Lab 18: Generating clocks on FPGAs

In this lab, we'll see how to generate derived clocks properly on a AMD/Xilinx FPGAs. We use the same design as in Lab 17.

In the previous lab, the derived clock (1/5 speed of the system) was generated using a self-coded clock divider module, which was sensitive to the rising edge of the input system clock. The problem with this approach is that we could not be change the clock at half cycle, resulting in an output clock with a duty cycle of 40%.

To generate a proper half-duty clock, the best way is to use the Mixed Mode Clock Manager (MMCM) module on the FPGA.

Exercise: Using the Vivado Clocking Wizard

Go to ~/labs/lab18/ and open the lab18.xpr Vivado project.

```
vivado lab18.xpr &
```

This is the same project as lab17. Have a look at the clock_divider.vhd file, to see how we created the derived clock. Run the Simulation, to see the clock behaviour. You might have to zoom in.

If you look at the divided_clk signal, you can immediately see the discrepancy in the time the signal is high and low.

GPIO_Demo.vhd × gpio_tb.vh	d × tb_GPIO_demo_beha	v.wcfg ×						? 🗆 🖸	
Q 🔛 🔍 Q 🔀 📈	+ [± ±r	+ Г ⇒						¢	
		<mark>75</mark>	.000 ns					^	
Name	Value		80.000 ns	100.000 ns	120.000 ns	140.000 ns	160.000 ns	180.000	
> 😼 SW_tb[15:0]	0000				0000				
16 CLK_tb	1								
> 😼 BTC_tb[0:0]	0				G				
> 😼 LED_tb[15:0]	0000		0000						
U CLK_PERIOD	10000 ps		10000 ps						
le divided_clk	1								

Let's now generate our clock properly. Open the IP Catalog and search for *Clocking Wizard*. Double click on it, to open the configuration window.

coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Coding Ward (4) Codin				Customize IP				- 0
P Symbol Resource Composert Nam @ dut 0.0 Composert Nam @ dut 0.0 Control Clock Sections P Symbol Resource Control Clock Sections P Symbol Resource Control Clock Sections Control Clock Sections Control Clock Sections Control Clock Sections P Symbol Resource Control Clock Sections Provide Clock Monitoring Provide Clock Monitoring P Symbol Resource Clock Sections Provide Clock Monitoring Provide Clock Monitoring P Symbol Resource Clock Sections Provide Clock Monitoring Provide Clock Monitoring P Symbol Resource Clock Sections Provide Clock Monitoring Provide Clock Monitoring P Societ Algorithmic Resource Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring P Societ Algorithmic Resource B Balanced Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring Provide Clock Monitoring	Clocking Wizard (6.0)							4
	🚯 Documentation 📄 IP Location C Switch to Defa	ults						
<pre>view of the control of the cont</pre>	IP Symbol Resource		Component Name clk_wiz_0					6
<pre>ct.kcline + s.axl,lite + cl.k(In1_D + cl.k(In2_D + cl.k(In2_D) + cl.k(In1_D) + cl</pre>	Show disabled ports		Clocking Options Output Clocks	ort Renaming MMCM Settings Summar	у			
<pre> Printite Printi</pre>			Clock Monitor					
<pre></pre>			Enable Clock Monitoring					
<pre></pre>			Primitive					
<pre> CLKFB_NLpD CLKFB_OUT_D + CLKFB</pre>			● MMCM ○ PLL					
<pre></pre>			Clocking Features	Jitter Optimization				
 s.add,adk s.add,aresetn s.add,aresetn dk_stop[12h]30 reset reset reset reset ref_dk dk_cort[3:0] ref_dk user_dk3 user_dk3 ref_dk re				ize Power Balanced				
 s.ad, areaseth reseth reseth reseth ref_dk user_dk2 user_dk3 user_dk3 user_dk3 ref_dk <	s_axi_aclk		Phase Alignment 🗌 Sprea	d Spectrum Okinimize Output Ji	tter			
Freet Interrupt	-O s_axi_aresetn		Dynamic Reconfig Dynamic	nic Phase Shift O Maximize Input Jit	ter filtering			
 Presen ref_dk user_dk2 user_dk3 user_dk4 user_dk4<td>- reset</td><td></td><td>Safe Clock Startup</td><td></td><td></td><td></td><td></td><td></td>	- reset		Safe Clock Startup					
 ref_clk user_clk2 user_clk2 user_clk3 user_clk3 user_clk4 user_clk4<	-O resetn							
 user_clk0 user_clk1 user_clk2 user_clk3 user_clk3 user_clk4 user_clk4	- ref_clk		Options	Phase Duty Cycle Config 🛛 Write DRP regi	sters			
user_clicit Input Cock Information Input Cock Information Input Cock Fort Name Input Cock Fort Name Input Cock Input Cock Source Input Cock Source Input Cock Source Input Cock Source Input Cock	user_clk0	clk_out1 —	AXI4Lite DRP					
user_ciki2 mmay dk_m1 10000 10000-800000 U1 0010 Srede ended clock capable prin • user_ciki3 secondary dk_m2 100000 0.000 0.010 Srede ended clock capable prin •	user clk1	locked -	Input Clock Information					
user_clk3	usor clk2							
						UI		
			Jecondary	Chight 100,000	0.000 120.000		0.010	authe curea cieci colisme hui
								OK Cancel
								on Calice

By default, it will show all the configurable port, that we don't need for our design. Untick *Show disabled ports`, to have a slimmer module.

In the *Clocking Options* page you can configure the input clock. By default, it is set to 100 MHz, which is what we are using, so we don't need to touch anything here.

Go now to the *Output Clocks* tab. There is by default one output clock enabled clk_out1. Modify the requested output frequency to be 20 MHz, having a duty cycle of 50 %. Untick the locked in the *Enable Options* area at the bottom of the window.

Have a look at the other options in the configuration. Once you are done, click on OK and generate the IP Output Products.

			Re-cus	comize IP								-
ocking Wizard (6.0)												
ocumentation 🛛 IP Location 🔿 Switch to Defaults												
Symbol Resource	Component Name clk	_wiz_0										
Show disabled ports	Clocking Options	Output Clocks	Port Renaming MMCM	A Settings Summary								
	The phase is calculat	ed relative to the	active input clock.									
	Output Clock	Port Name	Output Freq (MHz) Requested	Actual	Phase (degrees) Requested	Actual	Duty Cycle Requested	(%) Actual	Drives		Use Fine PS	Max Freq. of buffer
	Clk_out1	clk_out1		20.00000		0.000	50.000	S0.0	BUFG			464.037
	Clk_out2	clk_out2	100.000		0.000	N/A	50.000		BUFG	~		
	Clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	~		464.037
	Clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	~		464.037
	clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	~		464.037
			100.000		0.000		50.000		BUFG	~		
	Clk_out6	clk_out6	100.000									
	CIK_OUN6	clk_out7	100.000 100.000 Clocking Fe	N/A	0.000	N/A	50.000	N/A	BUFG	Ŧ		464.037
reset dk.out1 -	Clk_out7	clk_out7	100.000 Clocking Fe	N/A				N/A	BUFG	v		464.037
		clk_out7	100.000 Clocking Fe umber Source	N/A	0.000 Signaling			N/A	BUFG	·		464.037
clk_out1	Cutyut Clock	clk_out7 QUENCING Sequence N 1 1	100.000 Clocking Fo Source	edback	0.000 Signaling	N/A		N/A	BUFG	Ţ		464.037
clk_out1	Chronit Cleck Chronit Cleck Chronit C	QUENCING Sequence N 1 1 1 1	Clocking Fo	redback	0.000 Signaling hip ③ Sing	N/A		N/A	BUFG	v		464.037
clk_out1	Chickout Clock Sec Output Clock Chickout Chicko	CIR_OUT7	Clocking F4	N/A eedback) Automatic Control On-Cl) Automatic Control Off-C	0.000 Signaling hip © Sing hip Diff	N/A		N/A	BUFG	Ŧ		464.037
clk_out1	Chronit Cleck C	QUENCING Sequence N 1 1 1 1	Clocking F4	N/A eedback Automatic Control On-Cli) Automatic Control Off-C) User-Controlled On-Chip	0.000 Signaling hip © Sing hip Diff	N/A		NA	BUFG	*		464.037
clk_out1	USE CLOCK SEC USE CLOCK SEC dk.out dk.out dk.out	Sequence N 1 1 1 1 1 1	Clocking F4	N/A eedback Automatic Control On-Cli) Automatic Control Off-C) User-Controlled On-Chip	0.000 Signaling hip © Sing hip Diff	N/A		N/A	BUFG	•		464.037
clk_out1	dk_out Use cLock see Use cLock see dk_out dk_out dk_out dk_out dk_out dk_out dk_out dk_out	Sequence N 1 1 1 1 1 1 1 1 1	Clocking F4	N/A eedback Automatic Control On-Cli) Automatic Control Off-C) User-Controlled On-Chip	0.000 Signaling hip © Sing hip Diff	N/A		N/A	BUFG	•		464.037
clk_out1	dk_out Use cLock see Use cLock see dk_out dk_out dk_out dk_out dk_out dk_out dk_out dk_out	ck_out7 ck_out7 veencing sequence N 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	100.000 Clecking Fe	N/A eedback Automatic Control On-Cli) Automatic Control Off-C) User-Controlled On-Chip	0.000 Signaling hip © Sing hip Diff	N/A		N/A	BUFG	~		464.037
clk_out1	the coord second s	clk_out7 clk_out7 vencing Sequence N 1 1 1 1 1 1 1 1 1 1 1 1 vencing v	100.000 Clecking Fe	HVA Hedback Automatic Control On-Cl Automatic Control Off-C User-Controlled Off-Chip User-Controlled Off-Chip	5.000 Signaling hip @ Sing hip @ Diff	N/A		N/A	BUFG	•		464.037
clk_out1	the coord second s	ck_out7 VENCING Sequence N 1 1 1 1 1 1 puts / Outputs power_down	TOD.000 Clecking Fr Source Clecking Fr Clecking Fr Cle	edback Automatic Control On-Cl Automatic Control Off-Cl Automatic Control Off-Cl User-Controlled Off-Chip Reset Type	5.000 Signaling hip @ Sing hip @ Diff	N/A		N/A	BUFG	•		464.037

Now we need to replace the clock divider with the IP we just created. Open the GPIO_demo.vhd file from the Source Manager, and comment out the lines corresponding to the clock_divider block. Instantiate now the IP, using the generated template, as we did in Lab 16.

Save the file and run the simulation again.

You might notice that the even if the reset signal is down, it takes a while for the divided clock to be generated. This is because the simulation is accurately modelling the MMCM, and it takes a few microseconds to stabile, before starting to output the clock. This would have been signaled by the lock signal we disabled. Zoom in the wave, to actually verify that the generated clock has a period of 50 ns and is half-duty.

GPIO_Demo.vhd × gpio_tb.vhd	× clk_wiz_0.vho ×	tb_GPIO_demo_behav.v	vcfg* × debounce	r.vhd ×		? 🗆 🖸
Q 🔛 🔍 Q 💥 🗶	+ [↓ ↓ <u>+</u> r	+ Γ Γ⇔ →Γ −Γ				0
						<mark>10.000000 us</mark>
Name	Value	0.000000 us	2.000000 us	4.000000 us	6.000000 us	18.000000 us
> 😽 SW_tb[15:0]	0000			0000		
la CLK_tb	0					
> 😽 BTC_tb[0:0]	1	0		1		
> 😻 LED_tb[15:0]	0000			0000		
U CLK_PERIOD	10000 ps			10000 ps		0000
> 😽 RST[0:0]	0			0		
🔓 divided_clk	0					

Generate the bitstream and load it to the board, and check its functionalities.