RTR532 2018 Spring Lecture 06

- **Design clock frequency and synchronization**
- **Data transfer and protocol implementation**
- **FPGA hard blocks**

Design clock frequency and sync

Logic element timing parameters

- Input signal D must obey:
  - Setup time $\mathbf{T_{SU}}$ – D must be stable before clock edge
  - Hold time $\mathbf{T_H}$ – D must be stable after clock edge
- Output signal Q obeys:
  - Contamination delay $\mathbf{T_{CD}}$ – delay from clock edge to Q starts to change. Given in IC datasheet.
  - Propagation delay $\mathbf{T_{PD}}$ – delay from Q start to change to stable Q. Given in IC datasheet.
  - Clock to Q $\mathbf{T_{CtoQ}}$ – delay from clock pulse to stable Q
    $\boldsymbol{T_{CtoQ} = T_{CD} + T_{PD}}$

Relation among logic elements and flip flops (1)

- The number of logic elements through which signal propagates directly influence max frequency the design can work with
- $\boldsymbol{T_{PD} \leq T_{clk} - (T_{CtoQ} + T_{SU})}$
- Logic element caused delay: $\boldsymbol{T_{PD} = N_L * T_{PD(LUT)}}$ where $\boldsymbol{N_L}$ is the number of LUTs in series
- Clock frequency: $\boldsymbol{f_{clk} = \frac{1}{T_{clk}}}$

Relation among logic elements and flip flops (2)

- Max number of elements for a clock frequency $\boldsymbol{f_{clk}}$:

$$\boldsymbol{N_L = \frac{T_{clk} - T_{CtoQ} - T_{SU}}{T_{PD(LUT)}}}$$

- Max frequency for a flip flop and number of elements:

$$\boldsymbol{f_{clk(max)} = \frac{1}{N_L * T_{PD(LUT)} + T_{CtoQ} + T_{SU}}}$$

Casy study – Xilinx Spartan-6 and Virtex-6

- Configurable logic block – flip flop + mux + ...
- Datasheets:
  - Spartan 6 (page 48/89)

https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf

  - Virtex-6 (page 42/65)

https://www.xilinx.com/support/documentation/data_sheets/ds152.pdf

Casy study – Xilinx Spartan-6 and Virtex-6

	Param.	Xil. Symbol	-3	-1	Units
Spartan-6	$T_{PD(LUT)}$	T_{ILO}	0.21	0.46	ns, max
	T_{PCQ}	T_{CKO}	0.45	0.74	ns, max
	T_{SU}/T_H	T_{DICK}/T_{CKDI}	0.42 / 0.28	0.90 / 0.56	ns, max
Virtex-6	$T_{PD(LUT)}$	T_{ILO}	0.06	0.09	ns, max
	T_{PCQ}	T_{CKO}	0.29	0.44	ns, max
	T_{SU}/T_H	T_{DICK}/T_{CKDI}	0.30 / 0.17	0.44 / 0.25	ns, max

Whiteboard task

- We have 10 levels of logic elements between two flip-flops.
  - Will design work with $f_{clk} = 200MHz$?
  - What delay is caused by logic elements between two flip-flops ?
  - What is the max f_{clk} the design can work with ?

Additional delays

- IO buffer switching
- Routing delays – delay caused by IC interconnections (speed of light) – usually small delay
- Multiplexers used for routing

How can tools help us ?

- Manual calculation
- Analysis tools (part of Quartus / Vivado)
  - Altera – TimeQuest Analyzer
  - Xilinx – Timing Analyzer
- Tools report:
  - Max frequency for your design
  - Slowest places (levels of logic elements)

You can constrain the design with **timing constrains**

- Designer tells the software with what kind of frequency you want the design to work with (e.g. ZZZ clock frequency).
- This is super important if working with wide signals, not to have bit errors – there is no reason NOT to constrain the design.
- Tool tries to place elements with minimum number of logic elements (mux-es) to ensure the frequency.

- What is the key thing to have high frequency design ? What can you, as a designer, do to ensure the required frequency ?

Further reading

- Ortus pievienots materiāls – L06 resurss - Timing and Verification
- http://www.ece.ucsb.edu/Faculty/Johnson/ECE152A/L4%20-%20Propagation%20Delay,%20Circuit%20Timing%20&%20Adder%20Design.pdf

Data transfer and protocols

Data transfer

- Why we need data transfer ?

- Data transfer options
  - Within an integrated circuit
  - Within a circuit board among IC's
  - Within a device among circuit boards
  - Among multiple devices

- Data transfer is implemented by a **protocol**

Protocols – key characteristics

1. Architecture (how protocol is physically implemented)
2. Data transfer managers (who starts the communication?)
3. Data transfer directions

Protocol architecture

- Protocols are almost always – layered
- Physical layer (serial, parallel, speed) – number of signals (data, clock), voltages, such as
  - single ended SE
  - low voltage differential signaling LVDS, used in DisplayPort
  - pseudo open drain POD12, used in DDR4
  - others
- Logic layer – rules of communication
  - Length of one conversation
  - Contents of conversation
- Both ends must support the architecture

Protocol data transfer managers

- Master-slave configuration
  - Single master, single slave
  - Multi master, single slave
  - Single master, multi slave
  - Multi master, multi slave
- Master-master

Protocol data transfer directions

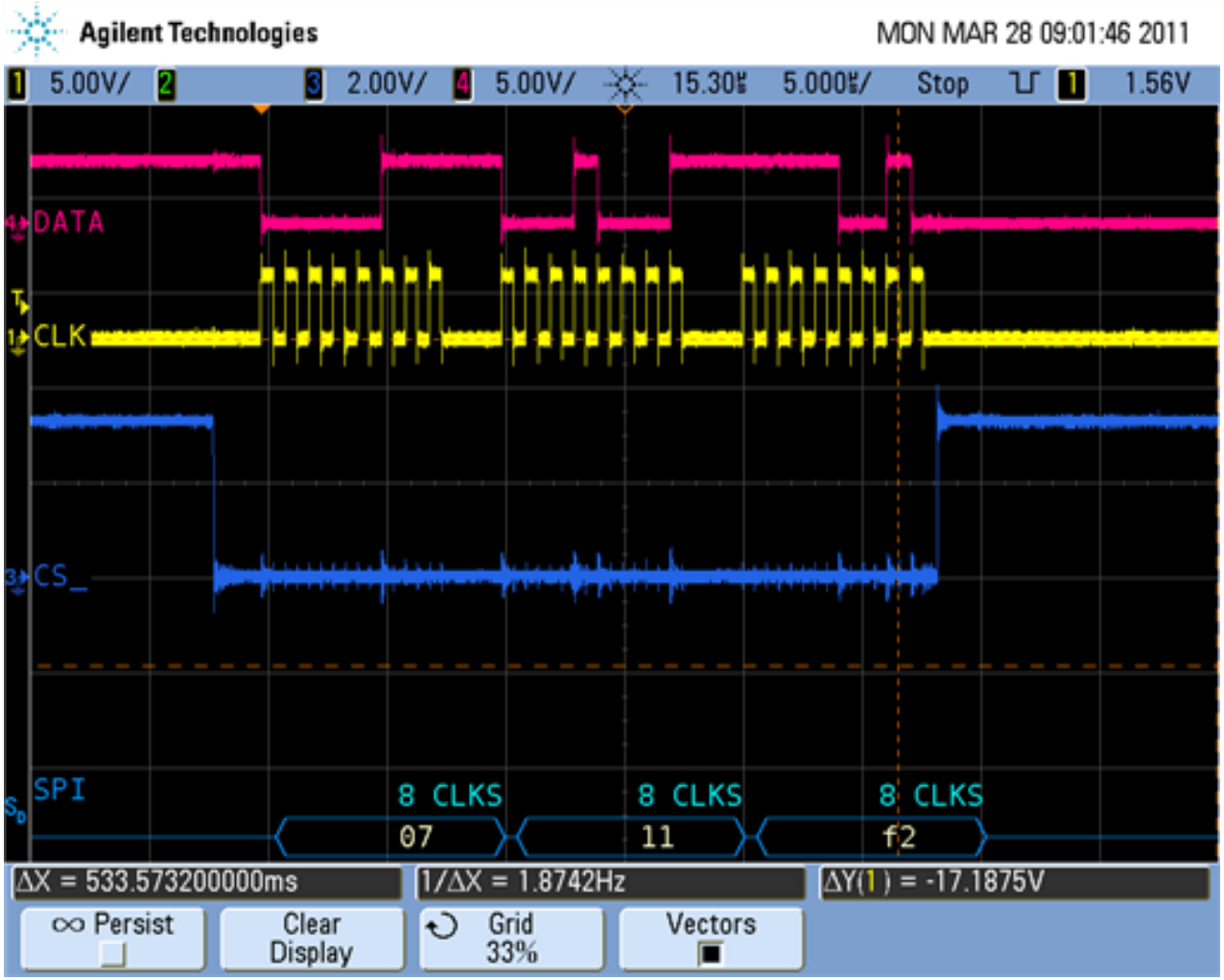
- One direction - simplex
- Bidirectional (one direction at a time) – half duplex
- Full bidirectional – full duplex

Protocols – popular

- Serial - SPI, I2C, 1-wire, JTAG, SATA, PCIe, RS232, USB, Firewire, CAN, Ethernet, AXI interconnect,
- Parallel – PATA (IDE), SCSI, GPIB, Printer port

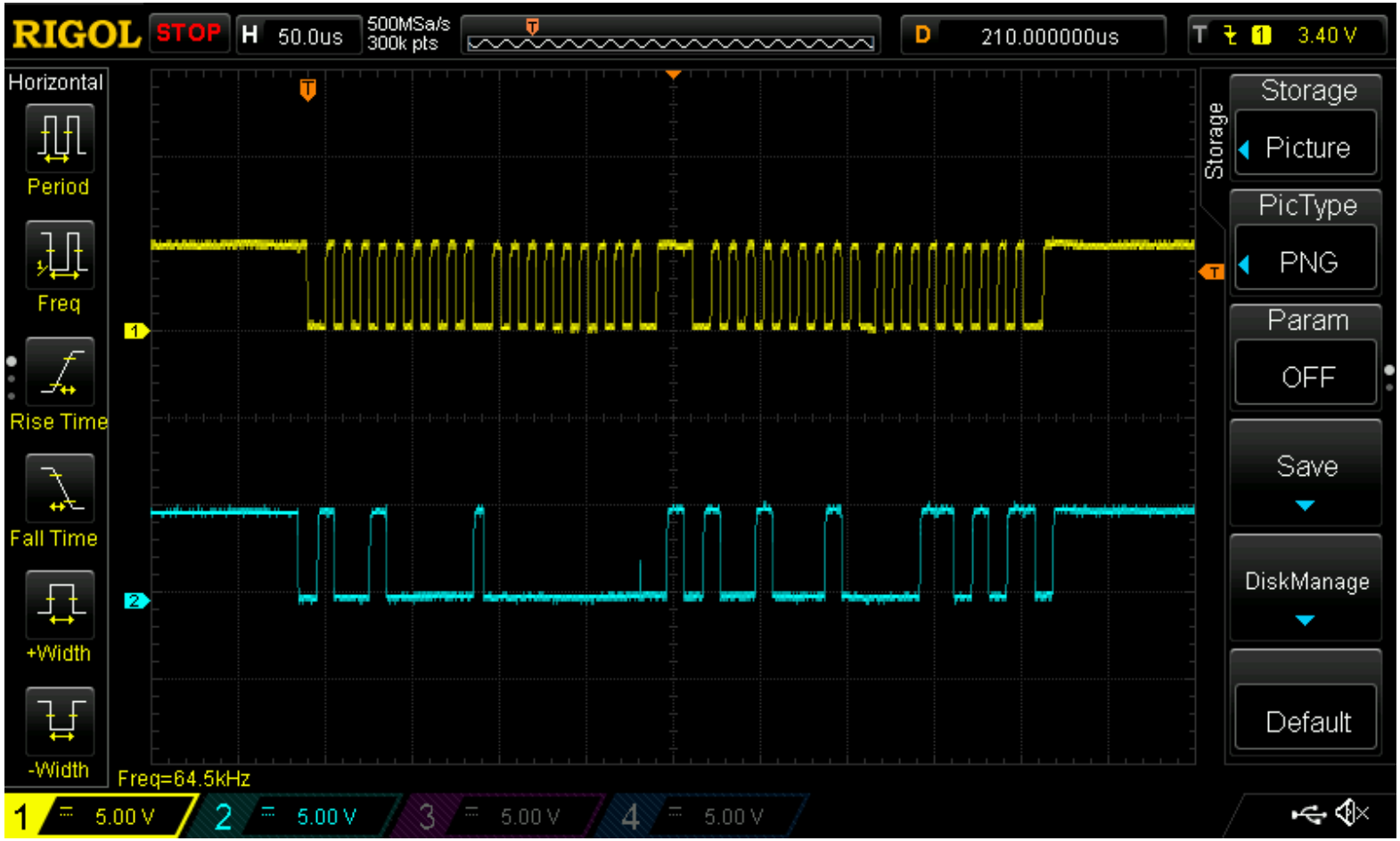
SPI protocol

- SPI – serial peripheral interface
- Three signals + control + return
  - SCL – serial clock
  - MOSI – master out, slave in
  - MISO – master in, slave out
  - CS/SS – chip select or slave select
  - (ground)
- Full duplex


Agilent Technologies
MON MAR 28 09:01:46 2011
1 5.00V/ 2 3 2.00V/ 4 5.00V/ 15.30ns 5.000ns/ Stop 1 1.56V
DATA
CLK
CS_
SPI
8 CLKS
07
8 CLKS
11
8 CLKS
f2
ΔX = 533.573200000ms
1/ΔX = 1.8742Hz
ΔY(1) = -17.1875V
∞ Persist
Clear Display
Grid 33%
Vectors

I2C Protocol

- I2C – Inter integrated circuit protcol
- Two signals = return
  - SCL – serial clock
  - SDA – serial data
  - (ground)
- Half-duplex (because of shared data signal)
- Multiple masters, multiple slaves possible
- Slave selection by serial slave address


RIGOL
STOP
H 50.0us
500MSa/s
300k pts
D 210.000000us
T 1 3.40 V
Horizontal
Period
Freq
Rise Time
Fall Time
+Width
-Width
Freq=64.5kHz
Storage
Storage
Picture
PicType
PNG
Param
OFF
Save
DiskManage
Default
1 5.00 V
2 5.00 V
3 5.00 V
4 5.00 V

Quartus task

Let us implement simple SPI with VHDL:

- 32 clock cycles
- Chip select active low (usually)
- 8 bit command, 8 bit data, 16 bit return
- Clock out data on rising edge
- Clock in data on falling edge

- Start with new project !

FPGA Hard Blocks

...or getting out all of your FPGAs features

FPGA resources (recap)

https://www.altera.com/en_US/pdfs/literature/hb/cyclone-v/cv_51001.pdf

- Breadboard – registers and multiplexers
- Phased lock loop modules (frequency synthesizers)
- Internal Memory
- Memory controllers (DDR2, DDR3)
- Transceivers
- PCIe hard block
- ...others

How to use such resources?

1. Infer them in VHDL code (not all can be inferred)
2. Use IP core generator (Megafunction wizard for Altera, Core Generator for Xilinx) – usually good enough + configuration wizard
3. Call them as modules in VHDL code (advanced) using libraries

* Infer – synthesizes puts your VHDL code in appropriate structure, such as internal memory.

IP Core generators

- Lets you configure part of FPGA registers/multiplexers/memory with specific function, such as first in – first out buffer (very useful !!)
- Lets you use hard blocks of FPGA, such as
  - Phased lock loop
  - Transceivers
  - Specific IO buffers

Multipliers

- Xilinx – DSP48
- Altera – Embedded Multipliers

- Guaranteed frequency (e.g. 250MHz, 500MHz, depending on an FPGA)
- Fixed multiplication width
- Accumulator

Internal memory

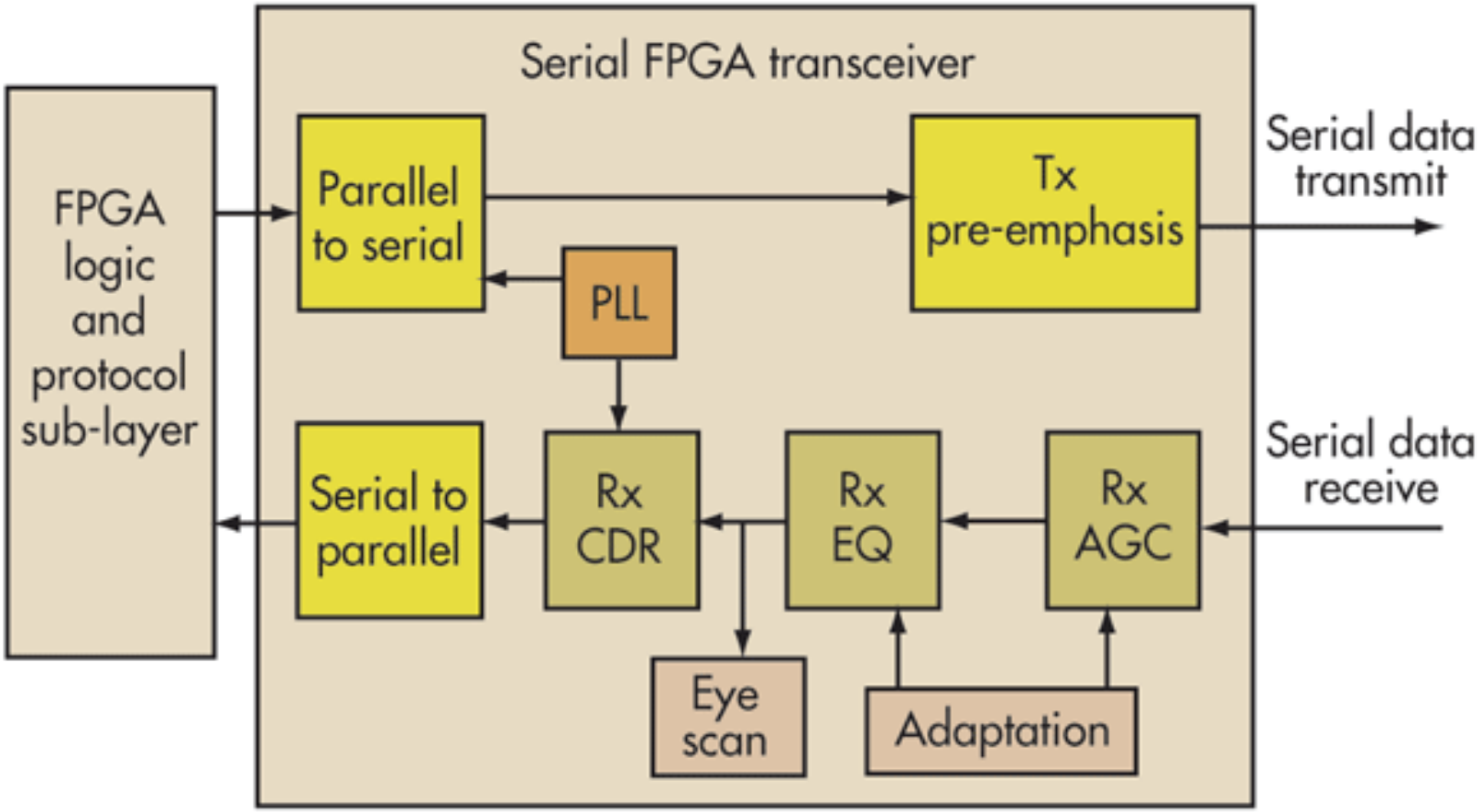
- Specific FPGA part to store volatile data (only when power is applied)
- As base – single memory element (such as M4K)
- Guaranteed speed
- Wide configuration options (as example, 4k x 1bit, 256 x 16b, ...)
- Can be used in smarter configurations with registers, such as:
  - Single-port, dual-port, shift registers, fifo buffers (generate with IP Core generator!)

Phased lock loop

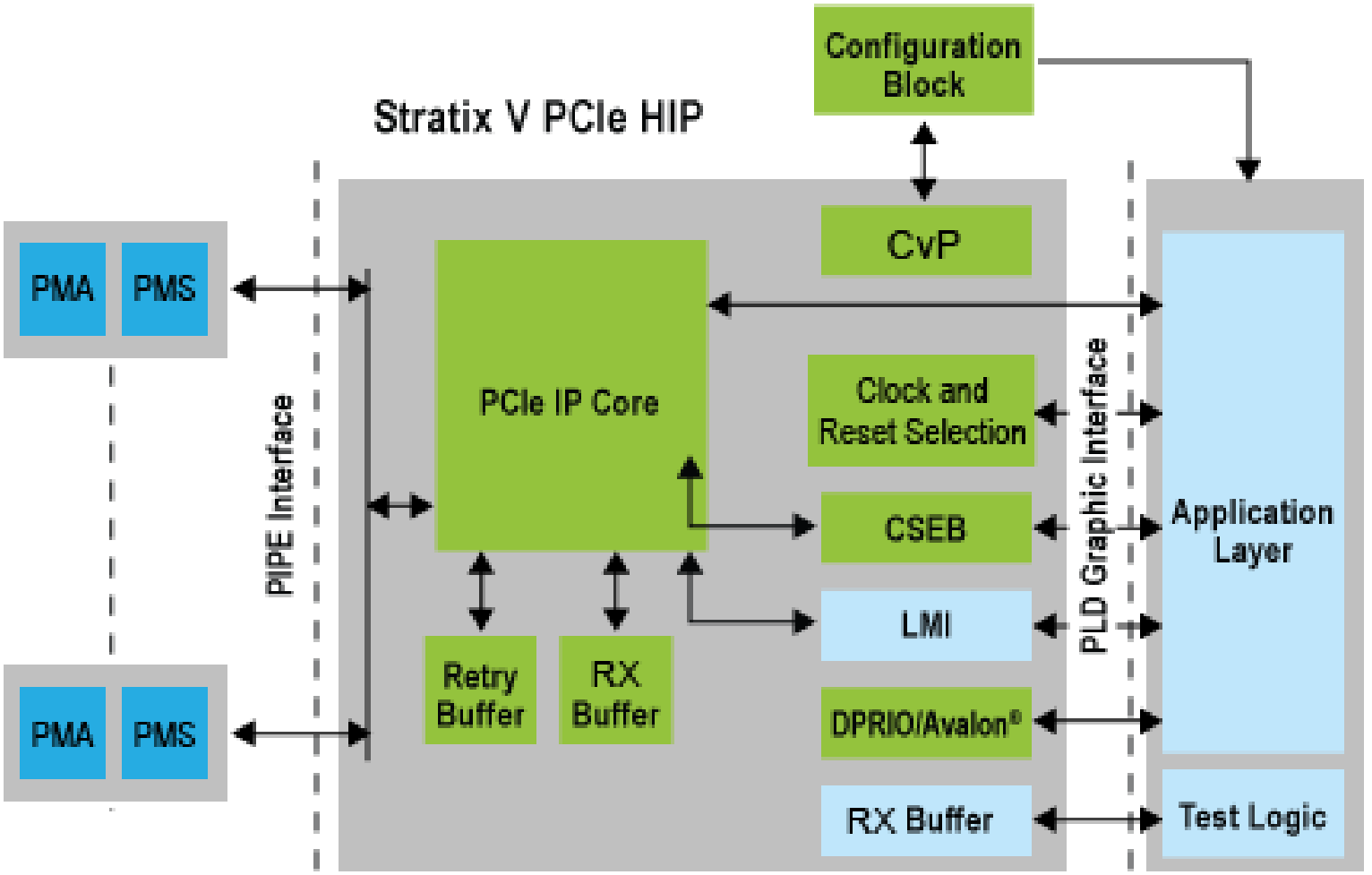
- Generate clock with known
  - Frequency
  - Phase
  - Duty cycle
  - Jitter
- Generate multiple clock frequencies within an FPGA from a single source
- Generate clock frequencies for transceivers

Transceivers

- HUGE bandwidth serial data
- Vastly configurable for commonly used protocols (DisplayPort, PCIe, SATA, ...)
- A way to load serial data to an FPGA
- At simple representation – a serializer and a deserializer.


Serial FPGA transceiver
FPGA logic and protocol sub-layer
Parallel to serial
PLL
Tx pre-emphasis
Serial data transmit
Serial to parallel
Rx CDR
Rx EQ
Rx AGC
Serial data receive
Eye scan
Adaptation

Transceiver as part of PCIe hard block



Practical part – using Megawizard Plug-in manager

- Create new project
- Call Megawizard Plug-in manager and generate:
  - LPM_MULT
  - FIFO
  - Memory element