Question

Please design a RTL module that computes hamming weight of a binary sequence of length 1024. The hamming weight of the sequence is always less than or equal to 31. The module also need to output the locations of 1's after receiving all the bits of a sequence in no more than 31 clock cycles (one location at a time). There is a start packet signal and this is asserted every 129 clock cycles. After the assertion of start packet signal, the bits would be input for 128 clock cycles at the rate of 8 bits/clk. This process is continuous.

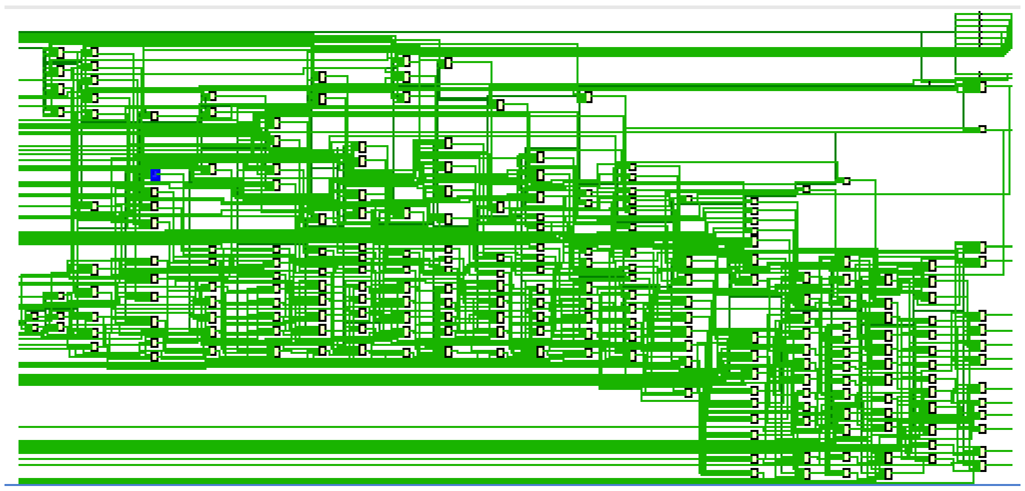
A standard algorithm was followed to locate the indices of ones and was shifted to calculate the indices of where the ones are located in the vector to calculate the hamming weight as well as output the positions of ones according to the input received.

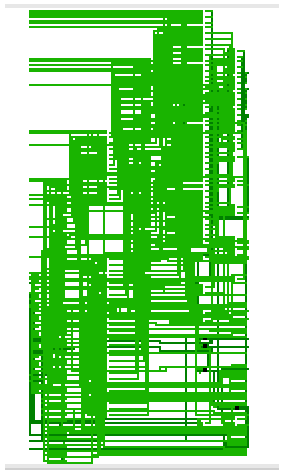
Keeping this in mind, this was the Verilog code used and simulated using modelsim:

```
module hamwt_locations(
clk,
clear,
pkt_starts,
bin_data,
ham_wt,
locn_ones
);


input clk;
input clear;
input pkt_starts;
input [7:0] bin_data; //packets of 8bit data 128 times (128*8=1024 is the entire data sequence)


output reg [4:0] ham_wt;            //max is 31, i.e 5 bit
output reg [9:0] locn_ones; //1024 possible locations, i.e 10 bit output for 1's in the sequence



wire clk;
wire clear;
wire pkt_starts;
wire [7:0] bin_data;


reg i;

reg locnofones;

reg hw;

reg [31:0] index;


always @(posedge clk) begin

    if((pkt_starts == 1) || (clear == 1)) begin

      index = 0;

      locnofones = 0;

             hw = 0;

    end

    else begin

      for (i=0; i < bin_data; i = i + 1) begin

        if(bin_data[i] == 1) begin

//every time it sees a 1, add for the hamming weight

          locnofones = locnofones << 10 + (index+i);

                          hw=hw+1;

        end

      end

      index <= index + 1;

    end

      ham_wt <= hw;

   locn_ones <= locnofones;

    end

endmodule
```

The design was synthesized using the xilinx vivado tool and the constraints were set. Some of the snapshots of the implemented design are as shown below.

Synthesis was checked at multiple timing constraints with the clock input at 10 to 100ns. The highest operating frequency achieved was 0.1 GHz.





```
Report Cell Usage:
+------+-------+------+
|      |Cell   |Count |
+------+-------+------+
|1     |BUFG   |     1|
|2     |CARRY8 |    39|
|3     |LUT1   |     1|
|4     |LUT2   |    10|
|5     |LUT3   |    41|
|6     |LUT4   |    13|
|7     |LUT5   |  1175|
|8     |LUT6   |    76|
|9     |FDRE   |   322|
|10    |IBUF   |    11|
|11    |OBUF   |   315|
+------+-------+------+

Report Instance Areas:
+------+---------+-------+------+
|      |Instance |Module |Cells |
+------+---------+-------+------+
|1     |top      |       |  2004|
+------+---------+-------+------+
```

Using matlab, and functions of random generator, a test vector was generated for the hamming weight and the locations of ones. A testbench was created in verilog to observe the outputs. However, given the time constraints, the outputs comparing the RTL and matlab results are not shown here. In terms of time taken, one hour was spent to understand the requirements and write a pseudo code for the question. A total of 5 hours were taken to compile, synthesise and create a testbench to see the outputs. Linking the matlab results and RTL results took more time than expected, hence the comparison has not been shown here.