

Constraints

Constraints file can be found in:

```
.    (Pondababa project directory)
|
|— Constraints/
|   └─ Pondababa.xdc
|   └─ Pondababa_Impl.xdc
|   └─ Pondababa_Btstrm.xdc
|
```

Below mentioned and described constraint files are or can be used when implementing the FPGA design with the Xilinx Vivado tool.

This is the default text in this ReadMe file:

“The files have constraint lines turned on or commented out. This is because the design has been used and tested for different types of FPGA, and to allow one to easily modify the design to the needs of a new design.”

For this design and in these files lines are also commented out or not but that is because the nature of the design is demonstration and testing. For both activities it is possible that logic is added or removed from the design and therefore constraints are already prepared but commented out while others are used.

Files

Bitstream_Cnstrnts.xdc

File containing constraints that have only to do with bit stream generation. The file is checked by the Vivado synthesis tool (syntax errors, ...) but is only used when the software creates a bitstream for downloading of the FPGA.

\${FILE_NAME}_Impl.xdc

File containing timing and implementation constraints.

The file should contain the design's timing constraints as far as different hierarchical modules do not have their own timing and implementation constraints. If that is the case, make sure Vivado uses the constraint files of the hierarchical modules too.

The Vivado synthesis tool uses the file during synthesis and checks the file for implementation constraint syntax errors which are further only used during implementation (place & route).

Remark:

When there are multiple timing and/or implementation (placement) constraint files, MAKE SURE to check the files for duplicate and overlapping constraints. Duplicate and overlapping constraints are most of the time hurting the design more than they do where they are set for.

`${FILE_NAME}.xdc`

This is the designs constraint file containing pin locking information. This is the constraints file that (probably) needs to be modified when another FPGA part type is used or the design is placed into another IOBank, or ...

VIO related constraints

- `set_property C_CLK_INPUT_FREQ_HZ (125000000, 300000000, or other)`
 - ILA/VIO related constraint.
 - Constraint defining the input frequency of the ILA/debug core in Hz.
 - The debug hub requires a free running clock.
That free running clock can be the cause of many timing related issues, from timing related warnings to a design missing the requested timing due to the addition of the ILA/debug core.
When after implementation of the design the timing has degraded or a lot of warnings are generated about the debug core then this constraint can be used.
- `set_property C_ENABLE_CLK_DIVIDER (TRUE or FALSE)`
 - This constraint must be set to TRUE when the `C_CLK_INPUT_FREQ_HZ` constraint is used. It puts a divider in place to adjust the clock in the ILA/debug core to the requested clock
- `set_property CLOCK_DEDICATED_ROUTE < Clock route option> (SAME_CMT_COLUMN, FALSE, ...)`
 - When a the clock input of a MMCM and/or PLL is not coming from a source (IO) in the same clock area as used by the MMCM and/or PLL, Vivado will error out during implementation of the design.
 - This constraint make sure Vivado does not see the IO/Clock buffer placement and the MMCM/PLL placement as an error but as a warning. When this constraint is used Vivado will thus issue a warning about sub optimal placement of the clock source and the MMCM/PLL it is connected to.
 - This attribute must be placed on a net segment at the highest level of the design hierarchy.
 - By default this constraint is turned on (TRUE), forcing the software to use dedicated and local to the clock area clock routes.
 - This design uses the options `SAME_CMT_COLUMN` or `FALSE`
 - `SAME_CMT_COLUMN`
Clock driver and its load must be placed in the same Clock Management Tile (CMT) column. The clock routing uses dedicated global clock routing resources.
 - `FALSE`
Clock driver and its load can be placed anywhere in the device.
The clock net can be routed using global clock routing resources, standard fabric routing resources or a mix of both. This can adversely affect the timing and performance of the clock net.
Thsi is meanly the only option to make Vivado place and route a design when the source and destination of the clock route are place in total different parts of the FPGA (Clock input and clock buffer at the left side and the MMCM and/or PLL at the right side of the FPGA die.)

Global Timing Constraints

- **create_clock -period 1.600 -name -waveform {0.000 0.800}**

Create a clock object with the specified period or waveform defined in nanoseconds (ns). This command defines primary clocks which are used by the timing engine as the delay propagation starting point of any clock edge. The defined clock can be added to the definition of an existing clock, or overwrite the existing clock.

- **set_clock_groups -asynchronous -group**

Define clocks, or groups of clocks, that are exclusive with or asynchronous to other clocks in the design. Exclusive clocks are not active at the same time, and paths between them can be ignored during timing analysis. Asynchronous clocks are clocks with no known phase relationship, which typically happens when they do not share the same primary clock or do not have a common period.

Using this command is similar to defining false path constraints for data paths moving between exclusive or asynchronous clock domains.

- **set_max_delay**

Sets the maximum delay allowed on a timing path, specified in nanoseconds (ns).

The specified delay value is assigned to both the rising and falling edges of the defined timing paths unless the -rise or -fall arguments are specified.

Here it is used to minimise the delay between certain asynchronously clocked FFs.

- **set_false_path**

Set this constraint on nets that are ignored during timing analysis.

- Next are a set of constraints that must be used together in order to force the software to place logic, primitives and other in a user defined area.

- **create_pblock <pblock_name>**

Defines a Pblock to allow one to add logic instances for floorplanning purposes.

A Pblock consists of a group of cells that can be assigned to one or more independent or overlapping rectangles. <pblock_name> is a unique user defined name.

- **add_cells_to_pblock [get_pblocks <pblock_name>] [get_cells**

Adds specified logic instances to a pblock in an open implemented design. Once cells have been add to a pblock, you can place the pblocks onto the fabric of the FPGA using the resize_pblock command.

The <pblock_name> needed by this constant is the same name as given to the create_pblock constraint.

The are hierarchical parts of the design.

- **resize_pblock [get_pblocks <pblock_name>] -add {<left_bottom>:<top_right>}**

Place, resize, move, or remove the specified Pblock.

The <pblock_name> needed by this constant is the same name as given to the create_pblock constraint.

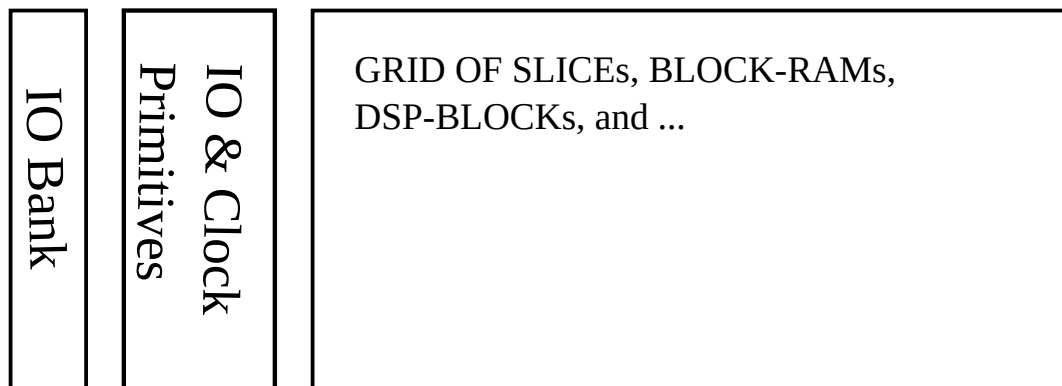
This <left_bottom>:<top_right> specifies the location in the FPGA logic area.

Example: *SLICE_X66Y180:SLICE_X77Y239*

- For convenience and to make the software not place logic all over the place in the FPGA, the different parts of the design are placed and routed in user restricted areas of an Io-Bank. Pblocks are in this design not only used to keep the design elements nicely together, but also used to make the design meet timing.

- An IO-Bank fits to a clock area (References for 7-Series FPGA).

Find just behind the IO-Bank a column of IO and clock primitives (ISERDES, IDELAY, OSERDES, ODELAY, MMCM and PLL). Then find a grid of SLICES, Block-RAM, DSP-Blocks and other. All is arranged as shown in the figure.



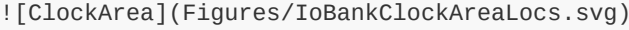
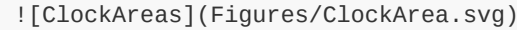
All the IOB, IO-primitives and grid of logic elements in a clock area in an FPGA have XY location information. That information can be found when opening the device view in the Vivado software. Some examples:

IOB_X0Y0	IOB is the location for input and output buffers.
HPIOBDIFFINBUF_X0Y0	Use this when a differential IO is used in an HP bank.
OLOGIC_X1Y33	Is a OSERDES
ILOGIC_X1Y15	This is a ISERDES
IDELAY_X1Y15	
ODELAY_X1Y33	
SLICE_X49Y254	
DSP48_X8Y101	
RAMB36_X6Y50	One name for all RAM derivatives (FIFO, ...)
.....	
.....	

- The XY grid starts down-left in the FPGA logic area and end top-right with XY coordinates depending the size of the FPGA.

- General rules for 7-Series FPGA
 - An IO-Bank column has 50 IO divided in two halve columns'
 - The 50 single ended IO, can be used as 24 differential IO provided. The two remaining IO stay single ended (one on top and one in the bottom of the IO_Bank).
 - For each IOB there is one IDELAY, ILOGIC (ISERDES), OLOGIC (OSERDES) and ODELAY available.
 - A PLLs and a MMCM are available per IO-bank, easy to figure out the grid of those.
 - Behind the IO primitives columns the grid of logic starts.
 - The logic of a clock area behind an IO-Bank column takes an Y (vertical) range of 50 slices
and an X size of 80 slices on the right side and 74 slices on the left side of the FPGA die.
 - A Block-RAM column takes up to 10 RAMB36 primitives, were each RAMB36 can be divided in two RAMB18 blocks. A clock area has 3 Block-RAM columns.
 - The same applies to the DSP blocks, a column takes up to 20 DSP primitives and the

clock area has 3 of these columns.

- The XY coordinates in the grid change depending the position of the IO-Bank and clock area in the device. In most device there are two IO-Bank columns and IO-Banks are stacked up. The bottom-left IO-Bank has thus a column of IO starting from *IO_X0Y0 to IOB_X0Y49*. The IO-Bank on top of this bottom-left IO-Bank starts at *IOB_X0Y50* to location and counts 50 IO up to *X0Y99, and so on.
- The clock area as described in above text shown in a figure:

- The IO-Bank structure for a 7-Series FPGA as discussed in above text in a figure:

- This is the theory behind the pblock constraints in the XDC constraints file.
- How to setup pblocks for a user create design
 - Start Vivado and create a new IO-Planning project for for device that is going to be used.
 - Open the device view and browse in the view to the IO-Bank that is going to be used for the design.
 - Record the position of the IO-Bank in the FPGA logic.
 - Select the left-bottom SLICE of the clock area and record the XY position.
 - Select the top-right SLICE of the clock area and record the Xy value of that SLICE.
 - With these values it is possible, using the given "General Rules", to position every primitive inside the given clock area

◦ These locations change thus when:

- Changing the FPGA part type.
 - Changing the IO-Bank the interface is placed in.
- set_property DIFF_TERM_ADV TERM_100
It is used to enable or disable the built-in, 100 Ohm, differential termination.
 - set_property DQS_BIAS FALSE
DQS_BIAS must be set to TRUE for AC coupled LVDS standards, else it must be set to FALSE.
 - set_property EQUALIZATION EQ_NONE
Must be set to an EQ_LEVELn when the design uses and AC coupled LVDS setup else it must be set to EQ_NONE. Determine the correct equalization factor experimentally.
 - set_property LVDS_PRE_EMPHASIS FALSE
 - set_property IOSTANDARD LVDS
Define the IO standard for IO not created by the synthesis tool from constraints set in the VHDL source code.