RTAX-S Design Checklist

Minh Uyen Nguyen
Applications Engineer
ACTEL Corporation

Topic

- Introduction
- Preparation and Creation of an RTAX-S Design
- Synthesis
- I/O Setting and Placement
- Board Level Consideration
- Design Optimization
- Timing Analysis in SmartTime
- FloorPlanning Tips

Introduction

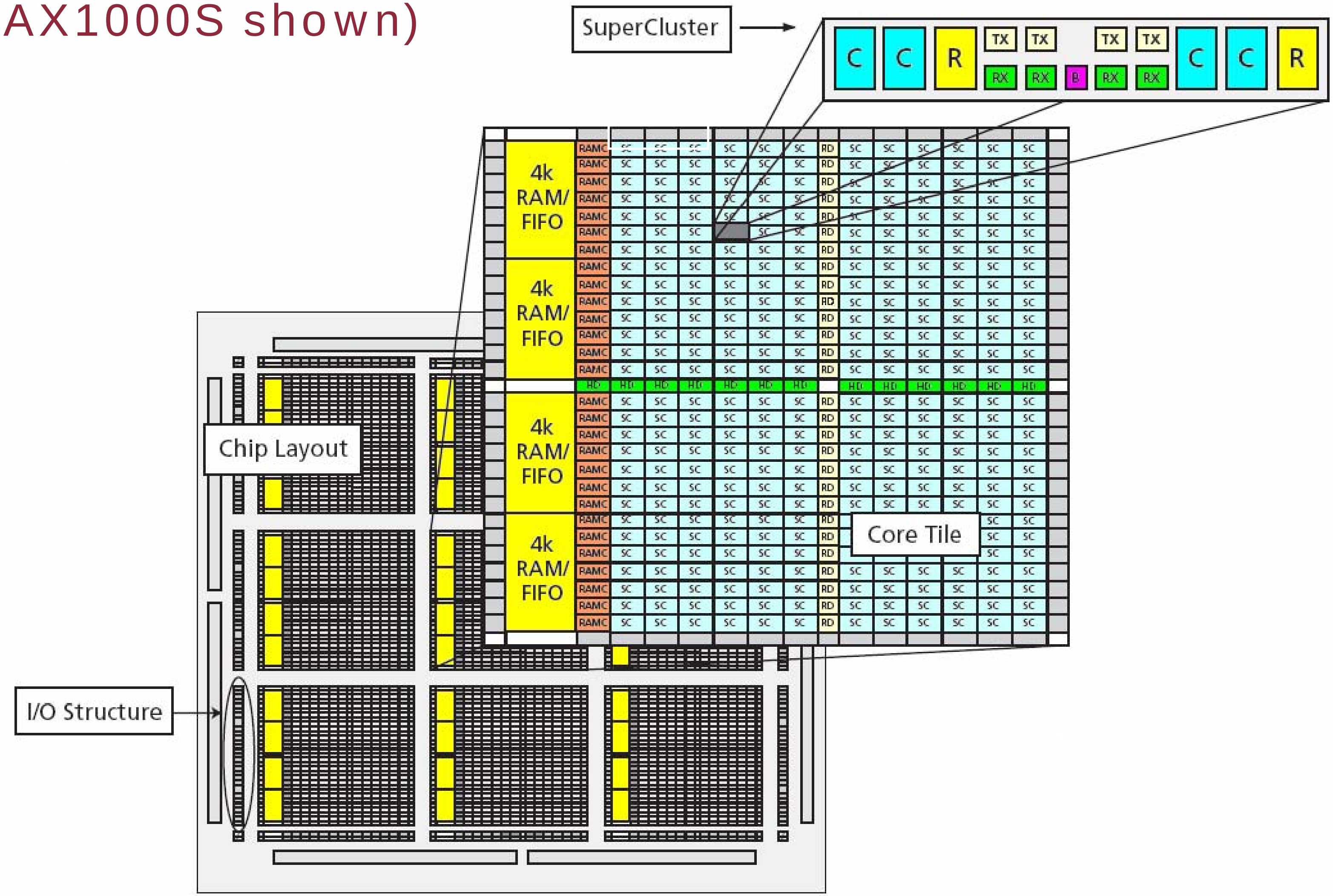
RTAX-S Family Overview

- 0.15µm, 7-layer-metal CMOS with Antifuse Fabbed at UMC
- High Performance Architecture, Enhanced to Meet Radiation Requirements of Space Flight
  - 350 MHz System Performance
  - Up to 840 I/Os
  - Up to 540 kbit User Memory
  - Memory Error Detection and Correct (EDAC) using Soft IP
  - Hardware TMR of All Registers
    - Logic Array
    - I/O Registers – Input, Output, and Enable
  - Enhanced C-cell with Carry Chain for Improved Arithmetic Performance
  - Power-on-reset (POR) Circuit
  - Clock Trees
    - Dedicated Clock Drivers
    - Low-skew Clock Tracks
  - Charge Pump

RTAX-S Family Features

Device Layout

0.15µ, 7-Layer-Metal CMOS Process
(RTAX1000S shown)



Design Challenge with RTAX-S

Design Challenge

- High Reliability Applications Allow ZERO Error
- High Utilization Requirement
- High Performance Requirement
- Fixed Pinout due to Interposer Mapping
- Design Tool Limitation

Purpose of RTAX-S Design Checklist

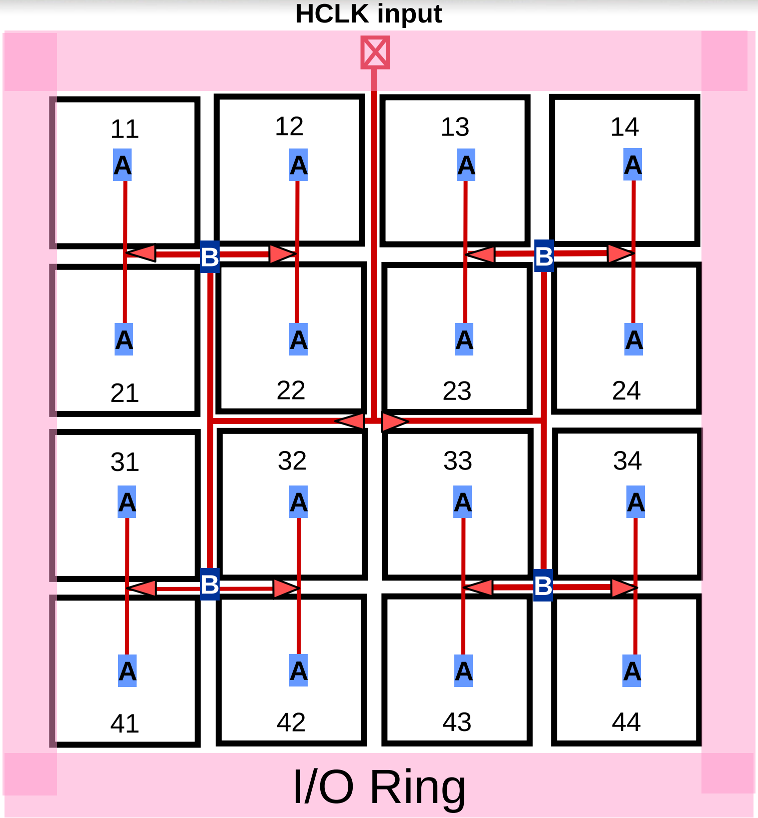
- Help designer get the most out of the RTAX-S product's capabilities and Build the design correctly the first time
- Provide Design Guidelines and Criteria to design and evaluate an RTAX-S design
- List items that are often overlooked at the system-level and during the design process using Actel's design tools
- List items that should be considered for implementing a successful, reliable and robust design.

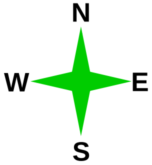
Preparation and Creation of an RTAX-S Design

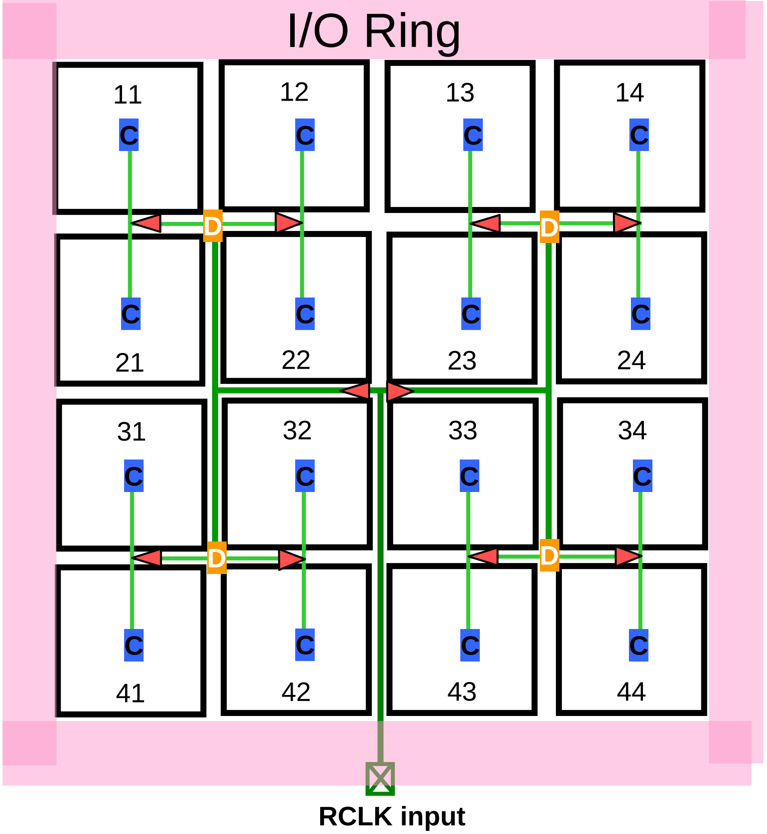
Design Preparation and Creation

- Always use the latest Software Versions if possible
- Consider Block Flow Advantages
  - **Timing Closure - Focus on critical designer block to ensure timing meets requirements**
  - **Efficiency - Re-use same designer block in multiple designs**
  - **Predictability - Change unrelated parts of design and the key designer block remains unchanged**
  - **Refer to Libero User Guide for more information http://www.actel.com/documents/libero_ug.pdf**
- Improve RTL Coding Style
  - **Reduce number of logic level to improve performance**
    - **Investigate Adding Pipeline Stages or Explicit Register Balancing**
  - **Minimize number of arithmetic units by investigating re-writing operations**
  - **Dealing with many buses:**
    - **Revisit Design Architecture**
    - **Implement Bus Sharing, if Possible**
    - **Bury Some of the Buses in a Larger Component / Module**

RTAX-S
Core-level Clock Trees







H-Clock delays

- All tile inputs see same clock delay
- A > B

R-Clock delays

- All tile inputs see same clock delay
- C > D

RTAX-S Global Clock Network

- 4 Hard-Wired Clocks (HCLK) directly drive all clock inputs
  - **Minimal clock skew**
  - **Immune to Internal Hold Violations by Construction**
  - **Put All Clocks on HCLK if possible to save CLK for others which require more physical flexibility than HCLK can provide**
  - **Note that HCLK cannot be output off the chip**
- 4 Routed Clocks (CLK) drive CLK, PRE and CLR inputs of R-Cells and any input of C-Cells
  - **Immune to Internal Hold Violations by Construction when CLK Drives R-Cells**
  - **Put Reset and High Fanout Nets on CLK**
- Global Clock Networks Are SEU Resistant
- Global Clock Networks Are *Segmented*
  - **The maximum number of clock segments can be found in AC310**

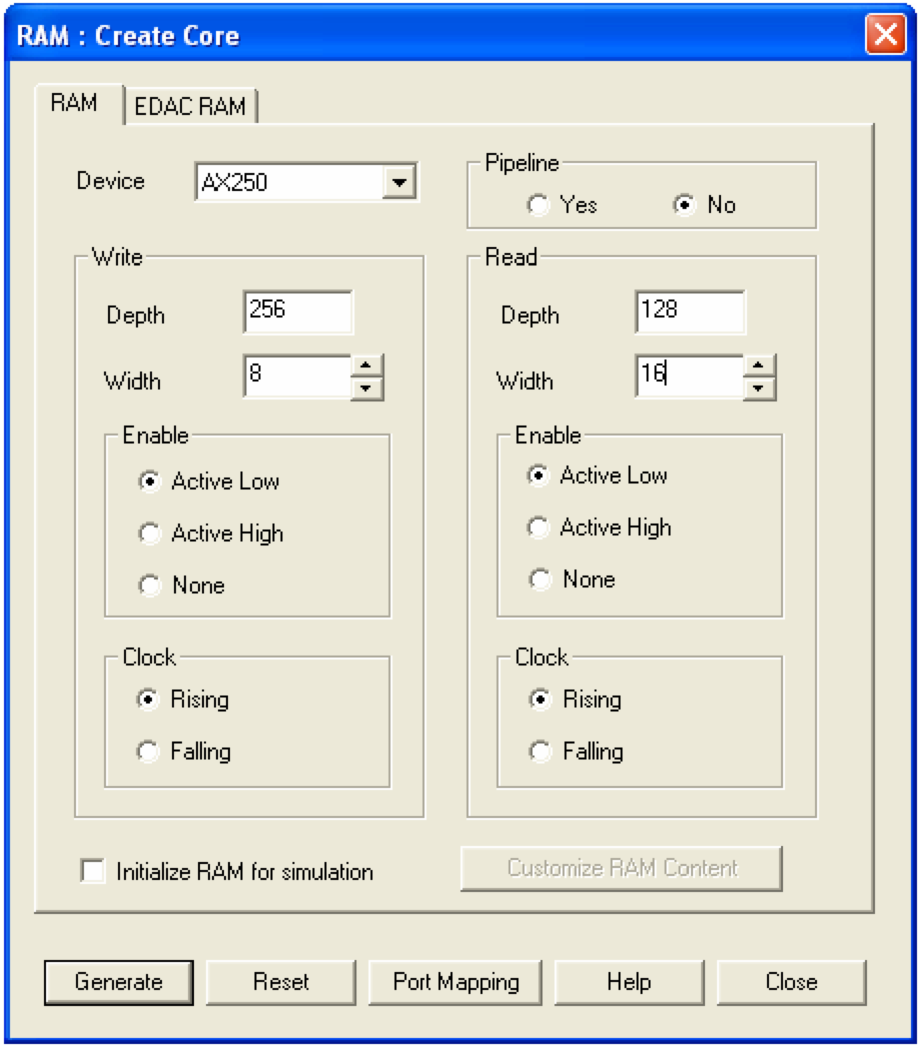
    http://www.actel.com/documents/RTAXS_Clocking_AN.pdf

Clock Buffer Instantiation

- RTAX-S Clock Buffer
  - Generic: HCLKBUF, HCLKINT, CLKBUF, CLKINT
  - I/O standard specific: CLKBUF_LVCMOS25, HCLKBUF_LVDS, etc..
- Recommend to Instantiate Clock Buffers in HDL Source Files Before Synthesis
  - Better design practice for high reliability applications
  - Synplify has difficulty recognizing the clock MUXes and clocks in the design. Do not rely on Synplify to infer clock buffer
  - It will be difficult to specify which clock network to use in later design stages using Actel design tool
- Use clock buffer for high fanout nets which are truly global
  - Nets which go to several places all over the chip and cannot be confined to a small part of the chip

RAM/FIFO Instantiation

- SmartGen core found under View\ Toolbars\ Catalog\ Memory&Controllers
- RAM Pipeline Register is not SEU Immune
  - **Not recommended for critical applications**
- Recommend *FIFO Controller With and Without Memory*
  - **Soft FIFO controllers built with TMR registers**
- *FIFO-Synchronous*
  - **Built-in FIFO controller is not SEU Immune**
  - **Not Recommended for Critical Applications**



RAM : Create Core

RAM | EDAC RAM

Device: AX250

Pipeline: ○ Yes ● No

Write
- Depth: 256
- Width: 8
- Enable: ● Active Low ○ Active High ○ None
- Clock: ● Rising ○ Falling

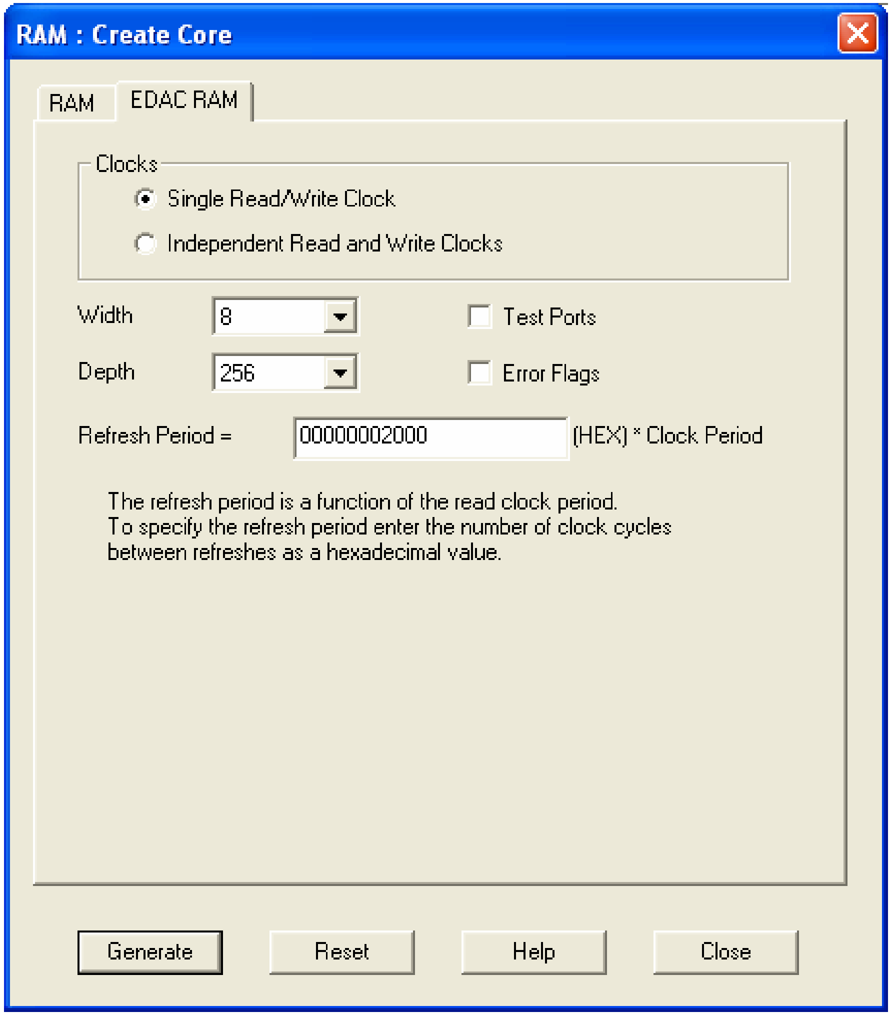
Read
- Depth: 128
- Width: 16
- Enable: ● Active Low ○ Active High ○ None
- Clock: ● Rising ○ Falling

□ Initialize RAM for simulation

EDAC RAM Instantiation

- Recommended Minimum Refresh Period
  - Allow Refresh State Machine 10 Cycles per Address to account for external accesses
  - Clock Cycles = 10 x RAM_DEPTH
  - Example: 256x8 RAM

    256 x 10 = 2560d = A00h
- Do not access EDAC RAM while scrubber is on
- Ensure No Simultaneous External Read/Write to Same Address
  - Check sample code to detect simultaneous read and write to the same address in AC319



RAM : Create Core

RAM | EDAC RAM

Clocks

● Single Read/Write Clock

○ Independent Read and Write Clocks

Width: 8

Depth: 256

□ Test Ports

□ Error Flags

Refresh Period = 00000002000 (HEX) * Clock Period

The refresh period is a function of the read clock period.
To specify the refresh period enter the number of clock cycles between refreshes as a hexadecimal value.

http://www.actel.com/documents/EDAC_AN.pdf

Synthesis

Recommended Fanout Setting

FPGA Utilization	Max Fanout Setting for Synplify
<50%	10
50%-75%	18
75%-90%	24
90%-100%	32

- Use this table as a guideline not as a rule, because every design has its own connectivity/congestion profile.

Synthesis Constraint SDC File

- Set Clock Constraints with ~10% margin
- Set syn_noclockbuf attribute to 1
  - Prevent Synplify from putting any unintentional clock buffers into the synthesized netlist, especially on high fanout nets
- Set syn_maxfan with a very large number to avoid buffer insertion
  - This step is for analysis purpose only. After running Place and Route, use ChipPlanner to have a feel of cell placement and high fanout nets
  - Help revealing true fanout of all the nets in the design
  - Exclude nets intended for Local Clocks from Buffering
- Set Multicycle and False Path constraints
  - Synplify save resources and focus effort on actual critical paths
- Actel synthesis Attribute and Directive
  - http://www.actel.com/documents/synplify_ref_ug.pdf

Synthesis Compile Report

- Examine the Report after Synthesis
  - Check utilization of R-cells, C-cells, clock buffers, and block RAMs
  - Check whether clock constraints have been applied correctly
    - Sometimes Synplify does not find the specified clock nets in the SDC file, it will apply the default clock 100MHz
- Synthesis Estimated Timing and Logic Utilization is often inaccurate
  - Need to check Designer Compile Report for actual device utilization
  - Source of inaccuracy: Designer Compile optimizes the synthesized netlist some more using netlist combiner

I/O Setting and Placement

I/O Setting and Placement

- Keep in mind dataflow to/from the chip and design topology when assigning I/O pins
  - Use dedicated global pins for clock/reset
  - Put clock/reset pins close to clock buffer on chip to reduce the clock insertion delays significantly
- Check die pad proximity to improve chanel-chanel skew using ChipPlanner
- Simultaneous Switching Output (SSO)
  - Minimal SSO Effects on RTAX-S
  - Here are some tips to prevent SSO:
    - Identify potential SSOs and spread them around the die (not just the package), use ChipPlanner to assign die pad
    - Avoid placement of asynchronous pins near SSOs
    - Place SSOs away from clock pins/traces.
    - Use low slew outputs when possible
  - For more information, refer to AC263 http://www.actel.com/documents/SSN_AN.pdf

Board Level Consideration

Board Level Consideration

- Require very good core supply layout, I/O supply layout and good board design overall
  - Large die which requires sufficient decoupling capacitors
  - Large number of simultaneous switching registers due to TMR
  - For SSO board level recommendation, refer to AC263 http://www.actel.com/documents/SSN_AN.pdf
  - For more information, refer to AC276 http://www.actel.com/documents/BoardLevelCons_AN.pdf
- Regulate the supply to VREF to ensure minimal fluctuation from the typical value
- Recommended Connections for Prototyping with Axcelerator:
  - VCCPLx: Connect to 1.5V
  - VCOMPLx: Leave floating
- Consider JTAG design to accommodate debugging capability by Silicon Explorer

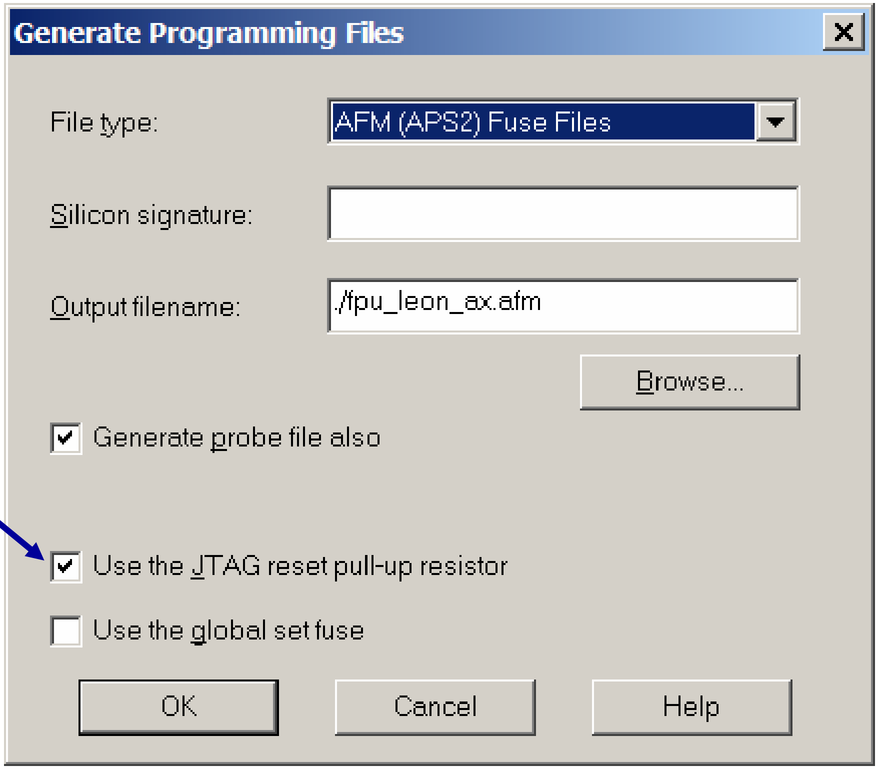
JTAG in RTAX-S

JTAG and Probe Pin Configurations

Pin Name	Configurations
TCK	MUST Be Driven (Can Be Hardwired to 0 or 1)
TDO	MUST Be Unconnected
TDI	Can Be Hardwired to 1 (Has Internal Pull-up)
TMS	Can Be Hardwired to 1 (Has Internal Pull-up)
TRST	*MUST Be Hardwired to GND*
PRA/B/C/D	MUST Be Unconnected

JTAG in RTAX-S

- Circuitry Is NOT Radiation Tolerant!
- Recommendation!
  - **Hardwire TRST Input (JTAG Reset) to Ground in Critical Applications**



Generate Programming Files

File type: AFM (APS2) Fuse Files

Silicon signature:

Output filename: ./fpu_leon_ax.afm

☑ Generate probe file also

☑ Use the JTAG reset pull-up resistor

☐ Use the global set fuse

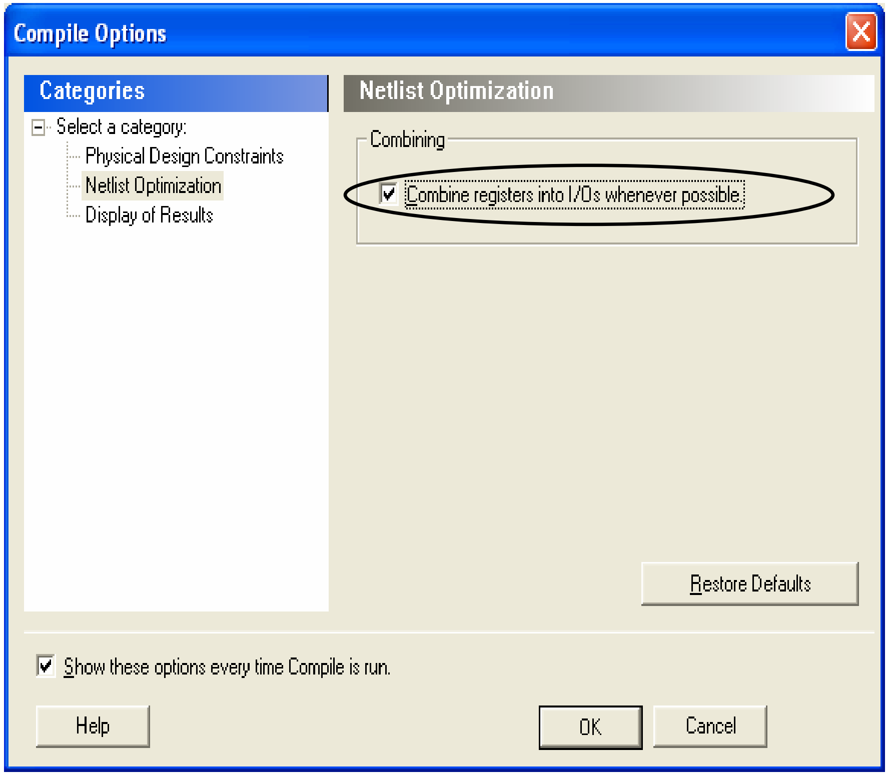
JTAG Reset Pull-Up Resistor Can Be Enabled (Default) or Disabled when Generating Programming File

Actel Recommends Disabling Pull-Up Resistor for Space Applications

Design Optimization

Compile Optimization

- Check Register Combining option. It is not checked by default
  - Save R-Cell Resources
  - I/O registers are TMR
- Watch out for I/O Register followed by long combinatorial path
  - In this case, consider combining each I/O separately by using I/O Attribute Editor
- Examine Compile Status Report
  - From Designer menu: Tools\ Reports\ Status
  - List of high fan-out nets and the number of I/O register combiners



Compile Options

Categories

Select a category:
- Physical Design Constraints
- Netlist Optimization
- Display of Results

Netlist Optimization

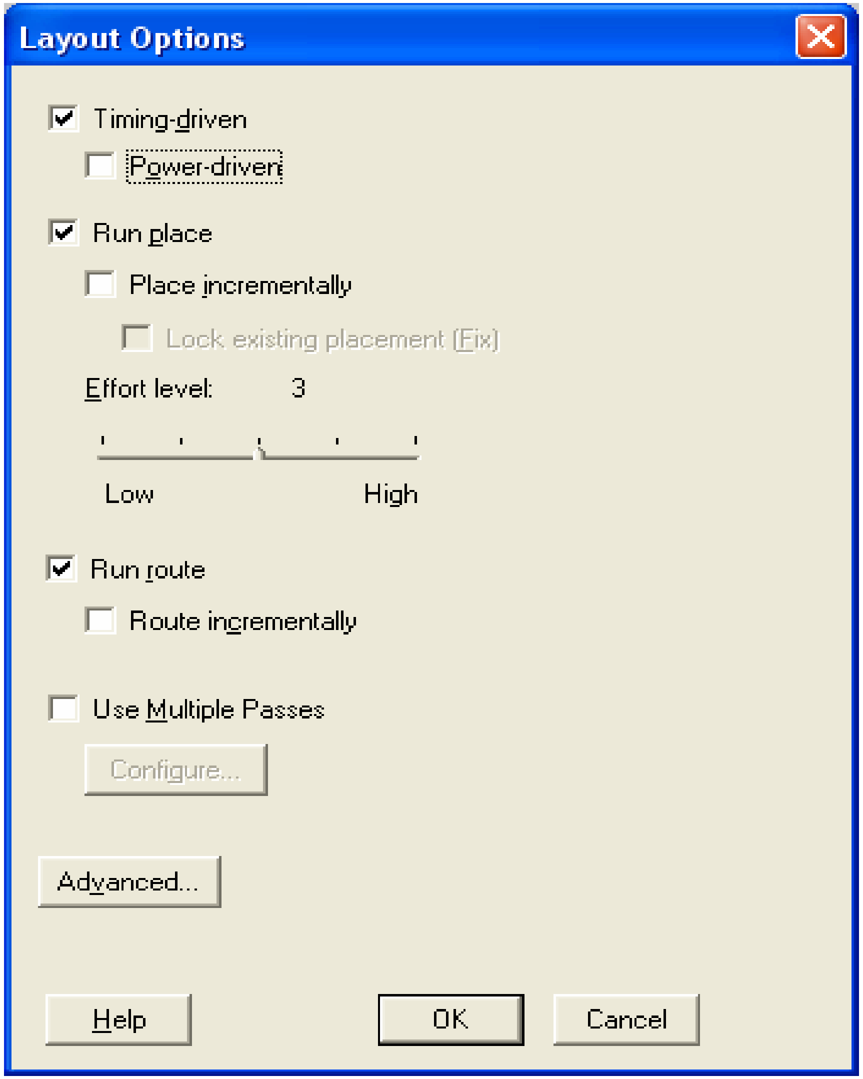
Combining

☑ Combine registers into I/Os whenever possible.

☑ Show these options every time Compile is run.

Layout Optimization

- Understand all Layout options is essential in optimizing a design



Layout Options

☑ Timing-driven

☐ Power-driven

☑ Run place

☐ Place incrementally

☐ Lock existing placement (Fix)

Effort level: 3

☑ Run route

☐ Route incrementally

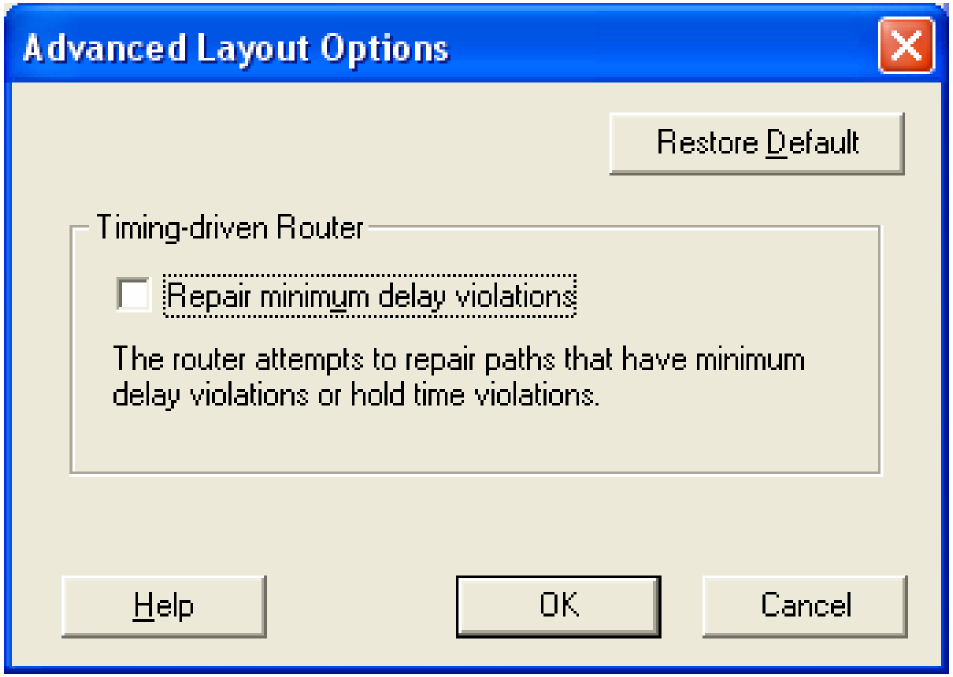
☐ Use Multiple Passes

Layout Options Recommendation

- Check Timing-driven option to take into account all the timing constraints
- Use incremental place to only work on changed instance and get faster run
- Use Incremental Routing only with extremely-limited physical changes
- Use Multiple Passes Layout to get the best timing result
  - **The seeds are randomly chosen by Designer**
  - **Another place and route (even on the same computer) might generate slightly different results => it might take more or less iterations to meet timing.**

Advanced Layout Options

- Select Repair Min Delay Violation in Advanced Layout option:
  - An additional route is performed by increasing the length of routing paths to add delay to paths
  - No additional logic is inserted
  - Best suited to repair paths with small (0 to 3 ns) hold and minimum delay violations



□ Repair minimum delay violations

The router attempts to repair paths that have minimum delay violations or hold time violations.

Timing Analysis with SmartTime

Timing Analysis Process

- A successful design has to meet timing
- After Layout complete, open SmartTime
- Check all the timing specifications
  - Clocks and their Frequencies
  - External Timing:
    - Input Delays
    - Expected Clock-to-Out
    - Late-arriving Inputs
  - False Paths
  - Multicycle Paths
- Read all timing reports
- From this data, a designer will need to find solutions to optimize the design better. Explore Timing Constraint options before venturing into Physical Constraints

Setting Timing Constraints

- Actel Designer uses industry standard SDC
  - **Synplify uses its own SDC format => Synplify SDC might not be accepted by Designer in some cases**
- Always use create_clock to define all the input clock sources
- Use create_generated_clock to define the relationship between internally-generated clock and the reference clock.
  - **This is very helpful for inter clock domain analysis**
- Use set_false_path and set_multicycle_path
  - **Prevent unreal violations, especially on high fanout nets**
  - **Free up resources so Layout can focus on optimizing actual critical paths**
- Use set_max_delay and set_min_delay to set Path Delay Requirement on Violating Paths

Timing Optimization Techniques

- Use Clock Segmentation
  - For design with many small clock networks
  - To map high fanout nets to local clock segments

  http://www.actel.com/documents/RTAXS_Clocking_AN.pdf (AC310)
- Use Inter-clock Domain Timing Analysis
  - SmartTime Tool\ Options \ General\ Clock Domains
  - This option is not enabled by default
  - To reveal potential violations across clock domains
- Use Bottleneck Analysis Report
  - Use Path_Costs to list instances causing the greatest amount of delay
  - Use Path_Count to list instances causing greatest number of path violations

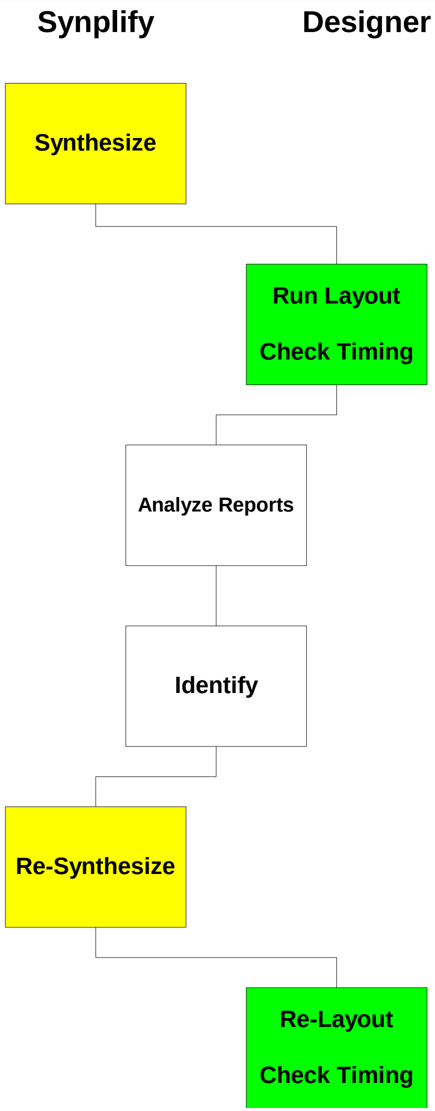
FloorPlanning Tips

FloorPlanning Tips

- Examine Data Dependency among Blocks
  - **Path-Based Optimization**
- Place Cells in Hierarchical Block as Close as Possible (without Routing Congestion)
- Specially Place Timing-Critical Blocks
  - **Wide Multiplexers / Decoders**
- Study Layout in ChipPlanner
  - **Ensure Region Is 15-20% Larger than Tile Count to Avoid Congestion**
- Place Memory Intelligently using ChipPlanner
  - **Depend on whether Memory Access is through I/O or Internal Logic**
  - **Leave Space between RAM Blocks to Accommodate Interconnection of Interacting Block**
- For Low-Utilization Designs
  - **Use `assign_region` or `define_region` Constraints to Avoid Scattered Placement**

Coping with High-Fanout Nets
Suggested Flow

1. Synthesize Design
   - Enter Timing Constraints
   - Set Global Fanout to Very High Value
2. Run Layout
   - Enter Timing Constraints
   - Compile
   - Place & Route
3. Analyze Reports
   - Net Fanout (Internal Nets, Resets, Clocks)
   - Global Assignments from Synthesis
   - RAM Placement
   - Timing Reports
4. Identify:
   - Candidates for Global or Local Clock Assignments
   - Need for Manual Logic Replication or Shielding
   - Need for Manual RAM Placement
5. Re-synthesize
   - Return to Default or Near-default Fanout
   - Enter Timing Constraints
   - Use Synthesis Attributes on Selected Nets or Blocks
6. Run Layout and Analyze Timing Results


Synplify
Designer
Synthesize
Run Layout
Check Timing
Analyze Reports
Identify
Re-Synthesize
Re-Layout
Check Timing