The following code answers the quiz question.

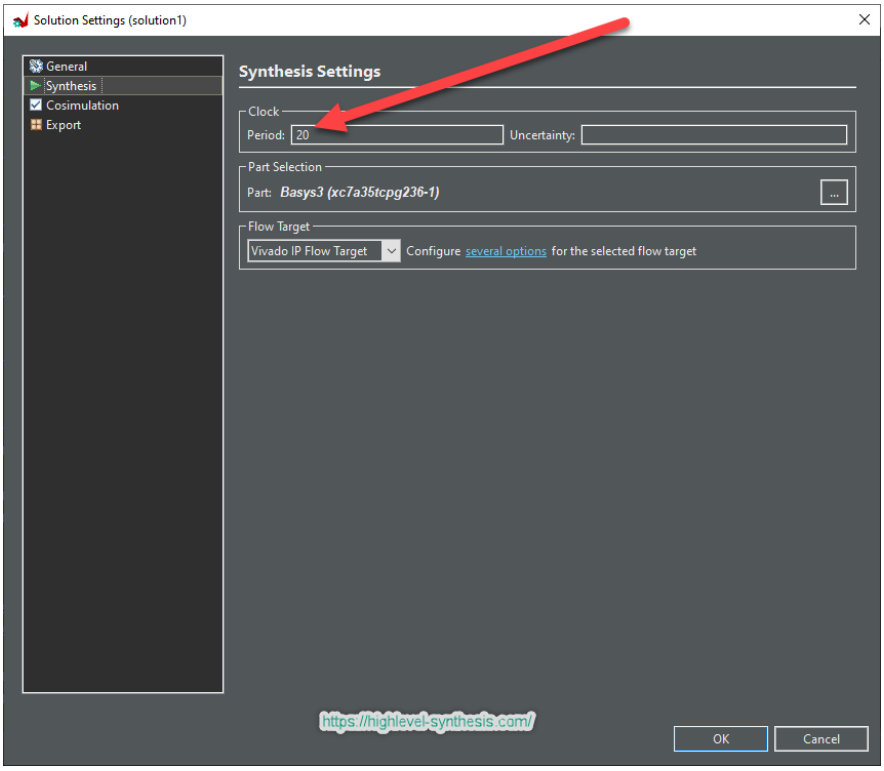
```
void updown_counter(
              bool        up_count_button,
              bool        down_count_button,
              ap_uint<8> &seven_segments_data,
              ap_uint<4> &seven_segments_enable) {
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=up_count_button
#pragma HLS INTERFACE ap_none port=down_count_button
#pragma HLS INTERFACE ap_none port=seven_segments_data
#pragma HLS INTERFACE ap_none port=seven_segments_enable


       static ap_uint<5>         number = 0;
       static counter_state_type state  = wait_for_one;



       ap_uint<5>         next_number;
       counter_state_type next_state;

       bool out_local;

       switch(state) {

       case wait_for_one:
              if (up_count_button == 1) {
                     if (number == 9) {
                            next_number = 0;
                     } else {
                            next_number = number+1;
                     }
                     next_state  = wait_for_zero;
              } else if (down_count_button == 1){
                     if (number == 0) {
                            next_number = 9;
                     } else {
                            next_number = number-1;
                     }
                     next_state  = wait_for_zero;
              } else {
                     next_number = number;
                     next_state  = wait_for_one;
              }
              break;
       case wait_for_zero:
              if (up_count_button == 1 || down_count_button == 1) {
                     next_number = number;
                     next_state  = wait_for_zero;
              } else {
                     next_number = number;
                     next_state  = wait_for_one;
              }
              break;
```

```
        default:
                break;
        }


        number = next_number;
        state  = next_state;

        seven_segments_data = get_seven_segment_code(number);
        seven_segments_enable = 0b1110;

}
```

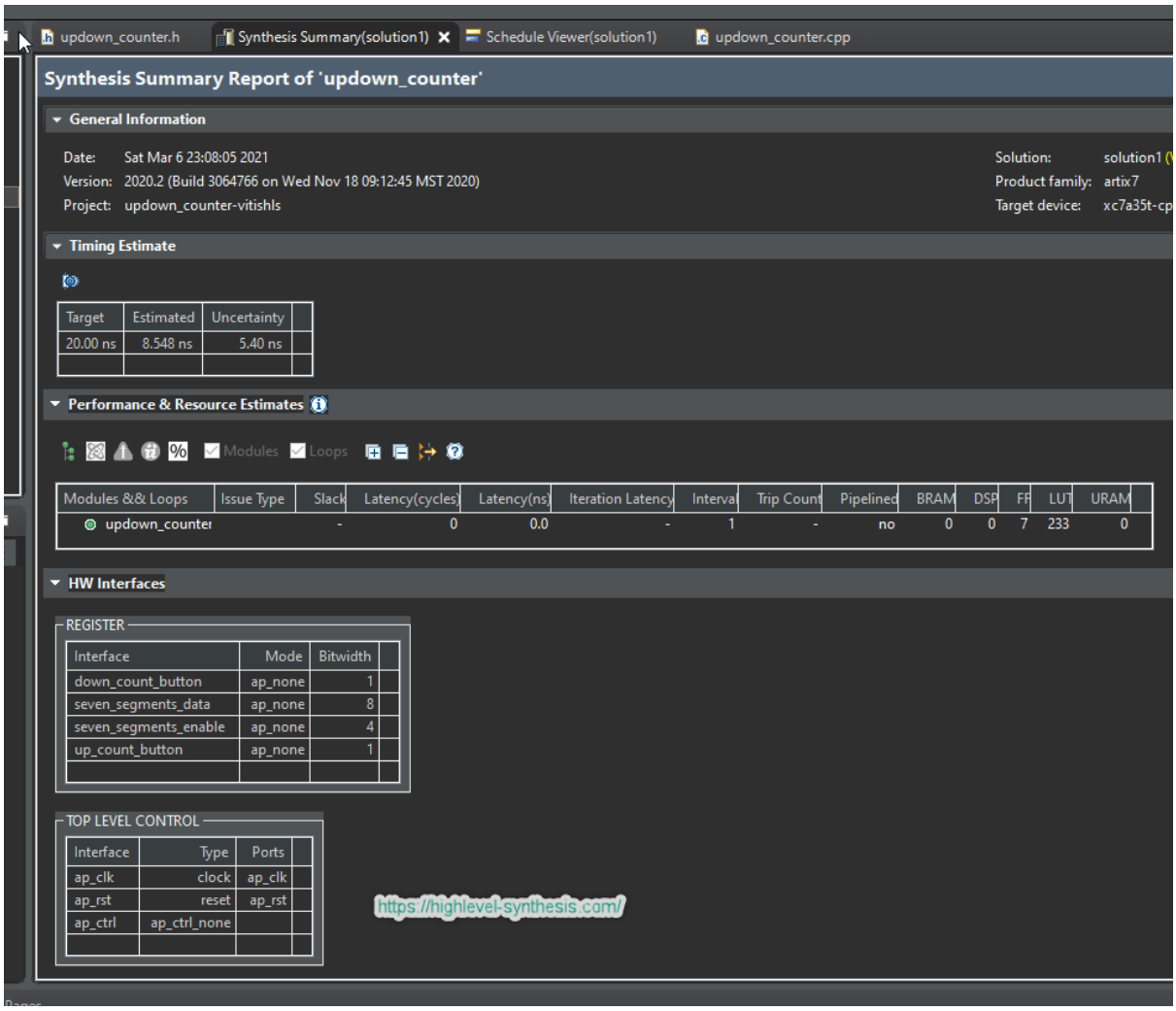
However, if you synthesis this code, the resulted hardware requires two clock cycles to finish, which is against our single-cycle design approach.

Change the design clock period from 10 to 20. This changes the design clock frequency from 100MHz to 50Mhz.

For doing this, right-click on the “Solution1” folder in the Explorer view, then select “Solution Settings…” go to Synthesis Settings and change the clock period.



Now synthesis the code. The following figure shows the synthesis summary report.



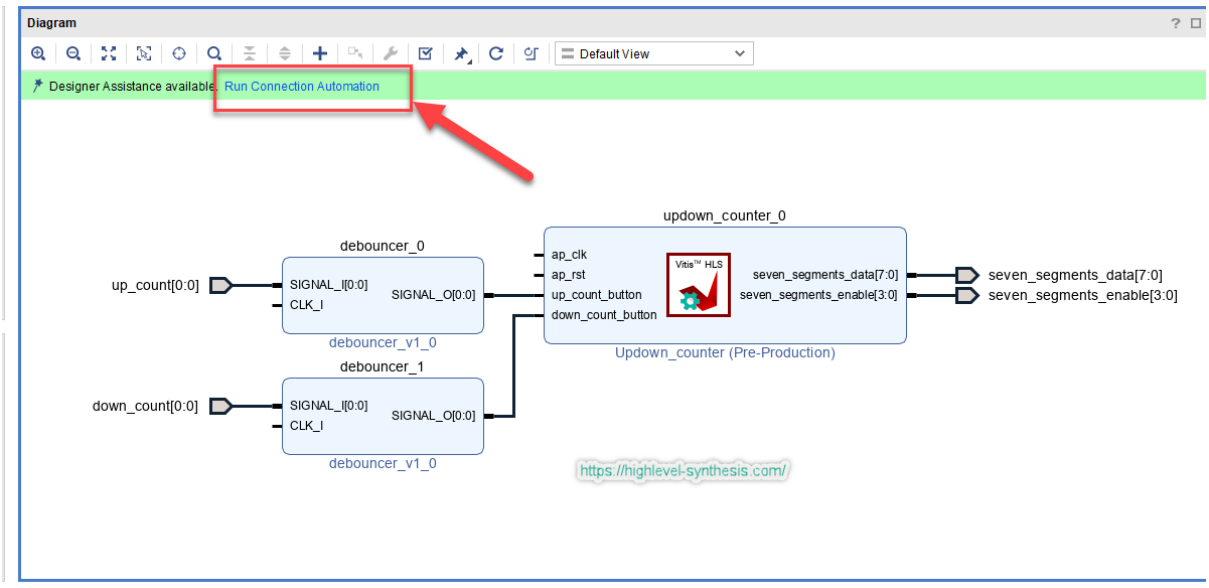
Now export the RTL IP ready for logic synthesis and bitstream generation.

Create a new Vivado project with the name of “updoune_counter-vivado” and select the Basys3 board as the target FPGA.

Then create a new block design. And add two IPs to the Vivado repository: *updown_counter* and *debouncer*. You can find the debouncer IP in the Resources folder attached to this lecture.

Add one updown_counter and two debouncer IPs to the design area.

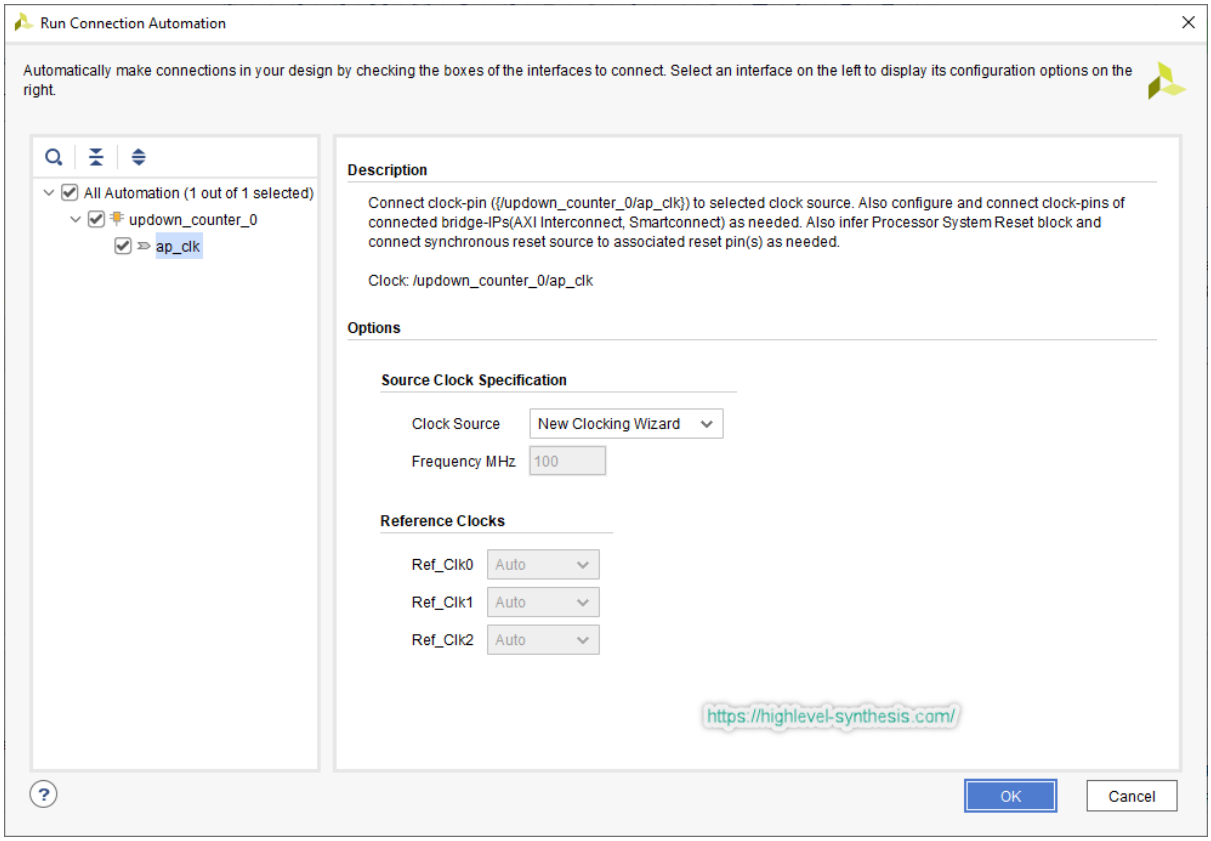
Connect them together as follows



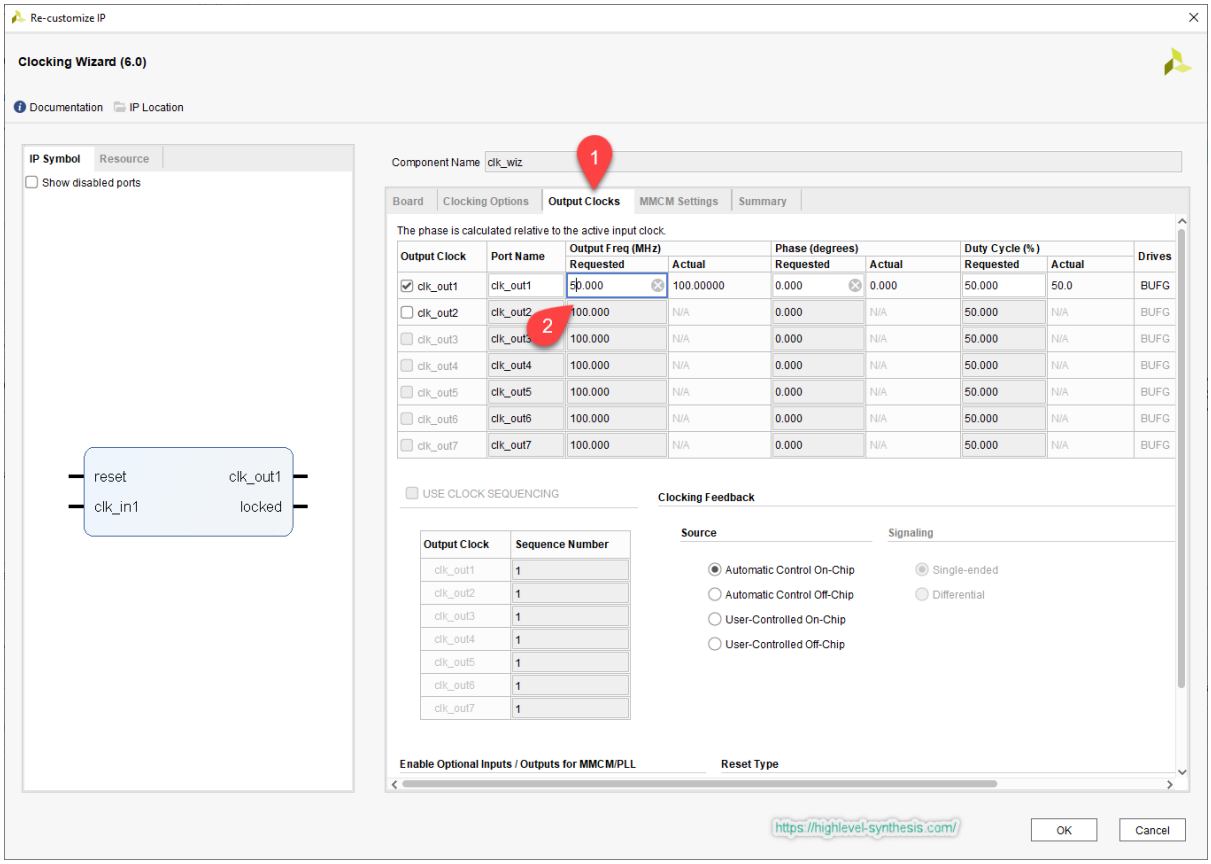
Then we should generate the 50MHz clock frequency for the updown_counter.

For this purpose, click on the “Run Connection Automation” link above the design area in the Diagram window.

Accept the default as the following figure and then press OK.



Double click on the clocking wizard IP to customise that. In the “Re-customize” dialog, go to the “Output clocks” tab and change the clock frequency from 100 to 50, as shown below. Then press OK.



Make the following ports external

clk_wiz→reset

clk_wiz→clk_in1

debouncer_0→SIGNAL_I

debouncer_1→SIGNAL_I

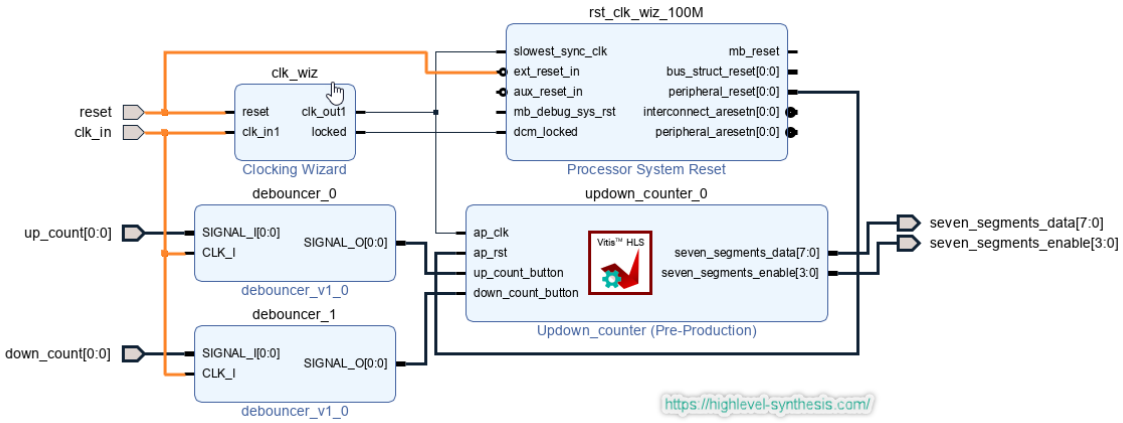
updown_counter→seven_segments_data

updown_counter→seven_segments_enable

Modify the exeternal ports as shown below.

Then connect debouncer_0→CLK_I and debouncer_1→CLK_I to clk_in.

Also connect rst_clk_wiz_100M→ext_rest_in to reset.



Download the constraint file from the Resources folder attached to this lecture and add that to the project.

Now generate the bitstream and program the board and play with the counter.