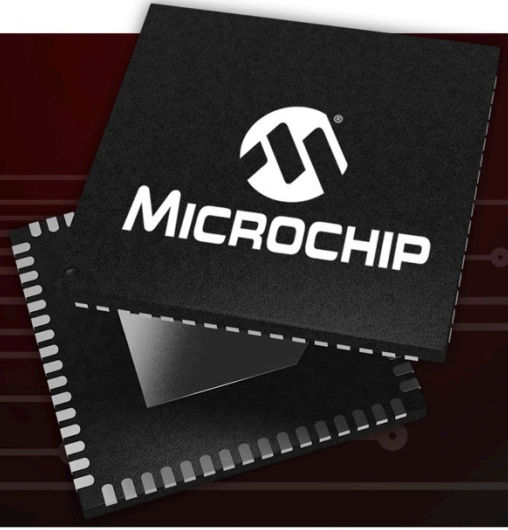
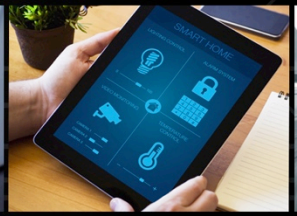
MICROCHIP



A Leading Provider of Microcontroller, Security, Mixed-Signal, Analog & Flash-IP Solutions






RTG4 Design Tips and Tricks
Ryan Mohan
Space Forum 2019

Introduction

- **Learn to use RTG4 features and design tools effectively and efficiently**
  - Discover latest design software features and device level guidelines
  - Confirm design intent and highlight improvement areas by reviewing key Libero reports
  - Remember key guidelines that dramatically improve quality of results for complex/large designs
  - Obtain pointers to appropriate/updated documentation for in-depth details

Agenda

- Considerations when using Fabric PLL
- Efficient usage of radiation mitigated clocks and resets
- Guidance when using fabric LSRAM and uSRAM blocks
- Accounting for clock jitter in design analysis
- Maximizing usage of the Libero SoC design flow features
- Designing robust multi-lane SerDes EPCS links when performing clock resource sharing

RTG4 PLL Tips

- **Determine if Fabric PLL will be used vs. Fabric CCC-only with PLL bypassed**
- **Decide if using single PLL or triplicated PLL (consult radiation test reports)**
  - If PLL is used, evaluate CCC option to employ automatic PLL reset logic upon loss of lock
    - If not using PLL auto-reset logic, then add user logic to monitor PLL lock to react to loss of lock events
    - E.g. If PLL lock is lost beyond expected self recovery time for single-PLL, place design in safe state and toggle PLL reset/powerdown input before re-acquiring lock and resuming normal operation
- **Libero SoC v11.9 SP5 adds enhanced PLL calibration soft logic**
  - New FCCC core resolves temp dependent PLL lock instability across full mil-temp range
    - Initial release supports single Fabric PLL, with support for Triplicated PLL in a future release
  - Select a free running clock source to feed the PLL calibration IP and optional auto-reset logic
    - Can use internal 50 MHz RC Osc, or any external oscillator, running at 50 MHz or below
  - Updated SerDes and FDDR cores with initialization will apply PLL calibration to SPLL (used for PCIe/XAUI) and FDDR FPLL

RTG4 PLL Tips (2)

- **Ensure Fabric PLL lock window setting adheres to Min LOCKWIN formula**
  - Consult section 5.4.3.6 of the RTG4 Clocking User's Guide (UG0586) Rev 7
  - 500 ppm or 1000 ppm lock window settings can result in unstable lock (removed in v11.9 SP5)
  - Lock window limits the allowable phase error at the PLL Phase and Frequency Detector (PFD)
    - Upon device start-up / reset release, phase error of the divided refclk vs. the feedback clock is measured
    - Once phase error is within ppm lock window, lock assertion waits for a counter to reach the LockCnt value
    - LockCnt defaults to 1024 cycles of refclk to allow PLL to reach normal output jitter, while minimizing PLL Lock acquisition time (for a typical reference clock)
  - The Default Lockwin setting in Libero is 6000 ppm
    - Default strives to maintain PLL lock in the presence of normal jitter on a typical reference clock and feedback clock routed through the global clock network

Min Lockwin Setting (ppm) =

MIN (ROUNDUP-TO-NEXT-SETTING (650 + 52.34*PFD Rate), 6000) ppm

Lockwin (ppm)	PFD Range Allowed (MHz)
16000	16000 ppm Lockwin setting allowed: 10 MHz to 200 MHz
6000	6000 ppm Lockwin setting allowed: 10 MHz to 200 MHz
4000	4000 ppm Lockwin setting allowed: 10 MHz to 63 MHz
3000	3000 ppm Lockwin setting allowed: 10 MHz to 44.5 MHz
2000	2000 ppm Lockwin setting allowed: 10 MHz to 25.5 MHz
1500	1500 ppm Lockwin setting allowed: 10 MHz to 16.5 MHz

Effective Use of Global and Reset Resources

- **Review "Global Nets Report" and "Compile Netlist Resources Report"**
  - See Report Pane in GUI or the XML files in project's /designer/<top_level>/ subfolder:
    - <top_level>_glb_net_report.xml and <top_level>_compile_netlist_resources.xml
    - These XML files can be opened in a Web Browser if rptstyle.xsl style-sheet is in the same folder
  - Analyze if clocks use local routing or global routing, and how they arrive on the global network
  - Confirm CCC CLKI pin assignment and Hardwired routing
    - E.g. If CCC_SE1_CLKI3 feeds CCC reference clock, confirm CCC location matches IO pin naming (SE1) and that a Hardwired Net Type is selected for the reference connection
  - Select clock input pins and CCC location early-on, especially if using all CCCs and SpW circuits
    - Prevents unexpected competition for global input pins and avoids resorting to locally routed clock inputs
- **Minimize number of async reset domains**
  - Analyze reset interpretation (sync or async) and search for unnecessary async resets
    - Review load for async reset and decide if a unique reset is really needed or if consolidation is possible
    - For async-on, sync-off resets, consolidate across all sync clocks and time release to slowest clock
- **Refer to Application Note AC463: RTG4 Clock and Reset Network Usage**

Effective Use of Fabric LSRAMs and uSRAMs

- **Use the LSRAM and uSRAM configuration cores in Libero SoC's Catalog pane**
  - Inference via synthesis provides access to basic LSRAM and uSRAM configurations
    - E.g. Managing ECC flags is difficult when Inferring ECC RAMs, and can't infer pipelined-ECC RAM mode
  - Libero cores support advanced RAM block configurations, such as ECC or depth cascading
  - Direct primitive macro instantiation not recommended due to complex rules and potential pitfalls
- **Avoid simultaneous read and write to the same address**
  - Consider the synthesis directive syn_ramstyle="rw_check" when inferring RAM blocks
- **Enable all RAM pipelining options to achieve higher system clock rate targets**
  - Compare RTG4 DS LSRAM/uSRAM read access time (T_{CLK2Q}) against design requirements
- **Enable SET mitigation for all ECC enabled LSRAM/uSRAM blocks**
  - Can use NDC constraint with "set_mitigation" command applied at hierarchical instance level
- **Review Ch. 4 of updated RTG4 Fabric User's Guide (UG0574) rev 5 (Aug 2019)**

Account for Clock Jitter

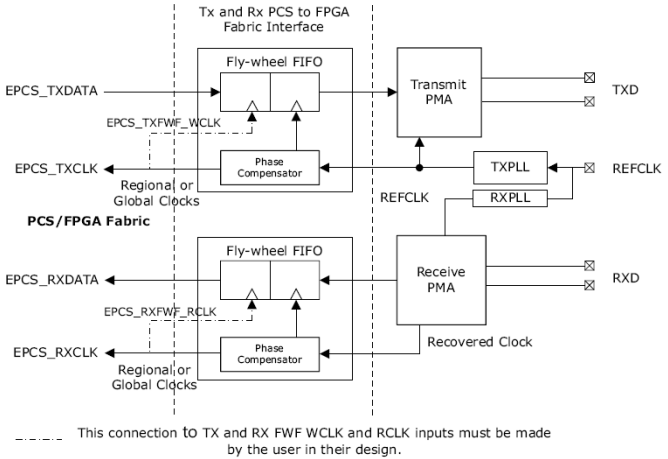
- **Select best Input IO standards for clocks with respect to input buffer jitter**
  - Design differential inputs to meet the RTG4 datasheet VID and VICM limits
    - Select appropriate ext. termination resistor (or use LVDS25 in range supporting internal 100-ohm ODT)
  - RTG4 datasheet specifies:
    - MSIO Input buffer jitter across various IO standards
    - PLL output jitter across various reference clock input IO standards
    - SerDes TX jitter for various SerDes reference clock input buffer IO standards
- **RTG4 DS includes RC Oscillator jitter, with / without fabric switching noise**
- **Account for known jitter via uncertainty constraints in Libero SoC v12.2**
  - Can be applied within a single clock domain (simple uncertainty) or between clock domains
  - Applies to **both** setup and hold by default, but –setup and –hold arguments are also available
  - Libero SoC v12.2 Timing Driven Place and Route adds support for uncertainty constraint application during max delay optimization and min delay violation repair

Efficient Tool Usage

- **Incorporate ‘Derived Constraints’ into user design flow**
  - Use Constraint Manager in Libero SoC to derive constraints for generated IP components
    - E.g. CCC and Mi-V components include clock and false-path constraints to aid with timing analysis
  - Project’s Design Hierarchy must contain the catalog component (versus imported HDL)
  - Add the derived SDC file into the list of constraint files, or manually merge into user SDC
    - Remember to re-derive constraints when modifying existing component configuration
- **Review ‘Constraints Coverage’ report**
  - Highlights missing constraints or paths not being covered by existing constraints
  - Enabled by configuring tool options for the ‘Verify Timing’ design flow step
- **Review and understand hold time violations before running minimum delay repair**
  - Identify required logic/HDL changes, multi-cycle paths, false path constraints, etc
  - Avoids masking design issues and saves tool effort and runtime during Place and Route
  - Enable Min Delay Repair in Place and Route options after addressing large violations

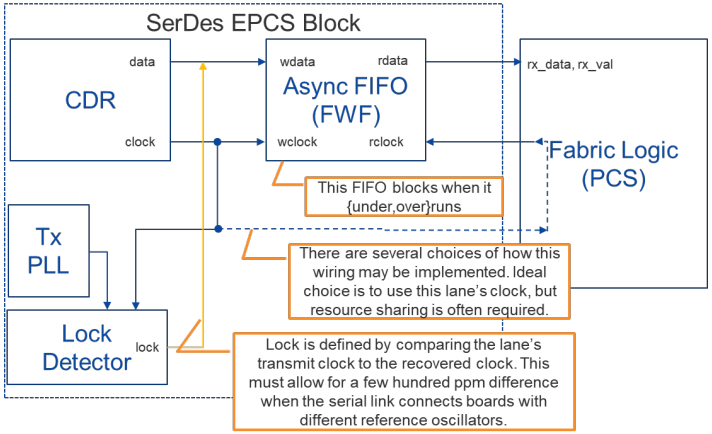
Managing SerDes EPCS Clocks and Resets

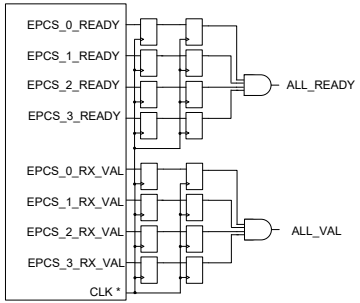
- **RTG4 SerDes quads contain EPCS TX and RX interface Flywheel FIFOs (FWFs) in the data path**
  - FWFs relax the clock phase relationship requirement between FPGA fabric and SerDes block in EPCS mode
  - FWFs enable TX/RX_CLK clock resource sharing for multi-lane links
  - All four EPCS_#_TXFWF_WCLK SerDes inputs can be driven by EPCS_0_TX_CLK or REFCLK through one GLOBAL_#_OUT
  - All four EPCS_#_RXFWF_RCLK SerDes inputs can be driven by EPCS_0_RX_CLK through one GLOBAL_#_OUT, if common clocking (with 0ppm frequency offset) exists at both sides of the link
  - When the lane providing the shared clock is reset, it affects the clock to all four TX/RX FWF and the serial protocol IP in the FPGA fabric
  - With shared RX_CLK, symbol alignment for lane sourcing the RX_CLK must be performed by fabric PCS IP instead of EPCS_#_ARXSKIPBIT
  - The built-in PMA ARXSKIPBIT functionality requires that each lane's RX FWF read clock be clocked with its respective recovered RX_CLK



Managing SerDes EPCS Clocks and Resets (2)

- **Link-up with shared RX_CLKs requires special attn:**
  - RX FWF released from reset when RX_READY rises (calib. finished)
  - Then RX PLL in each lane starts acquiring steady state freq. lock
  - Across 4 lanes, each lane's RX FWF write clk could be different freq.
  - The FWF of the lane whose RX_CLK drives the fabric and shared FWF read clock will operate normally
  - FWFs in other lanes will {under,over}run until all lane CDRs lock
  - FWF underrun or overrun are sticky and require FWF reset to clear
  - **Clean start up of all FWFs across multi-lane RX links using a shared RX clock topology requires a periodic reset circuit**
  - **Goal**: Reset the FWFs once all CDRs reach steady-state to negate any accumulated over/under run conditions and ensure proper data synchronization at link startup, before data sent to fabric PCS logic
  - Asserting EPCS_{0,1,2,3}_ARXSKIPBIT resets respective RX FWF
  - The design should include a state machine to monitor ALL_READY, toggle ARXSKIPBIT, and wait for ALL_VAL or repeat the process
  - Refer to rev 6 of RTG4 High Speed Serial UG0567 – ***coming soon!***





Conclusion

- The considerations discussed during this presentation can be overlooked during initial design development and thus specific documents have been referenced
- This presentation highlights some common design decisions required early in the design cycle to prevent lengthy design analysis and debug during design testing
- Any ongoing studies and new information presented will soon be finalized and added to a future release of the appropriate RTG4 and Libero SoC documentation

Thank You