STARGAZER SYSTEMS INC
1783 Forest Drive, Suite 291
Annapolis, MD 21401

Preparing for FPGA Design Reviews

Brian Smith
BSmith@StargazerSystems.com 301-358-2582

Intro

- Brian Smith
- Stargazer Systems Inc. Provides Design and Design Review services to the Military and Aerospace communities
- FPGA and ASIC Designers
- Hi-Rel Electronics since 1986
- Participated in >50 FPGA design reviews
- Helping to develop Design and Review Methodology for NASA GSFC, and consulting to the NASA NESC on FPGA-related matters

Intro

STARGAZER SYSTEMS INC

- Purpose of this Seminar
  - To discuss an FPGA development methodology that prepares the designer for success … and a successful design review
- Format of Seminar
  - Interactive – Speak up if:
    - You find something wrong with what I've said
    - You have something substantive to add
    - You'd like to explore another route with the audience
    - You want to compliment the speaker ;)

Agenda

- Purpose of Design Reviews
- The Design Process
  - DESIGN
  - IMPLEMENTATION
  - VERIFICATION
- The Review Process
  - Charts
  - Presentations
  - Checklists

Purpose of Design Reviews

STARGAZER SYSTEMS INC

- Introspection
- Vetting of implementation with dependants
- Extra Eyes
- Record of the verification
- Final Design Validation
- To review the readiness of the design … not the designer!

DESIGN

- DOCS□
  - Document the FPGA requirements
    - Functions to be implemented
    - Performance (speed, critical timing, throughput)
    - Interface description (signal levels, timing, software, data formats)
    - Environmental constraints (thermal, radiation level at part, mission duration)
    - Testability requirements (JTAG, board scan, software, observable internal points).

FPGA Requirements Document

Select FPGA Part

- Select Early
- Consider:
  - Package style
  - Reliability / Flight qualification status / Heritage
  - Radiation specs (Total Dose and Single Event Effects)
  - Estimate of utilization:
    - Use prior experience
    - Find similar design and get gate count for target technology
    - Overestimate if a guess is necessary
    - Quantity needed
  - Consider Speed and Power Rating

Project/Board Parts List

Design Guidelines

- Gather and Discuss design guidelines and review requirements with project at the beginning … you don't want to be surprised later!
  - Official Documents
  - Project Documents
  - Branch/Division/Group Standards
  - Common knowledge "do's and don'ts" (klabs.org)
  - Manufacturers seminars and applications notes

FPGA Design Specification

- The Requirements Doc tells you what you must do.
- The Design Spec *IS* the design that implements those requirements … How are you doing it?
- Includes:
  - Block diagrams of the FPGA architecture
  - Specific implementation for scrubbing, triple module redundancy, etc, to meet requirements.
  - Specific methods for the ability to inject errors in order to test mitigation or error correction techniques
  - Constraints on board-level implementation (critical pins for board routing, proximity to other devices, input slew-rate limitations, etc.)

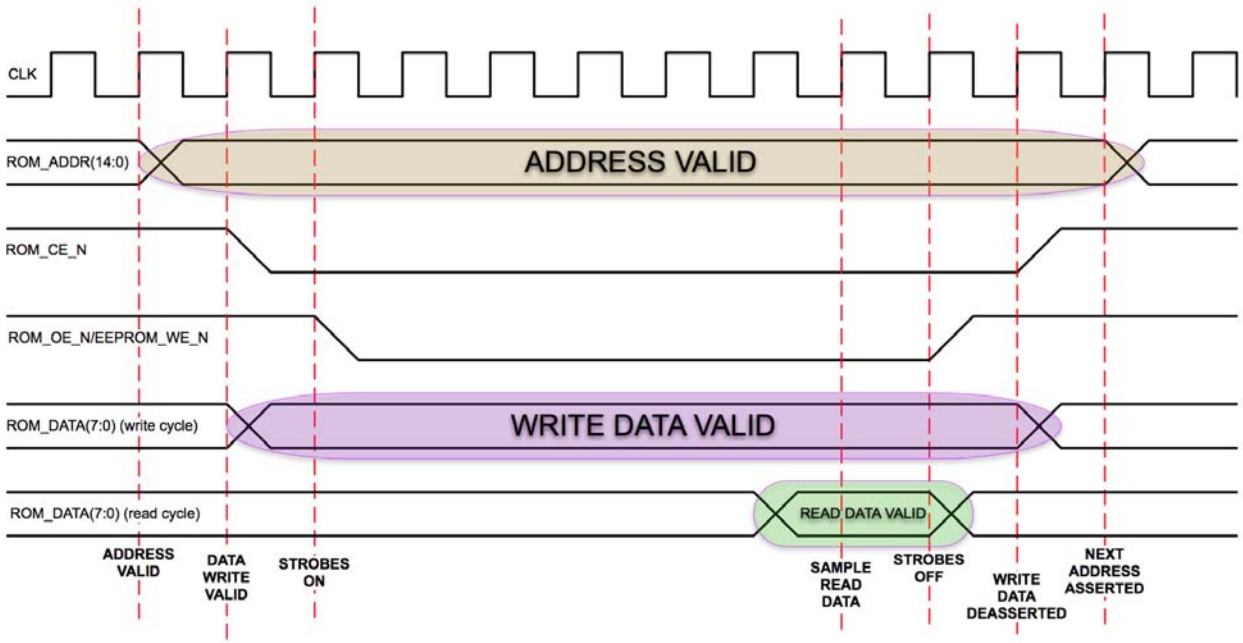
FPGA Design Spec – cont'd

- Includes:
  - A description of all interfaces including pin-out assignment
  - A functional description of the device
  - If there is a software interface to the FPGA, a Software User's Guide. Depending on the complexity of this interface a separate document may be required for this purpose.
  - Timing information (requirements on clocks from board, etc)
  - Place and Route Guidelines
  - A list of all files needed to create the FPGA (appendix)
  - A list of the data sheets and relevant application notes used to implement the design

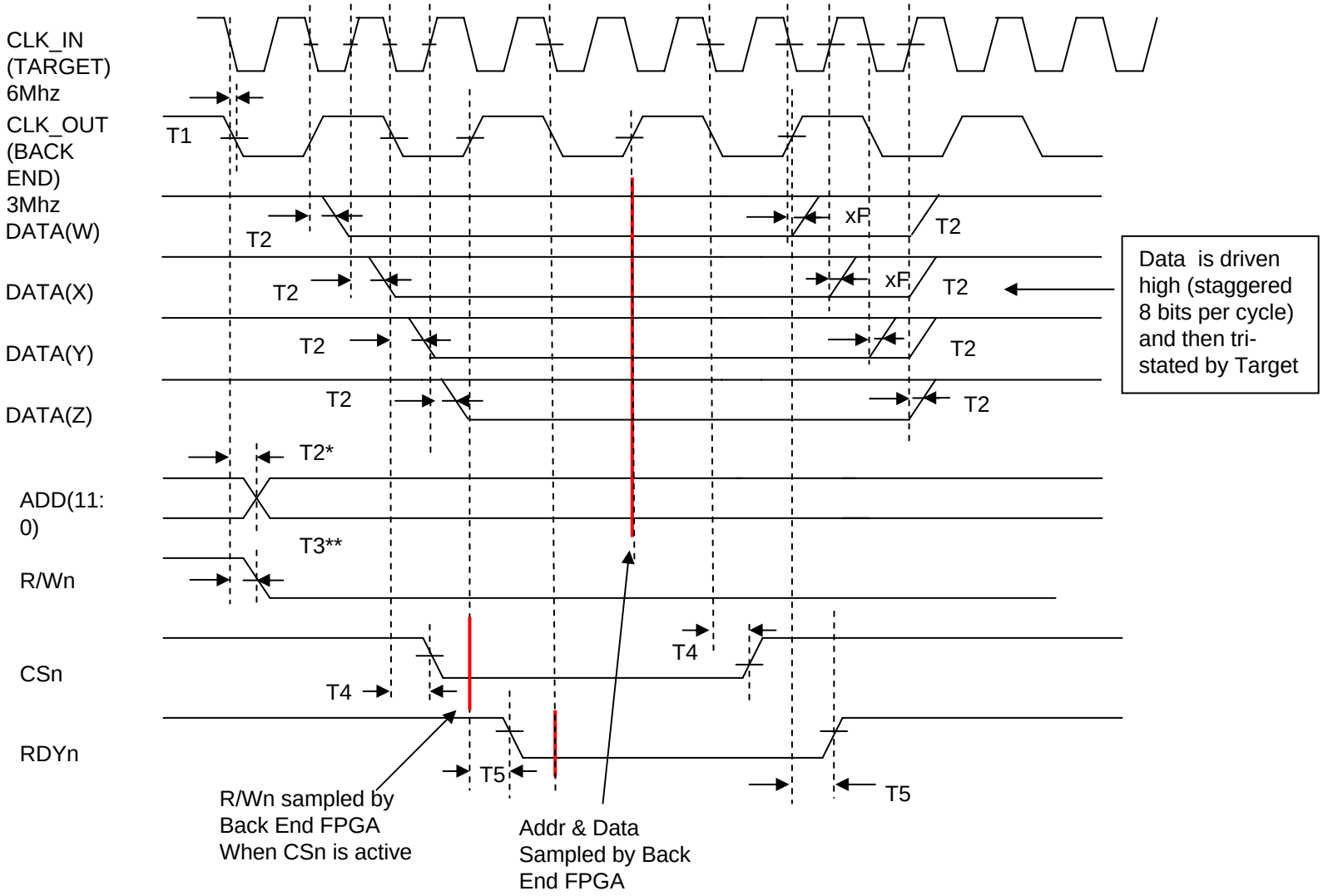
FPGA Specification

Timing Diagram

- Example Memory Access (Read)



Example Timing Diagram



Test Plan – More DOCS

- In order of decreasing adjustability/variability/coverage:
  - Simulation environment testing
  - Breadboard testing
    - Flight Software. Typically tests only normal modes, positive testing
    - Special Test Code. Plan early for in-situ debugging using special software
  - ETU testing
    - Temperature testing
    - Verification Suite and Flight Software
  - Flight Unit testing
    - Plan for observability of functions while in a chamber

Test Plan – More DOCS

- Functionality and Timing
  - Detailed instructions on how to test each function in the FPGA.
  - Details on how to test the mitigation or error correction techniques.
  - Link the tests to each item in the specification (which follows requirements).
  - Positive and Negative tests. Make sure it works how it is intended, and reacts safely to unintended inputs.
  - Number the tests in the document. These numbers are referred to in the testbench code.
- Write enough detail that someone else can do the work

Test Plan Example

- **7.7 N/S IOB Switch Raw Memory Tests**
  - 7.7.1 0x00/0xFF Test -
    a. write 0xff to all registers
    b. read/check 0xff from all registers
    c. write 0x00 to all registers
    d. read/check 0x00 from all registers
    e. repeat steps a through b.
  - 7.7.2 0xAA/0x55 Test -
    a. write 0xAA to all registers
    b. read/check 0xAA from all registers
    c. write 0x55 to all registers
    d. read/check 0x55 from all registers
    e. repeat steps a through b.

Implementation … Finally!

- This is usually the shortest portion of the design process!
- VHDL or Verilog preferred (try to stick with one language per chip if possible)
- Schematics for hierarchy if desired, or for small designs.
- Document your code properly. Inline documentation will help you later, and will help the next engineer if a personnel change is made in the middle of the Project.
- Write the purpose of each procedure or function.
- If you are using any tricks to achieve the design, use inline comments to explain why and how.

Version Control

- For ease of tracking changes.
- Can back up after 'oops' changes.
- For backup purposes (on another machine).
- RCS/SVN/CVS (tortoiseCVS/others).
- Follow project guidelines for entry into CM (i.e. CM at beginning of ETU build).

CM Copy should be ADB and Fuse File, plus DOCS and all source/sim code, plus constraint files … everything needed to re-create the chip.

Good Code Development

STARGAZER SYSTEMS INC

- Consider the electrical implications of your code.
- An FPGA design is a **hardware** implementation, not software.
- Some of the points on the following slides require action at the board level, outside the part.

Communicate issues with the board designer!

Style/Coding Standards

STARGAZER SYSTEMS INC

- They help you and help reviewers
  - Follow some conventions to allow you to recognize the function of signals by their name. Project defined coding standards are preferred, but self-made conventions are acceptable if allowed by the project. (as an example, refer to MMS PSE Coding Style Guidelines).
  - Document the standards, either in the header of the code itself, or refer to an existing document.
  - Use modular design to ease testability, readability, and simulation.

EXAMPLE – coding guides

STARGAZER SYSTEMS INC

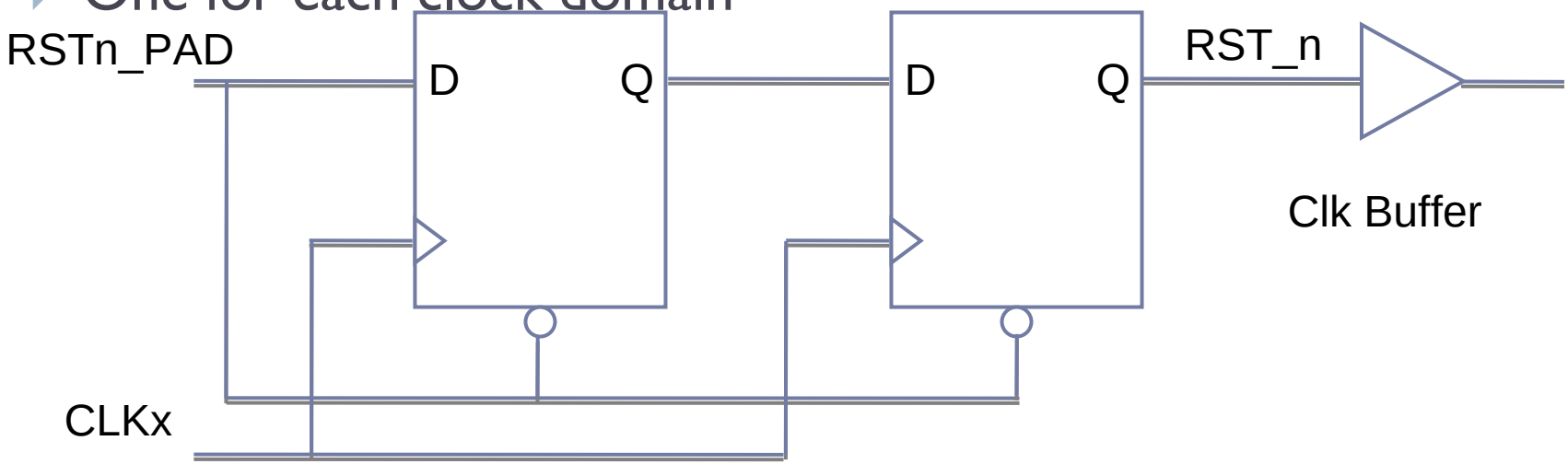
Signal Naming

From MMS PSE Guidelines

- Avoid carpal tunnel, avoid capitalization for all things except state machine type definitions
  - Corollary: mixing of lower case and upper case is acceptable if it visually helps to identify signals.(example: ctrl_UpperRam_n)
- Internal system level clocks should be named clk, clk_<freq>, clk_i2c, etc.
- I/O signals should be named pin_<I/O name> after the input pad or before the output pad.
  - In case you are using synthesis for pad insertion, use the I/O signal name.
  - I/O signal naming has precedence over all other signal name conventions
- Inversions of signals will be named <signal_name>_n.
  - Inversions of an inversion, will be named <signal_name>_n_n or <signal_name> if said name does not exist already.
- Outputs of an entity used internally before assigned to an output shall be named int_<output_signal_name>.
- Signals that are synchronized internally shall be named sync_<signal_name>.
  - In cases of conflict or code clarity, presync_<signal_name> can be used before synchronization.
- Internal reset should be named rst, rst_<block_name>, rst_<reset_group>

Code - Reset

- Reset Practices
  - Typically asynchronously applied and synchronously removed
  - One for each clock domain



Refer to MAPLD04 *"Reset Circuit Topologies"* by Melanie Berg for the pros and cons of different implementations of reset

Code - Timing

- Timing Practices
  - Synchronous design
    - SET AND RESET PINS ARE FOR INITIALIZATION ONLY!
  - Asynchronous inputs (synchronize them)
    - Synch them to new clock domain to minimize metastable conditions.
  - Clock Domain Crossings (CDCs)
    - Treat the same as other asynchronous inputs unless the two clocks are divided from the same source and the relationship can be guaranteed to meet setup/hold timing

Code – Error Handling

STARGAZER SYSTEMS INC

- Error Handling
  - Design for return to safe state if the unexpected occurs in inputs … and EXPECT THE UNEXPECTED
  - Consider Error-handling in every circuit … "what happens if…" and design the circuit to get to a safe state and continue. This is flight hardware, usually unmanned, and can't wait for user intervention.

Implementation– Electrical

- Power Related
  - Communicate supply requirements with board/box people. May need to run power estimation to get necessary info.
  - Proper power supply decoupling
  - Power supply sequencing (see part data sheet or app notes)
  - Distribute simultaneously switching output pins around periphery to avoid overloading supplies and causing ground-bounce. *Be wary of manufacturer specs, which may give a max # with an unacceptable droop in supply voltage*

Implementation– Electrical

- Interfacing
  - Verify correct I/O levels are being used. Choose best I/O drivers.
  - Use the slowest edge rates possible given the design constraints.
  - Handle power-up/power-down where I/O may not be valid, to prevent an invalid state.
  - Don't allow bus contention.
  - Don't allow tri-state buses to float in the center region.

Implementation– Electrical

- Interfacing – con't
  - Input slew rate specification must be met.
  - Perform signal integrity analysis of all the interfaces to determine the need for external impedance matching termination.
  - De-bounce and de-glitch interfaces from mechanical devices.

Design - Testability

- Testability
  - Plan your design with testing in mind and incorporate the resources needed to facilitate it. Consider observability as you implement your design. Think about how you will debug the circuit while the part is on the (BB/ETU/Flight) board.
  - Reserve test pins as test-only pins. Buffer the signals provided to the test pins from the internal circuitry.

Verification

- Develop Test Code – An independent FPGA tester is preferred, but is not mandatory. The following guidelines should be observed:
  - Follow the test sequence identified in the test procedure. Refer to the assigned test number for each test.
  - Use Self-Checking/Documenting test-benches.
  - Analyze code coverage of simulation and test vectors.
  - Automate tests using scripts for repeatability and unattended runs. (Bash/CSH/TCL/Perl/Python, etc.)

Simulate

STARGAZER SYSTEMS INC

- Simulate Functional code using test-benches
  - Review tests – Have others look at them for completeness
  - Review waveforms for sanity check.
  - Capture I/O to other chips/systems
  - Share with interfacing design engineers.
  - Take the time to discuss results at this point, it can save lots of hassles later.
  - Chase down all warnings and errors reported by simulator.
    - Understand why they are there.
    - Document any decision to ignore them.

Synthesis

- Synthesize the design
  - Use flight equivalent part from the beginning.
  - Set timing constraints in synthesis using constraint files.
  - Specify loading for each pin by reviewing schematics and specs for each interfacing part.
  - Set critical paths if pushing part speed in any particular path.
    - Begins familiarity with critical paths.
  - Using constraint files assists with self-documenting design.
  - Review output files and logs for synthesis.
    - Understand all warnings and if you decide to ignore any, document the reason why.

Place and Route

- Typically done on vendor tool
  - Set timing constraints. Document and archive constraints files for reproducibility and review.
  - Double-check false paths / multi-clock paths.
  - Set proper flight part
    - Package
    - Temp range (MIL range suggested to ensure sufficient timing margin)
    - Voltages (Core, I/O)
    - Radiation level (use max unless/until you need to borrow from margin for timing closure)
  - Fix pin locations – Can be done after first run for best timing if project schedule allows (rarely).

Place and Route

- Run Place and Route.
  - Export Min-Typ-Max Standard Delay Format (SDF) files for simulation
    - Min/Max delays will be contained within this file, ranging from the best case to the worst case.
  - Search through the netlist for issues.
    - For example, search for flip-flops with both asynchronous preset and clear. These should not be used, and point to interpretation issues in the code. Search for latches; may be unintended result of coding style.

Post-Route Verification

- This is where MOST of the time is spent
  - Review all logs from vendor tools for errors, warnings, and notes.
    - Clear the error/warning conditions and re-run until clean, or document why you are ignoring them
  - Review timing report to verify that the longest routes make sense.

Post-Route Verification

STARGAZER SYSTEMS INC

- Back-Annotated Simulations
  - Re-run simulations that were run on RTL (golden vectors)
  - Add tests to get coverage up … use coverage tools
  - Run at least these two conditions:
    - Best Case beginning of life (BOL) sim: (Max Voltage, Min Temp), Zero Radiation, Highest Speed.
    - Worst Case EOL sim: (Min Voltage, Max Temp), Max Radiation, Lowest Speed (process).
  - Read every warning and error the tools generate. If you decide to ignore a warning, document the reason.
- Verify that timing and functionality are both met.

Post-Route Verification

- Timing Analysis
  - Use the vendor's Static Timing Analysis (STA) Tool
  - Include delays to/from pads on board
  - Consider clock source and delays
  - Include loading on outputs
  - Get min/max data for any device interfacing with FPGA
  - Enter all constraints into the STA tool
    - Don't just look at the unconstrained clock-to-clock speed and declare success!
  - Set clock constraints at 20% higher than actual to ensure required timing margin

Static Timing Analysis

STARGAZER SYSTEMS INC

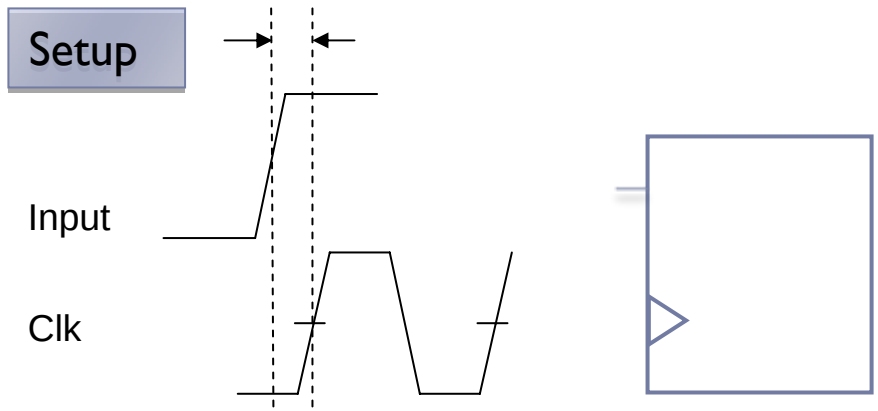
- Commonly performed using vendor-specific tools
  - Actel Smarttime
  - Altera Quartus (Synopsis PrimeTime)
  - Xilinx Timing Analyzer
- Why use this … it's too much work to set up. I already do back-annotated sims!
  - Static Timing Analysis tools, once set up properly, check ALL paths in the design.
  - Simulations are only as good as the test vectors … tough to achieve 100% coverage. Mistakes in vectors may cause us to miss timing paths.

Static Timing Analysis

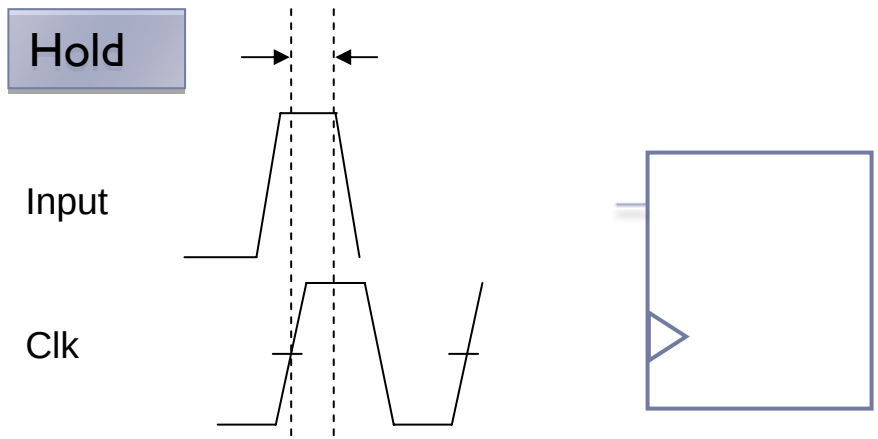
STARGAZER SYSTEMS INC

- What is Static Timing Analysis?
  - Primarily, an automatic check of setup and hold times on our designs for every clocked path.
  - Also checks recovery and removal times for asynchronous inputs to flops.
- Requires adequate constraints to be useful

Setup and Hold



Input stable before clock edge



Input remains stable After clock edge

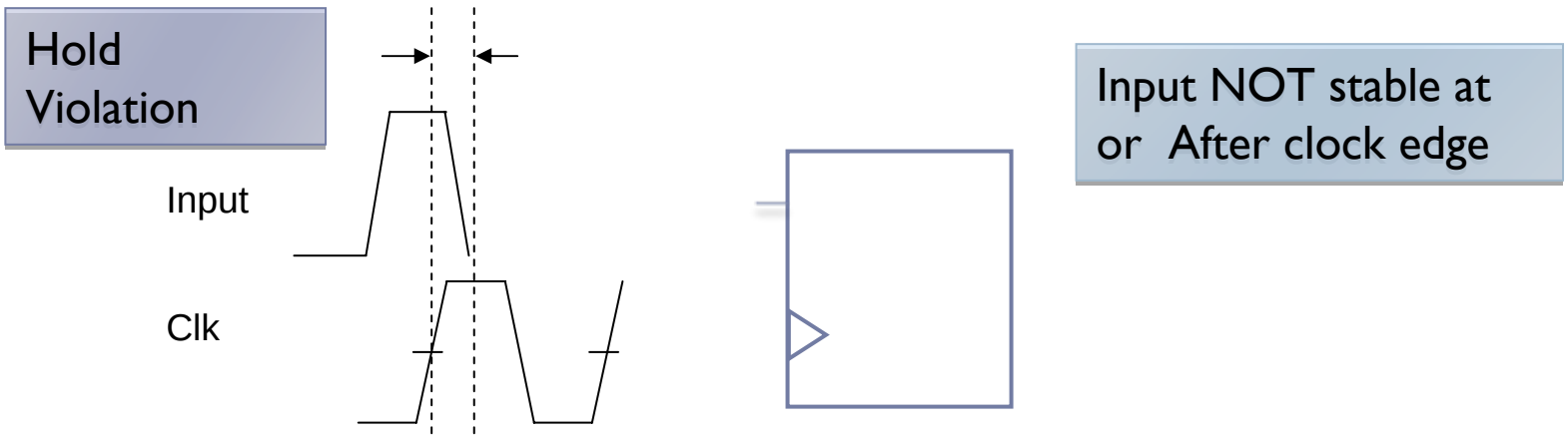
Static Timing Analysis

STARGAZER SYSTEMS INC

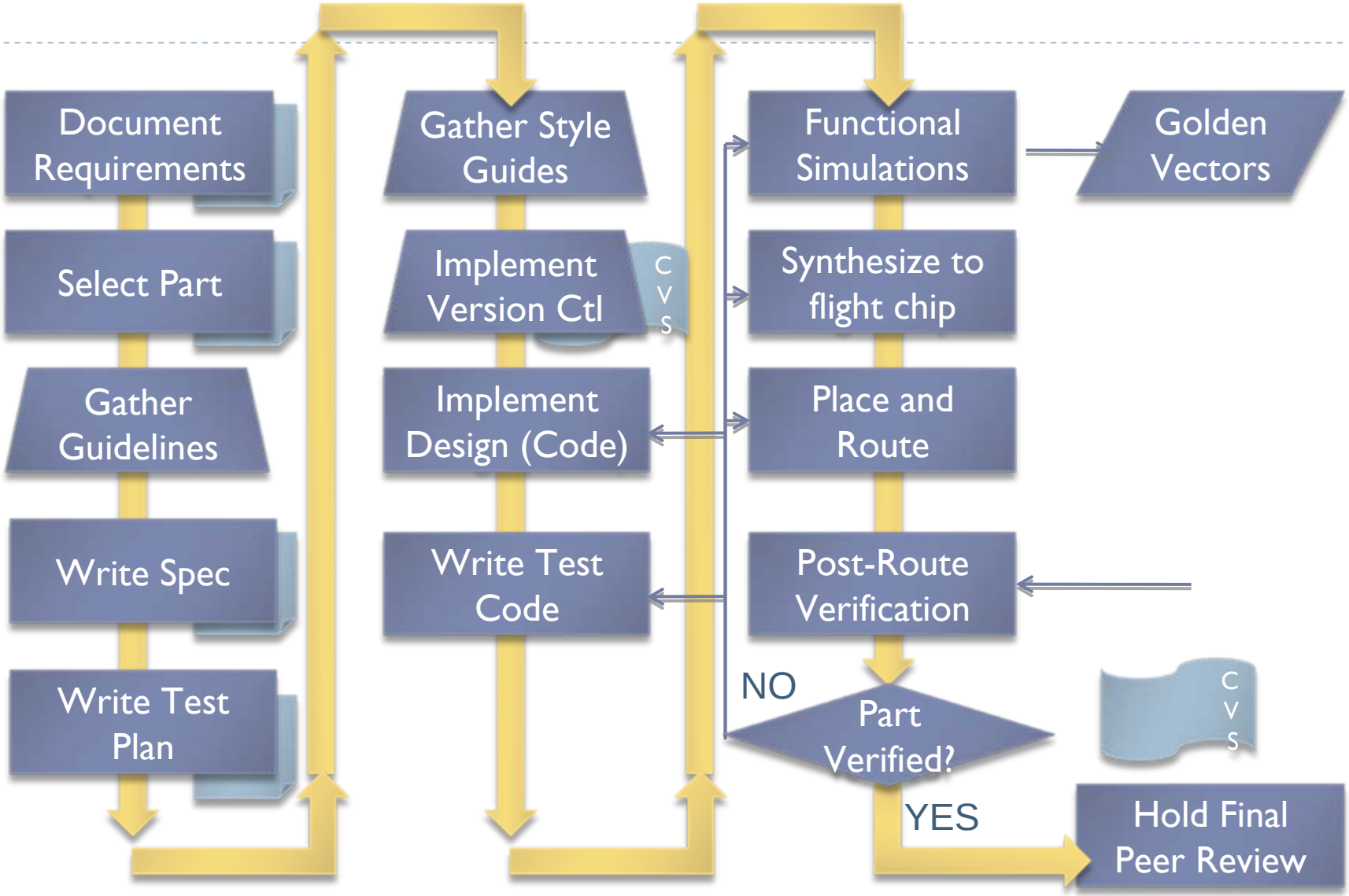
- What are constraints, which do I need?
  - Capacitance of output pins
  - External path times into and out of the FPGA (including board delay)
  - Relationships between clocks
  - Duty cycle of clocks (if tool does not allow duty cycle adjustment, use the shortest time as ½ the period)
- Don't I only need to analyze 'critical' signals?
  - Yes!
  - Critical Signal is defined as anything that needs to work!

Static Timing Analysis

- My design is really slow … why do I need to check timing?
  - Clock speed doesn't matter …
  - It the relative path timing that matters
  - FPGAs often fail at cold (best case). This is usually due to hold-time violations.



Development Flow



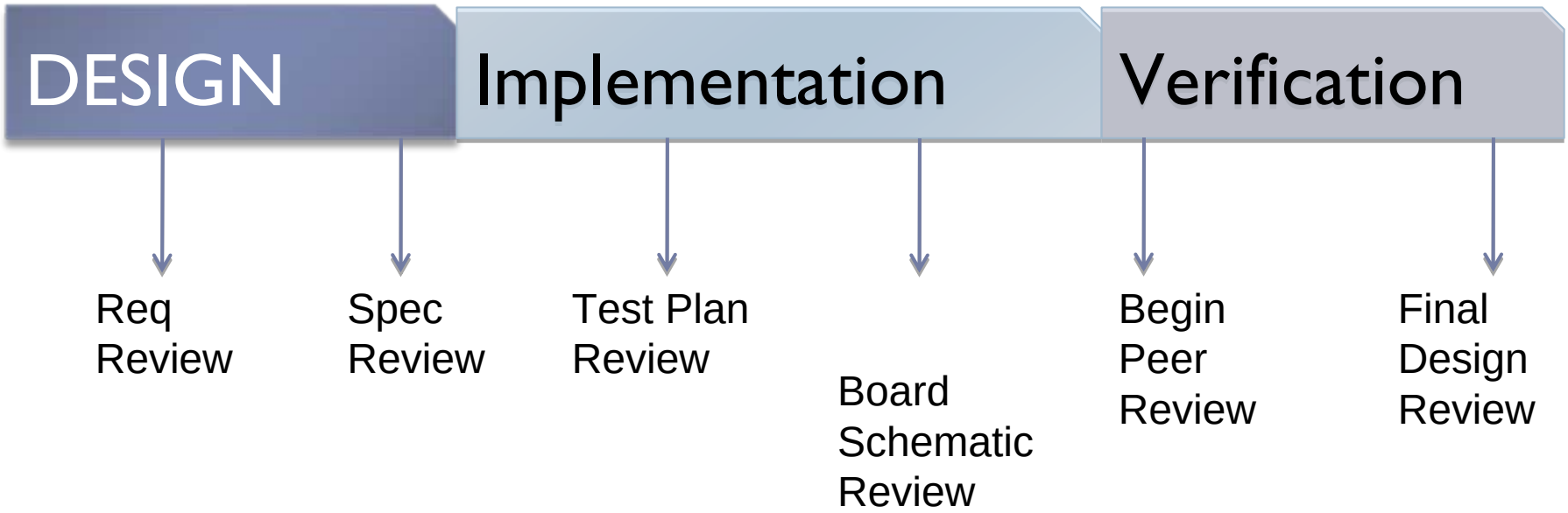
Design Review !!!

STARGAZER SYSTEMS INC

- Yes, the title of this presentation had '*Review*' in it.
  - Following the preceeding steps make the design review easy!
  - Just show the results of those steps … show how you convinced yourself that the design is ready.

Reviews

- Hold reviews early … and often!



Peer Review

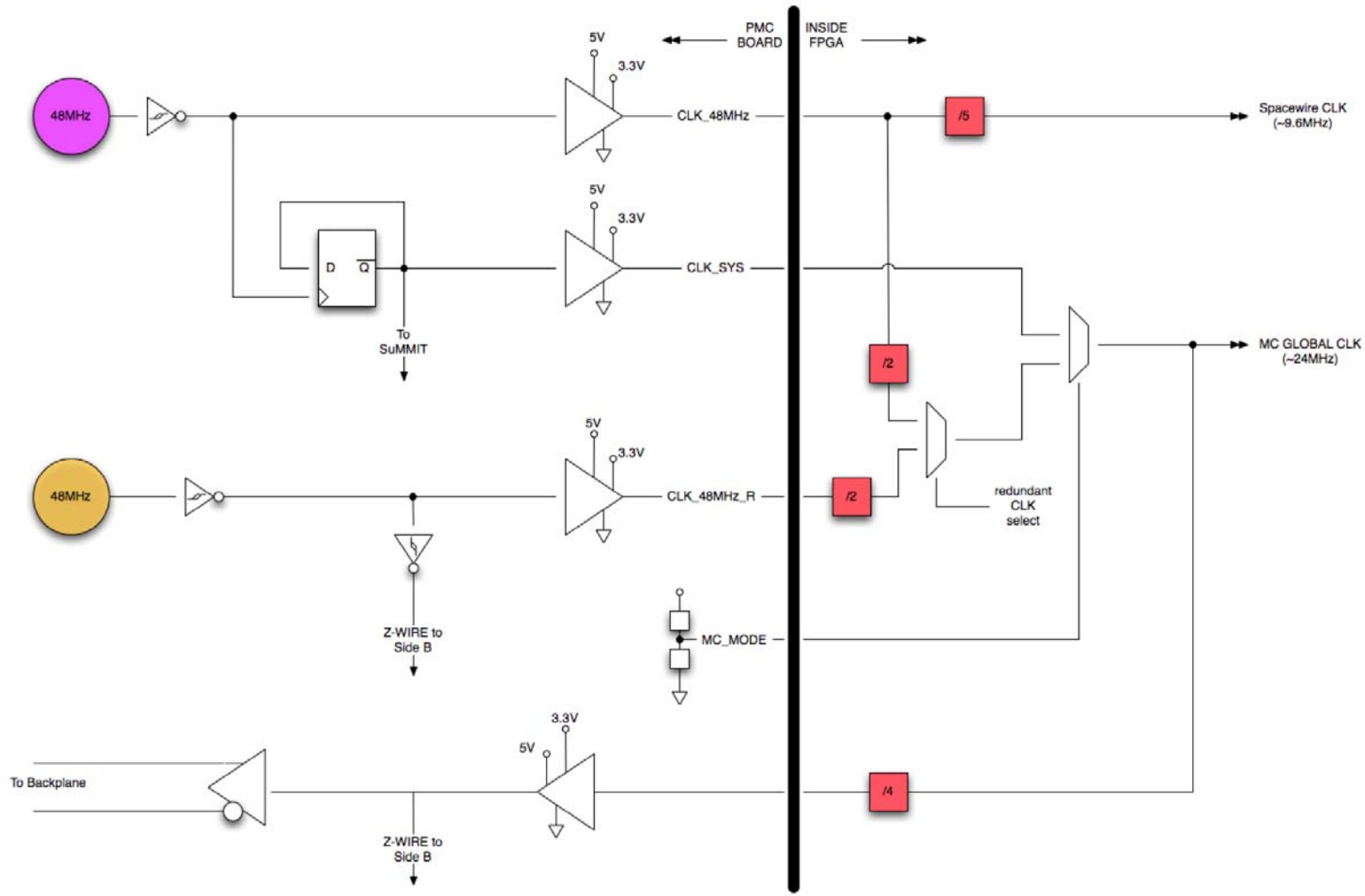
- The peer review is the most important review of the design process.
- Goals: demonstrate to the review panel:
  - the design meets all its requirements
  - has been designed properly
  - all analyses and simulations have been performed to verify that there are adequate margins so that:
    - it will work in the intended application
    - over the environmental range
    - for the life of the mission

Peer Review – Initial Mtg

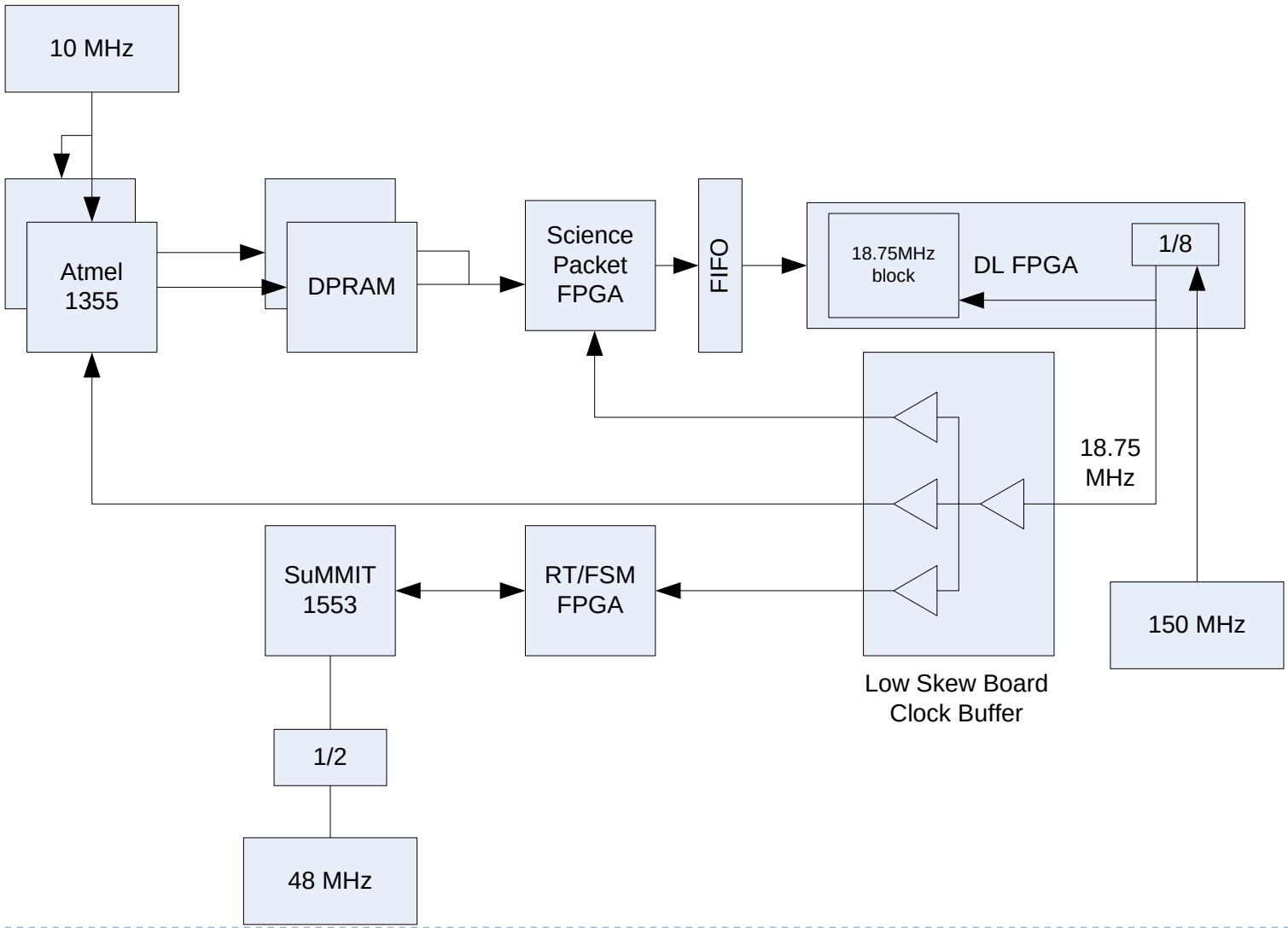
STARGAZER SYSTEMS INC

- Implementation discussion:
  - Pinouts
  - I/O Selection (Type/Drive Strength/Slew Rate)
  - External clocks (draw **clock tree** for each oscillator)
  - Clocking (rates, routing resources, distribution)
  - Reset (source, location, duration, recovery)
  - Utilization percentages of combinatorial and sequential modules and memory
- Test Plan – Walk through test procedure document and test sequence flowchart.

Clock Tree

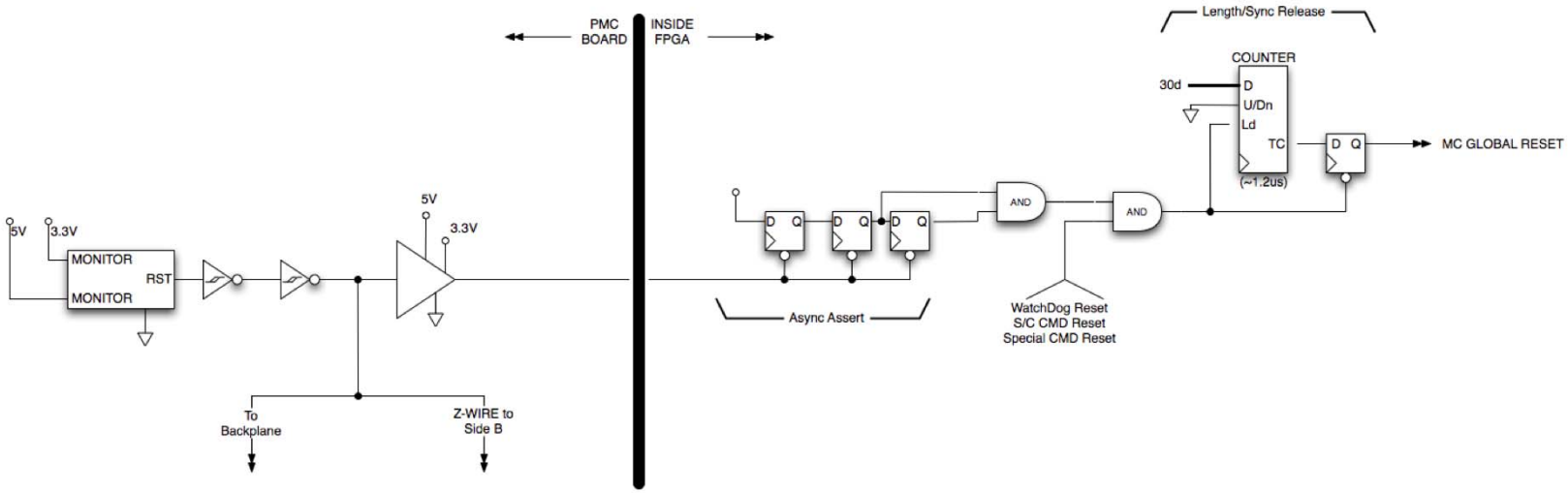

PMC BOARD
INSIDE FPGA
48MHz
5V
3.3V
CLK_48MHz
/5
Spacewire CLK (~9.6MHz)
CLK_SYS
D
Q
To SuMMIT
/2
MC GLOBAL CLK (~24MHz)
48MHz
CLK_48MHz_R
/2
redundant CLK select
Z-WIRE to Side B
MC_MODE
To Backplane
/4
Z-WIRE to Side B

Clock Tree


10 MHz
Atmel 1355
DPRAM
Science Packet FPGA
FIFO
18.75MHz block
DL FPGA
1/8
18.75 MHz
SuMMIT 1553
RT/FSM FPGA
150 MHz
Low Skew Board Clock Buffer
1/2
48 MHz

Reset Diagram

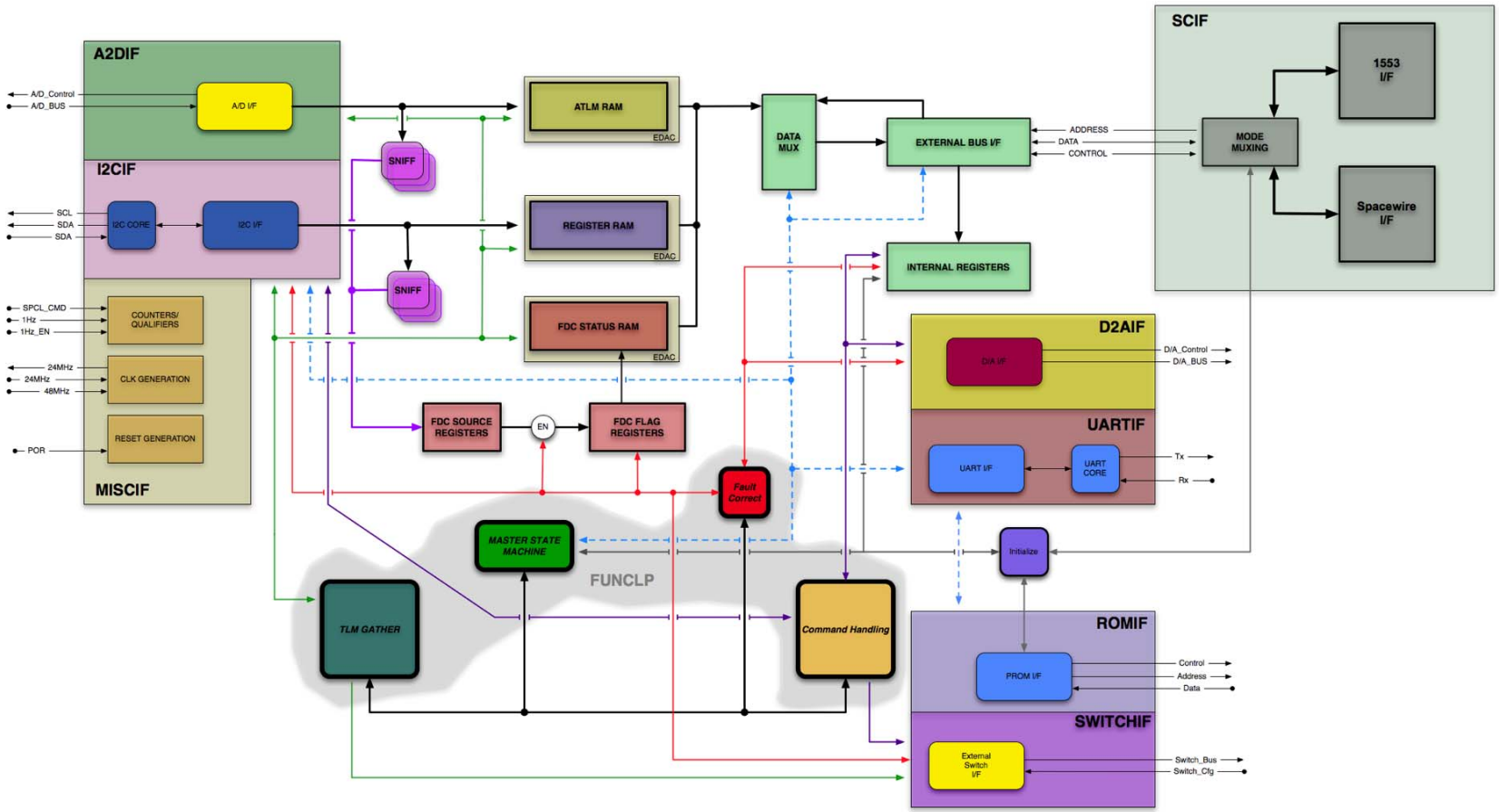
- Shows board source(s) and internal distribution.



Peer Review Materials

- Requirements Review
- Design Overview – Include context drawings or schematics
  - Include System/Box/Board-Level Diagrams, whatever is needed to give 'cold' reviewers a feel for what the FPGA does
  - Chip-level block diagrams (discussed earlier)
  - Include State Machine Design Philosophy and diagrams
- Interface Descriptions. Discuss timing/ functionality of external interfaces

Block Diagrams



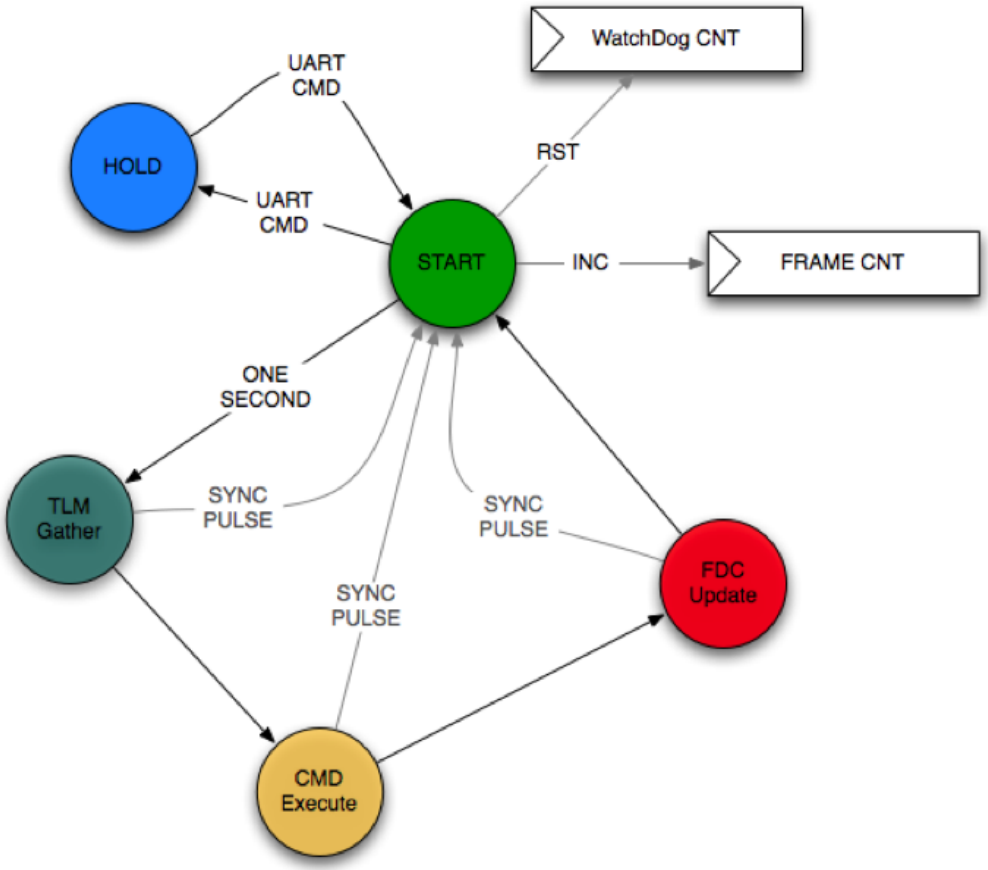
Peer Review Materials

STARGAZER SYSTEMS INC

- Code Structure – include block diagrams of each major block and how they relate to each other. Very nice to have block names match code entity names.
- Code Walkthrough – Discuss:
  - Structure/Organization – Include Directory Tree
  - Reset handling
  - How illegal states in each FSM are handled
  - Use of global vs. routed clock signals
  - Clock boundary signal resynchronization (CDC)

State Machine Diagrams

- Simple diagrams help explain functionality



Peer Review

STARGAZER SYSTEMS INC

- Present results:
  - Simulation results
  - Timing Analysis. Show how margins are met (20% margin)
  - Interface Analysis (drive strengths, I/O levels, power supply levels, sampling of input signals, no bus left floating)
  - Board Implementation (power supply decoupling, signal integrity analysis, routing)

Review Checklist

- No Checklist is comprehensive!
- Designers, complete the checklist before peer review
- Don't ignore things because they are not on the checklist
- Use your natural curiosity as an engineer to say "What If"? … then get the answers!

Review Checklist

- Requirements are met
- Simulations successfully completed for best and worst case conditions:
  - Best Case (Lowest Temperature, Highest Operating Voltage, Zero Radiation, Best Process)
  - Worst Case (Highest Temperature, Lowest Operating Voltage, Maximum Radiation, Slowest Process)

  (These are both usually contained in the one SDF generated)
- Simulation adequately tests design (tests all sections of code and circuitry)

Review Checklist

- Timing analysis completed successfully
- Resets handled properly
- Clocking handled properly
- All clock-domain crossings are handled properly
- Asynchronous inputs are filtered for meta-stability issues

Review Checklist

- I/Os properly selected (SSO, levels, slew rates, etc.)
- Relevant manufacturer recommendations and app notes followed
- Asynchronous circuits clearly identified and analyzed for robust operation
- Board-level issues addressed (decoupling, routing, signal integrity, etc.)
- Margins have been demonstrated and are acceptable (timing, utilization)
- Reviewer issues satisfied

Conclusion

- The road to a smooth design review is paved with a well-thought-out development plan.
- Design on paper, then implement in hardware.
- Plan your tests as you create the design
  - Co-develop the test plan and part spec

- Digital Design is simple …
  - Functionality
  - Timing
  - Electrical

Discussion – Reconfig

- Most of my recent experience has been with One-Time-Programmable (OTP) devices.
- Are there any differences in design flow or review prep for reprogrammable devices?
  - I say no … the same issues apply to ASICs/OTP/PLD/Reconfigurable …
  - What does the group here say?

Discussion

- What did I miss?

Follow-Up Discussions

- I've set up a blog at http://www.StargazerSystems.com/FPGAblog
- Please write comments to this presentation or other ideas about the design and review process
- Use it as a place to ask questions of your colleagues.

Brian Smith

301-358-2582

Brian.Smith@StargazerSystems.com