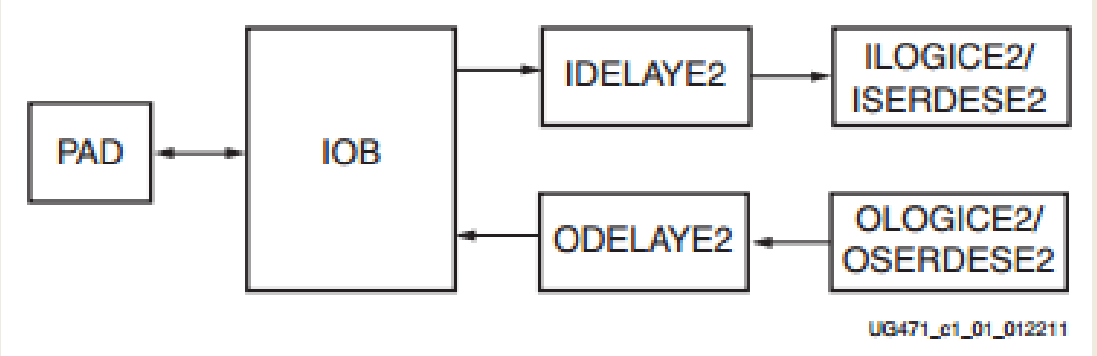
FPGA IO Resources

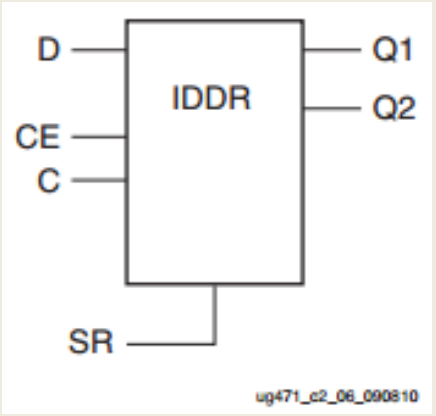
- FPGA pin buffers can be configured for
  - Voltage
  - impedance,
  - Slew rate
  - Pull(up/down)

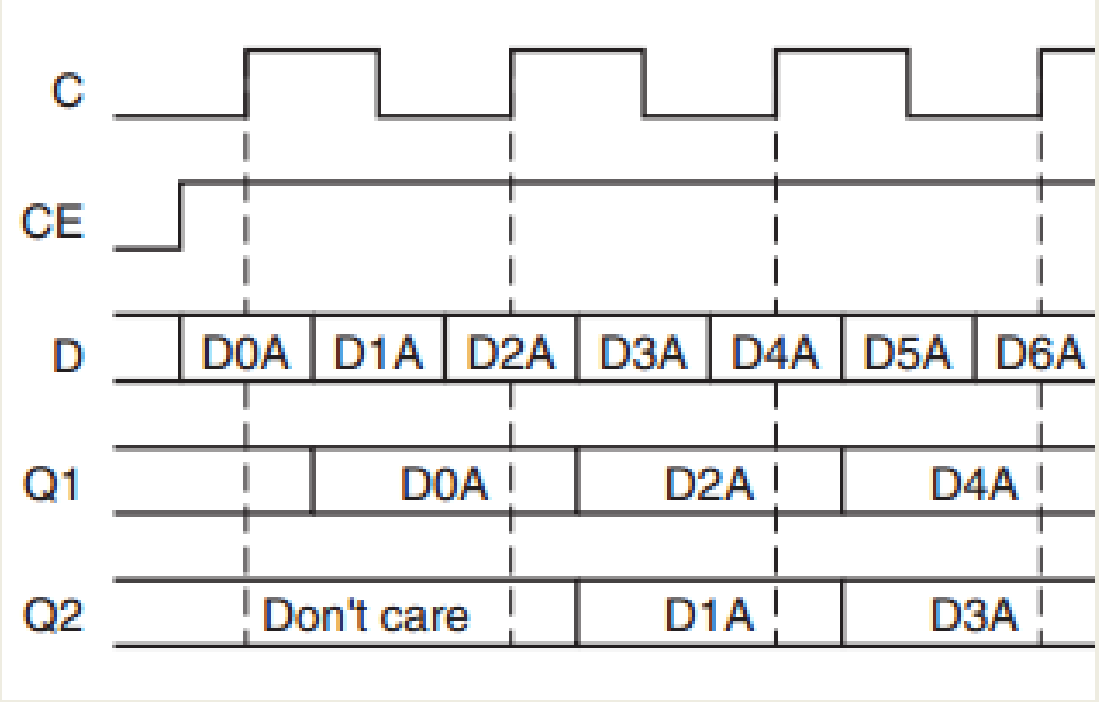


- IO columns include Xilinx SelectIO resources
  - Register (SDR/DDR in IOLOGIC)
  - Fine-grained delay adjustment (IODELAY)
  - Serializer/Deserializer (IOSERDES)

FPGA IO Resources

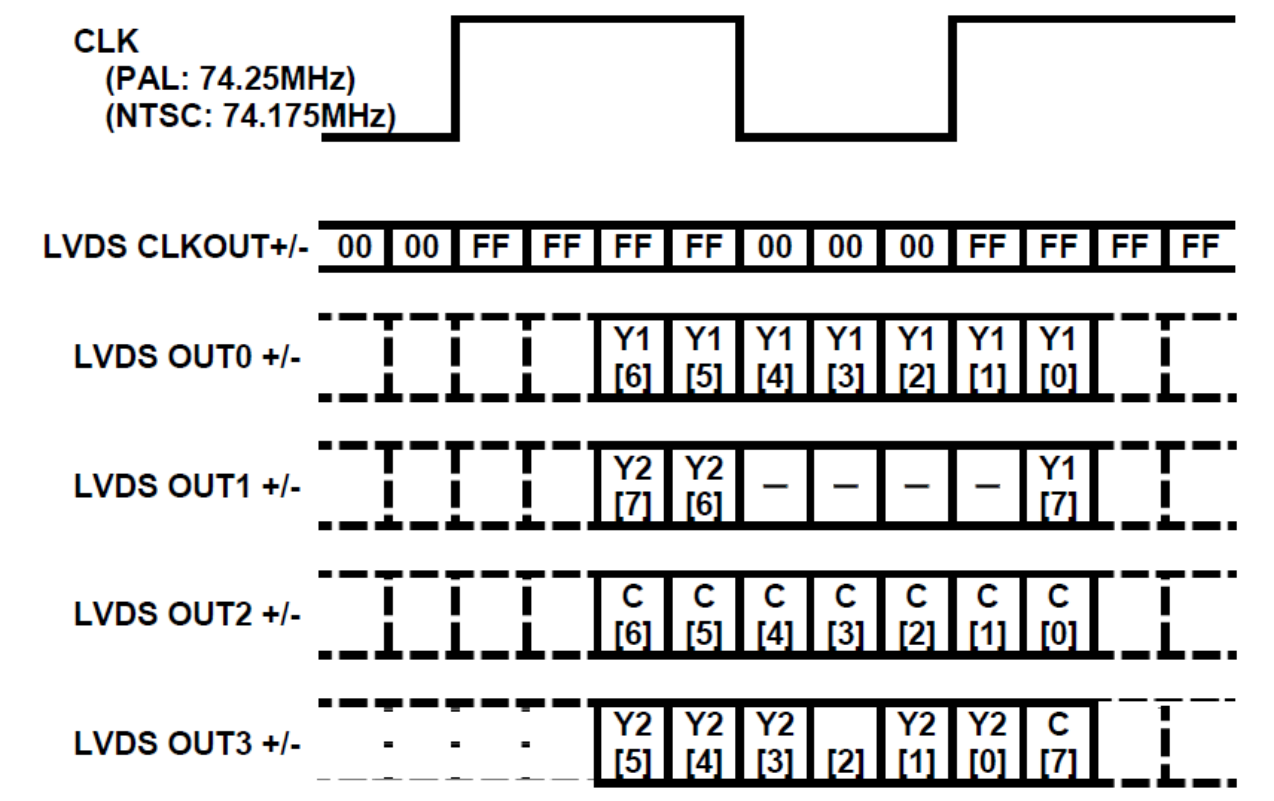
- Challenge: High Data Rates, Low Power (!)
- IODDR (IDDR/ODDR)





FPGA IO Resources

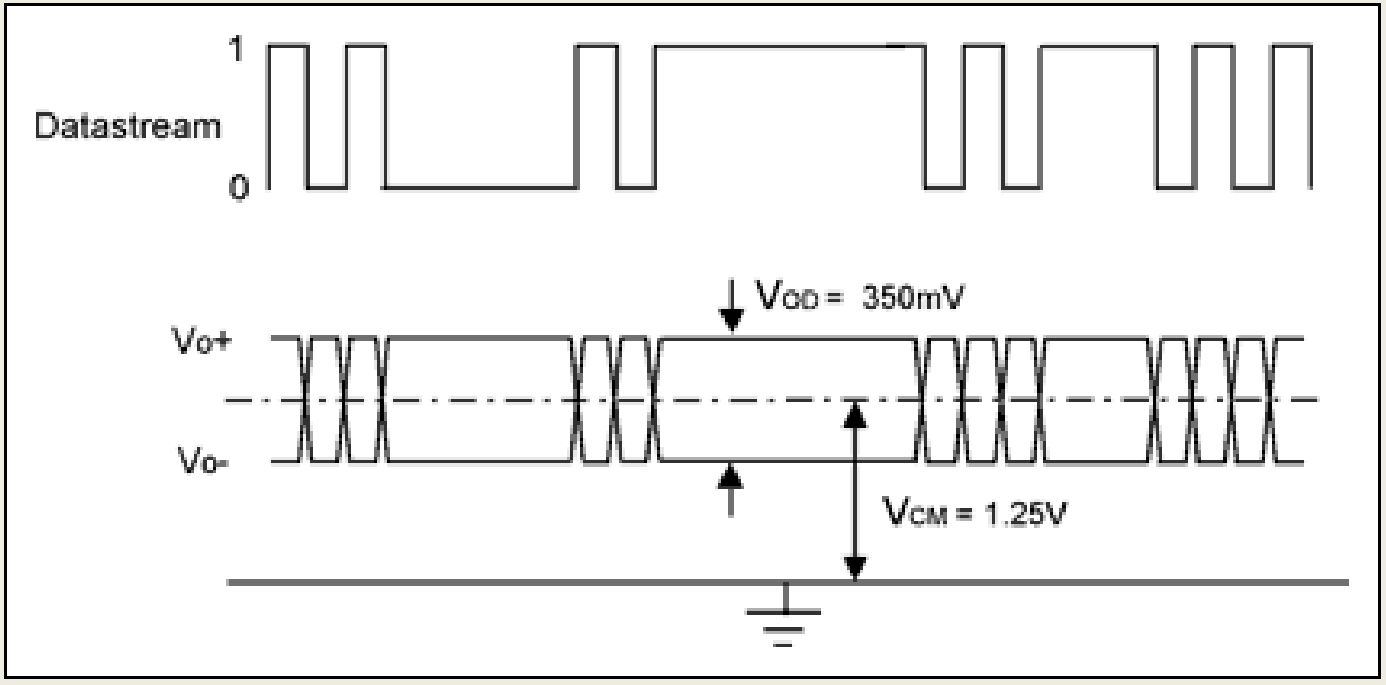
- Even Higher Rates: LVDS Serial Transmission



Panasonic GP-MH322 LVDS Transmission

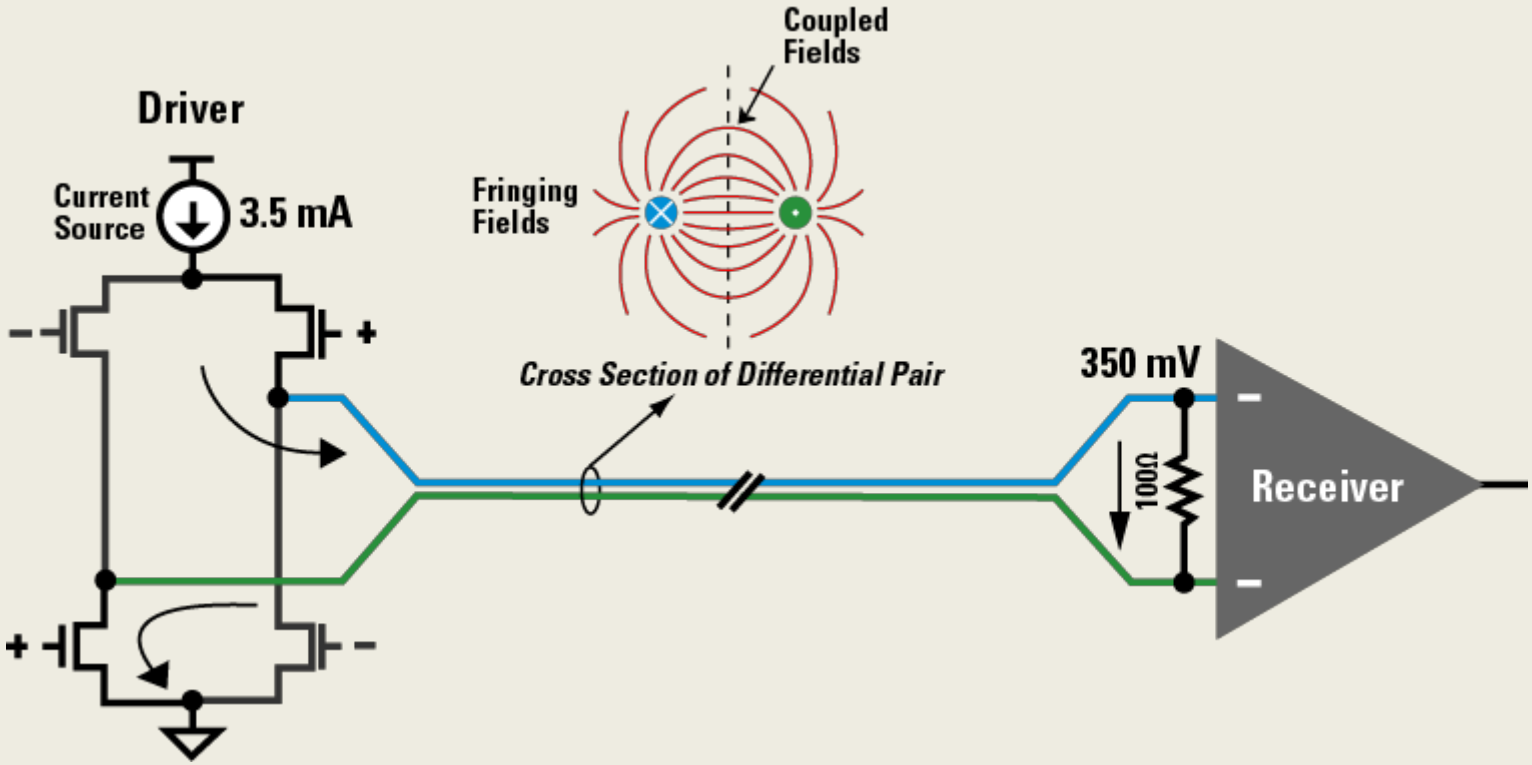
FPGA IO Resources

- LVDS: Low Voltage Differential Signalling



FPGA IO Resources

- LVDS: Low Voltage Differential Signalling



LVDS Deserializer

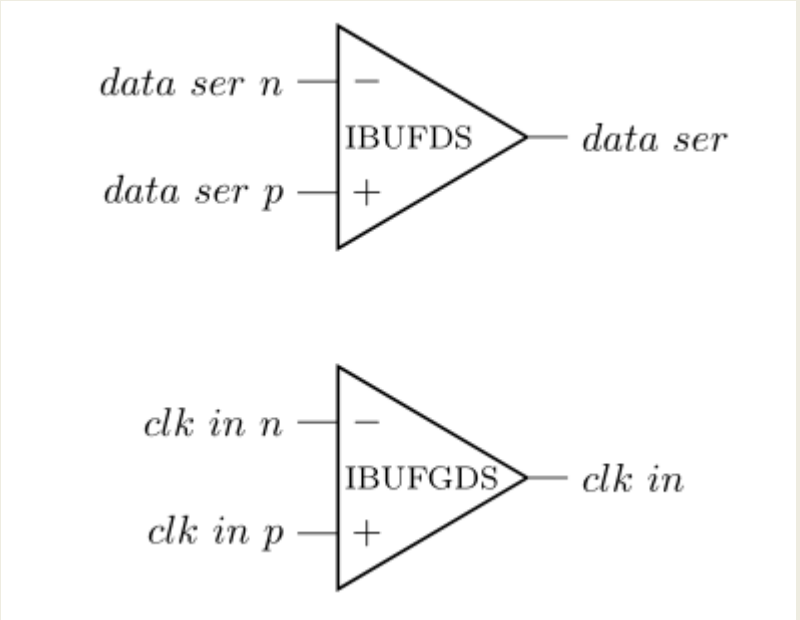
- Steps for deserialization
  - Receive signal with proper termination and convert from differential to single-ended
  - Recover transmission clock from LVDS clock
  - Align input data to recovered clock (sub-bit)
  - Convert serial data to parallel data
  - Align parallel data at word level

LVDS Deserializer

- Steps for deserialization
  - Receive signal with proper termination and convert from differential to single-ended
  - Recover transmission clock from LVDS clock
  - Align input data to recovered clock (sub-bit)
  - Convert serial data to parallel data
  - Align parallel data at word level

FPGA IO Resources

- IBUF(G)DS
  - Primitive for buffering input differential signals
  - G variant must be used when signal is clock and output goes to MMCM/PLL
  - Optional 100 Ohm termination (for LVDS)



LVDS Deserializer

- Steps for deserialization
  - Receive signal with proper termination and convert from differential to single-ended
  - Recover transmission clock from LVDS clock
  - Align input data to recovered clock (sub-bit)
  - Convert serial data to parallel data
  - Align parallel data at word level

Reminder: FPGA Clocking

- MMCM
  - Multiply input clock by the serialization factor
  - Feedback not required, we don't care about phase alignment
  - Outputs go to Ios (fast clock) and internal logic (slow clock)
  - Which buffers to use? (BUFG/BUFR/BUFIO)

Reminder: FPGA Clocking

- MMCM
  - Multiply input clock by the serialization factor
  - Feedback not required, we don't care about phase alignment
  - Outputs go to IOs (fast clock) and internal logic (slow clock)
  - Which buffers to use? (BUFG/BUFR/BUFIO)
    - Use BUFG whenever possible (global reach)
    - Use BUFR/BUFIO when BUFG is not fast enough

LVDS Deserializer

- Steps for deserialization
  - Receive signal with proper termination and convert from differential to single-ended
  - Recover transmission clock from LVDS clock
  - Align input data to recovered clock (sub-bit)
  - Convert serial data to parallel data
  - Align parallel data at word level

FPGA IO Resources

- IODELAY (IDELAY/ODELAY)
  - Primitive for fine-grained delay on pin-to-fabric or fabric-to-pin paths
  - 31 “taps” (delay values from original signal)
  - Delay increments are PVT independent
  - Can change delay during operation
    - Useful for calibration of delay to daughterboards etc

LVDS Deserializer

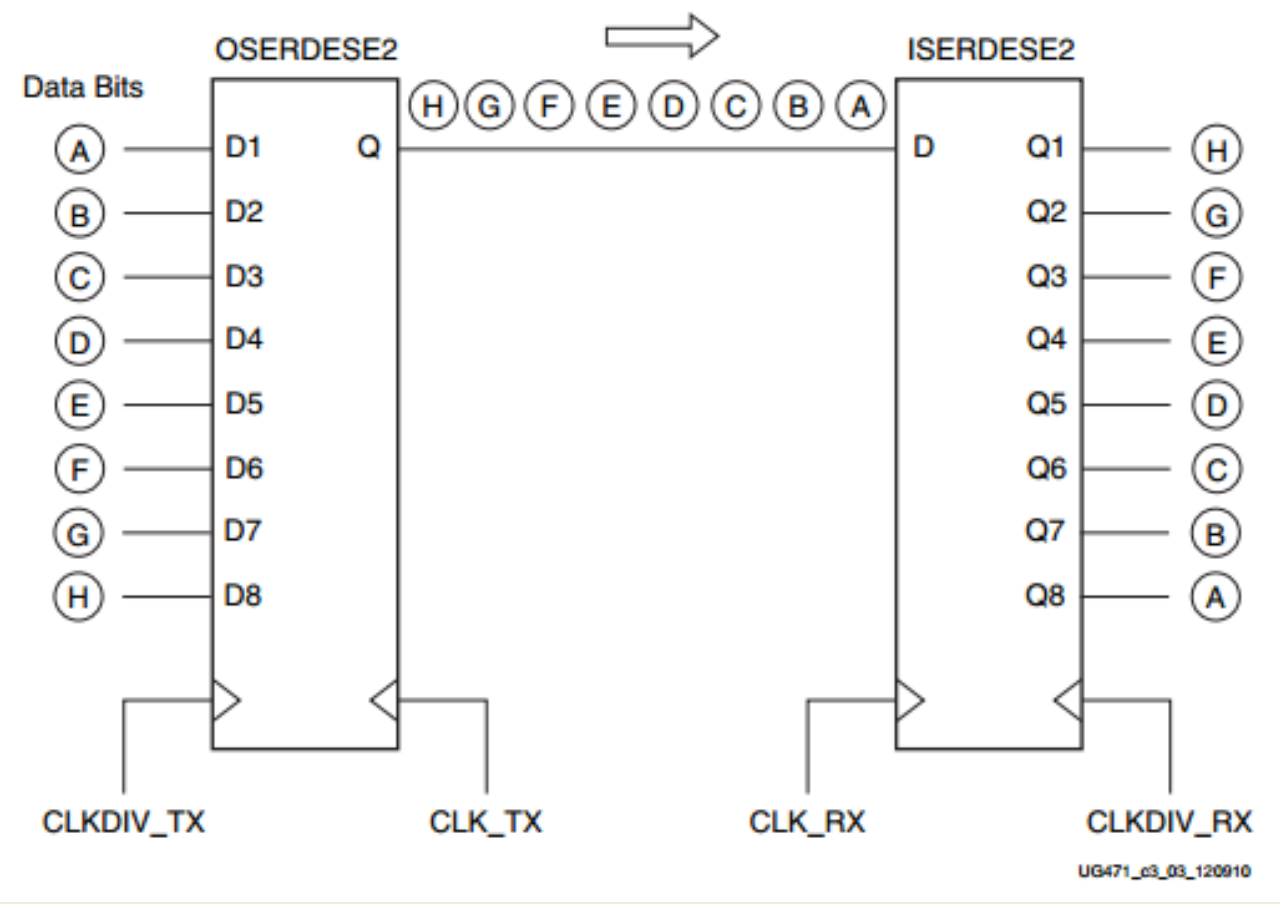
- IDELAY Calibration
  - Span range of IDELAY, looking for a transition in the output
  - When a transition is encountered, we increase or decrease the IDELAY by half a bit period
  - Afterwards, the sampling is in the middle of the data eye
  - Discussion: limitations for this type of calibration

LVDS Deserializer

- Steps for deserialization
  - Receive signal with proper termination and convert from differential to single-ended
  - Recover transmission clock from LVDS clock
  - Align input data to recovered clock (sub-bit)
  - Convert serial data to parallel data
  - Align parallel data at word level

FPGA IO Resources

- SERDES (Serializer/Deserializer)



FPGA IO Resources

- SERDES (Serializer/Deserializer)
  - SDR/DDR input/output
  - Up to 8 bits SDR/DDR with a single SERDES primitive, up to 14 bits DDR with two SERDES primitives (master/slave)
  - CLK and CLKDIV must be phase-aligned
    - Outputs of parallel BUFR and BUFIO are phase-aligned by design, for this purpose
  - Bitslip mechanism for aligning data at word level

Implementation Constraints

- Implementation constraints affect
  - IO placement (pin where signal is connected)
  - IO buffer type (voltage of pin)
  - IO buffer properties (slew, pullup etc)
  - Timing
    - Clock period
    - Offsets before and after clock
    - Multi-cycle paths etc.
  - Resource placement

Implementation Constraints

- IO constraints
  - Placement: LOC constraint
    - NET “<signal>” LOC=<pin name>
  - Buffer type (IOSTANDARD=<pin type>)
    - LVTTL, LVCMOS12/15/18/25/33, LVDS
    - Many others…
    - Same type for all pins in a bank
  - Drive strength (DRIVE=<mA>)
  - PULLUP/PULLDOWN

Implementation Constraints

- Resource placement
  - LOC constraint can also place Slices, BRAMs, DSPs
  - AREA_GROUP and RANGE constraints
    - Define a placement constraint for all logic of a certain type within the block
    - May be specified separately for each resource type: Logic/Distributed RAM/Block RAM/DSP
    - Example: constrain all logic driven by a BUFR to the clock region of the BUFR:

```
AREA_GROUP "pblock_bufr"
RANGE=CLOCKREGION_X1Y2:CLOCKREGION_X1Y2;
```

Further Reading

- Xilinx 7-Series IO Resources User Guide
- Xilinx Constraints Guide
- Xilinx PlanAhead Floorplanning Tutorial