Mmux_ip[4]_ip[4]_MUX_11_o1
error_detection