1- The testbench can be like this

```
#include <iostream>

void simple_design_02(bool start, bool input1, bool input2, bool &output);

#define N 10
#define M 16


int main() {
    int status = 0;

    bool input_1;
    bool input_2;
    bool output;
    bool start;

    bool data1[M] = {1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0};
    bool data2[M] = {1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0};

    std::cout << std::endl;

    start = 1;
    input_1 = 0;
    input_2 = 0;
    simple_design_02(start, input_1, input_2, output);
    std::cout << output;

    for (int i = 0; i < N; i++) {
          for (int j = 0; j < M; j++) {
                input_1 = data1[j];
                input_2 = data2[j];
                simple_design_02(start, input_1, input_2, output);
                std::cout << output;
          }
    }


    return status;
```

2- The following code solves the problem.

```
void parallel_serial_shift_register(ap_uint<4> d, bool load, bool din, bool
&dout)
{
#pragma HLS INTERFACE ap_none port=d
#pragma HLS INTERFACE ap_none port=din
#pragma HLS INTERFACE ap_none port=dout
#pragma HLS INTERFACE ap_none port=load
#pragma HLS INTERFACE ap_ctrl_none port=return

    static ap_uint<4> d_register = 0;
    ap_uint<4>   next_d_register = 0;

    if (load == 1) {
          next_d_register = d;
    } else {
          next_d_register = d_register >> 1;
          next_d_register[3] = din;
    }

    d_register = next_d_register;
    dout = d_register[0];
}
```

```
int main() {

      int status = 0;

      ap_uint<4> d;
      bool       load;
      bool       din;
      bool       dout;



      load = 0; din = 1;
      parallel_serial_shift_register(d,  load,  din, dout);
      std::cout << " dout = " << dout << std::endl;

      load = 1; din = 0; d=0b1010;
      parallel_serial_shift_register(d,  load,  din, dout);
      std::cout << " dout = " << dout << std::endl;

      load = 0; din = 0;
      parallel_serial_shift_register(d,  load,  din, dout);
      std::cout << " dout = " << dout << std::endl;

      load = 0; din = 1;
      parallel_serial_shift_register(d,  load,  din, dout);
      std::cout << " dout = " << dout << std::endl;

      load = 0; din = 1;
      parallel_serial_shift_register(d,  load,  din, dout);
```

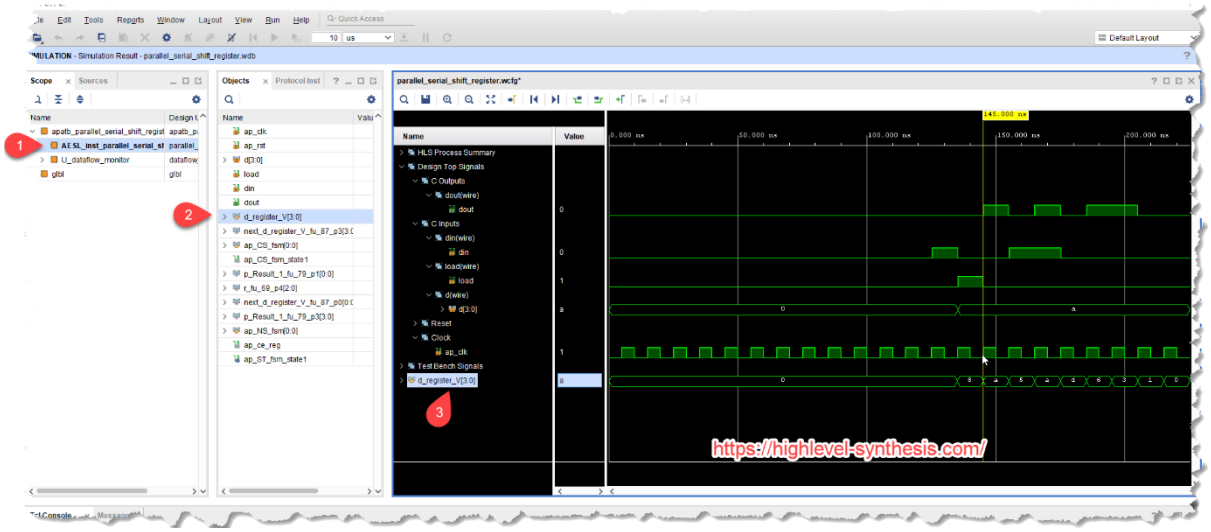
```
        std::cout << " dout = " << dout << std::endl;

        load = 0; din = 0;
        parallel_serial_shift_register(d,  load,  din, dout);
        std::cout << " dout = " << dout << std::endl;




        return status;
}
```

The following figure shows the waveform after RTL/C co-simulation



You can add the d_resgister to the waveform to inspect the design's internal nodes.