



*Synplicity*®

Simply Better Results...

---

**Secret Synthesis Recipes for  
Performance and Cost**

SOPC World

November, 2004

# Agenda

- ♦ **Company Introduction**
- ♦ Recipes for achieving high performance
- ♦ Recipes for area (cost) saving
- ♦ Physical Synthesis
- ♦ DSP Synthesis



Synplicity

Simply Better Results™

# Synplicity, Inc.

- ♦ **A Global EDA company founded in 1994**
- ♦ **Unique company philosophy**
  - ♦ Best results don't have to come from hard to use tools
  - ♦ Flexible and easy to work with company
  - ♦ Dedicated to providing the best technical support
- ♦ **The Market Leader in FPGA Synthesis**
- ♦ **Innovative Solutions For FPGA Designers**
  - ♦ Physical synthesis
  - ♦ Debug
  - ♦ DSP Synthesis

*Simply Better Results®*

# Agenda

- ♦ Company Introduction
- ♦ **Recipes for achieving high performance**
- ♦ Recipes for area (cost) saving
- ♦ Physical Synthesis
- ♦ DSP Synthesis

# Performance Tips

- ◆ **Out of the box synthesis yields good results**
  - ◆ Even when running with default settings, the synthesis algorithms are very good at making FPGA specific decisions
- ◆ **Applying real constraints will boost performance**
  - ◆ Constraints define critical areas for the synthesis algorithms
- ◆ **Setup Tips**
  - ◆ Setting up the project correctly can impact performance and area greatly. Some tips to follow.
- ◆ **Controlling Implementation Tips**
  - ◆ After set-up, what can the user do to change the implementation to reduce Area further and increase performance.



# Setup Tip #1 The Optimum Constraint

- ♦ Create the right amount of negative slack for constraining Synthesis and Place & Route
  - Best to exactly constrain Synthesis
  - Ensure that the most critical path is a “real” path
  - Ensure that Synplify doesn't have positive slack on the critical path

Worst slack in design: 0.107

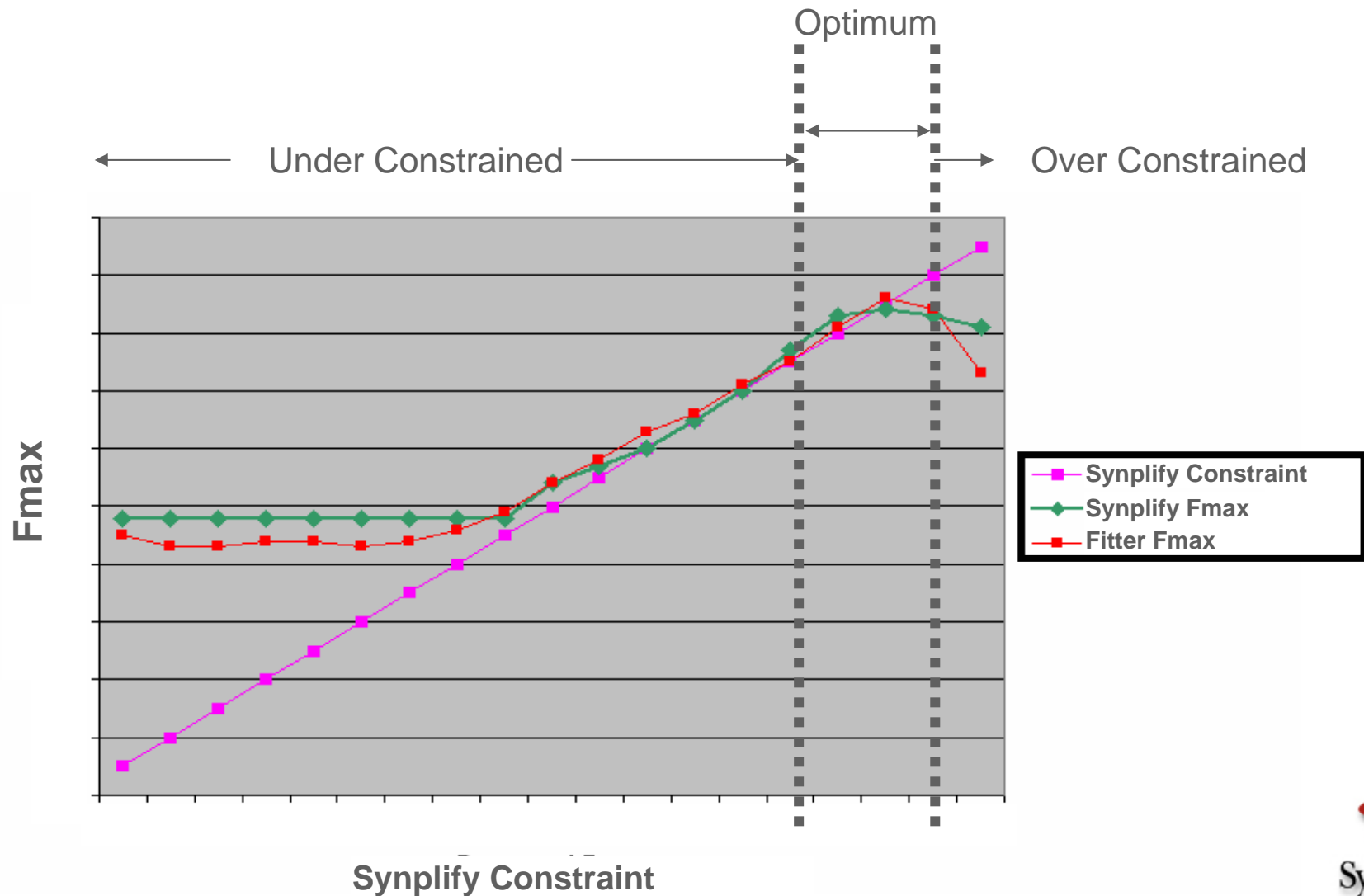
Starting Clock	Requested Frequency	Estimated Frequency	Requested Period	Estimated Period	Slack	Clock Type	Clock Group
clk	96.2 MHz	97.2 MHz	10.400	10.293	0.107	declared	g1

Positive Slack - bad

- If Synplify indicates less than ~10% of cycle time negative slack, some optimizations may not kick in
- Performance will likely degrade if over-constrained by over 15%



# The Optimum Constraint for Best Results



# Setup Tip #2 – Clock Constraints

- ◆ **Enter all timing constraints**
  - ◆ Define real individual clock constraints
  - ◆ If the clocks are unrelated, always put them into different clock groups
- ◆ **Using the global frequency field can harm results**
  - ◆ Only useful for single clock, small benchmarks & demonstrations
- ◆ **Always specify the correct clock groupings**

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock	
1	<input checked="" type="checkbox"/>	clock1	285.714	3.5	group1			50	1.2	<input type="checkbox"/>	
2	<input checked="" type="checkbox"/>	clock2	232.558	4.3	group2			50	0.5	<input type="checkbox"/>	
3	<input checked="" type="checkbox"/>	clock3	178.571	5.6	group3			50	0	<input type="checkbox"/>	
4	<input checked="" type="checkbox"/>									<input type="checkbox"/>	
◀ ▶ \ <b>Clocks</b> / Clock to Clock / Inputs/Outputs / Registers / Multi-Cycle Paths / False Paths / Max Delay Paths / Attributes / Compile Points / Other / ▶											



# More Setup Tips

- ◆ **I/O Timing**

- ◆ When I/O timing is on
  - ◆ The critical path will likely be through an I/O
  - ◆ Other logic may therefore not be optimal
- ◆ If I/O timing is not required turn it *off*
  - ◆ *Un-selecting the 'Use clock period for unconstrained IO' in the implementation options'*
  - ◆ *Off by default for new projects*

- ◆ **Be sure to specify false and multi-cycle paths**

- ◆ Enables Synplify to focus on real critical paths

- ◆ **Add Clearbox to the Synplify project**

# Tip #3 – Pipelining & Retiming

- ◆ **Significant Performance Increase**
  - ◆ Up to 50% better timing
  - ◆ Extremely design dependant
- ◆ **Pipelining (a la Synplicity)**
  - ◆ Applies to arithmetic datapath
    - ◆ Multipliers, Adders, ROMs
  - ◆ Moves existing registers to balance delays
  - ◆ Timing driven
  - ◆ On by default
- ◆ **Retiming (a la Synplicity)**
  - ◆ Applies to the entire design
  - ◆ Moves existing registers to balance delays
  - ◆ Timing driven
  - ◆ Off by default



Synplicity

Simply Better Results™

# Tip #4 – State Machines

## ◆ FSM Compiler

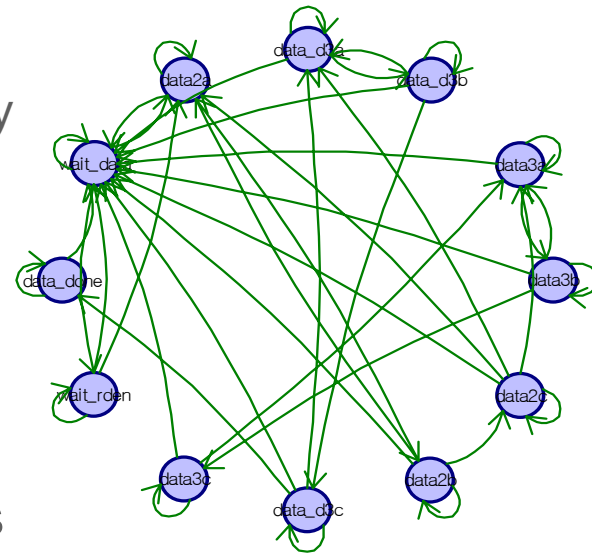
- ◆ ON by default
- ◆ Extracts and optimizes FSMs
- ◆ State encoding based on number of states
  - ◆ 2-4: sequential / 5-40: onehot / more than 40: gray

## ◆ FSM Explorer

- ◆ OFF by default
- ◆ Timing-driven state encoding

## ◆ User can force state encoding

- ◆ syn\_encoding attribute on modules or instances
  - ◆ sequential, onehot, gray and even user defined!
- ◆ Encoding could be viewed as a retiming across the FSM
  - ◆ Output decoding logic vs FSM logic



Synplicity

Simply Better Results™

# Tip #5 – Resource Allocation

- ♦ **Macro block is not always the fastest implementation**
  - ♦ A well pipelined LUT mult is faster than a combination of blockmults
- ♦ **Use selected attributes to control resource usage**
- ♦ **Multiplier**
  - ♦ `syn_multstyle {logic | lpm_mult}`
- ♦ **RAM**
  - ♦ `syn_ramstyle {registers | M512 | M4K | M-RAM | block_ram | no_rw_check}`
- ♦ **ROM**
  - ♦ `syn_romstyle {logic | lpm | block_rom}`
- ♦ **Shift Register**
  - ♦ `syn_srlstyle {registers | altshift_tap}`



Synplicity

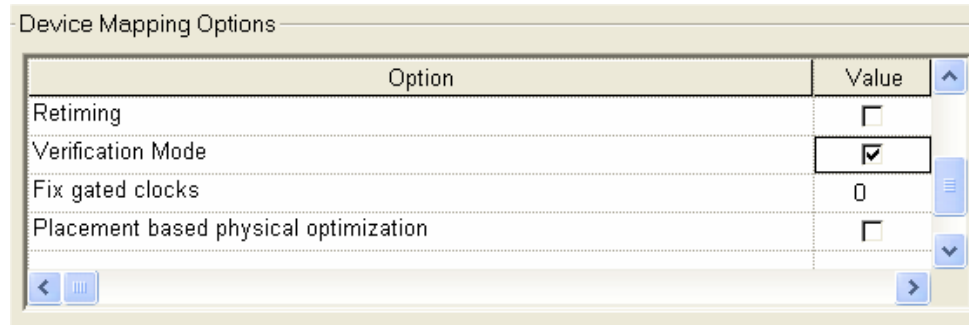
Simply Better Results™

# Tip #6 – Optimization Control

- ◆ **Power users can control synthesis optimizations**
- ◆ **syn\_keep (in source code)**
  - ◆ Preserves a RTL net throughout synthesis
  - ◆ Prevents LUT packing, replication, removal, etc
  - ◆ Allows –thru constraints
- ◆ **syn\_preserve (in source code)**
  - ◆ Disables sequential optimizations on FFs
  - ◆ Prevents removal, merging, inverter push-thru, FSM extraction
- ◆ **syn\_replicate (in constraint file)**
  - ◆ Prevents replication of FFs
- ◆ **syn\_maxfan (in constraint file)**
  - ◆ Hard fanout limit on module or instances
  - ◆ Triggers replication and buffering
- ◆ **Details and examples in on-line documentation**

# Formal Verification Tool Support

- ◆ **Enable Verification Mode**



The screenshot shows a dialog box titled "Device Mapping Options". It contains a table with two columns: "Option" and "Value". The "Verification Mode" option is checked. There are also buttons for "Retiming", "Fix gated clocks", and "Placement based physical optimization".

Option	Value
Retiming	<input type="checkbox"/>
Verification Mode	<input checked="" type="checkbox"/>
Fix gated clocks	0
Placement based physical optimization	<input type="checkbox"/>

- ◆ **VIF (Verification Interface File)**

- ◆ Automatically generated during synthesis
- ◆ Tool independent ASCII file
- ◆ Contains information needed by FV/LEC tools
- ◆ Synplify Pro feature only

- ◆ **Cadence's Conformal and Prover's ECheck supported**

- ◆ **Script to convert "VIF" to LEC compatible file(s) available**



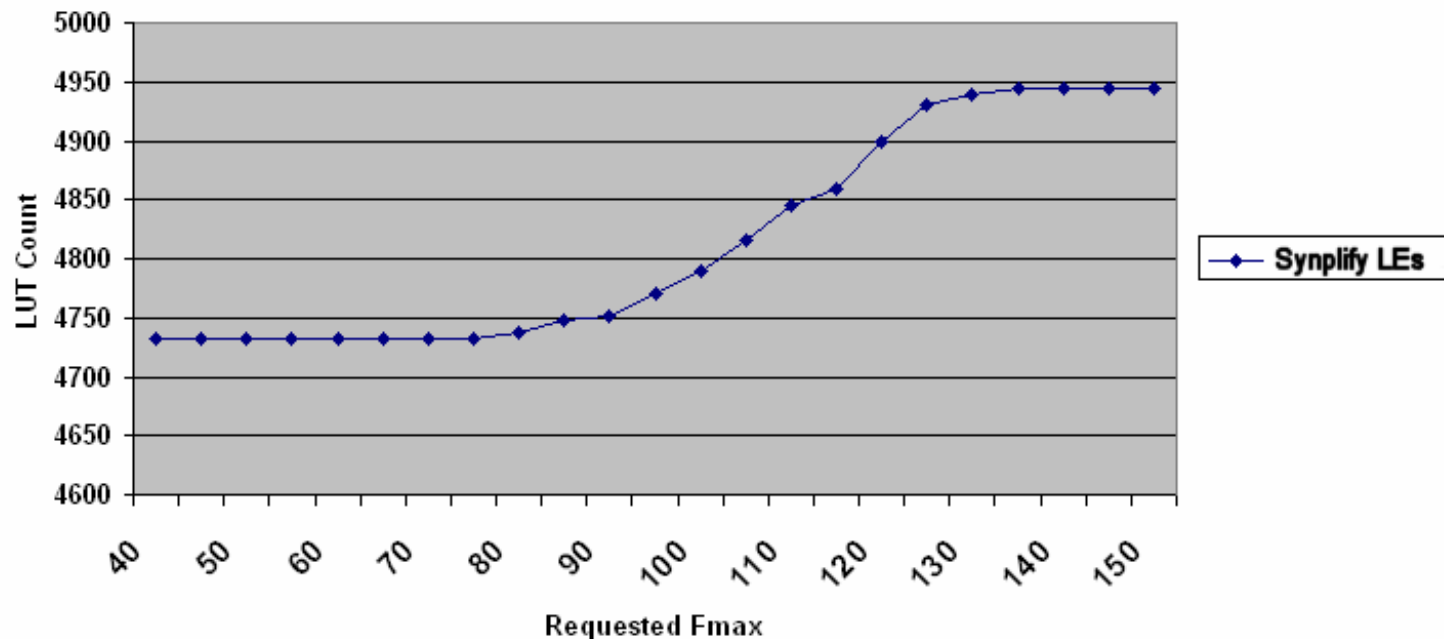
# Agenda

- ♦ Company Introduction
- ♦ Recipes for achieving high performance
- ♦ **Recipes for area (cost) saving**
- ♦ Physical Synthesis
- ♦ DSP Synthesis

# Synthesis For Area (Cost) Saving

- ◆ **Synplify is truly timing driven**

- ◆ If a path is non-critical, Synplify will try to save area while maintaining constraints
- ◆ Only when the path requires performance will Synplify start to increase area
- ◆ Performance-upon-demand





# Synthesis For Area

- ♦ **Turn resource sharing ON**
- ♦ **Use resource allocation attributes**
  - ♦ syn\_ramstyle
  - ♦ syn\_romstyle
  - ♦ syn\_multstyle
  - ♦ syn\_srlstyle
- ♦ **Explore FSM encodings**
  - ♦ FSM Explorer
  - ♦ Use syn\_encoding



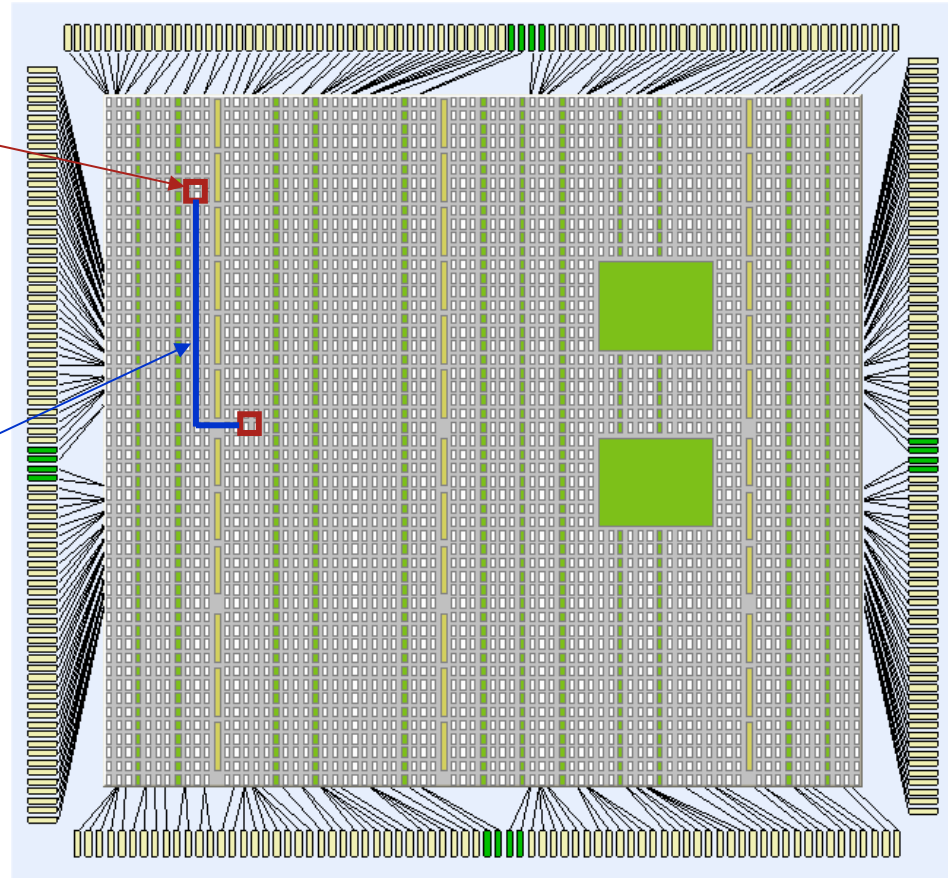
# Agenda

- ◆ Company Introduction
- ◆ Recipes for achieving high performance
- ◆ Recipes for area (cost) saving
- ◆ **Physical Synthesis**
- ◆ DSP Synthesis

# Route Delay Must be Considered During Synthesis

Delay inside  
logic block is  
25% of total

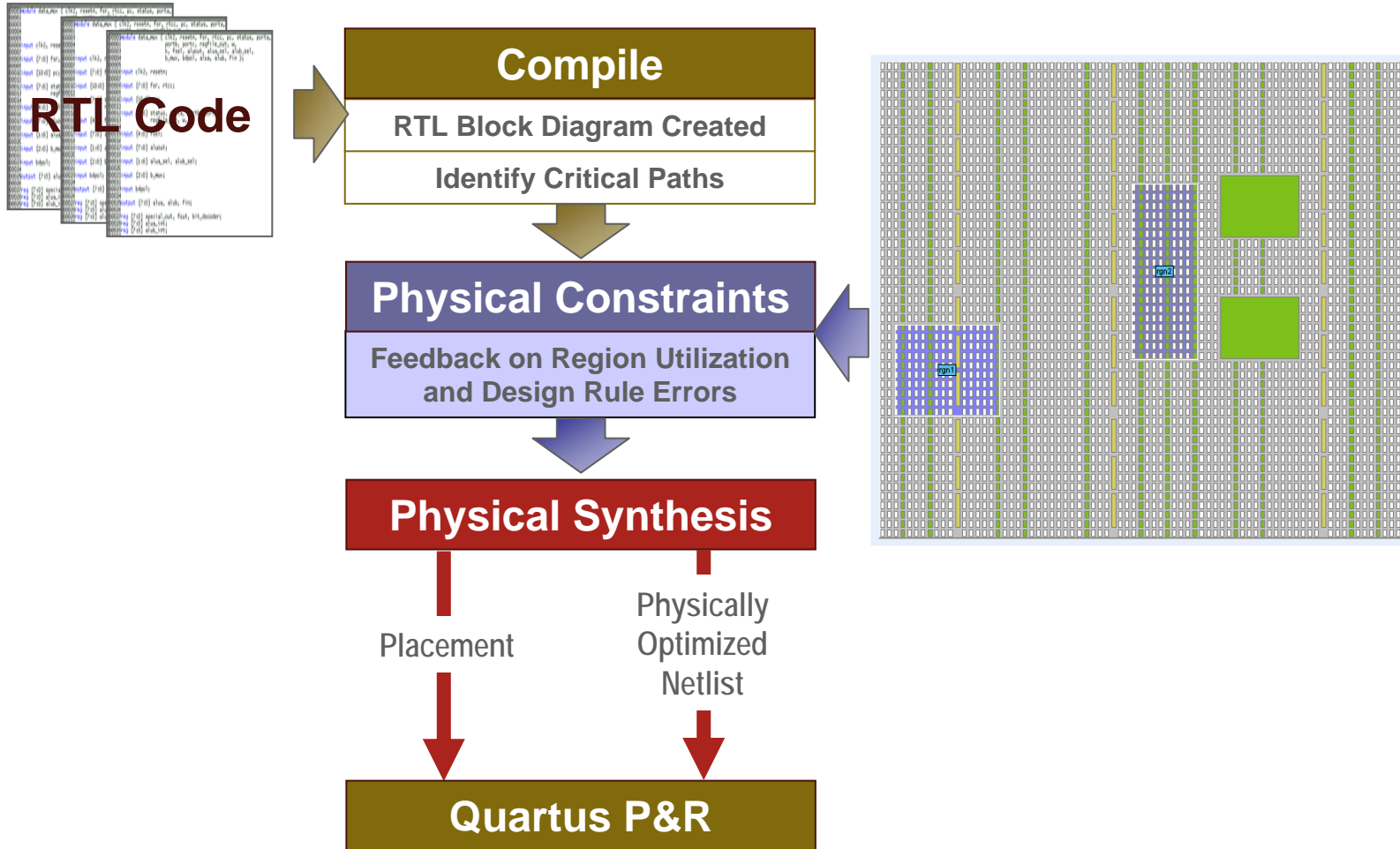
Routing between  
logic blocks is  
75% of total



Synplicity

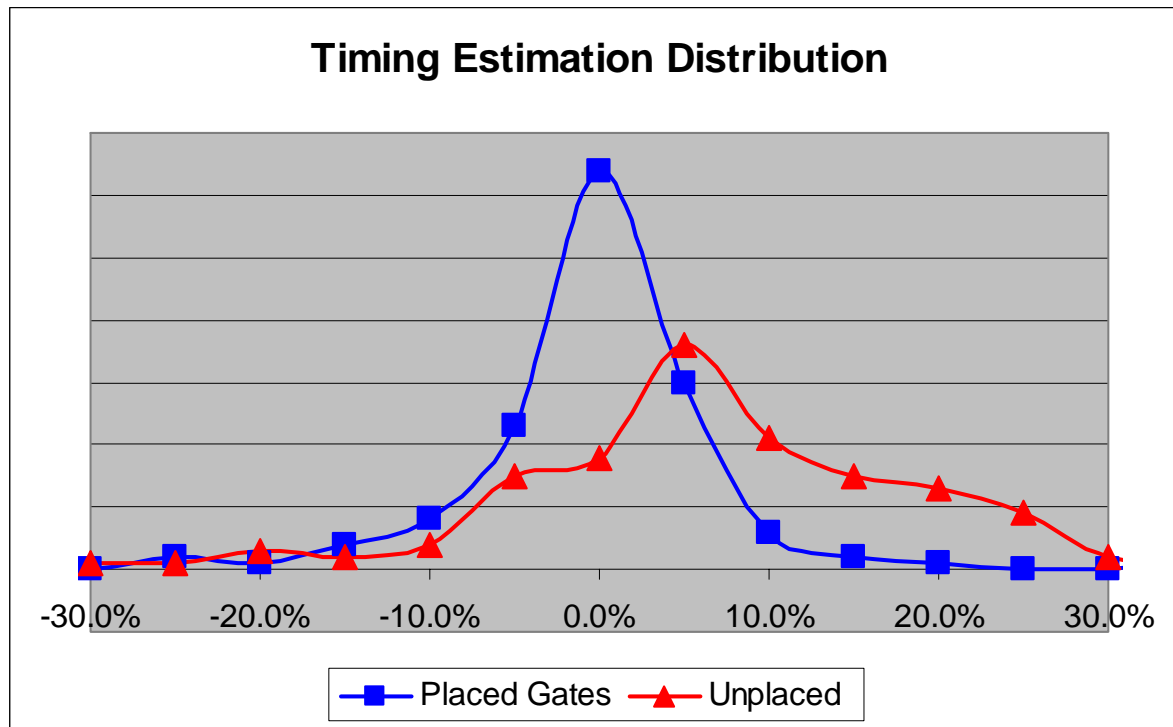
Simply Better Results™

# Amplify FPGA - Physical Synthesis



# Accurate Timing Correlation is a Must

Estimates must be accurate to ensure the tool is working on the right paths

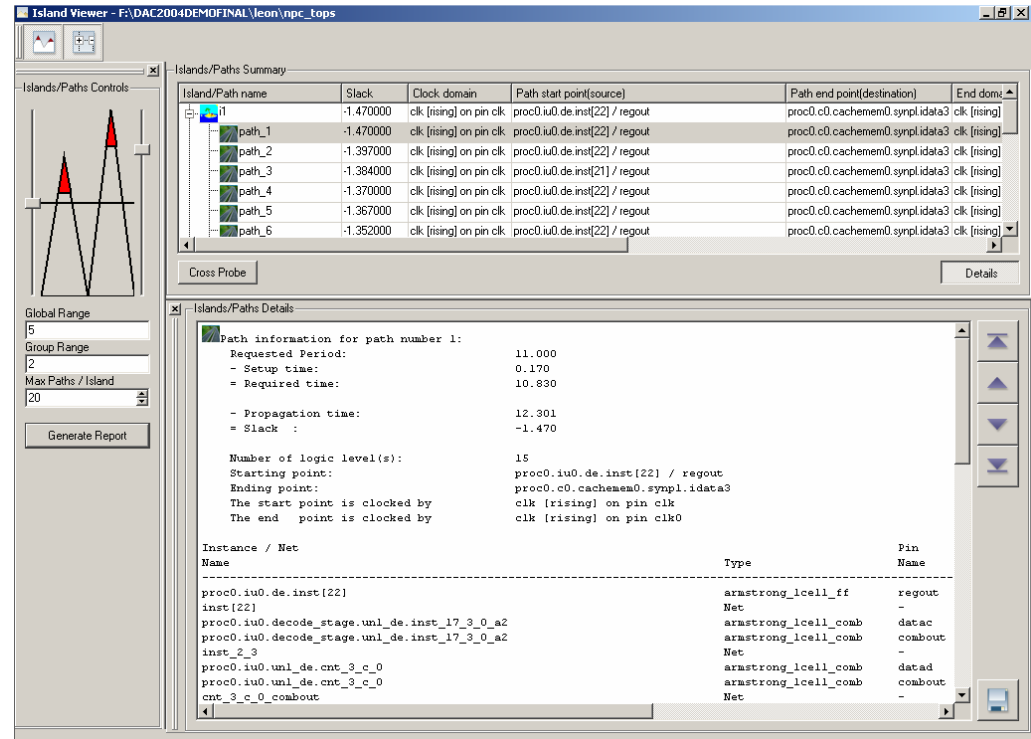


- ◆ 90% within 10% of actual timing
- ◆ 67% within 5% of actual timing
- ◆ 145 designs used



# Graphical Island Timing Viewer

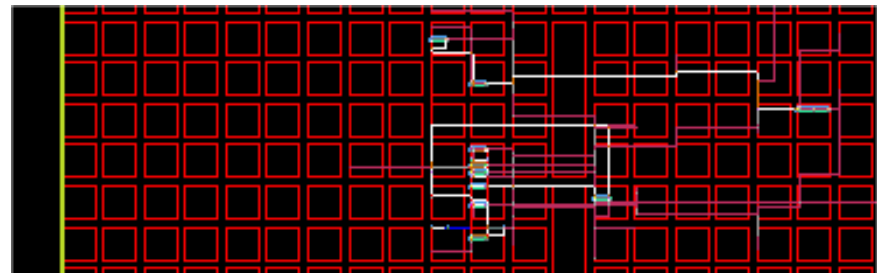
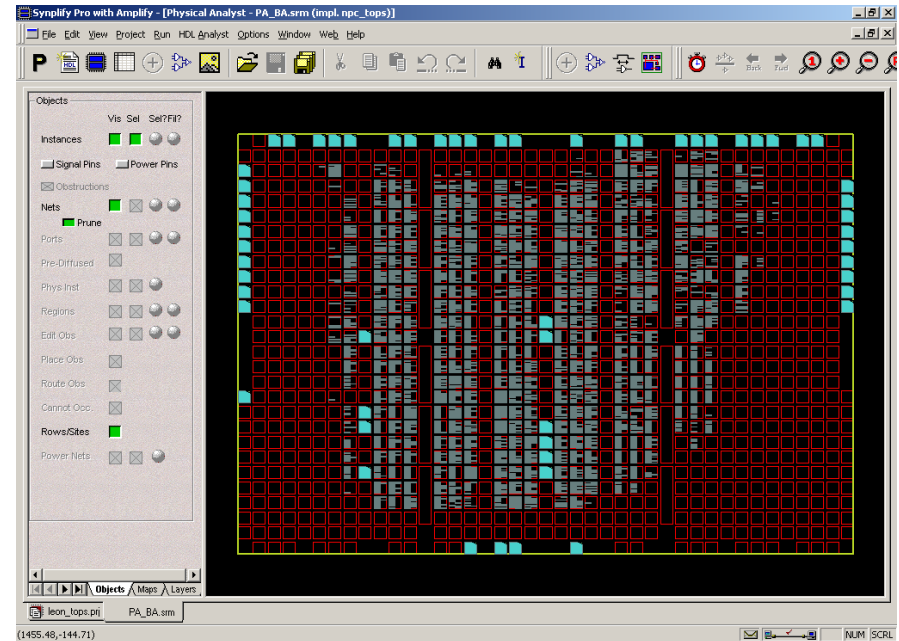
- ◆ Helps users create a good floor plan (physical constraints)
- ◆ Easily identify physically connected paths with negative slack
- ◆ Reduces iterations of synthesis
- ◆ New in 8.0



# Visual Feedback For Analysis & Design Planning

## *Physical Analyst*

- ◆ **Physical analysis**
  - ◆ Find, filter, expand commands
  - ◆ Cross-probing to source code, RTL view and Technology view
- ◆ **Improved timing analysis**
  - ◆ Critical path display
  - ◆ Cross-probing from timing report
- ◆ **Congestion analysis**
  - ◆ Global route estimator
  - ◆ Congestion maps
- ◆ **New in 8.0**



Synplicity  
Simply Better Results™

# Agenda

- ♦ Company Introduction
- ♦ Recipes for achieving high performance
- ♦ Recipes for area (cost) saving
- ♦ Physical Synthesis
- ♦ **DSP Synthesis**



Synplicity

Simply Better Results™



# **DSP Synthesis Automation**

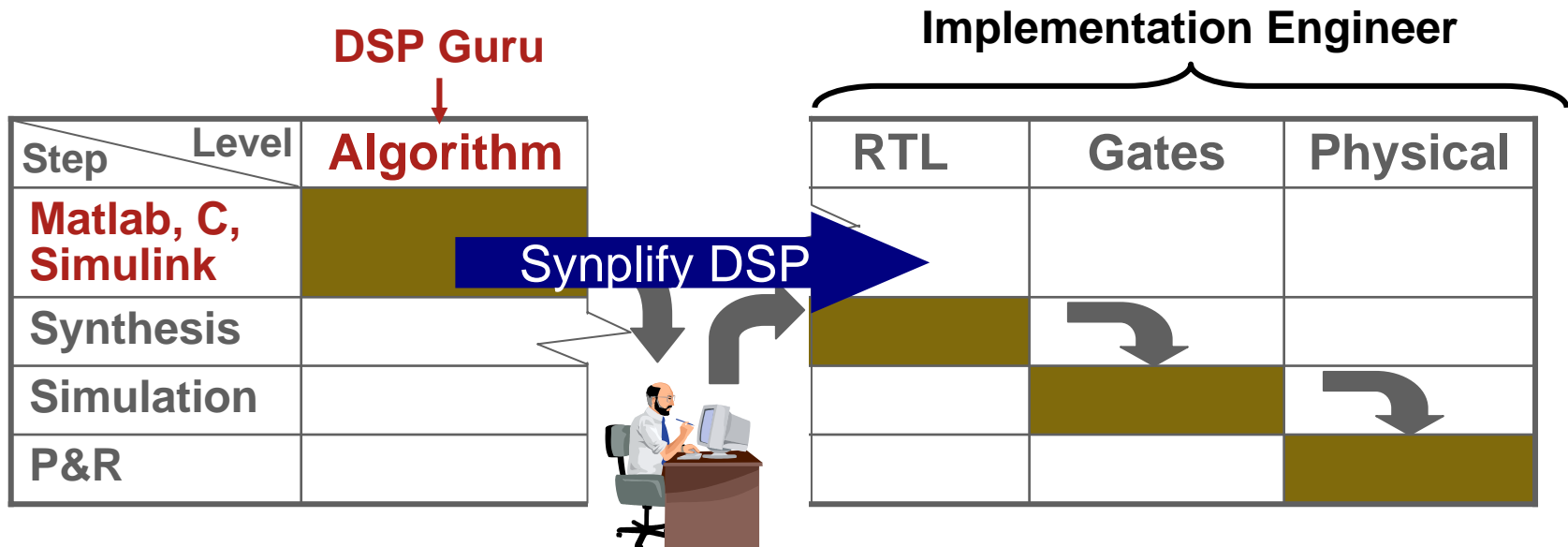
**Algorithm to RTL Implementation for FPGAs**



*Synplicity*

Simply Better Results™

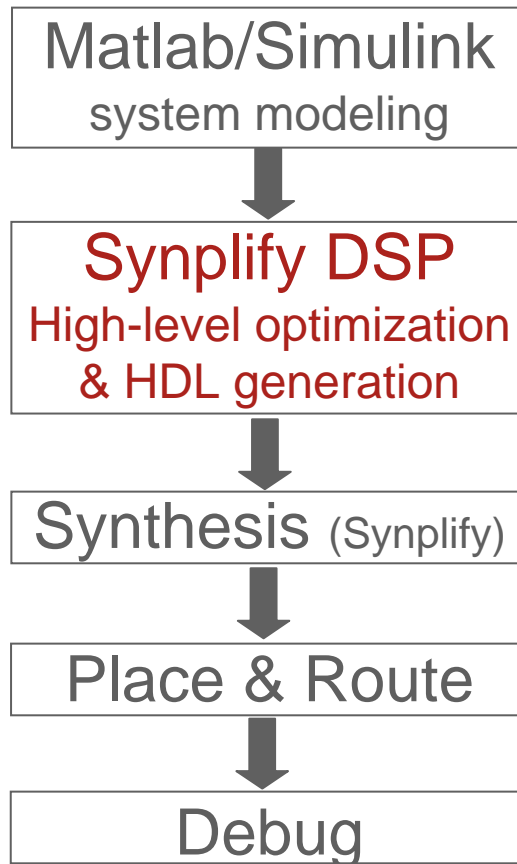
# DSP Designers & HDL Coders are Different



Algorithm tools (and designers) have no idea of implementation issues  
RTL is written by hand (redundant & error prone)

Synplify DSP addresses this by raising the level of abstraction in which an engineer operates  
(Algorithm to RTL)

# DSP Synthesis using Synplify DSP

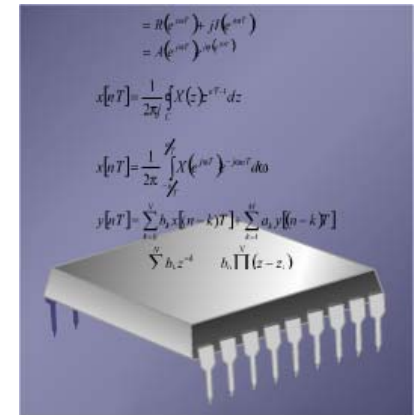


- ◆ **Front-End for**

- ◆ Synplify Pro
- ◆ Certify

- ◆ **Two Components**

- ◆ Blockset
- ◆ Toolbox



**Optimized DSP  
algorithms for hardware**



**Synplicity**

Simply Better Results™

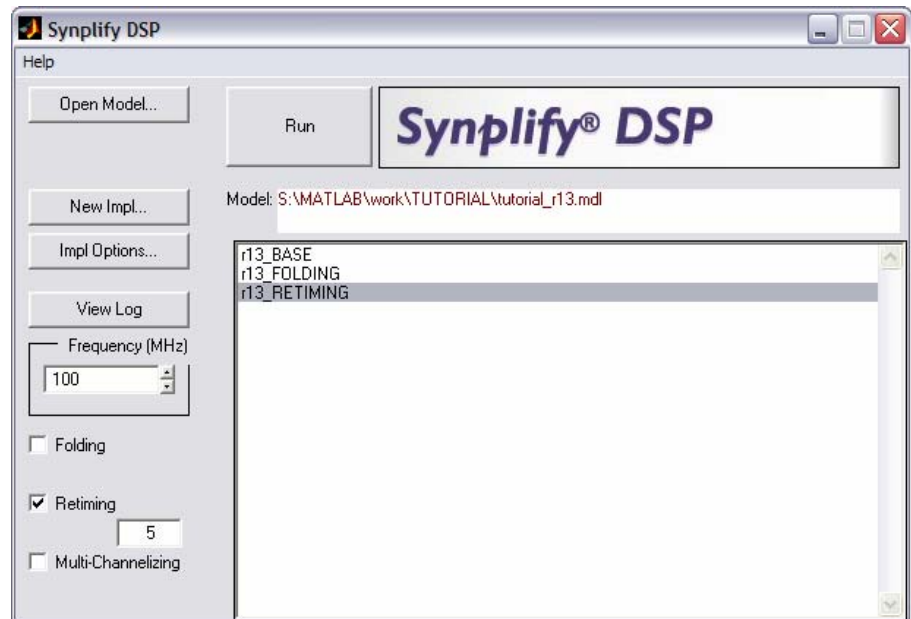
# The Synplify DSP Blockset

- ◆ **Blockset advantages**
  - ◆ Simulink Fixed Point discrete data type
  - ◆ Simulink Multi Rate discrete time management
  - ◆ Architecture details hidden
  - ◆ Latency free design
- ◆ **What's this all mean?**
  - ◆ High productivity
  - ◆ Days instead of weeks



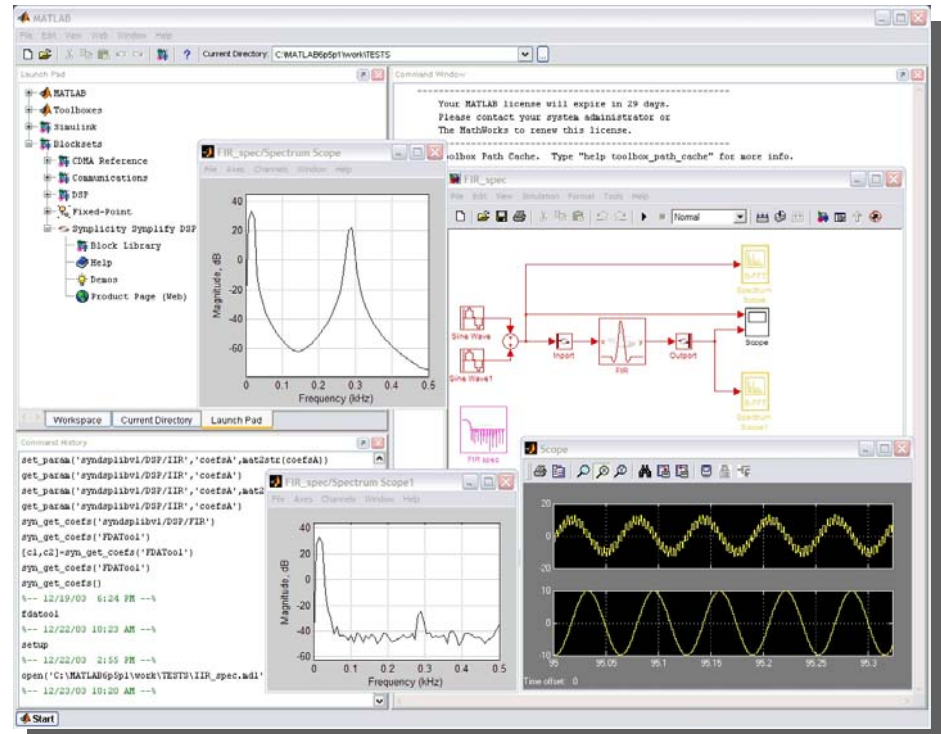
# The Synplify DSP Toolbox

- ◆ **Toolbox advantages**
  - ◆ Optimization technology
  - ◆ Decouples algorithm from architecture
  - ◆ Technology independent
- ◆ **Can be used by the DSP Guru**



# The Value of Synplify DSP

- ◆ **Large productivity gain**
  - ◆ Decouples algorithm from architecture
  - ◆ Area-Speed tradeoffs
  - ◆ Multi-channel system from single-channel spec (patent)
- ◆ **Single source with QoR**
  - ◆ DSP optimization gives faster and smaller designs
  - ◆ Technology Independence
- ◆ **Leverages familiar design environment**
  - ◆ Simulink – No learning curve
  - ◆ Integrated ToolBox



# Summary

- ♦ **The market leader in FPGA synthesis**
  - ♦ Synthesis impacts your customers bottom line
- ♦ **Innovation leader in FPGA design**
  - ♦ First with physical synthesis
  - ♦ First with RTL debug
  - ♦ First with DSP synthesis
- ♦ **Excellent working relationship with Altera**
  - ♦ New device support upon availability
- ♦ **Industry-leading post-sales technical support**



Synplicity

Simply Better Results™