

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;
}

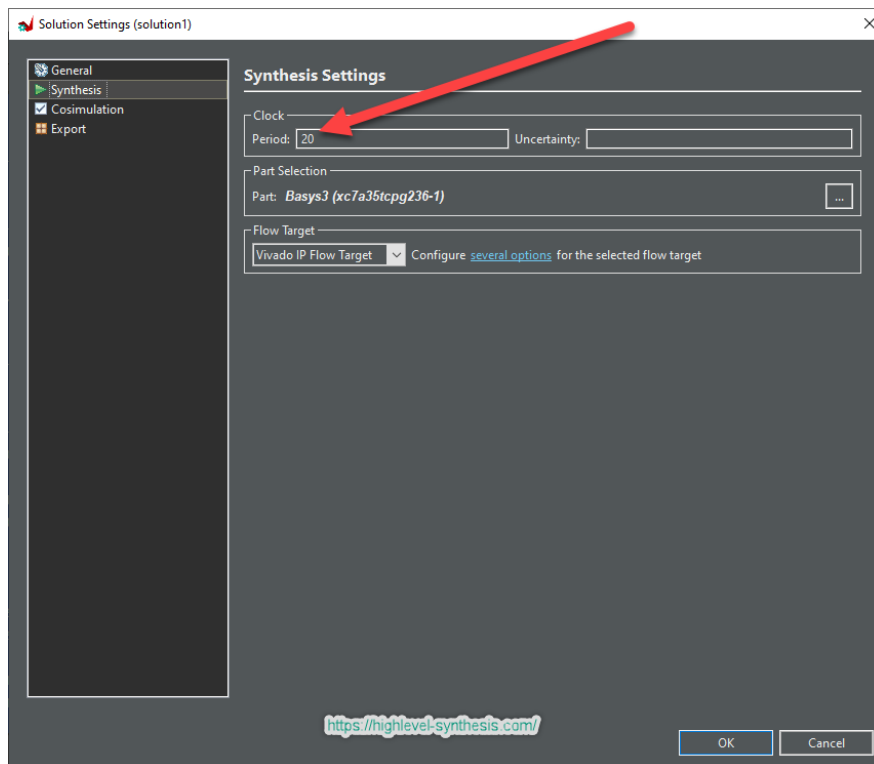
number = next_number;
state = next_state;

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.



3 Digital System Design with High-Level Synthesis for FPGA

Sequential Circuits

Utilities: counter: Quiz Solution

www.highlevel-synthesis.com

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

Synthesis Summary Report of 'updown_counter'

General Information

Date: Sat Mar 6 23:08:05 2021
Version: 2020.2 (Build 3064766 on Wed Nov 18 09:12:45 MST 2020)
Project: updown_counter-vitishls

Solution: solution1
Product family: artix7
Target device: xc7a35t-cp

Timing Estimate

Target	Estimated	Uncertainty
20.00 ns	8.548 ns	5.40 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
updown_counter		-	0	0.0	-	1	-	no	0	0	7	233	0

HW Interfaces

REGISTER

Interface	Mode	Bitwidth
down_count_button	ap_none	1
seven_segments_data	ap_none	8
seven_segments_enable	ap_none	4
up_count_button	ap_none	1

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst	reset	ap_rst
ap_ctrl	ap_ctrl_none	

<https://highlevel-synthesis.com/>

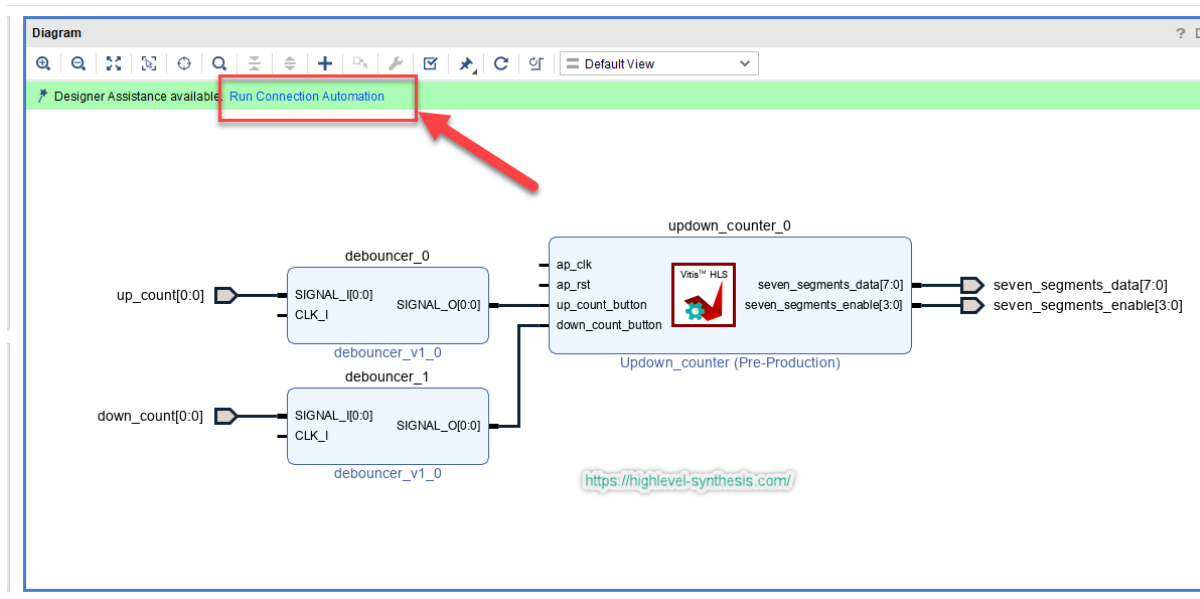
Now export the RTL IP ready for logic synthesis and bitstream generation.

Create a new Vivado project with the name of “updown_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.

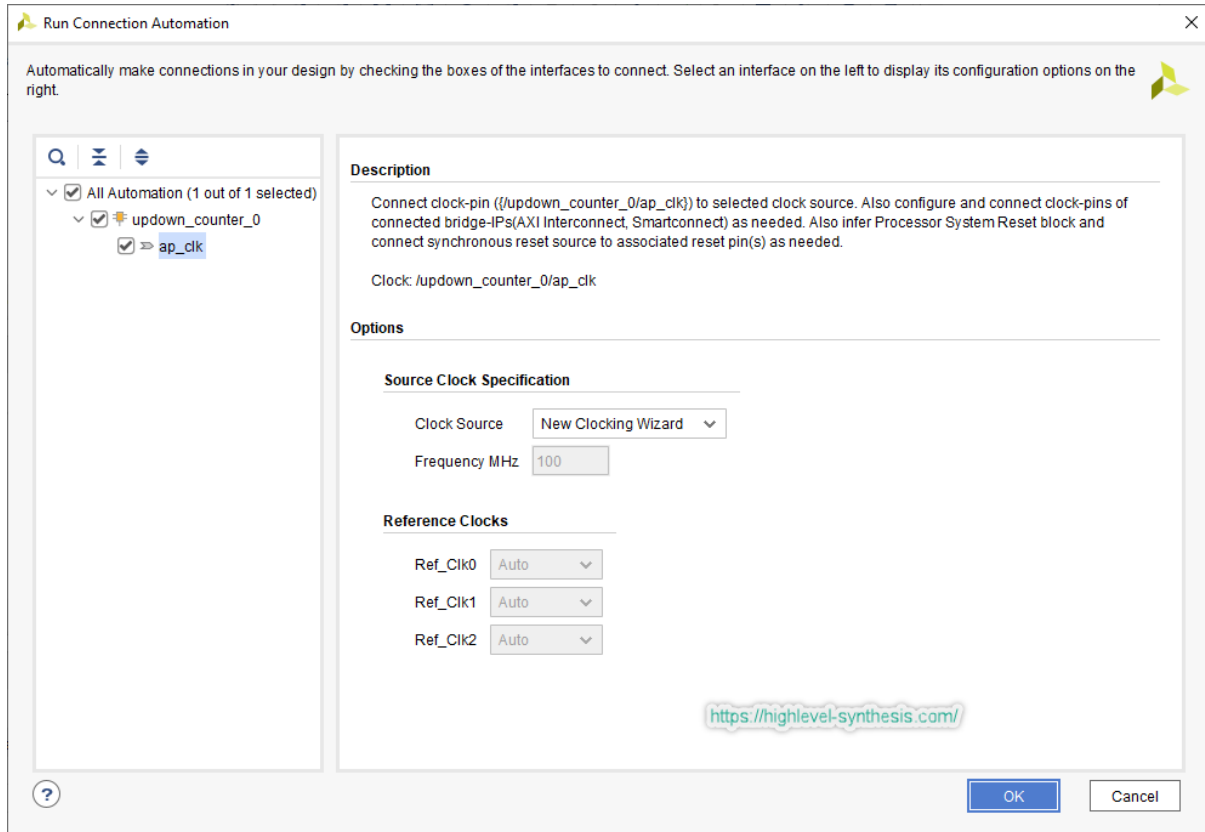
Connect them together as follows



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.

Component Name: clk_wiz

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
<input checked="" type="checkbox"/> clk_out1	clk_out1	50.000	100.000000	0.000	0.000	50.000	50.0	BUFG
<input type="checkbox"/> clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

USE CLOCK SEQUENCING

Reset Type

Enable Optional Inputs / Outputs for MMCM/PLL

OK Cancel

Make the following ports external

clk_wiz → reset

clk_wiz → clk_in1

debouncer_0 → SIGNAL_I

debouncer_1 → SIGNAL_I

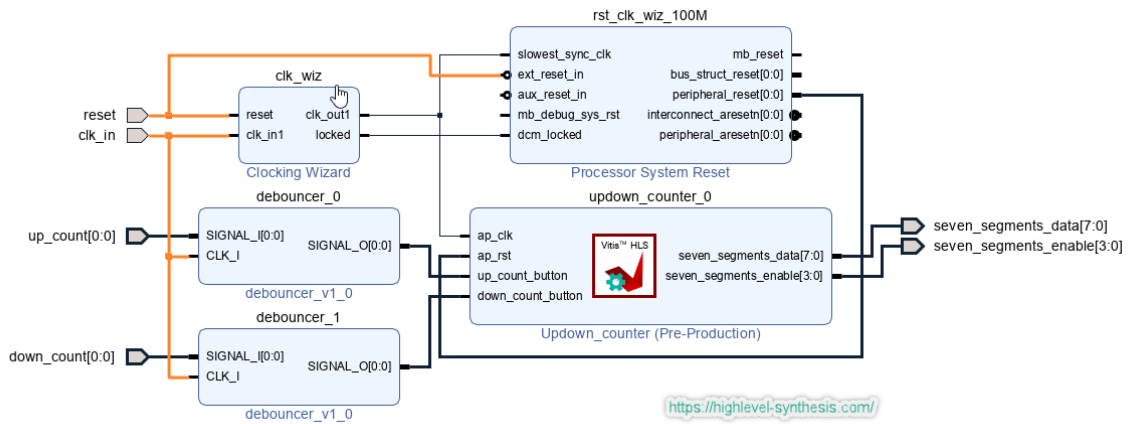
updown_counter → seven_segments_data

updown_counter → seven_segments_enable

Modify the external 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.