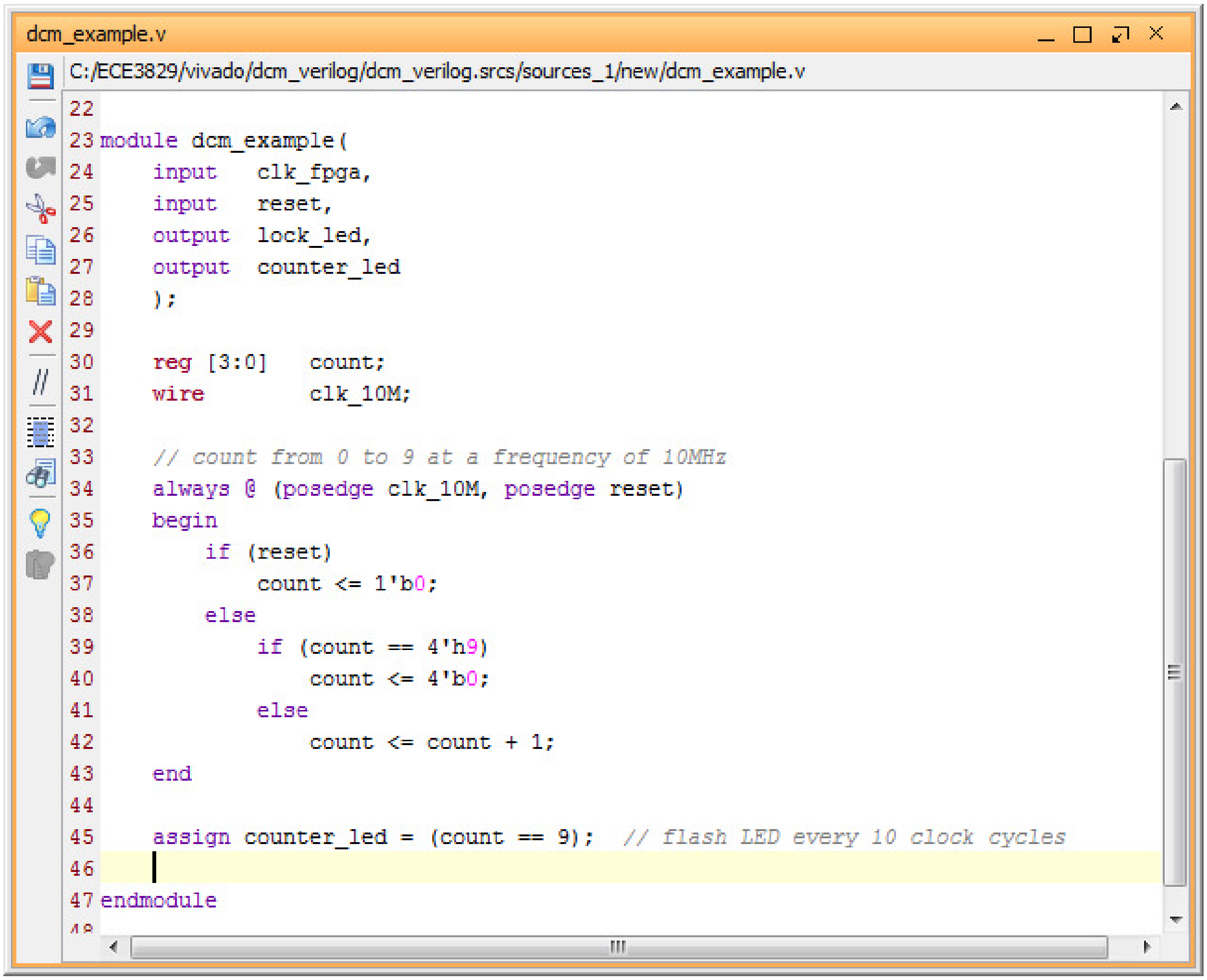
This tutorial shows how to create a simple project with a MMCM (Mixed-Mode Clock Manager) using Xilinx Vivado Design Suite. (Verilog Example)

In this example we instantiate an MMCM to generate a 10MHz clock from the 100MHz oscillator connected to the FPGA.

Create a new project and verify the Tools => Project Settings => General => Target Language is set to Verilog.

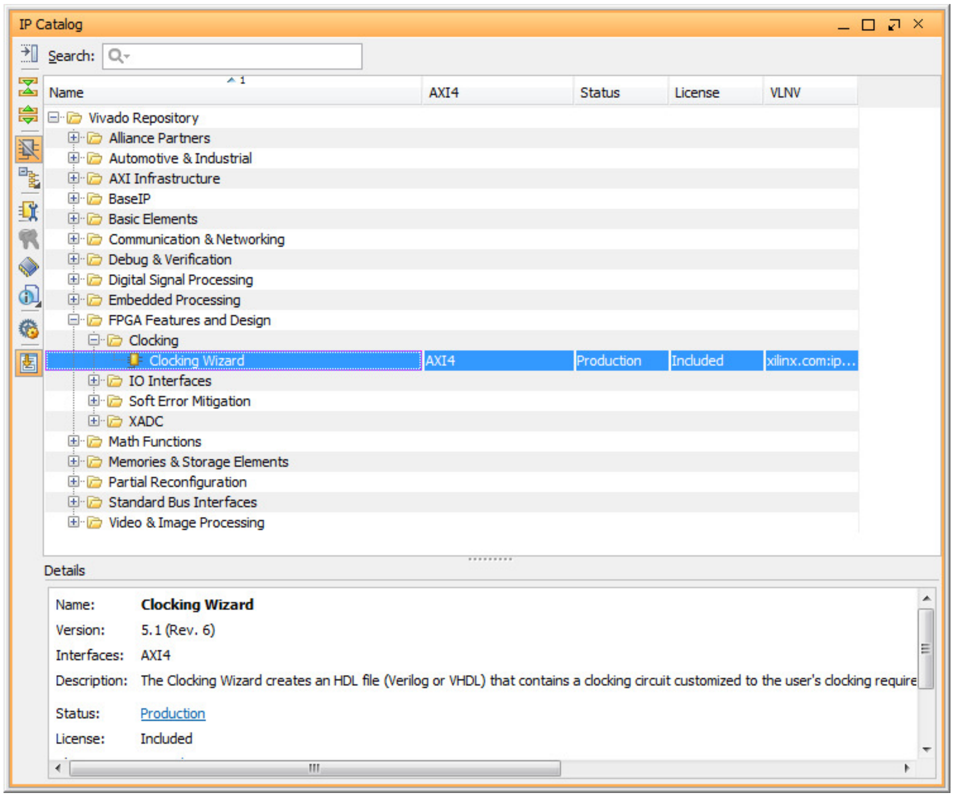
Create a simple module with the following ports and counter logic:



dcm_example.v
C:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/new/dcm_example.v

```verilog

module dcm_example(
    input   clk_fpga,
    input   reset,
    output  lock_led,
    output  counter_led
    );

    reg [3:0]   count;
    wire        clk_10M;

    // count from 0 to 9 at a frequency of 10MHz
    always @ (posedge clk_10M, posedge reset)
    begin
        if (reset)
            count <= 1'b0;
        else
            if (count == 4'h9)
                count <= 4'b0;
            else
                count <= count + 1;
    end

    assign counter_led = (count == 9);  // flash LED every 10 clock cycles

endmodule
```

Select the IP Catalog in the Project Manager and select the clocking wizard:

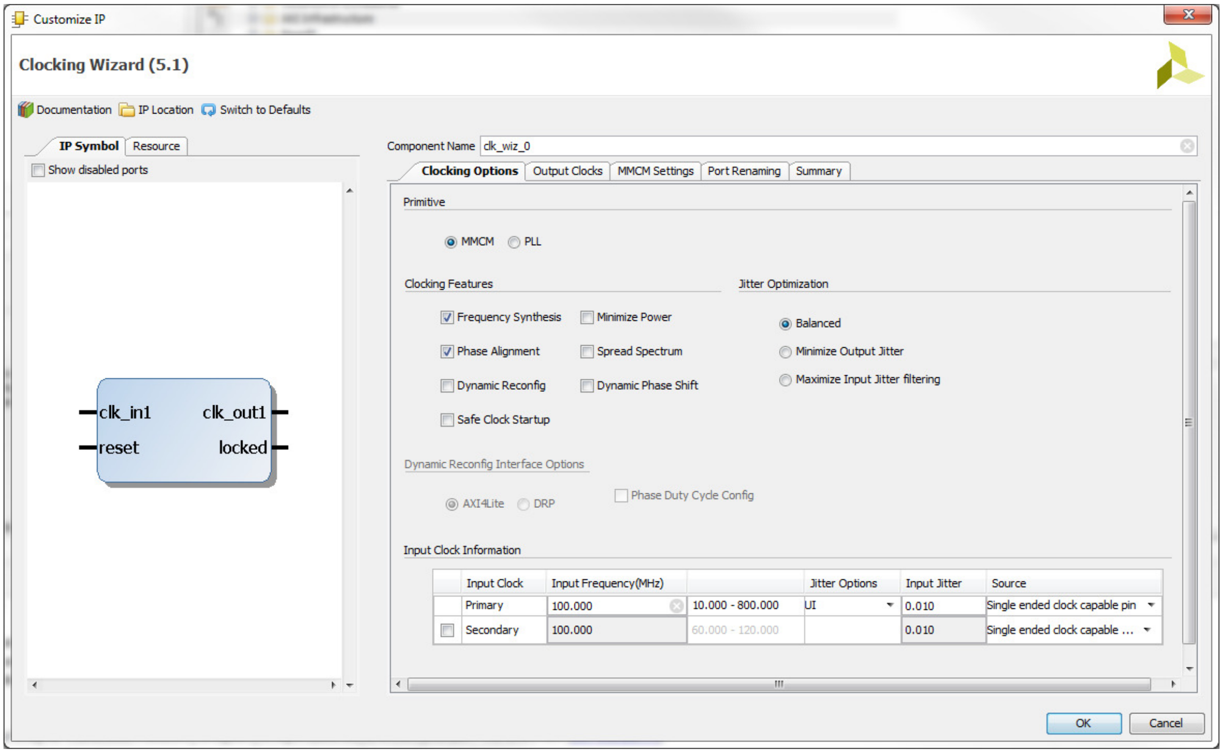


Name	AXI4	Status	License	VLNV
Vivado Repository				
Alliance Partners				
Automotive & Industrial				
AXI Infrastructure				
BaseIP				
Basic Elements				
Communication & Networking				
Debug & Verification				
Digital Signal Processing				
Embedded Processing				
FPGA Features and Design				
Clocking				
Clocking Wizard	AXI4	Production	Included	xilinx.com:ip…
IO Interfaces				
Soft Error Mitigation				
XADC				
Math Functions				
Memories & Storage Elements				
Partial Reconfiguration				
Standard Bus Interfaces				
Video & Image Processing				

Details

Name: **Clocking Wizard**
Version: 5.1 (Rev. 6)
Interfaces: AXI4
Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) that contains a clocking circuit customized to the user's clocking require
Status: Production
License: Included

The Clocking Wizard - Customize IP window opens:



Component Name: clk_wiz_0

Primitive

● MMCM ○ PLL

Clocking Features

☑ Frequency Synthesis ☐ Minimize Power
☑ Phase Alignment ☐ Spread Spectrum
☐ Dynamic Reconfig ☐ Dynamic Phase Shift
☐ Safe Clock Startup

Jitter Optimization

● Balanced
○ Minimize Output Jitter
○ Maximize Input Jitter filtering

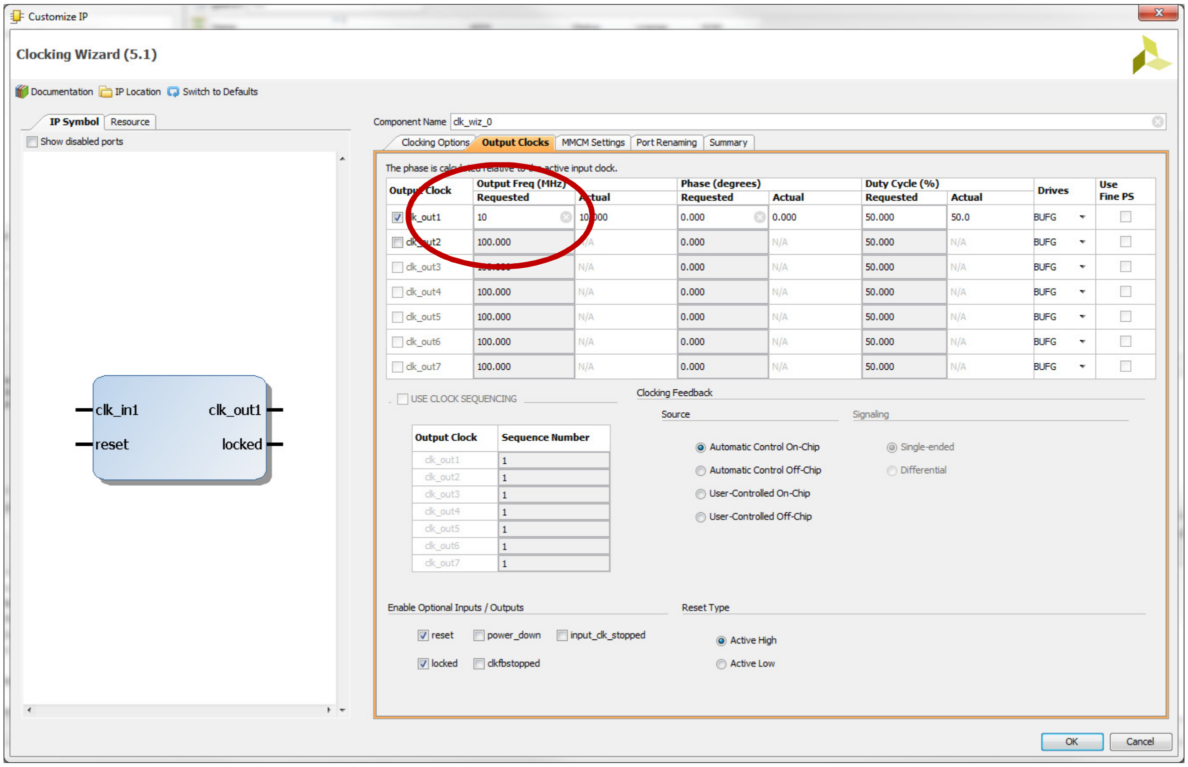
Dynamic Reconfig Interface Options

● AXI4Lite ○ DRP ☐ Phase Duty Cycle Config

Input Clock Information

	Input Clock	Input Frequency(MHz)		Jitter Options	Input Jitter	Source
	Primary	100.000	10.000 - 800.000	UI	0.010	Single ended clock capable pin
☐	Secondary	100.000	60.000 - 120.000		0.010	Single ended clock capable …

In the Output Clocks tab select 10MHz for the clk_out1 frequency:



Component Name clk_wiz_0

The phase is calculated relative to the active input clock.

Output Clock	Output Freq (MHz) Requested	Actual	Phase (degrees) Requested	Actual	Duty Cycle (%) Requested	Actual	Drives	Use Fine PS
☑ clk_out1	10	10.000	0.000	0.000	50.000	50.0	BUFG	☐
☐ clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐
☐ clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐
☐ clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐
☐ clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐
☐ clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐
☐ clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	☐

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

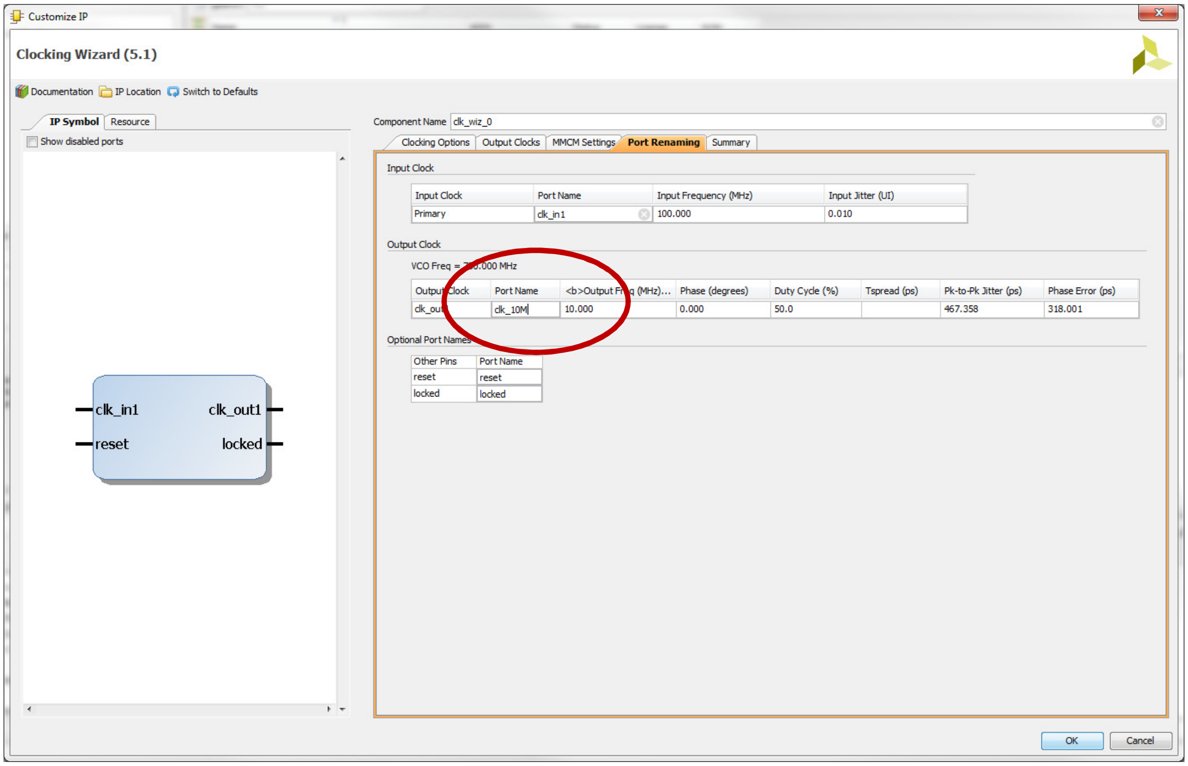
Source
- ● Automatic Control On-Chip
- ○ Automatic Control Off-Chip
- ○ User-Controlled On-Chip
- ○ User-Controlled Off-Chip

Signaling
- ● Single-ended
- ○ Differential

Enable Optional Inputs / Outputs
- ☑ reset ☐ power_down ☐ input_clk_stopped
- ☑ locked ☐ clkfbstopped

Reset Type
- ● Active High
- ○ Active Low

In the Port Renaming tab, change the port name to clk_10M:



Component Name clk_wiz_0

Input Clock

Input Clock	Port Name	Input Frequency (MHz)	Input Jitter (UI)
Primary	clk_in1	100.000	0.010

Output Clock

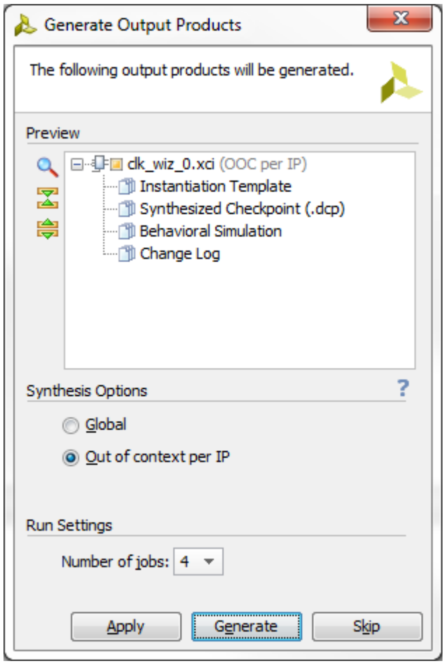
VCO Freq = [illegible].000 MHz

Output Clock	Port Name	<b>Output Freq (MHz)...	Phase (degrees)	Duty Cycle (%)	Tspread (ps)	Pk-to-Pk Jitter (ps)	Phase Error (ps)
clk_out1	clk_10M	10.000	0.000	50.0		467.358	318.001

Optional Port Names

Other Pins	Port Name
reset	reset
locked	locked

Click **OK**



The following output products will be generated.

- clk_wiz_0.xci (OOC per IP)
  - Instantiation Template
  - Synthesized Checkpoint (.dcp)
  - Behavioral Simulation
  - Change Log

Synthesis Options

○ Global

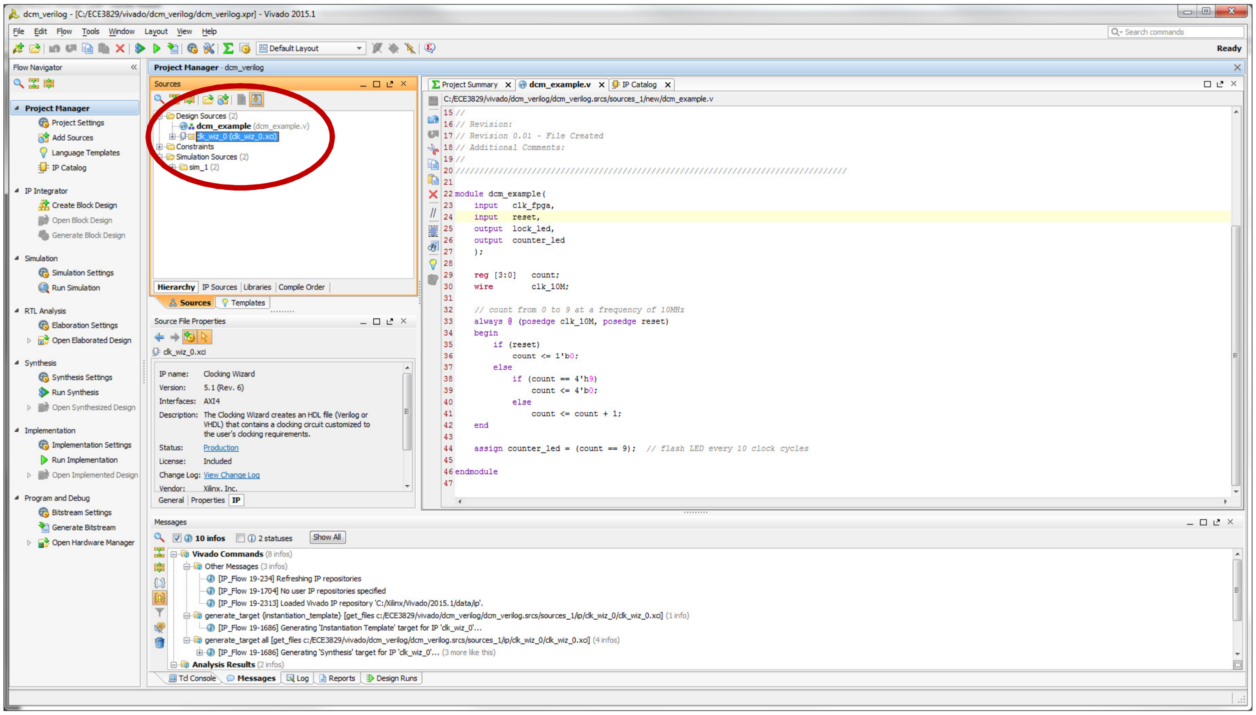
● Out of context per IP

Run Settings

Number of jobs: 4

Click on **Generate.**

The MMCM is now added to the available design sources:



- Design Sources (2)
  - dcm_example (dcm_example.v)
  - clk_wiz_0 (clk_wiz_0.xci)
- Constraints
- Simulation Sources (2)
  - sim_1 (2)

clk_wiz_0.xci

IP name: Clocking Wizard

Version: 5.1 (Rev. 6)

Interfaces: AXI4

Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) that contains a clocking circuit customized to the user's clocking requirements.

Status: Production

License: Included

Change Log: View Change Log

Vendor: Xilinx, Inc.

C:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/new/dcm_example.v

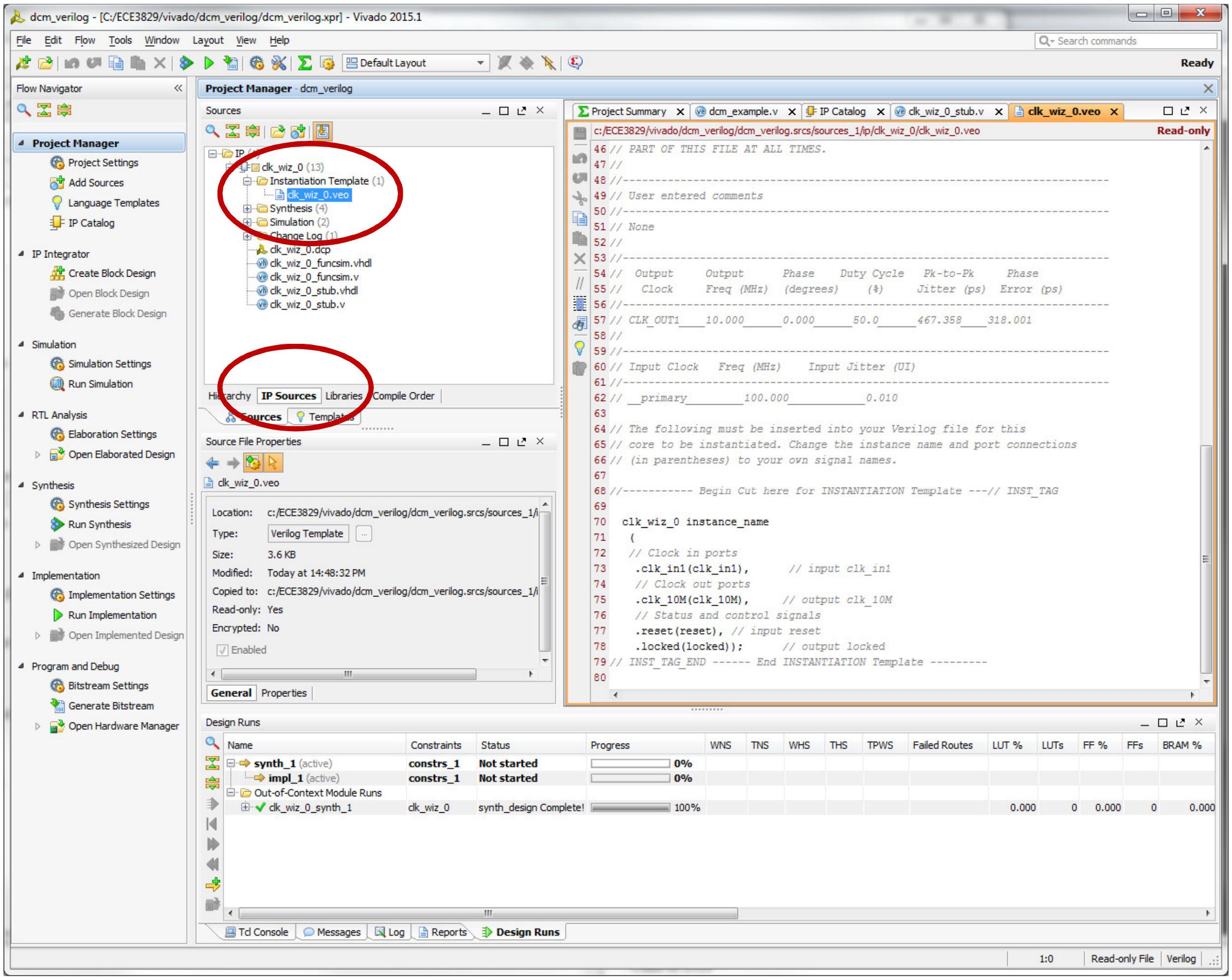
```verilog
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module dcm_example(
    input   clk_fpga,
    input   reset,
    output  lock_led,
    output  counter_led
    );

    reg [3:0]   count;
    wire        clk_10M;

    // count from 0 to 9 at a frequency of 10MHz
    always @ (posedge clk_10M, posedge reset)
    begin
        if (reset)
            count <= 1'b0;
        else
            if (count == 4'h9)
                count <= 4'b0;
            else
                count <= count + 1;
    end

    assign counter_led = (count == 9);  // flash LED every 10 clock cycles

endmodule
```

☑ 10 infos ☐ 2 statuses

```
Vivado Commands (8 infos)
  Other Messages (3 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2015.1/data/ip'.
  generate_target {instantiation_template} [get_files c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/ip/clk_wiz_0/clk_wiz_0.xci] (1 info)
    [IP_Flow 19-1686] Generating 'Instantiation Template' target for IP 'clk_wiz_0'...
  generate_target all [get_files c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/ip/clk_wiz_0/clk_wiz_0.xci] (4 infos)
    [IP_Flow 19-1686] Generating 'Synthesis' target for IP 'clk_wiz_0'... (3 more like this)
Analysis Results (2 infos)
```

We now need to add the MMCM to our top level design.

In the Sources window, Select IP Sources and then expand the Instantiation Template to see the clk_wix_veo file:



c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/ip/clk_wiz_0/clk_wiz_0.veo

```
// PART OF THIS FILE AT ALL TIMES.
//
//----------------------------------------------------------------------------
// User entered comments
//----------------------------------------------------------------------------
// None
//
//----------------------------------------------------------------------------
//  Output     Output      Phase    Duty Cycle   Pk-to-Pk     Phase
//   Clock     Freq (MHz)  (degrees)    (%)     Jitter (ps)  Error (ps)
//----------------------------------------------------------------------------
// CLK_OUT1____10.000______0.000______50.0______467.358____318.001
//
//----------------------------------------------------------------------------
// Input Clock   Freq (MHz)    Input Jitter (UI)
//----------------------------------------------------------------------------
// __primary_________100.000____________0.010

// The following must be inserted into your Verilog file for this
// core to be instantiated. Change the instance name and port connections
// (in parentheses) to your own signal names.

//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG

  clk_wiz_0 instance_name
   (
   // Clock in ports
    .clk_in1(clk_in1),      // input clk_in1
    // Clock out ports
    .clk_10M(clk_10M),     // output clk_10M
    // Status and control signals
    .reset(reset), // input reset
    .locked(locked));      // output locked
// INST_TAG_END ------ End INSTANTIATION Template ---------
```

Source File Properties

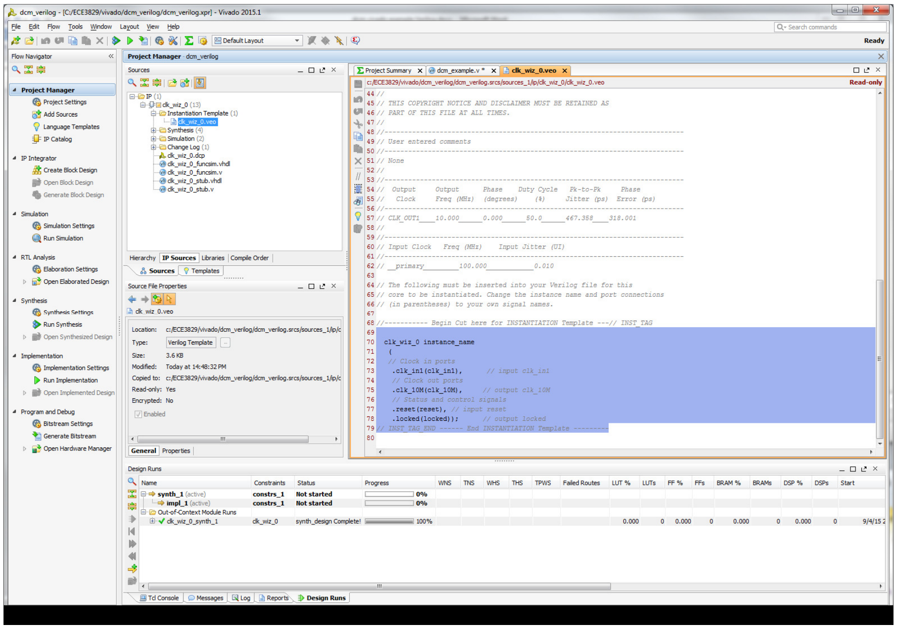
clk_wiz_0.veo

Location: c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/i
Type: Verilog Template
Size: 3.6 KB
Modified: Today at 14:48:32 PM
Copied to: c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/i
Read-only: Yes
Encrypted: No
☑ Enabled

Design Runs

Name	Constraints	Status	Progress	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT %	LUTs	FF %	FFs	BRAM %
synth_1 (active)	**constrs_1**	**Not started**	**0%**											
impl_1 (active)	**constrs_1**	**Not started**	**0%**											
Out-of-Context Module Runs														
✓ clk_wiz_0_synth_1	clk_wiz_0	synth_design Complete!	100%							0.000	0	0.000	0	0.000

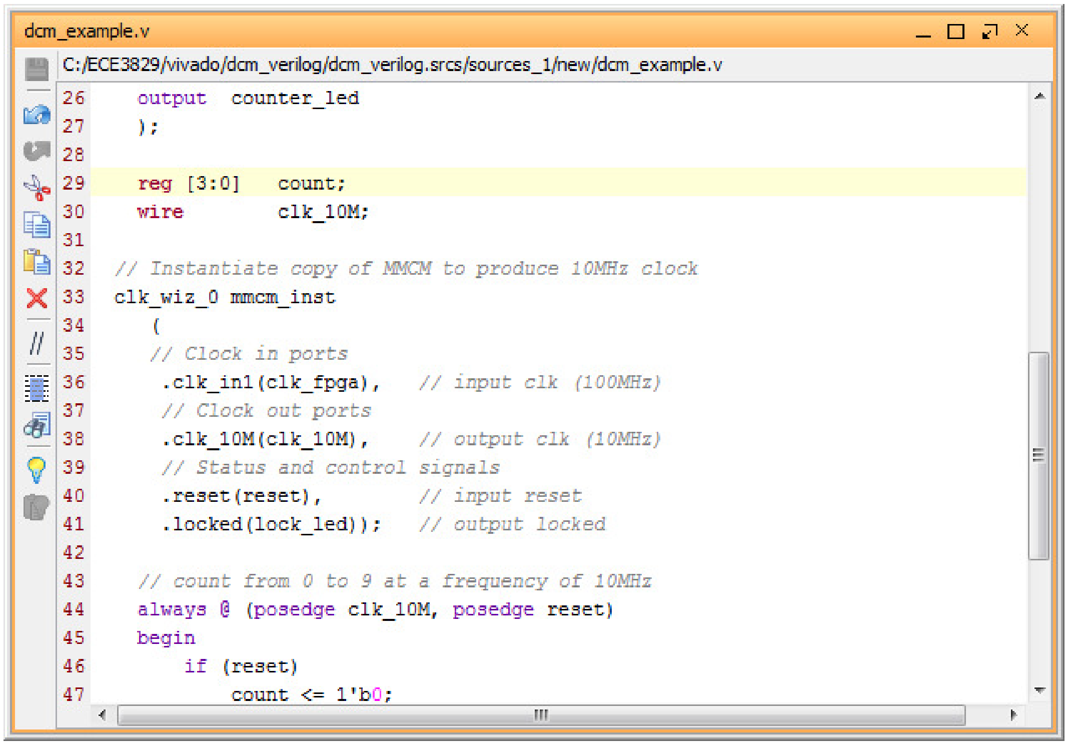
Select the lines at the bottom of the file for the instantiation template (select the lines and use ctrl C to copy):



c:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/ip/clk_wiz_0/clk_wiz_0.veo

```
//
// THIS COPYRIGHT NOTICE AND DISCLAIMER MUST BE RETAINED AS
// PART OF THIS FILE AT ALL TIMES.
//
//----------------------------------------------------------------------------
// User entered comments
//----------------------------------------------------------------------------
// None
//
//----------------------------------------------------------------------------
//  Output     Output      Phase    Duty Cycle   Pk-to-Pk     Phase
//   Clock     Freq (MHz)  (degrees)    (%)     Jitter (ps)  Error (ps)
//----------------------------------------------------------------------------
// CLK_OUT1____10.000______0.000______50.0______467.358____318.001
//
//----------------------------------------------------------------------------
// Input Clock   Freq (MHz)    Input Jitter (UI)
//----------------------------------------------------------------------------
// __primary_________100.000____________0.010

// The following must be inserted into your Verilog file for this
// core to be instantiated. Change the instance name and port connections
// (in parentheses) to your own signal names.

//----------- Begin Cut here for INSTANTIATION Template ---// INST_TAG

  clk_wiz_0 instance_name
   (
   // Clock in ports
    .clk_in1(clk_in1),      // input clk_in1
    // Clock out ports
    .clk_10M(clk_10M),     // output clk_10M
    // Status and control signals
    .reset(reset), // input reset
    .locked(locked));      // output locked
// INST_TAG_END ------ End INSTANTIATION Template ---------
```

Back in your top level file, paste the instantiation template and modify the signal names to match your connections:



C:/ECE3829/vivado/dcm_verilog/dcm_verilog.srcs/sources_1/new/dcm_example.v

```
    output   counter_led
    );

    reg [3:0]     count;
    wire          clk_10M;

  // Instantiate copy of MMCM to produce 10MHz clock
  clk_wiz_0 mmcm_inst
     (
     // Clock in ports
      .clk_in1(clk_fpga),     // input clk (100MHz)
      // Clock out ports
      .clk_10M(clk_10M),      // output clk (10MHz)
      // Status and control signals
      .reset(reset),          // input reset
      .locked(lock_led));     // output locked

    // count from 0 to 9 at a frequency of 10MHz
    always @ (posedge clk_10M, posedge reset)
    begin
        if (reset)
            count <= 1'b0;
```

You can now synthesize and implement this design.