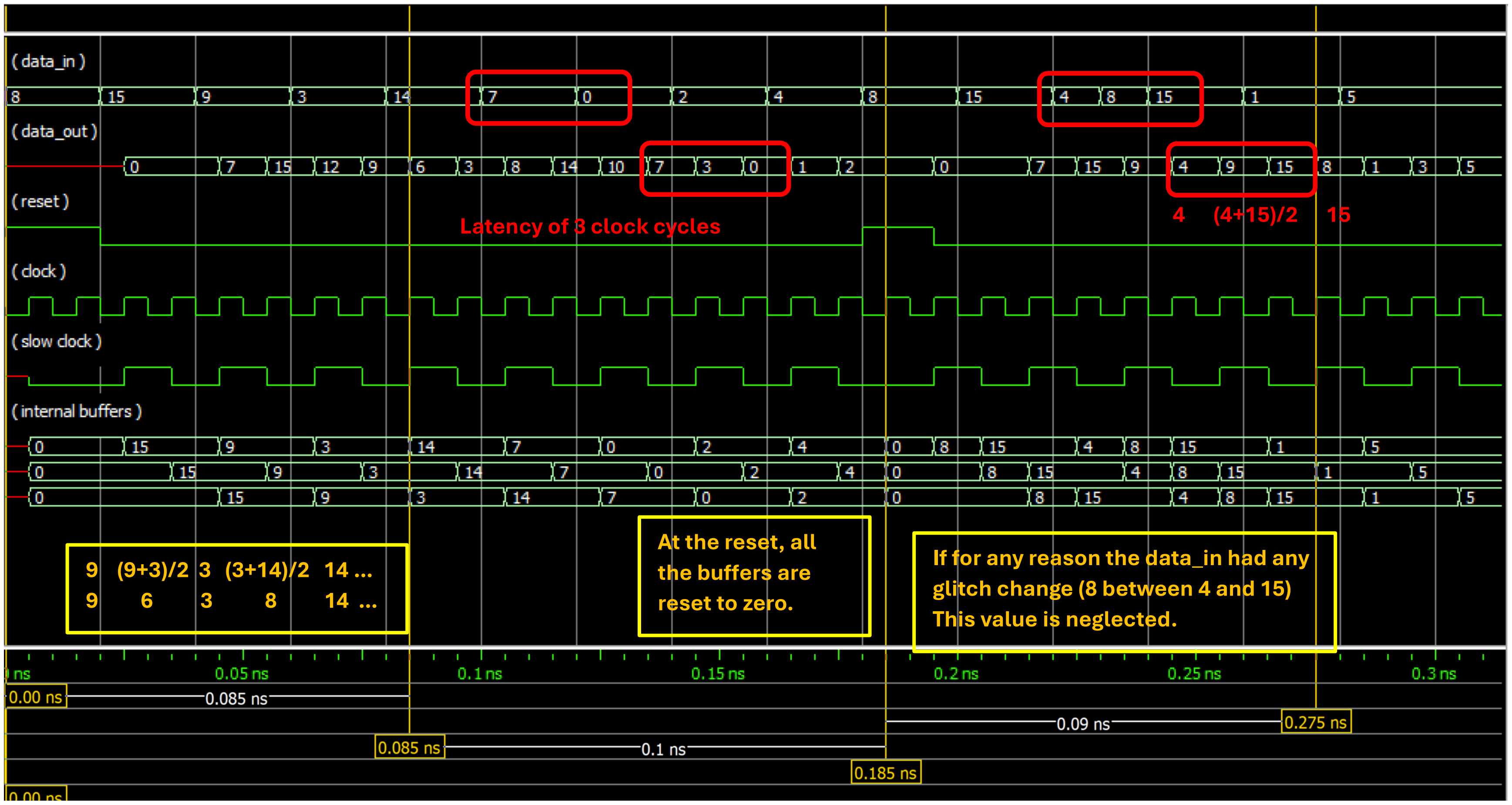
```verilog
/*------------------------------------------------------------------------------------------
           Lesson 9: Digital Signal Processing Applications
--------------------------------------------------------------------------------------------
Design a linear interpolator that would double the sampling frequency. The code should
be parametrizable with a flexible word size, but you can assume that it will always
double the sampling frequency by inserting one sample between adjacent samples received.
The inserted interpolated word should represent the average of the two received samples.
The output should be operating at the faster clock, and the input should update at the
slower clock frequency. This should be clear in the test bench. The module should
have a reset.
Combine your code for the interpolator, the testbench, and simulation results
in one pdf file.
------------------------------------------------------------------------------------------*/

module Interpolator (data_in, clk, reset, data_out);

parameter word_size = 4;
input     [word_size-1   : 0]   data_in;
output    [word_size-1   : 0]   data_out;
input                           reset, clk;

reg       [word_size    :  0]   data_reg;         // enough bits to calculate the sum
reg                             slow_clk;
reg       [word_size-1 :   0]   buffer [0:2];     // to guarantee sample alignment

assign  data_out = data_reg [word_size-1 :0];


always @ (posedge clk)
       if (reset) begin
                  slow_clk   <= 0;
                  buffer[0]  <= 0;
                  buffer[1]  <= 0;
                  buffer[2]  <= 0;  end
         else     begin
                  slow_clk   <= ~slow_clk;
                  buffer [0] <= data_in;             // input reg to guarantee sample alignment
                  buffer [1] <= buffer[0];           // even if the input is off sync with clock
                  buffer [2] <= buffer[1];
                  data_reg   <= slow_clk ? {1'b0,buffer[1]} : (( buffer[0] + buffer[2] )>>1) ;
                  end


endmodule
```


(data_in)
(data_out)
(reset)
(clock)
(slow clock)
(internal buffers)
Latency of 3 clock cycles
4 (4+15)/2 15
9 (9+3)/2 3 (3+14)/2 14 ...
9 6 3 8 14 ...
At the reset, all the buffers are reset to zero.
If for any reason the data_in had any glitch change (8 between 4 and 15) This value is neglected.
0.05 ns
0.1 ns
0.15 ns
0.2 ns
0.25 ns
0.3 ns
0.00 ns
0.085 ns
0.1 ns
0.185 ns
0.09 ns
0.275 ns