Quartus II Software Design Series: Foundation

Online Training

Supplemental Files to Download

- Complete presentation in PDF format
- Lab exercise manual in PDF format
- Lab exercise files (executable ZIP file)
- All files contained in single .zip file

- *Click link in email or go to **Attachments** button to download (may need to hold **Ctrl** to download)*

Objectives

- Create a new Quartus® II project
- Choose supported design entry methods
- Compile a design into an FPGA
- Locate resulting compilation information
- Assign design constraints (timing & pin)
- Perform timing analysis & obtain results
- Generate files for 3rd-party EDA simulation
- Configure an FPGA

Class Agenda

- Projects
  - Exercise 1
- Design Entry
  - Exercise 2
- Compilation
  - Exercise 3
- Settings & Assignments
  - Exercise 4
- I/O Planning
  - Exercise 5
- Timing Analysis
  - Exercise 6
- EDA Simulation
- Programming / configuration
  - Exercise 7 (optional)

Advanced Quartus II Courses

- Quartus II Software Design Series: Verification
  - Timing analysis
    - Thorough investigation of performing timing analysis on an Altera device with TimeQuest
  - Power analysis
  - Debugging solutions
    - SignalProbe incremental routing
    - Logic Analyzer Interface
    - In-System Memory Content Editor
    - In-System Sources & Probes
    - Chip Planner & Resource Property Editor
    - SignalTap II Embedded Logic Analyzer
- Quartus II Software Design Series: Optimization
  - Incremental Compilation
  - Quartus II optimization features & techniques

Quartus II Software Design Series: Foundation

Introduction to Altera & Altera Devices

The Programmable Solutions Company®

- Programmable Logic Devices
- Tools
  - Quartus® II software
  - SOPC Builder
  - DSP Builder
  - Nios® II IDE
- Intellectual Property (IP)
  - Signal processing
  - Communications
  - Embedded processors
    - Nios II embedded processor

Programmable Logic Families

- Structured ASIC
  - HardCopy® II & HardCopy® Stratix devices
- High & medium density FPGAs
  - Stratix® family devices
- Low-cost FPGAs
  - Cyclone® family devices
- FPGAs w/ high-speed transceivers
  - Stratix II GX, Stratix GX, & Arria® GX devices
- CPLDs
  - MAX® II, MAX 7000 & MAX 3000 devices
- Configuration devices
  - Serial (EPCS) & enhanced (EPC)

Software & Development Tools

- Quartus II Subscription Edition
  - Stratix III, Stratix, II & Stratix devices
  - Stratix II GX, Stratix GX, & Arria GX devices
  - Cyclone III, Cyclone II, & Cyclone devices
  - HardCopy II & HardCopy Stratix devices
  - MAX II, MAX 7000S/AE/B, MAX 3000A devices
  - Select older families
- Quartus II Web Edition
  - Free version
  - Not all features & devices included
    - See www.altera.com for feature comparison

Quartus II Software Design Series: Foundation

Quartus II Design Software Feature Overview

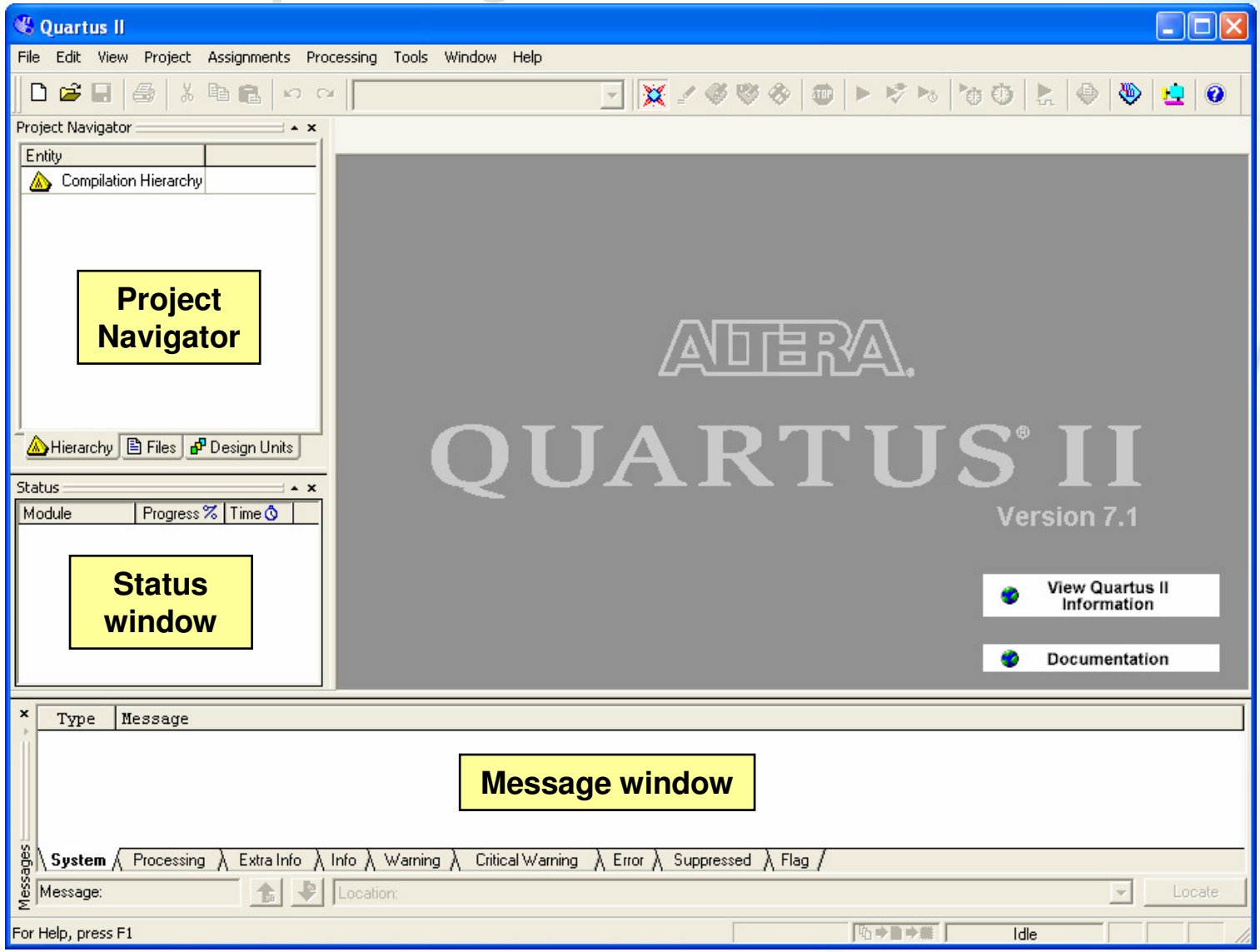
Quartus II Design Software

- Fully-integrated development tool
  - Multiple design entry methods
  - Logic synthesis
  - Place & route
  - Simulation
  - Timing & power analysis
  - Device programming

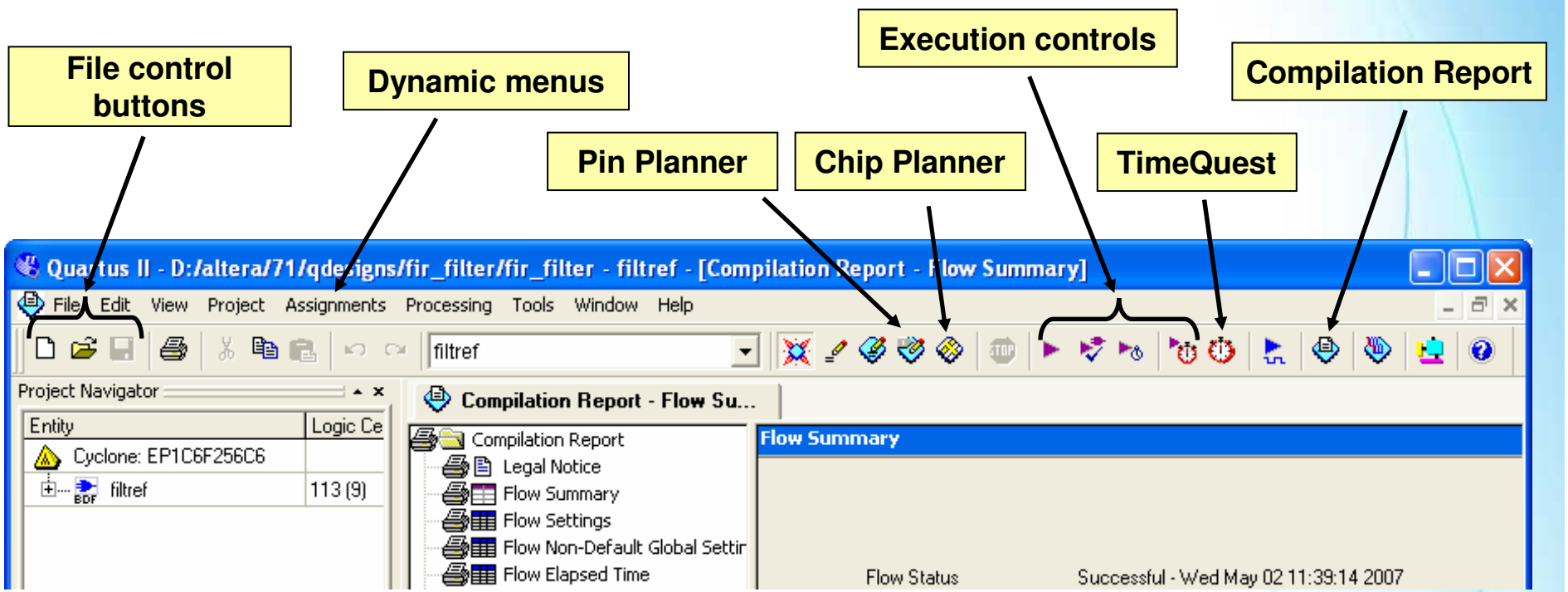
More Features

- MegaWizard® & SOPC Builder design tools
- TimeQuest Timing Analyzer
- Incremental Compilation feature
- PowerPlay Power Analyzer tool
- NativeLink® 3rd-party EDA tool integration
- Debugging capabilities
  - From HDL to device in-system
- 32 & 64-bit Windows, Solaris, & Linux support
- Multi-processor support
- Node-locked & network licensing options

Quartus II Operating Environment



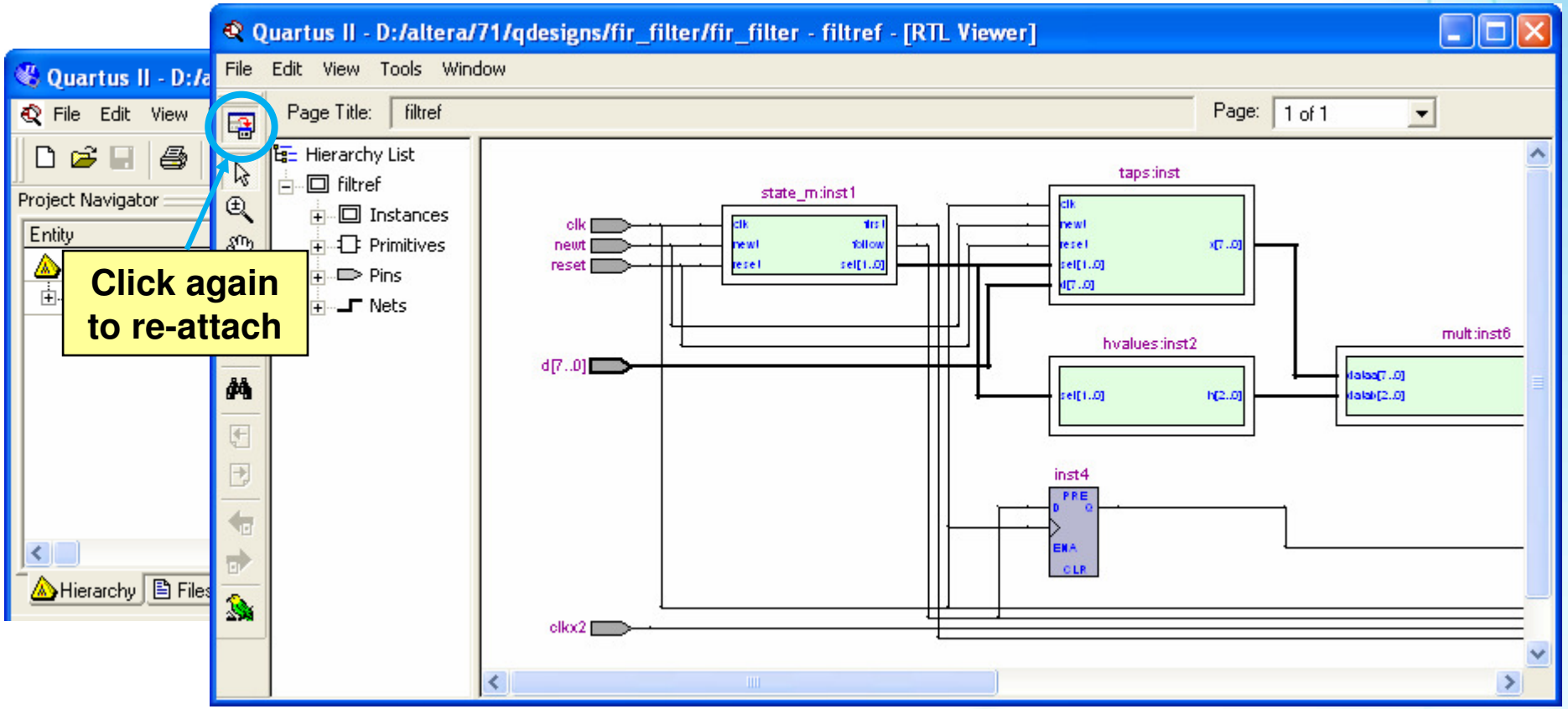
Main Toolbar



To reset views:
1. **Tools ⇒ Customize ⇒ Toolbars ⇒ Reset All**
2. **Restart Quartus II**

Detachable Windows

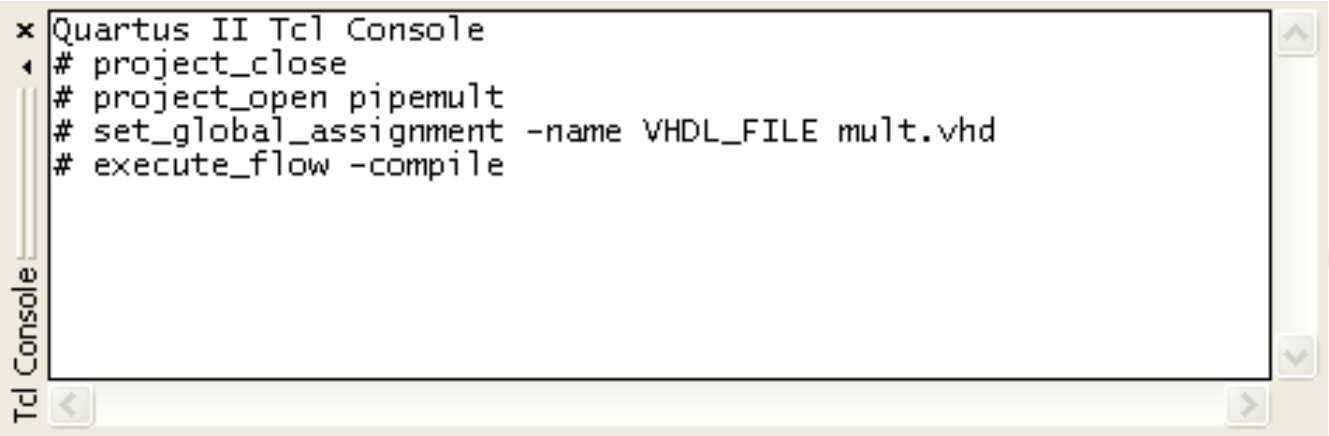
- Separate child windows from the Quartus II GUI frame (Window menu ⇒ Detach/Attach Window)



Tcl Console Window

- Enter and execute Tcl commands directly in the GUI

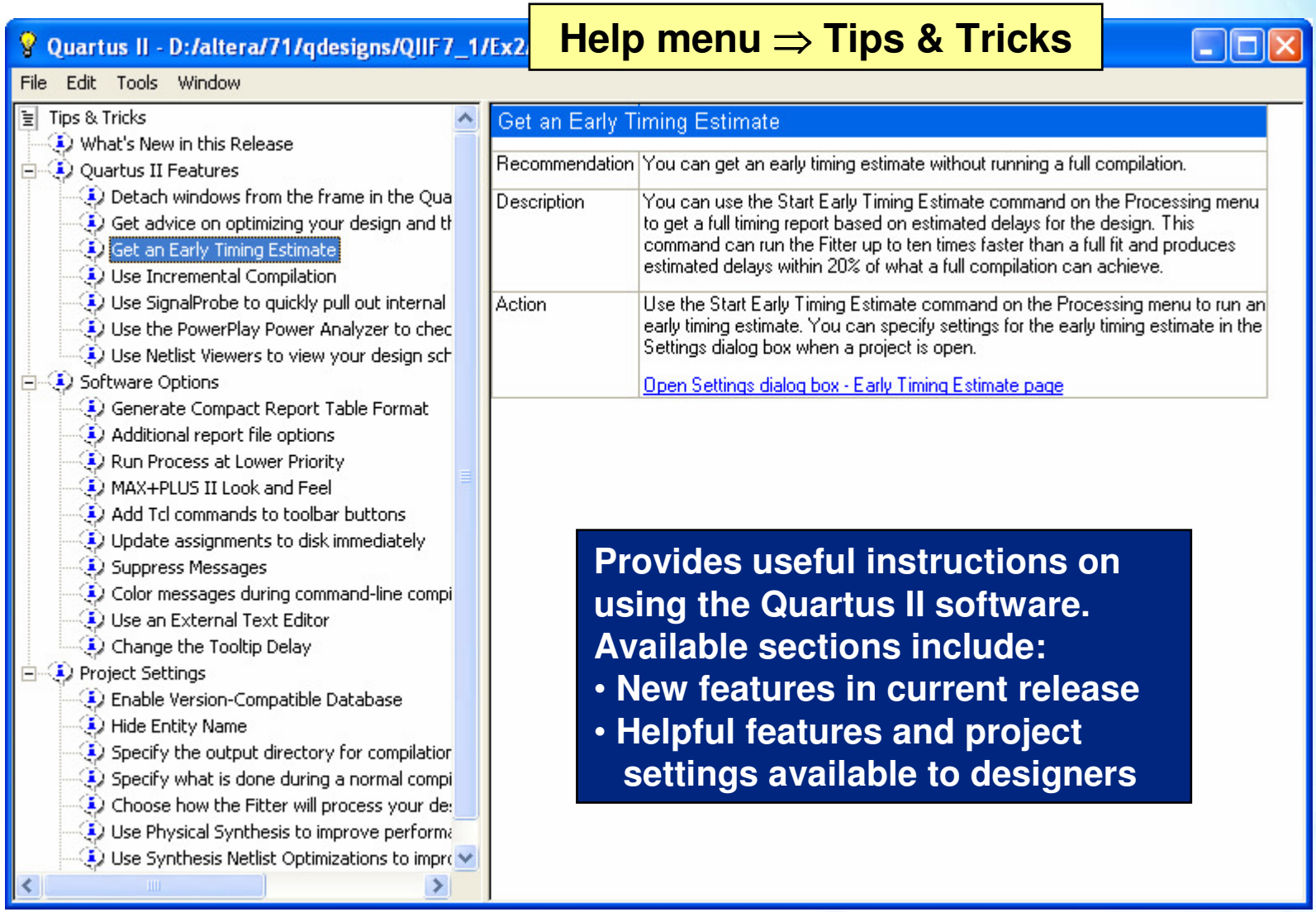
View menu ⇒ Utility Windows ⇒ Tcl Console



- Execute from command-line using Tcl shell
  - `quartus_sh --shell`

Tips & Tricks Advisor

Help menu ⇒ Tips & Tricks



Provides useful instructions on using the Quartus II software. Available sections include:

- **New features in current release**
- **Helpful features and project settings available to designers**

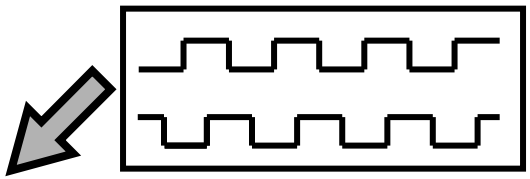
Quartus II Software Design Series: Foundation

Design Methodology

Typical PLD Design Flow

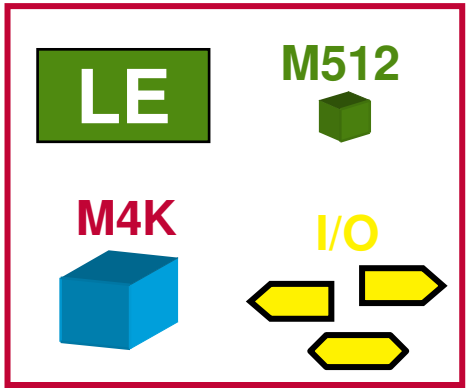
Design entry/RTL coding

- Behavioral or structural description of design



RTL simulation

- Functional simulation (ModelSim®, Quartus II Simulator)
- Verify logic model & data flow (no timing delays)



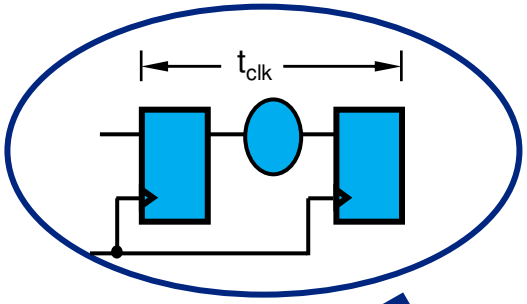
Synthesis (Mapping)

- Translate design into device specific primitives
- Optimization to meet required area & performance constraints
- Quartus II Synthesis, Precision Synthesis, Synplify/Synplify Pro, Design Compiler FPGA

Place & route (Fitting)

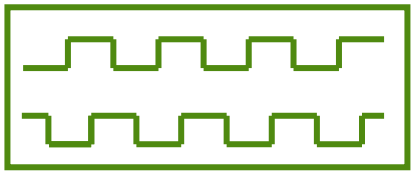
- Map primitives to specific locations inside Target technology with reference to area & performance constraints
- Specify routing resources to be used

Typical PLD Design Flow



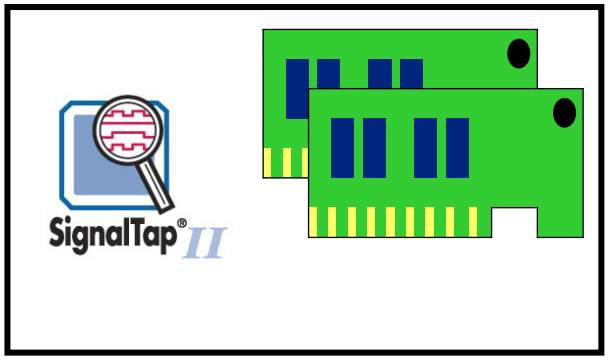
Timing analysis (Classic Analyzer or TimeQuest)

- Verify performance specifications were met
- Static timing analysis



Gate level simulation

- Timing simulation
- Verify design will work in target technology



PC board simulation & test

- Simulate board design
- Program & test device on board
- Use **SignalTap II** Logic Analyzer or other on-chip tools for debugging

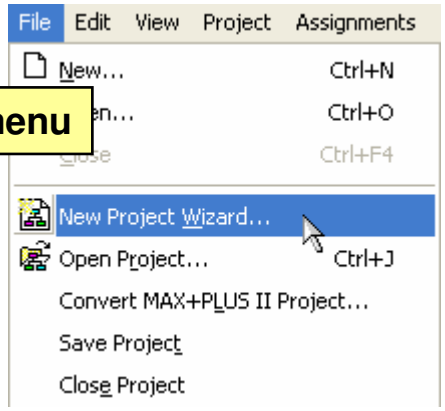
Quartus II Software Design Series: Foundation

Quartus II Projects

Quartus II Projects

- Description
  - Collection of related design files & libraries
  - Must have a designated top-level entity
  - Target a single device
  - Store settings in Quartus II Settings File (.QSF)
- Create new projects with New Project Wizard
  - Can be created using Tcl scripts

New Project Wizard



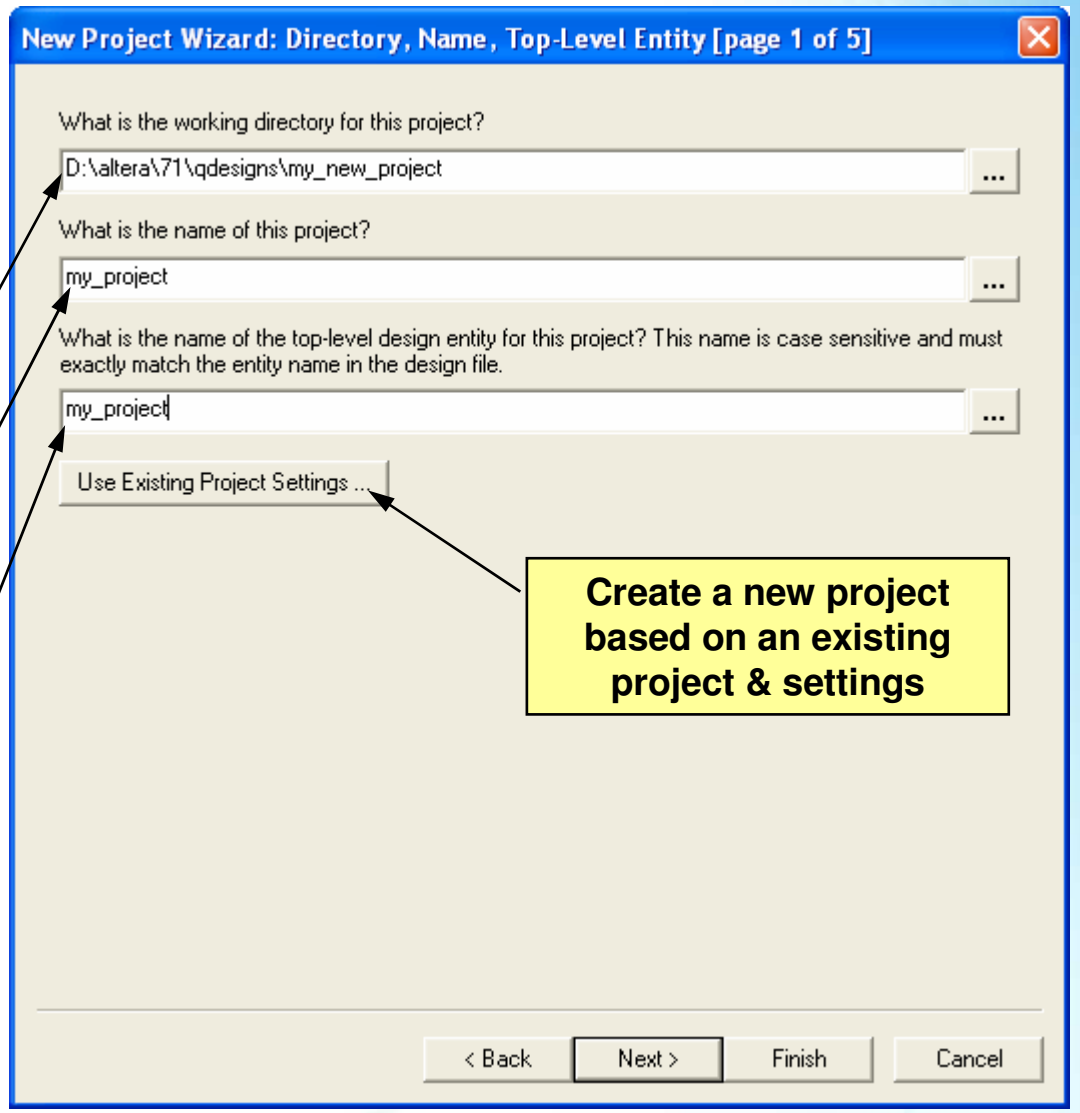
File menu

Select working directory

Name of project can be any name; recommend using top-level file name

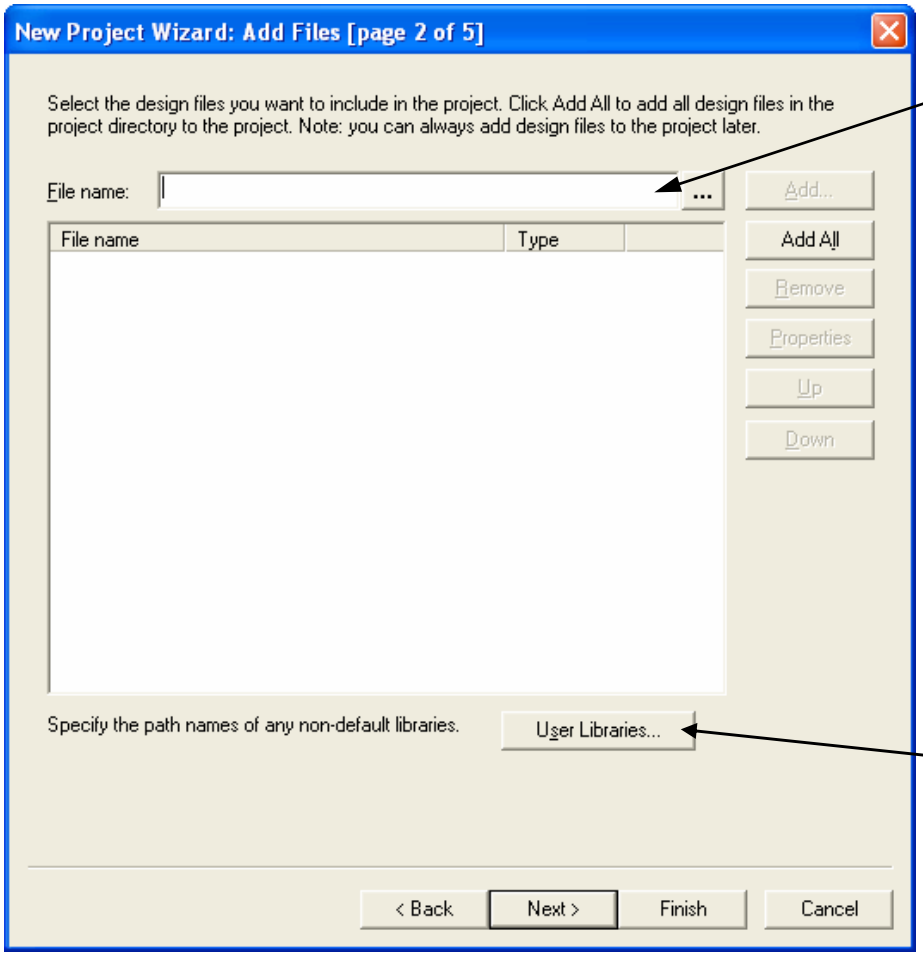
Top-level entity does not need to be the same name as top-level file name

Tcl: project_new <project_name>



Create a new project based on an existing project & settings

Add Files



Add design files

- Graphic (.BDF, .GDF)
- AHDL
- VHDL
- Verilog
- EDIF

Notes:

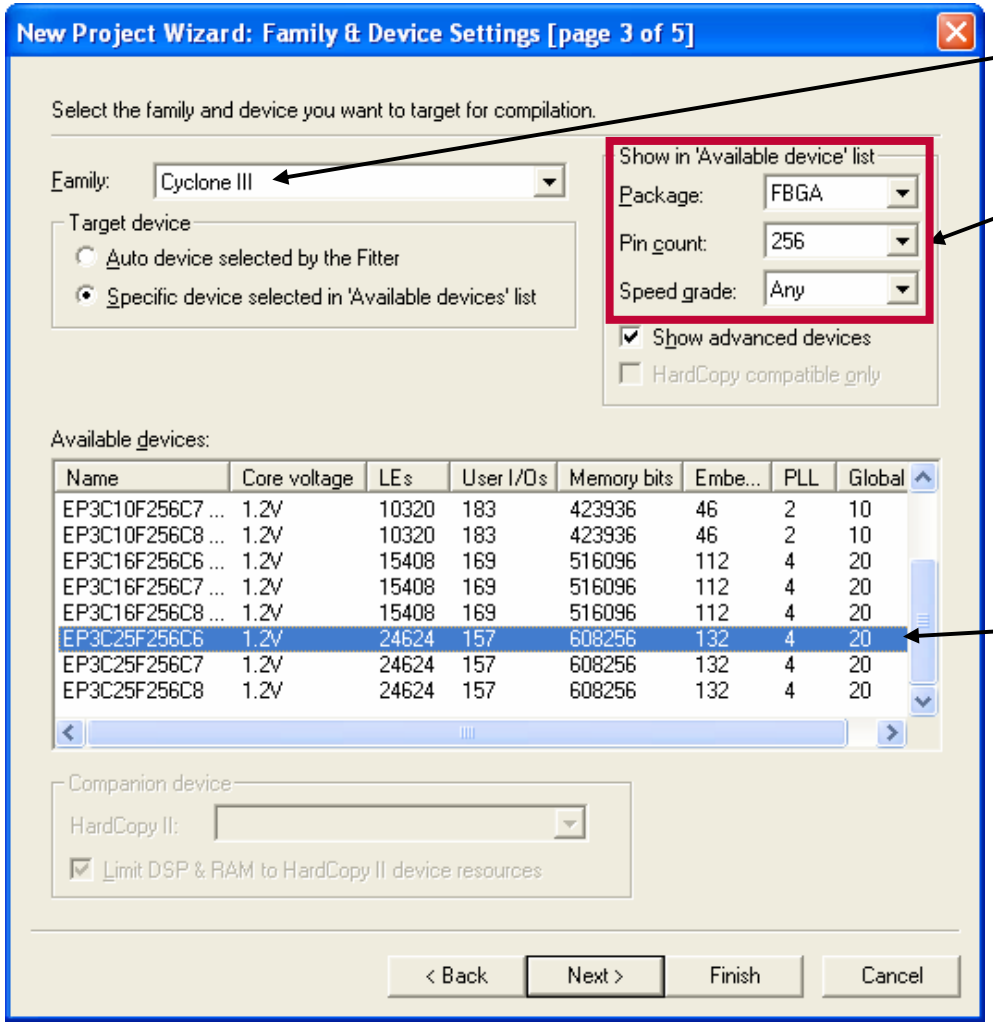
- *Files in project directory do not need to be added*
- *Add top-level file if filename & entity name are not the same*
- *Absolute & relative paths are supported*

Add user library pathnames

- User libraries (any directory containing files)
- MegaCore®/AMPP℠ libraries
- Pre-compiled VHDL packages

Tcl: set_global_assignment –name VHDL_FILE* <filename.vhd>
Tcl: set_global_assignment –name USER_LIBRARIES <library_path_name>
**** Replace with VERILOG_FILE, EDIF_FILE, AHDL_FILE or BDF_FILE***

Device Selection



Choose device family and filter results

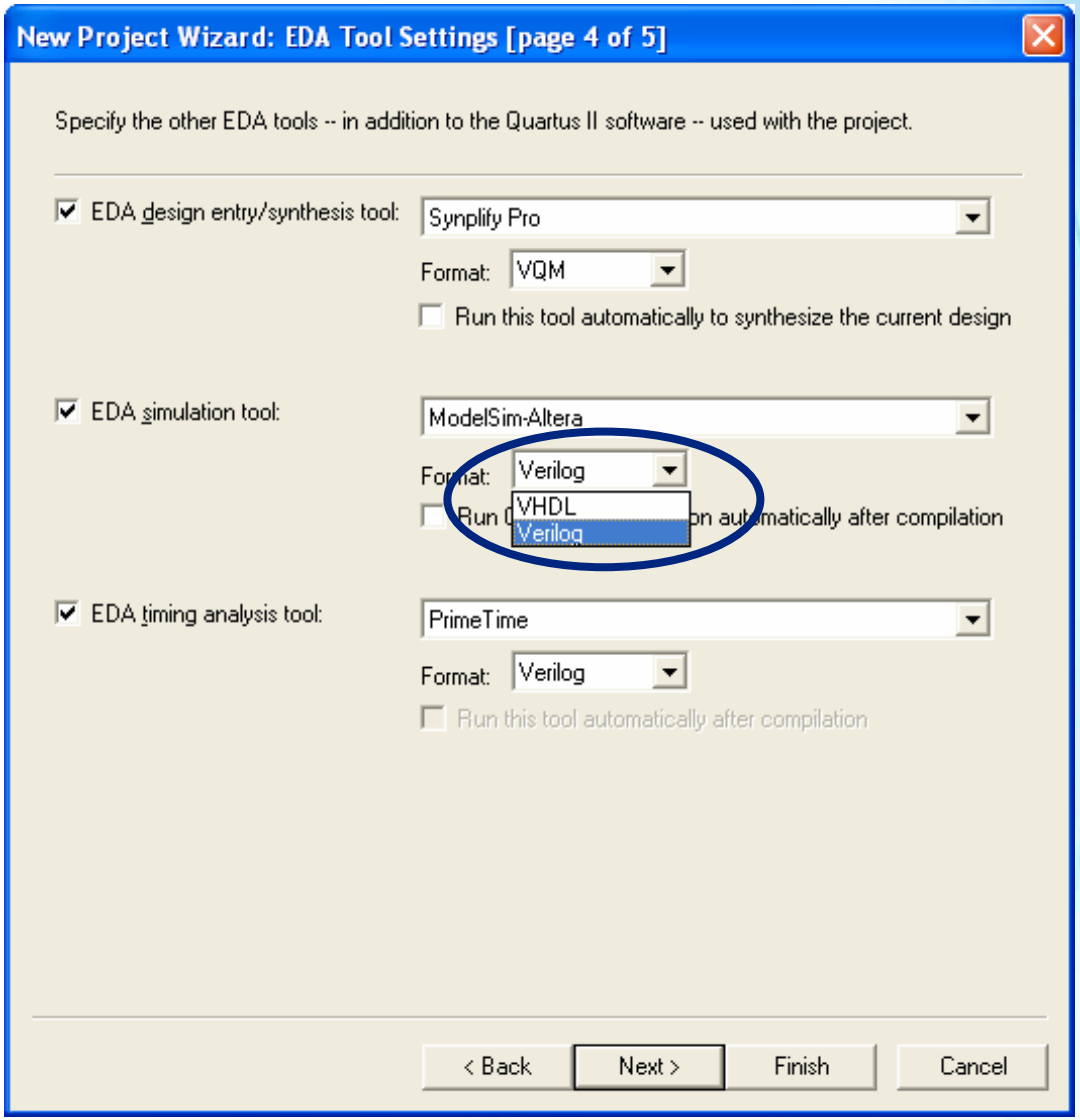
Choose specific part number from list or let Quartus II choose smallest, fastest device based on filter criteria

Tcl: set_global_assignment –name FAMILY "device family name"
Tcl: set_global_assignment –name DEVICE <part_number>

EDA Tool Settings

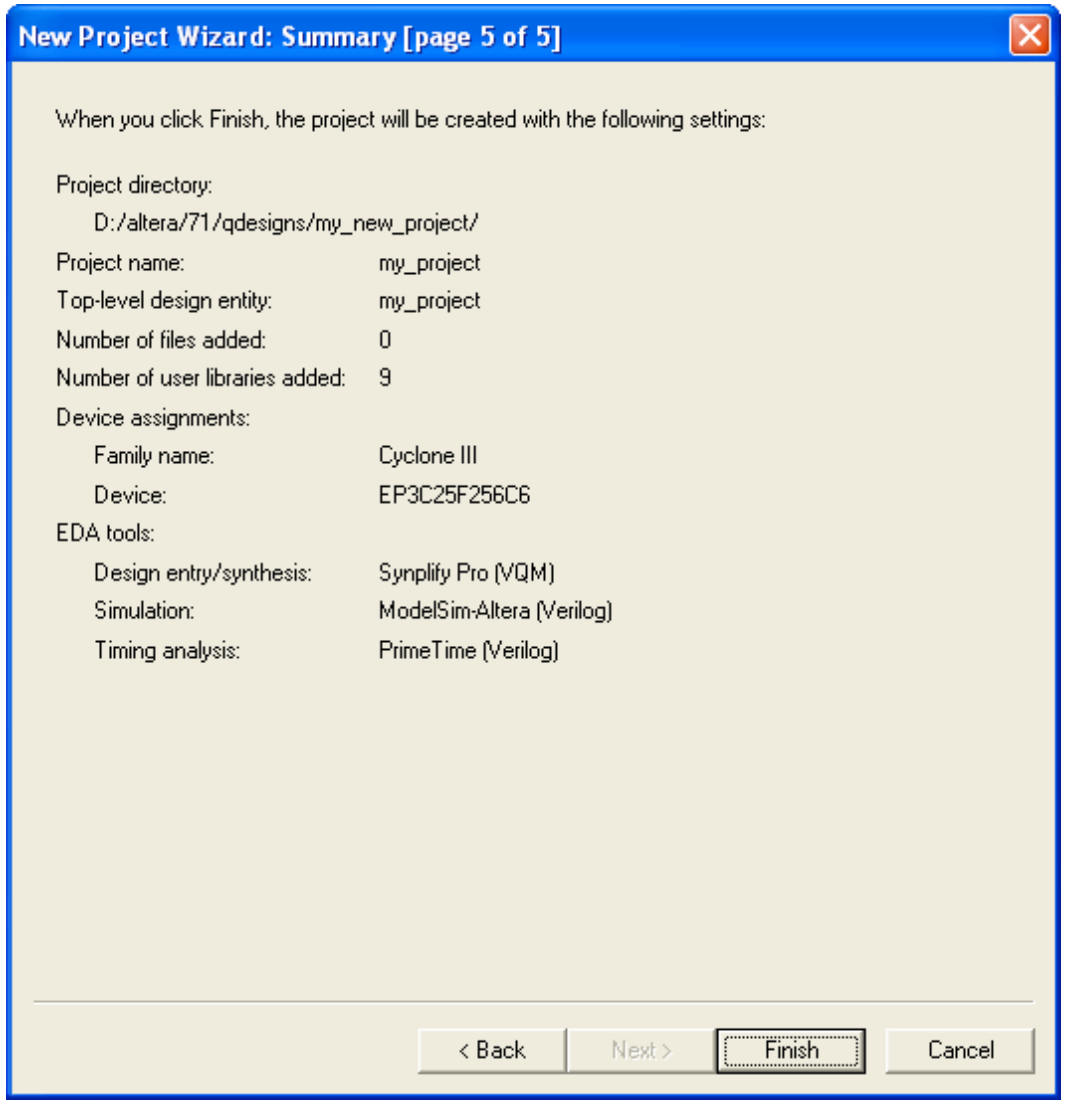
Choose EDA tools & file formats

Add or change settings later



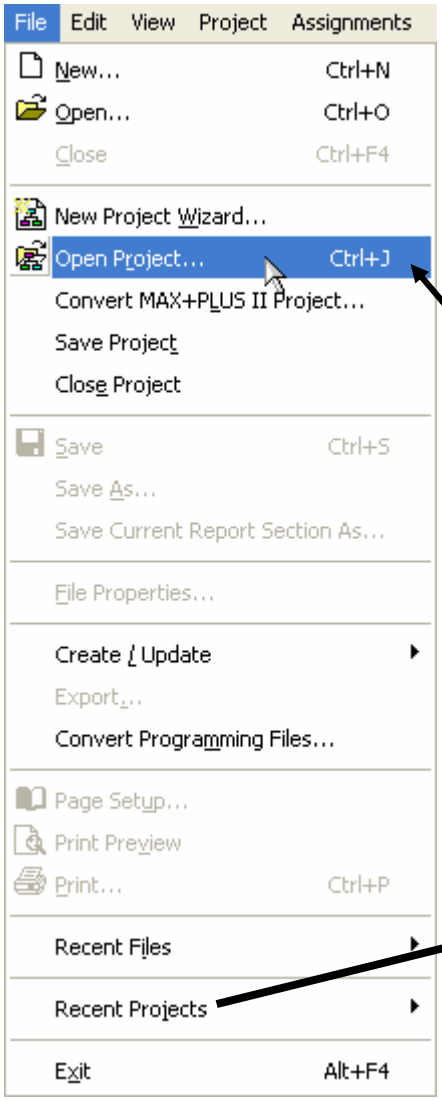
See handbook for Tcl command format

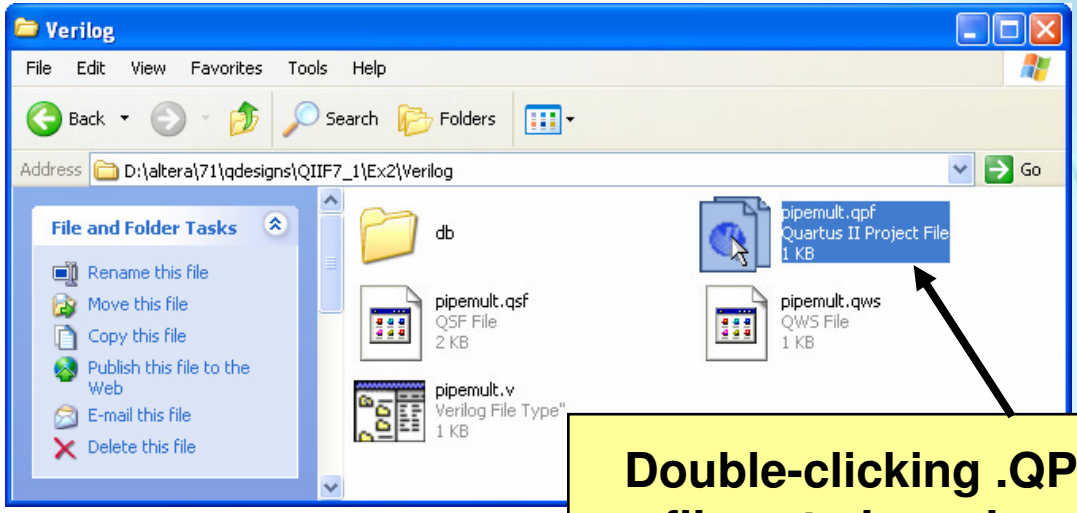
Done!



Review results & click Finish

Opening an Existing Project





Double-clicking .QPF file auto launches Quartus II software

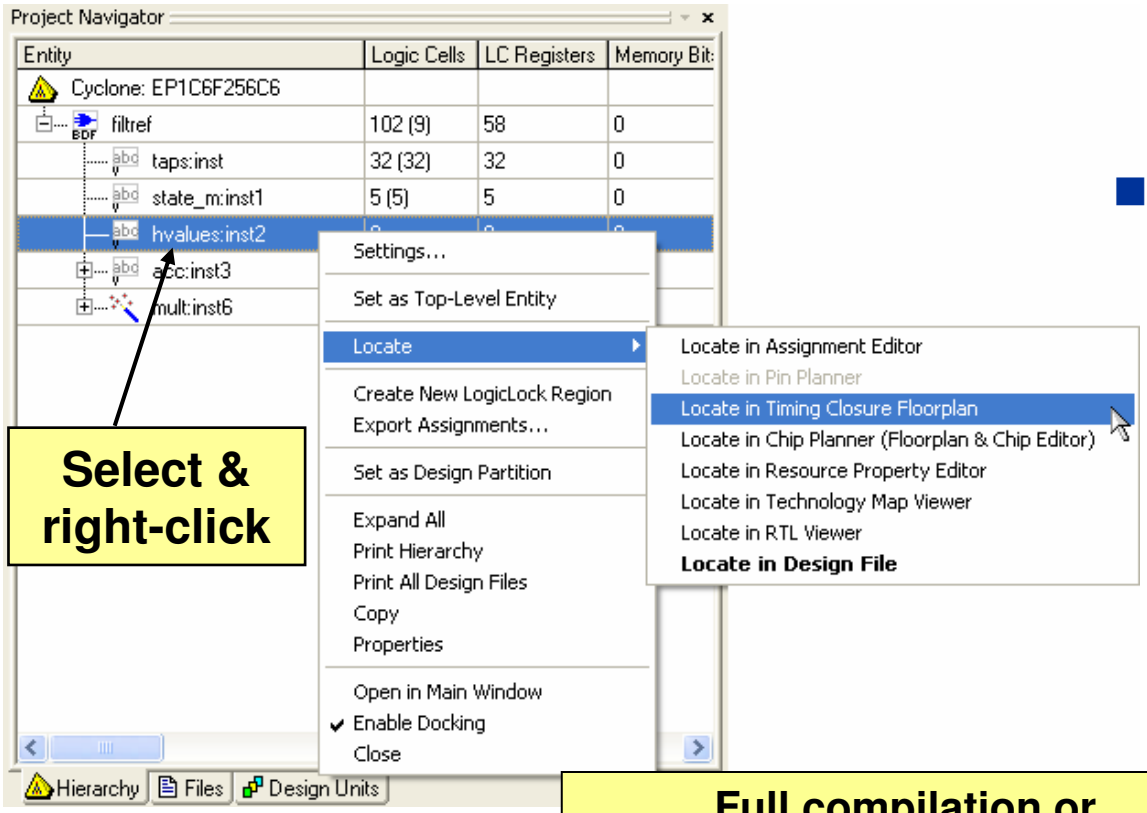
File ⇒ Open Project…

OR

Select from most recent projects list

Tcl: project_open <project_name>

Project Navigator – Hierarchy Tab

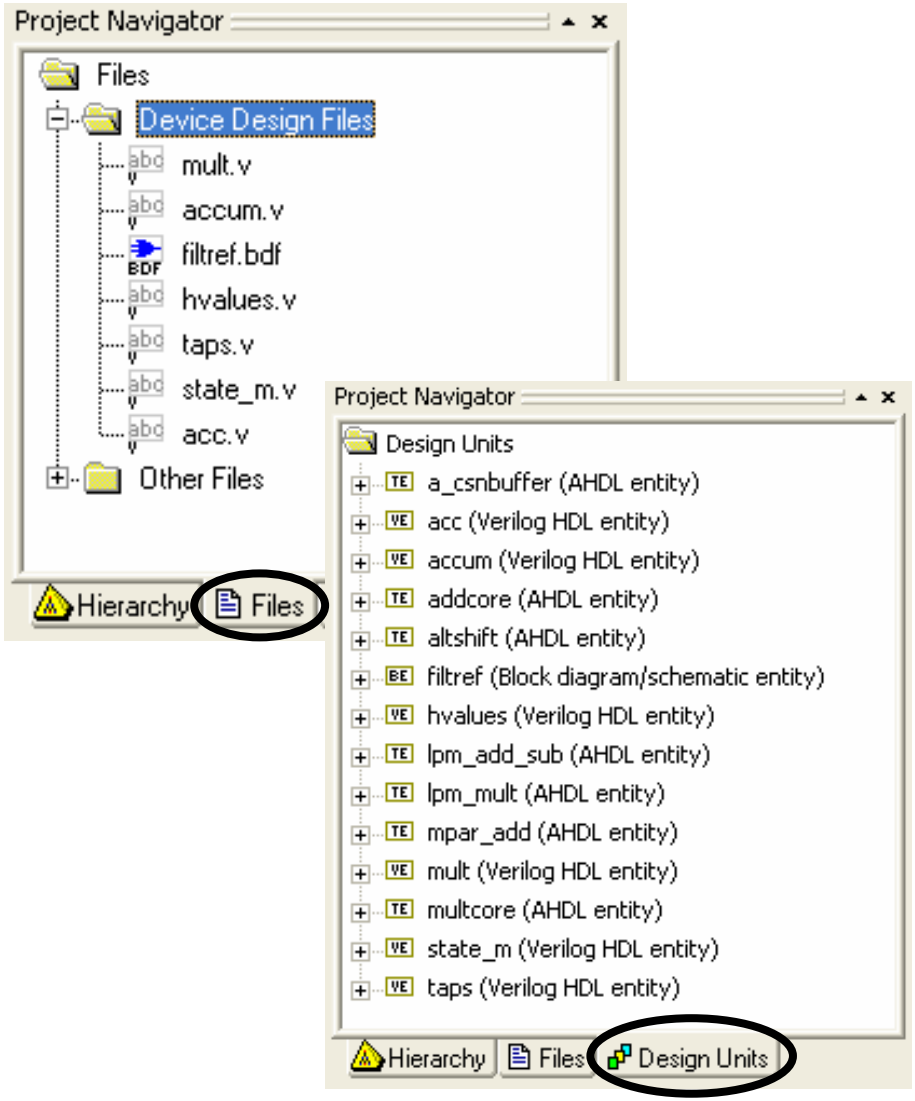


Select & right-click

Full compilation or Processing menu ⇒ Start ⇒ Start Analysis & Elaboration

- Displays project hierarchy after project is analyzed
- Uses
  - Set top-level entity
  - Set incremental design partition
  - Make entity-level assignments
  - Locate in design file or viewers/floorplans
  - View resource usage

Files & Design Units Tabs



- Files tab
  - Shows files explicitly added to project
  - Uses
    - Open files
    - Remove files from project
    - Set new top-level entity
    - Specify VHDL library
    - Select file-specific synthesis tool
  - Can also use **Project ⇒ Add/Remove Files in Project…**
- Design Units tab
  - Displays design unit & type
    - VHDL entity
    - VHDL architecture
    - Verilog module
    - AHDL subdesign
    - Block diagram filename
  - Expanded unit displays file which instantiates design unit

Quartus II Project Files

- Quartus II Project File (.QPF)
- Quartus II Defaults File (.QDF)
- Quartus II Settings File (.QSF)

- Synopsys Design Constraints (.SDC)
  - Holds timing constraints
  - Discussed later

Project & Default Files

- Quartus II Project File (QPF)
  - Quartus II version
  - Time stamp
  - Active revision(s)

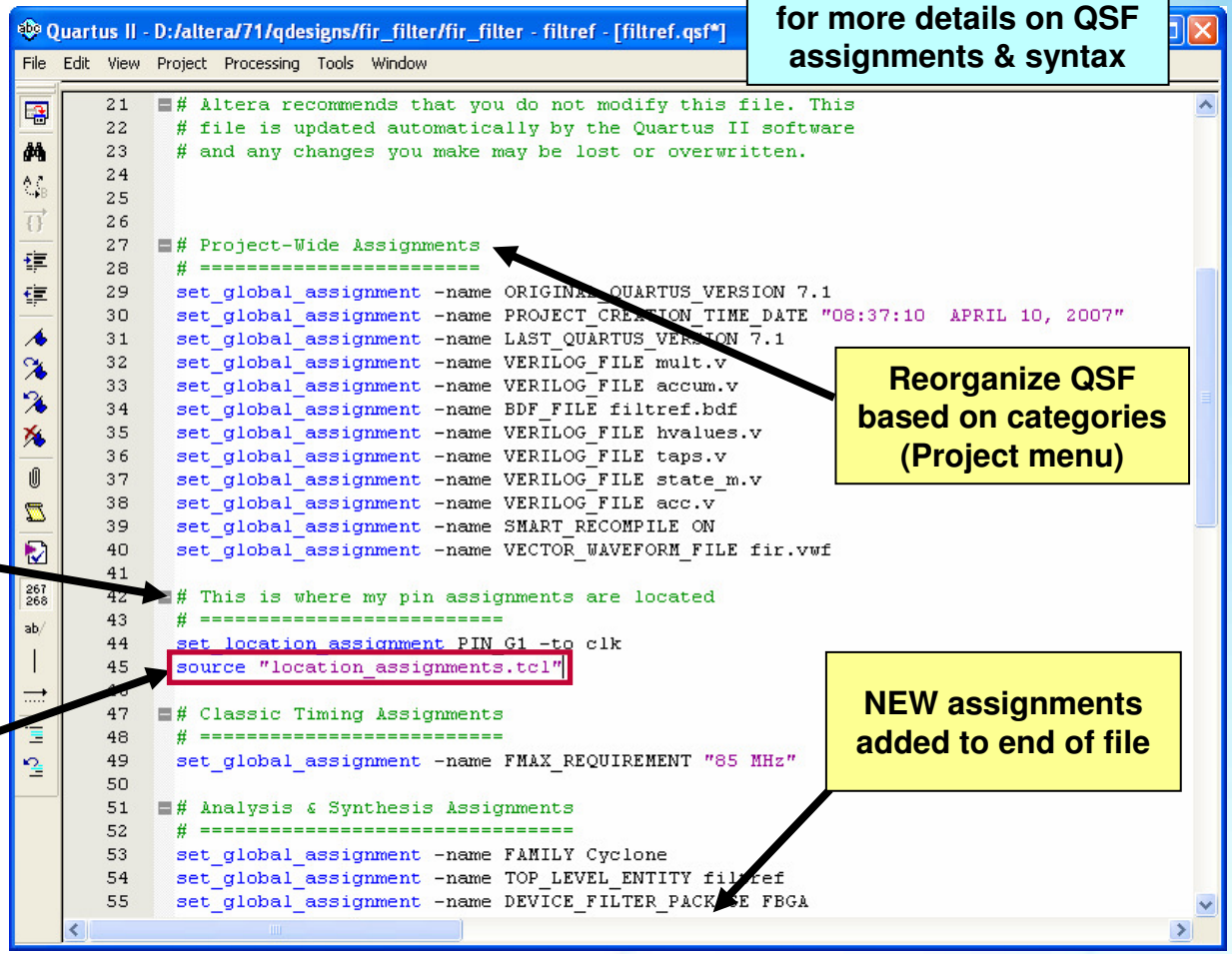
fir_filter.QPF

```
QUARTUS_VERSION = "7.1"
DATE = "14:31:04  May 02, 2007"


# Active Revisions

PROJECT_REVISION = "filtref"
PROJECT_REVISION = "filtref_new"
```

- Quartus II Defaults Files (QDF)
  - Stores Quartus II project setting & assignment defaults
  - Example names: *assignment_defaults.qdf* or *<revision_name>_assignment_defaults.qdf*
  - Found in local project or *altera\<version>\quartus\bin* directory
    - Copy in local project directory read before original in bin

Quartus II Settings File (QSF)

- Stores all settings & assignments (constraints)
- Uses Tcl syntax
- Can be edited manually by user



Note: See Appendix for more notes on using QSF file.

Constraint File Priority

1. QSF
2. Revision-specific QDF file located in project directory
   - *<revision_name>_ assignment_defaults.qdf*
   - Created automatically in project directory when revision opened in new version of the Quartus II software
3. QDF located in project directory
   - *assignment_defaults.qdf*
   - Created automatically in project directory when project archived & restored
4. QDF located in Quartus II *\bin* directory

Project Management

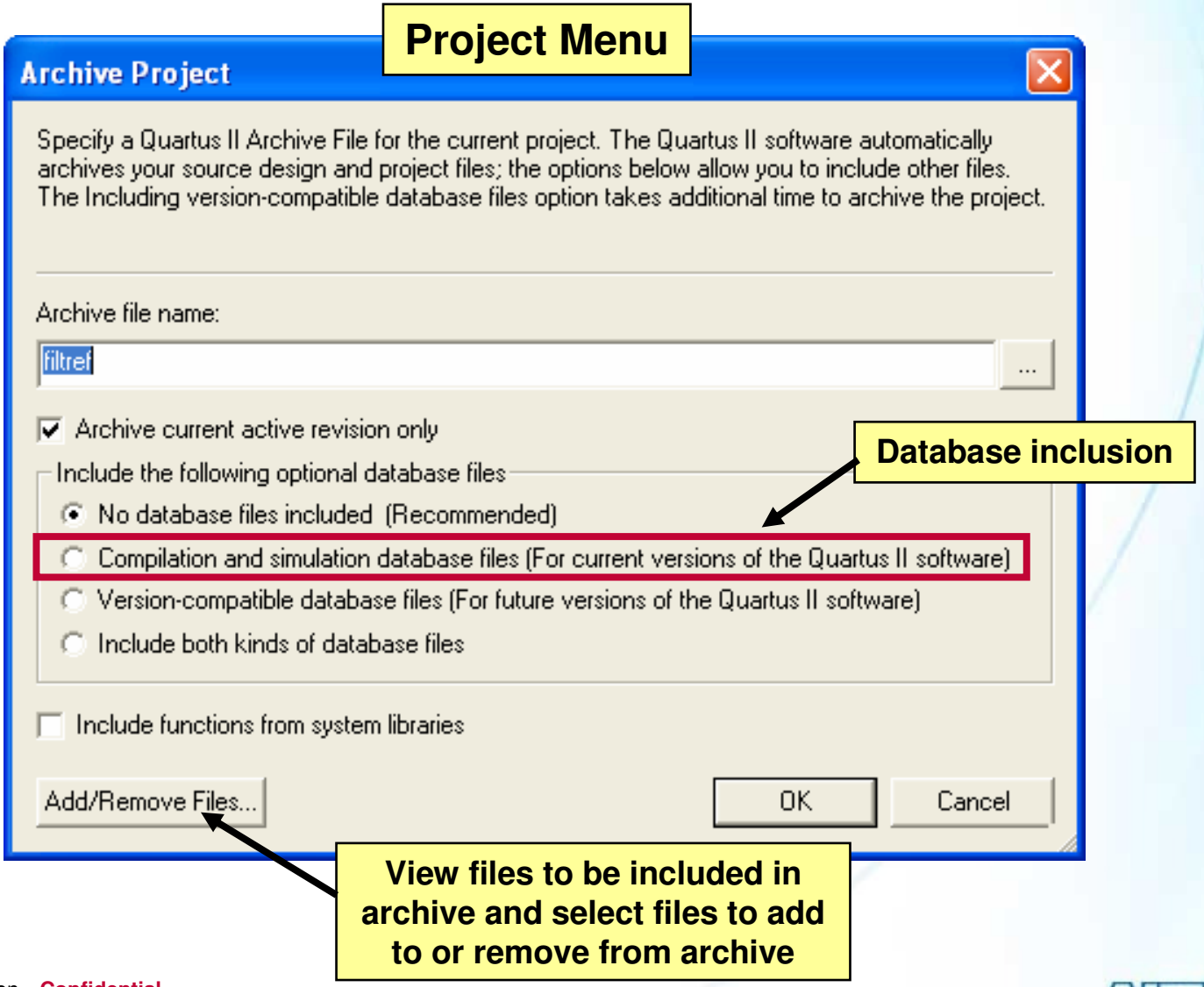
- Project archive & restore
- Project copy
- Revisions

Project Archive

- Creates 2 files
  - Compressed Quartus II Archive File (.QAR)
    - Includes design files, QPF file, & QSF file(s)
    - Option to include databases (db folder in project directory)
      - Recompile necessary if databases not included
    - Creates local QDF file for archive
  - Archive activity log (.QARLOG)
- Example Uses
  - File storage (e.g. version control)
  - Project handoff
    - Useful for sending to Altera support
- Design files referenced from user libraries are included in archive

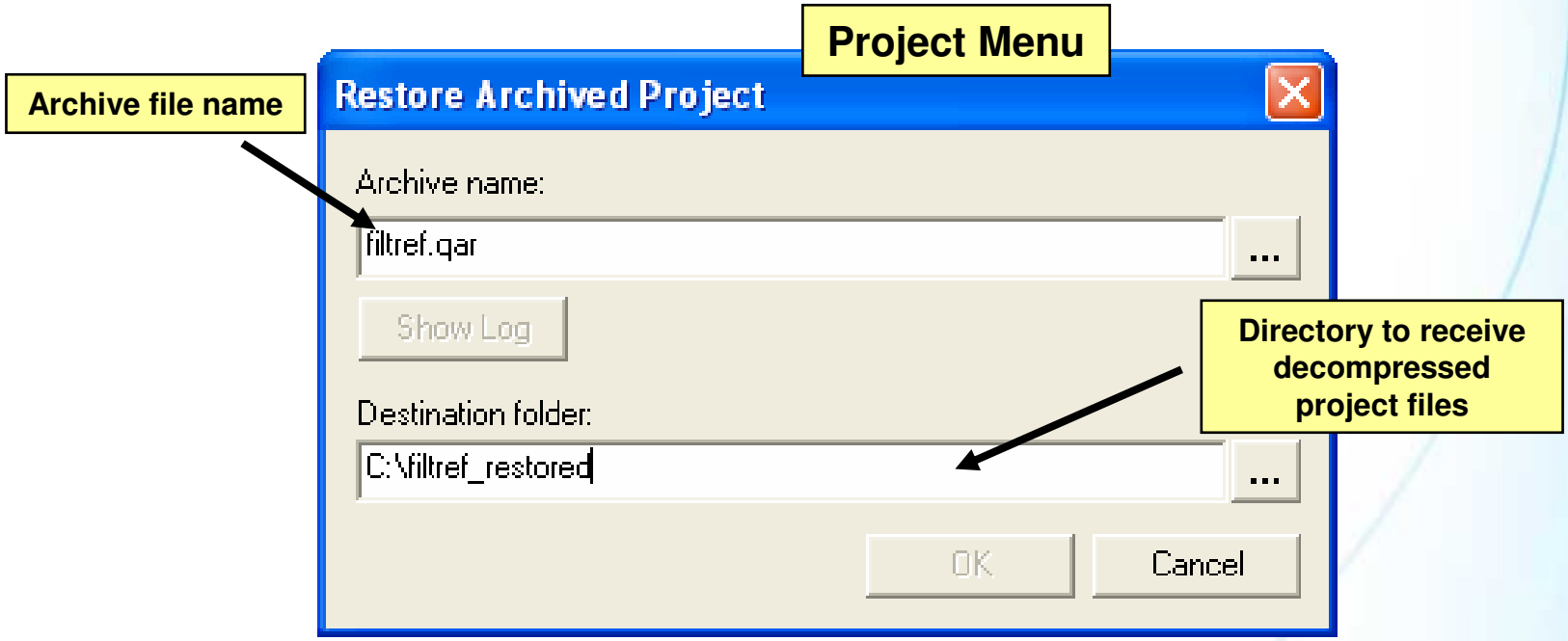
Tcl: project_archive <project_name>

Project Archive (cont.)



Project Restore

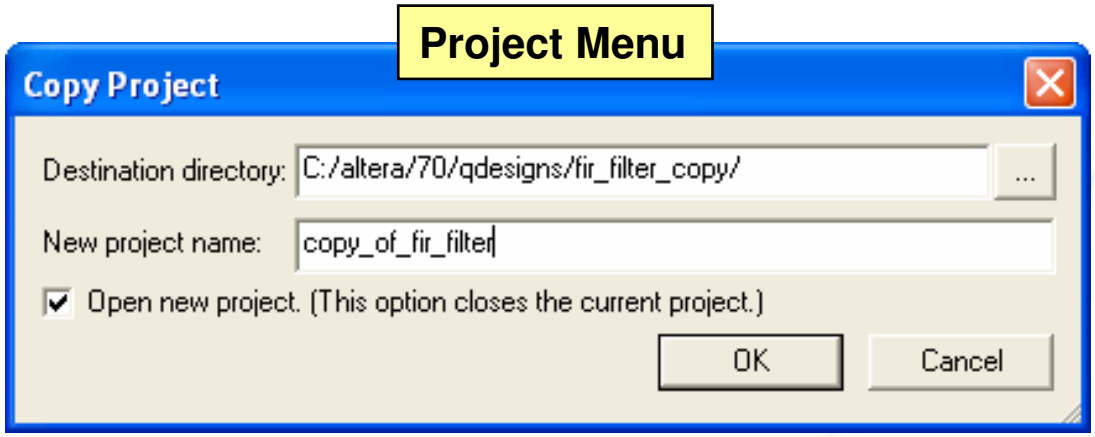
- Decompresses .QAR into specified directory



Tcl: project_restore <archive_file>

Project Copy

- Copies & save duplicate of project in new directory
  - Project file (.QPF)
  - Design files
  - Settings files
- Example Use
  - Duplicating work before editing design files
- User libraries are not copied
- New QDF not created; only copies QDF if it exists

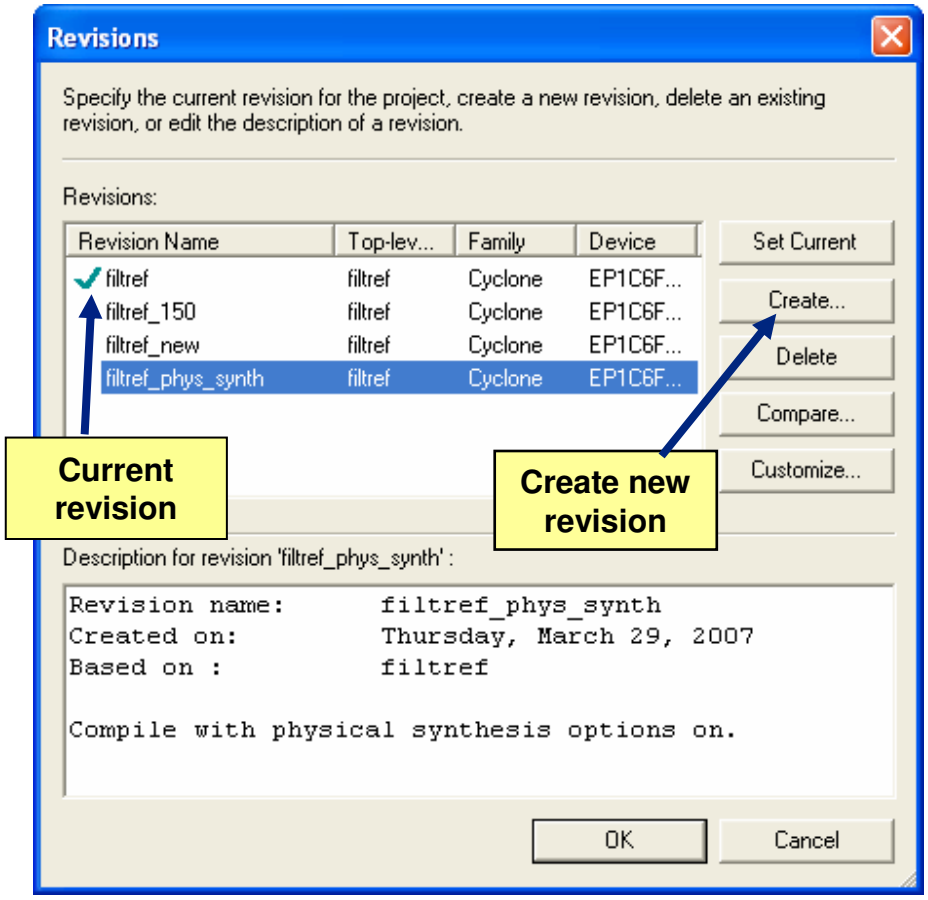


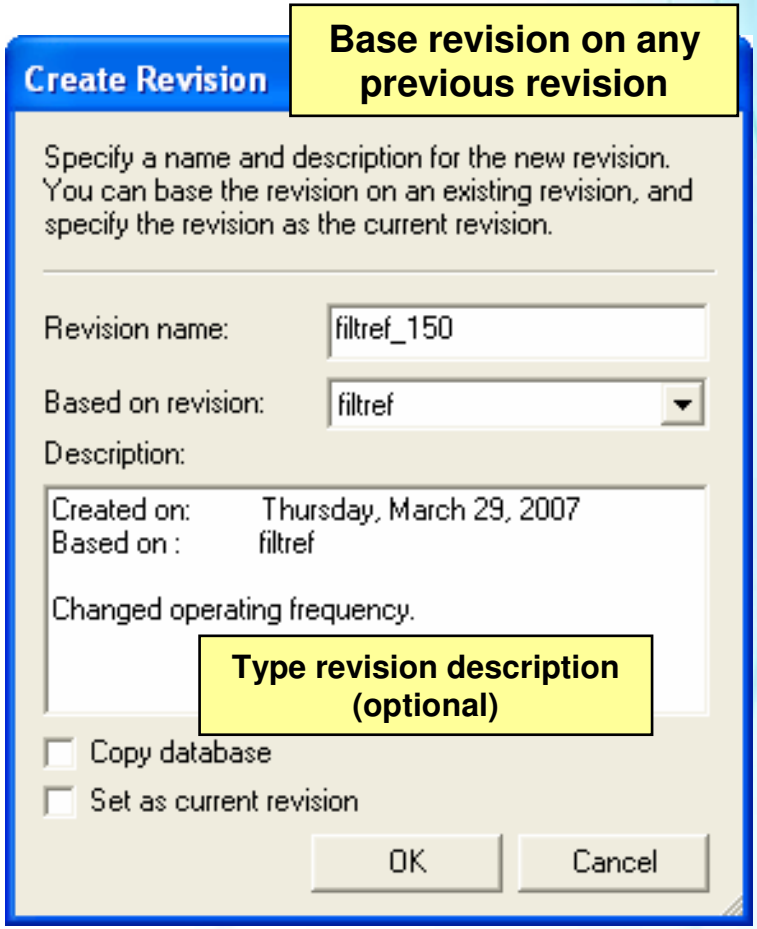
Revisions

- Explore new sets of constraints or compile options without losing previous work
  - Allows designer to try different options on same design files
- Compare results between revisions

Creating a Revision

- Project ⇒ Revisions

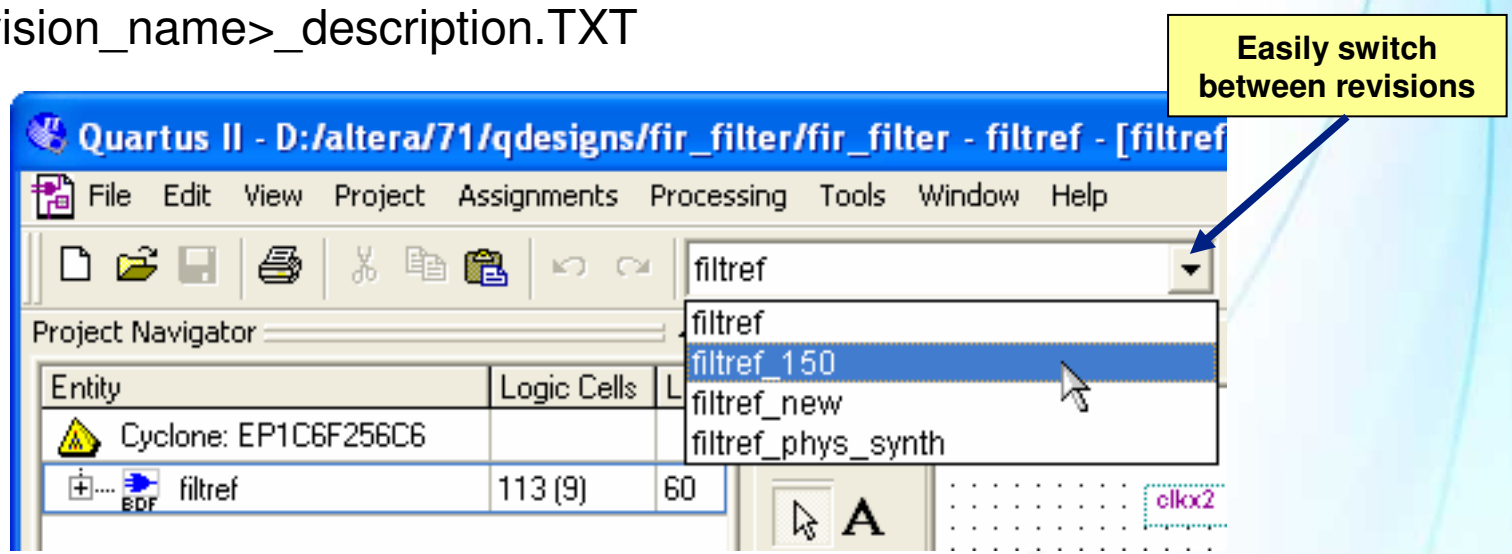




Tcl: create_revision <revision_name>

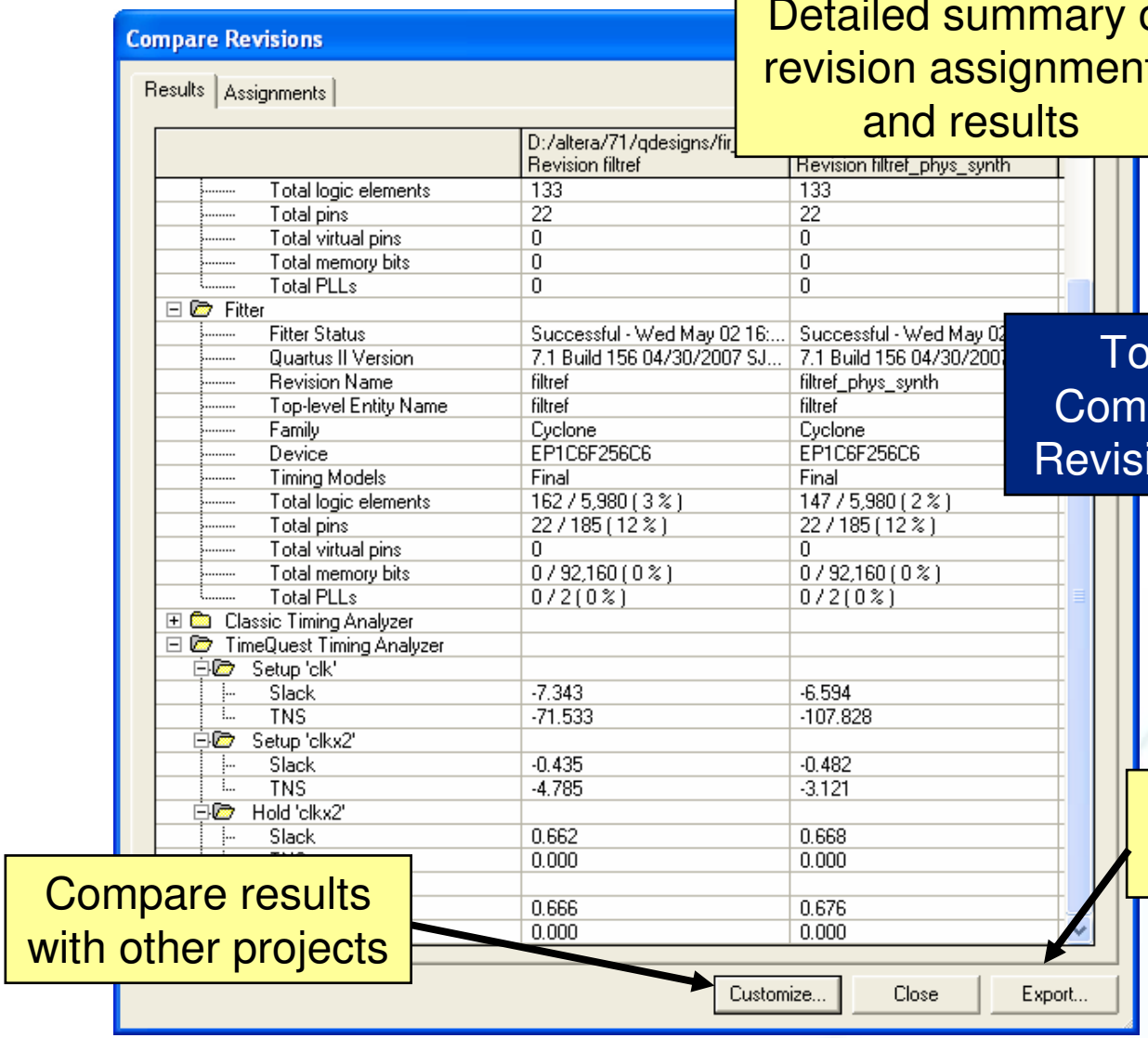
Project Revision Support

- QSF created for each revision
  - <revision_name>.QSF
- Active revision names stored in QPF
- Text file created for each revision
  - <revision_name>_description.TXT



Tcl: project_open –revision <revision_name> <project_name>
Tcl: set_current_revision <revision_name>

Compare Revisions



Exercise 1 Demonstration

Projects Entry Summary

- Projects necessary for design processing
- Use New Project Wizard to create new projects
- Use Project Navigator to study file & entity relationships within project
- Project archive, copy, and revisions provide easy-to-use project management

Project Support Resources

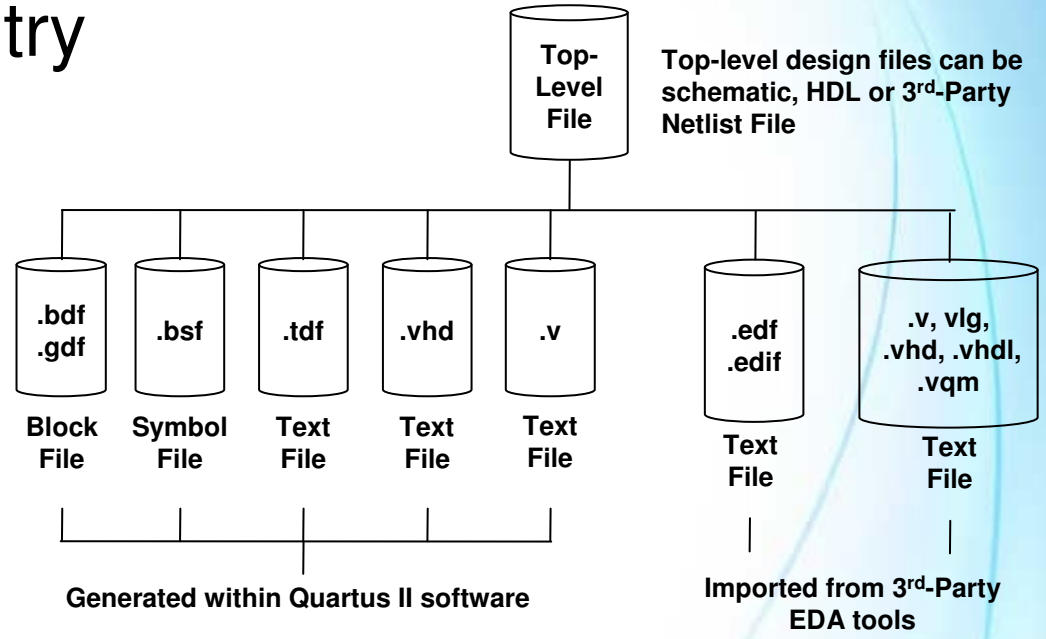
- "Managing Quartus II Projects" chapter in Volume 2 of the Quartus II Handbook

Quartus II Software Design Series: Foundation

Design Entry

Design Entry Methods

- Quartus II design entry
  - Text editor
    - AHDL
    - VHDL
    - Verilog
  - Schematic editor
    - Block Diagram File
    - Graphic Design File
  - Memory editor
    - HEX
    - MIF
- 3rd-party EDA tools
  - EDIF 2 0 0
  - Verilog Quartus Mapping (.VQM)
- Mixing & matching design files allowed



Text Design Entry

- Quartus II Text Editor features
  - Block commenting
  - Line numbering in HDL text files
  - Bookmarks
  - Preview/editing of full design and construct HDL templates
  - Syntax coloring
  - Find/replace text
  - Find and highlight matching delimiters
  - Function collapse/expand
  - Edited but unsaved filenames appear with an asterisk (*) next to the filename in the GUI
- Enter text description
  - AHDL (.tdf)
  - VHDL (.vhd, .vhdl)
  - Verilog (.v, .vlg, .Verilog, .vh)

Verilog & VHDL

- **VHDL**- VHSIC hardware description language
  - IEEE Std 1076 (1987 & 1993) supported
  - IEEE Std 1076.3 (1997) synthesis packages supported
- **Verilog**
  - IEEE Std 1364 (1995 & 2001) & 1800 (SystemVerilog) supported

- Create in the Quartus II editor or any standard text editor
- Use Quartus II integrated synthesis to synthesize
- View supported commands in on-line help

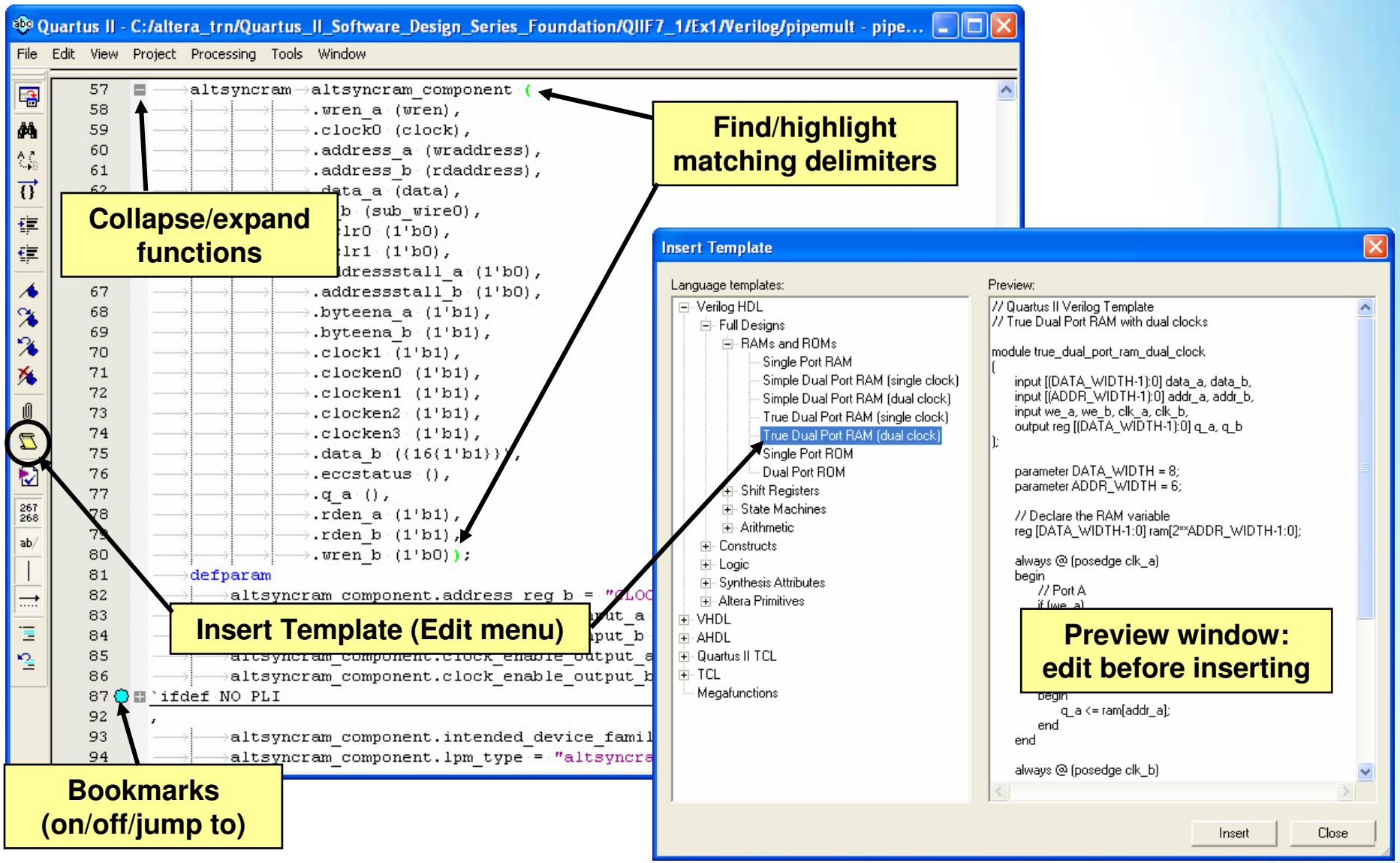
Learn more about HDL in Altera HDL customer training classes

AHDL

- Altera hardware description language
  - High-level hardware behavior description language
  - Used in Altera megafunctions
  - Uses boolean equations, arithmetic operators, truth tables, conditional statements, etc.

- Create in the Quartus II editor or any standard text editor

Text Editor Features



Schematic Design Entry

- Full-featured schematic design capability
- Schematic Editor uses
  - Create simple test designs to understand the functionality of an Altera megafunction
    - PLL, LVDS I/O, memory, etc…
  - Create top-level schematic for easy viewing & connection
    - Convert Block Diagram File (.BDF) to HDL file (VHDL/Verilog) or image file (.JPG or .BMP)

Note: ***Please see the Appendix for a more detailed discussion of the Block Diagram Editor and schematic entry.***

Altera Megafunctions

- Pre-made design blocks
- Benefits
  - Configurable settings add flexibility
  - “Drop-in” support to accelerate design entry
  - Pre-optimized for Altera architecture
- Two versions
  - Quartus II megafunctions
  - Intellectual Property (IP) megafunctions

Quartus II Megafunctions

- Free & installed with Quartus II software
  - Non-encrypted functions written in AHDL
  - HDL simulation models installed in Quartus II libraries
- Two types
  - Altera-specific megafunctions (begin with “ALT”)
  - Library of parameterized modules (LPMs)
    - Industry standard logic functions
    - See www.edif.org/lpmweb (EDIF.org archive) for more info
- Examples
  - Multiply-accumulate (ALTMULT_ACCUM)
  - On-chip RAM/ROM (ALTSYNCRAM)
  - PLL (ALTPLL)
  - DDR/QDR memory interface (ALTMEMPHY)
  - Counter (LPM_COUNTER)
  - Comparator (LPM_COMPARE)

IP Megafunctions

- Must purchase license to use in finished design
  - Logic for IP function is encrypted
- Two types
  - MegaCore® IP
    - Developed by Altera
    - Install with Quartus II software or download/install individually from www.altera.com
  - Altera Megafunctions Partner Program (AMPP℠) IP
    - Developed by 3rd-Party IP vendors & certified by Altera
    - Contact vendor for evaluating and licensing function
- All MegaCore functions & some AMPP functions support OpenCore® Plus feature
  - Develop design using free version of core
  - HDL simulation models provided with IP
  - Generate time-limited configuration/programming files

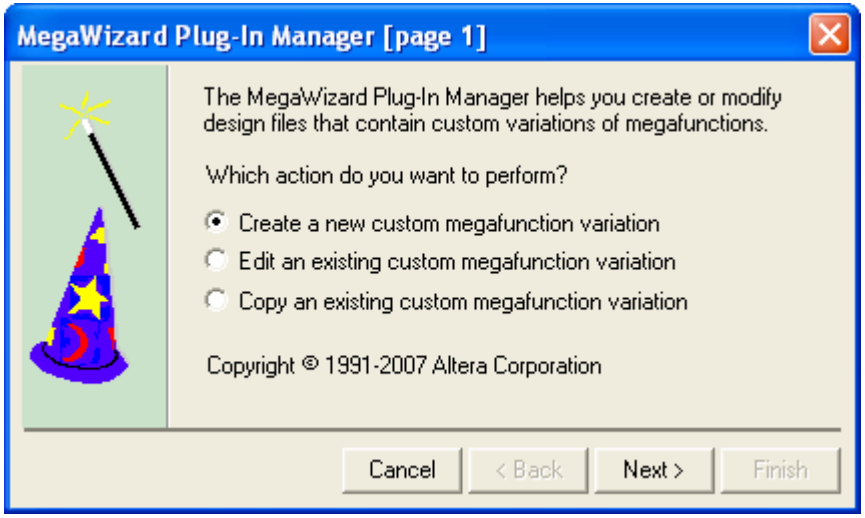
Example MegaCore IP

- Triple-Speed Ethernet MAC
- FIR Compiler
- Fast Fourier Transform
- DDR2 Memory Controller
- CRC Compiler
- PCI Compiler

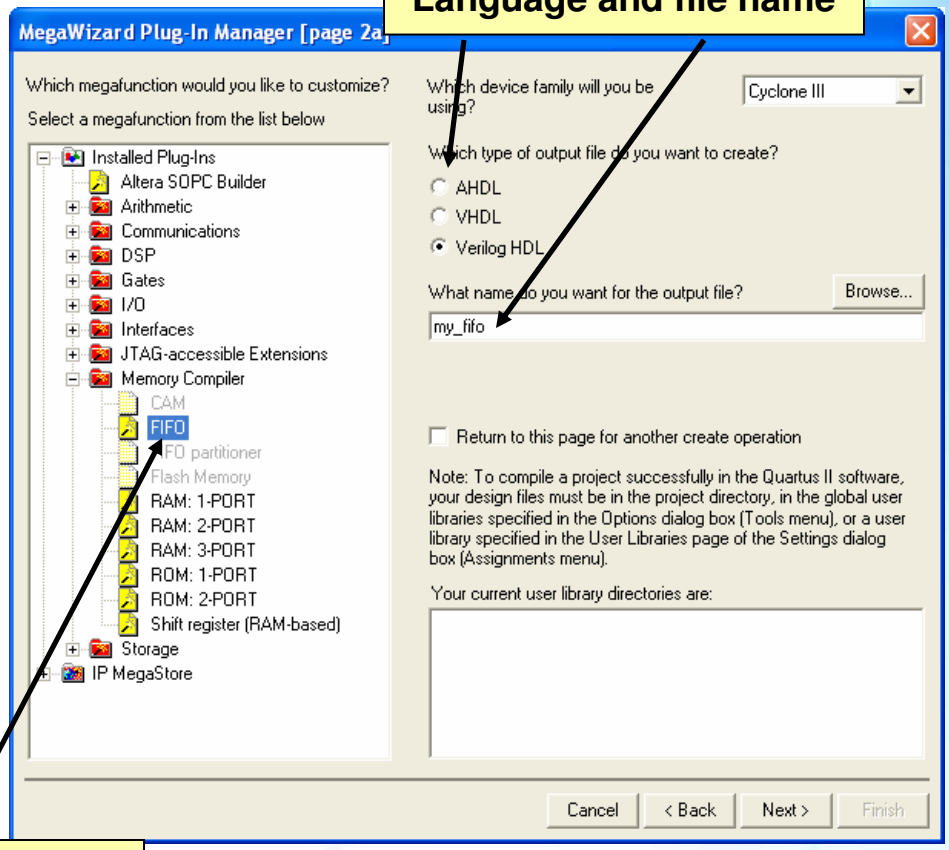
MegaWizard Plug-in Manager

- Eases implementation and configuration of megafunctions & IP

Tools ⇒ MegaWizard Plug-In Manager

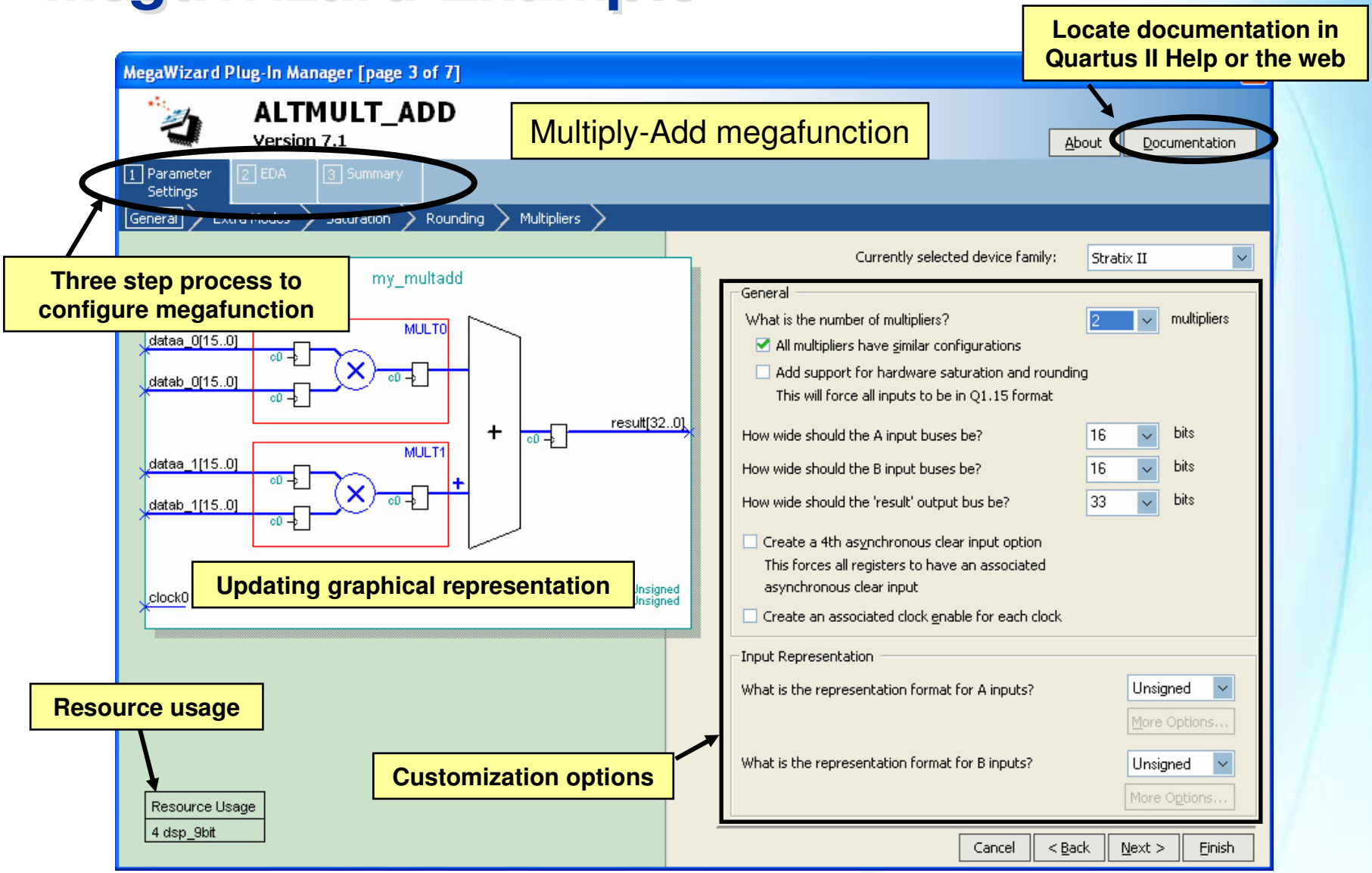


Language and file name

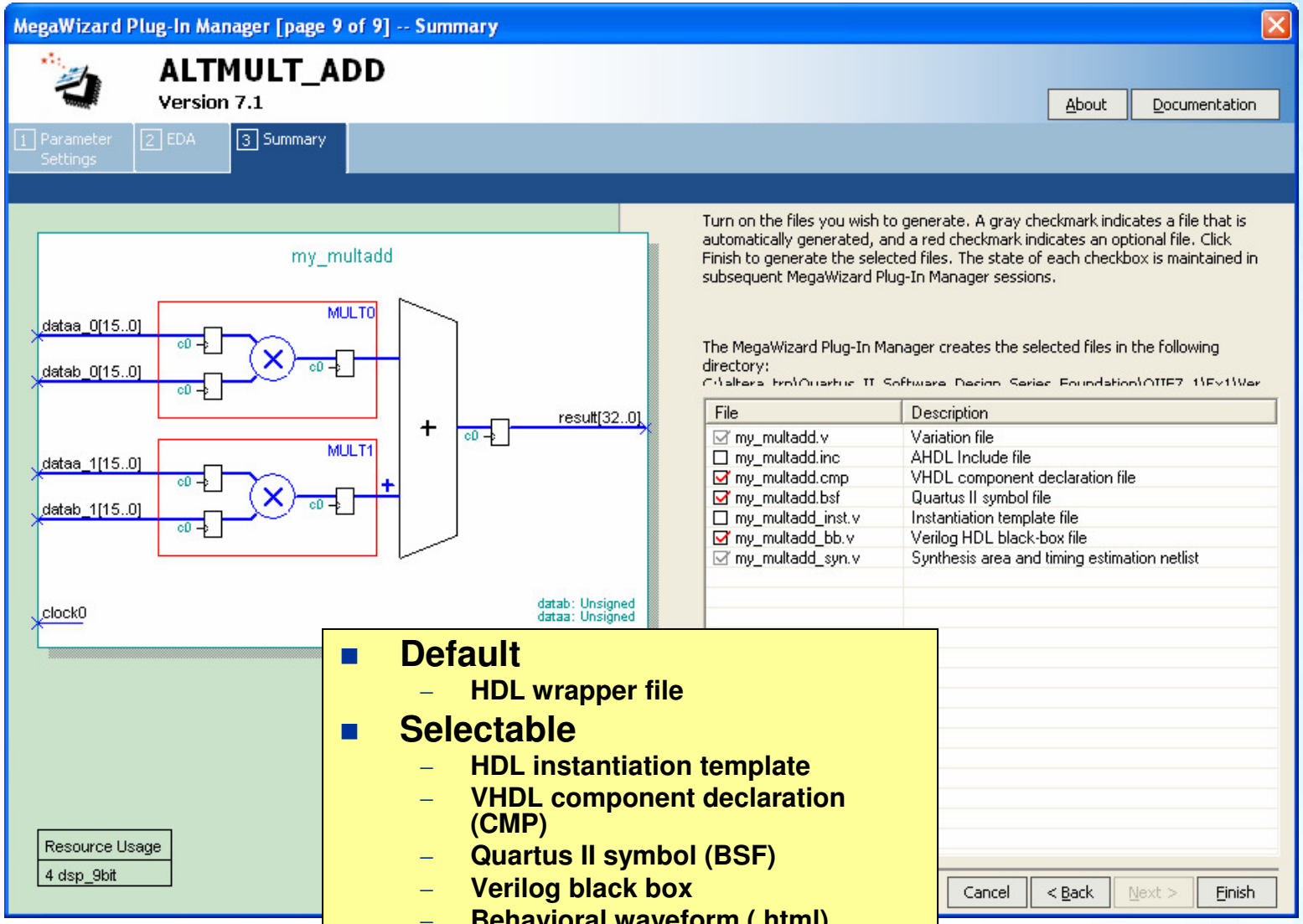


Select megafunction or IP

MegaWizard Example



MegaWizard Output File Selection



- **Default**
  - **HDL wrapper file**
- **Selectable**
  - **HDL instantiation template**
  - **VHDL component declaration (CMP)**
  - **Quartus II symbol (BSF)**
  - **Verilog black box**
  - **Behavioral waveform (.html)**

Memory Editor

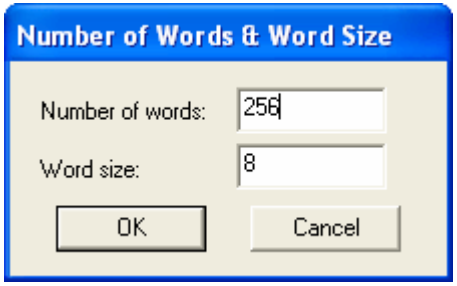
- Create or edit memory initialization files in Intel HEX (.HEX) or Altera-specific (.MIF) format
- Design entry
  - Use to initialize your memory block (ex. RAM, ROM) during power-up
- Simulation
  - Use to initialize memory blocks before simulation or after breakpoints

Create Memory Initialization File

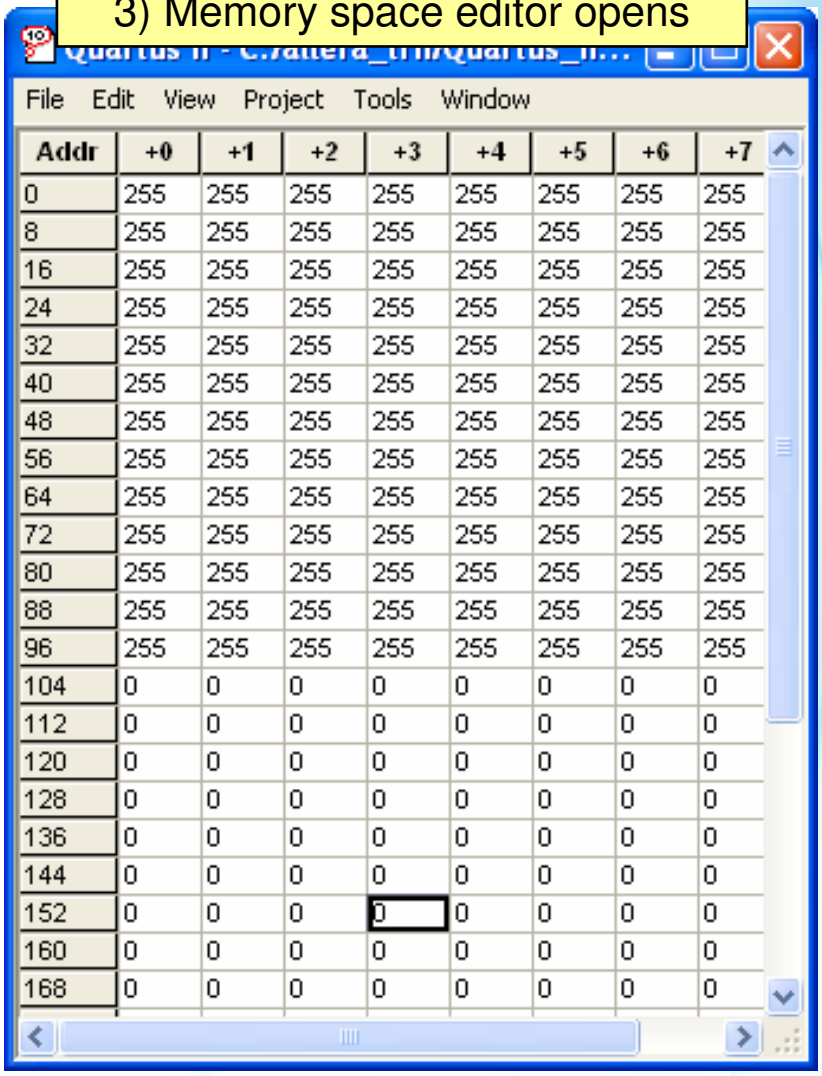
File ⇒ New ⇒ Other Files tab or 

1) HEX format or MIF format

2) Select Memory Size



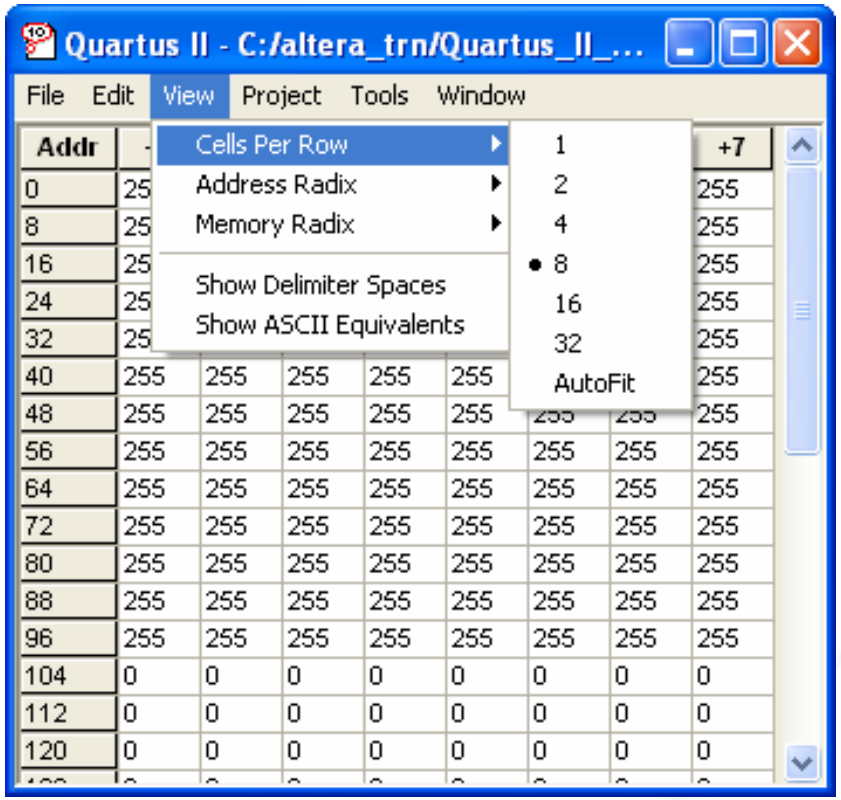
3) Memory space editor opens



Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255
16	255	255	255	255	255	255	255	255
24	255	255	255	255	255	255	255	255
32	255	255	255	255	255	255	255	255
40	255	255	255	255	255	255	255	255
48	255	255	255	255	255	255	255	255
56	255	255	255	255	255	255	255	255
64	255	255	255	255	255	255	255	255
72	255	255	255	255	255	255	255	255
80	255	255	255	255	255	255	255	255
88	255	255	255	255	255	255	255	255
96	255	255	255	255	255	255	255	255
104	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0
128	0	0	0	0	0	0	0	0
136	0	0	0	0	0	0	0	0
144	0	0	0	0	0	0	0	0
152	0	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0
168	0	0	0	0	0	0	0	0

Change Options

- View options of memory editor
  - View ⇒ select from available options



Edit Contents

- Edit contents of memory file
- Save memory file as .HEX or .MIF file

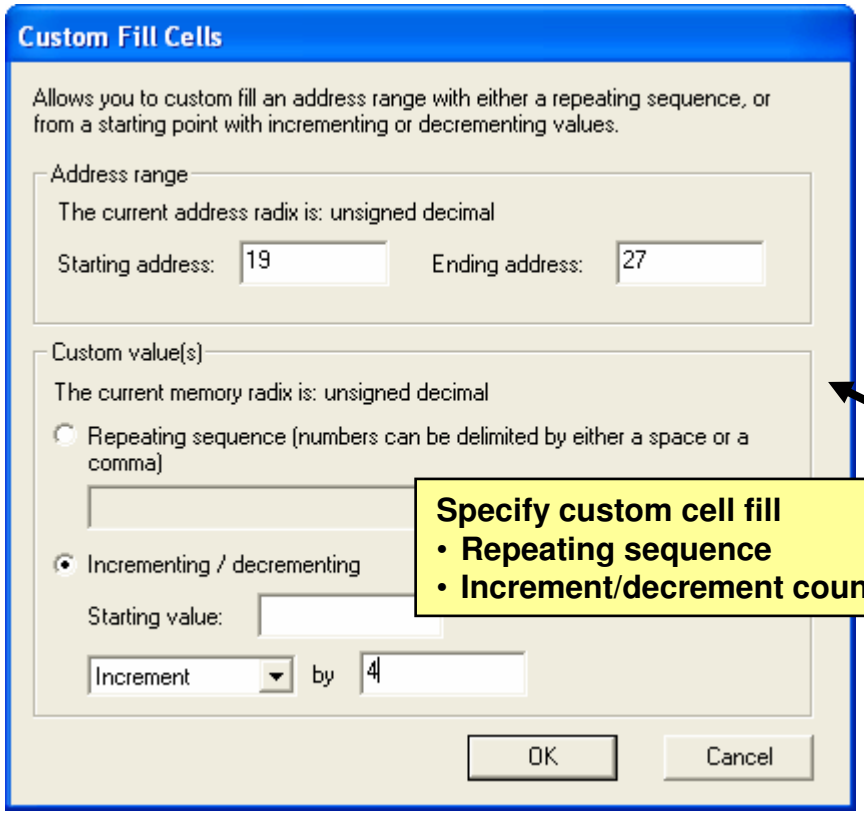
Select address location & type in a value

OR

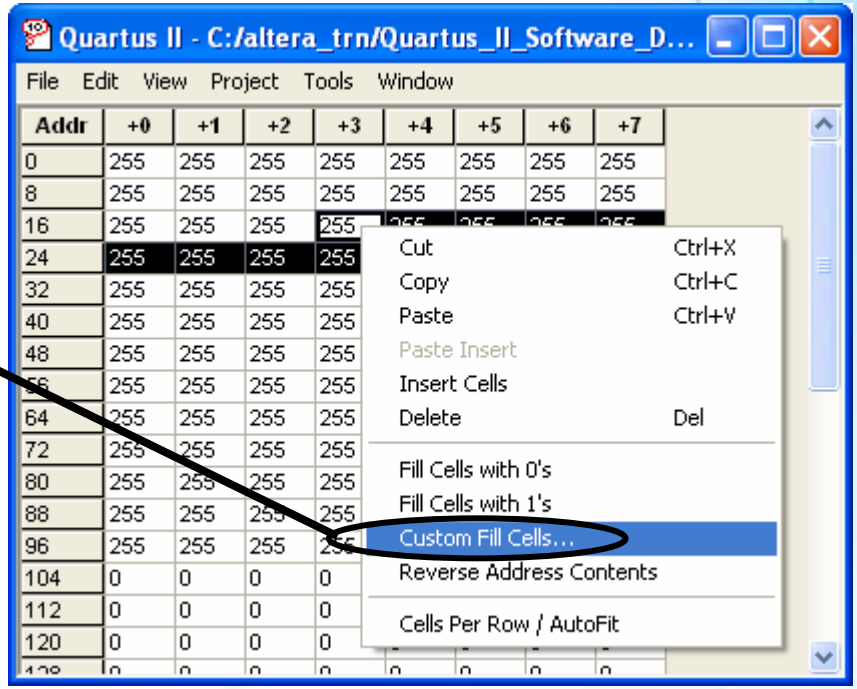
Select the address & right-click to select fill option from menu

OR

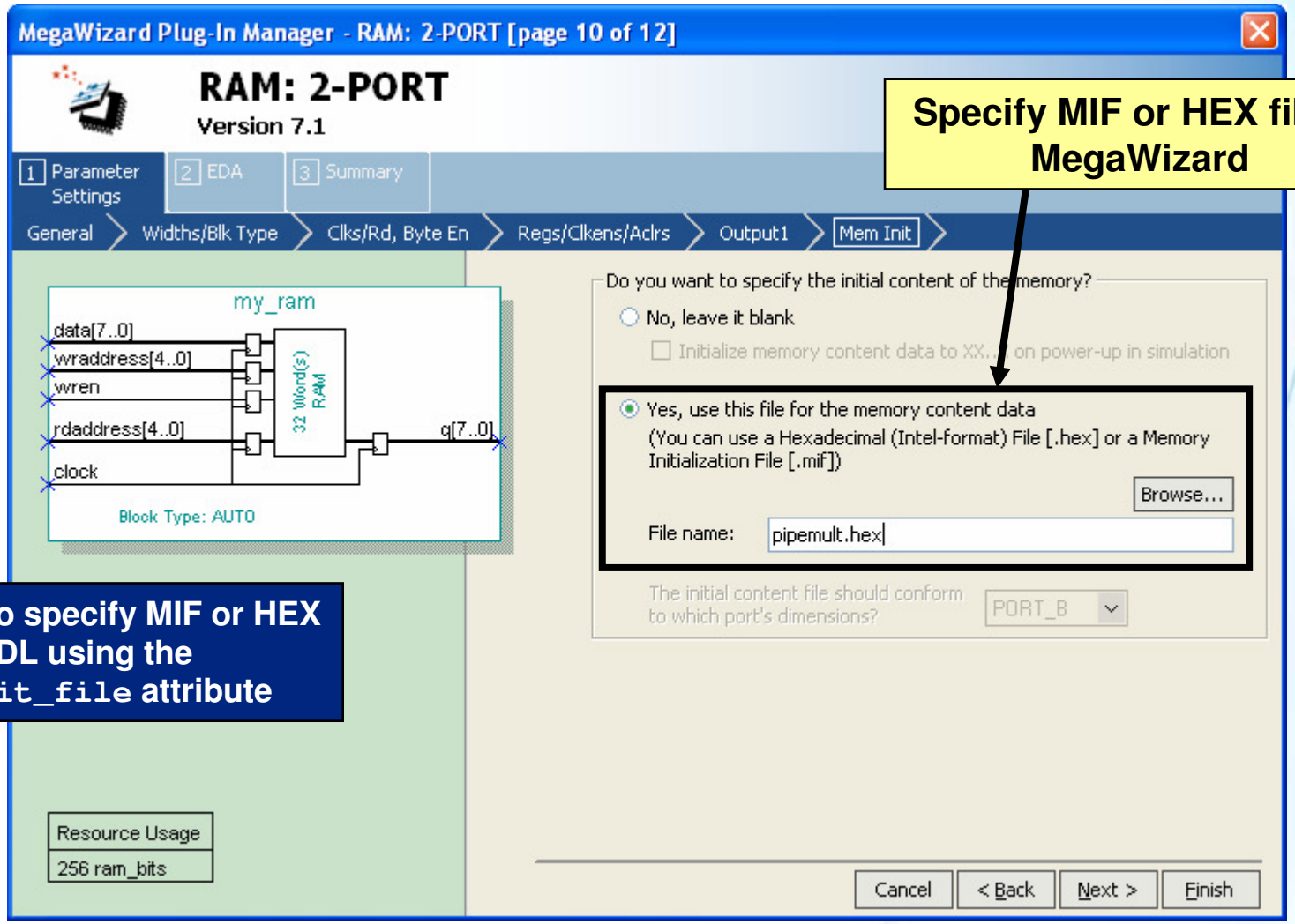
Copy & paste from spreadsheet



Specify custom cell fill
- **Repeating sequence**
- **Increment/decrement count**



Using Memory File In Design

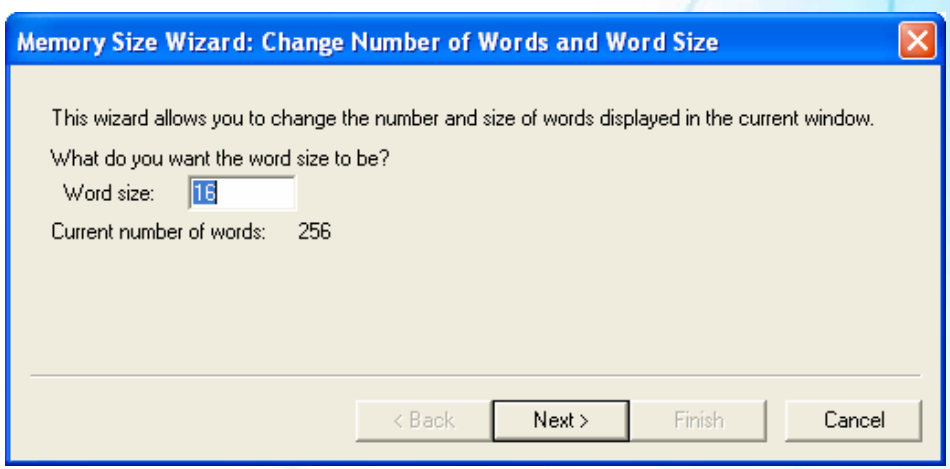


May also specify MIF or HEX file in HDL using the `ram_init_file` attribute

Memory Size Wizard

Need to edit size of memory file?

- Use the Memory Size Wizard (Edit menu)
  - Edit word size
  - Edit number of words
  - Specify how to handle word size change
    - Increasing word size
      - Pad words
      - Combine words
    - Decreasing word size
      - Truncate words from left
      - Truncate words from right



EDA Interfaces Introduction

- Interface with industry-standard EDA tools that generate a netlist file
  - EDIF 2 0 0 (.EDF)
  - Verilog Quartus Mapping (.VQM)
- To import netlist files
  - Specify EDA tool in the Quartus II software settings
  - Instantiate block(s) in design
  - Add .EDF/.VQM file(s) to Quartus II project

3rd-Party Design Entry Tool Support

- Mentor Graphics®
  - LeonardoSpectrum™
  - Precision RTL Synthesis™
- Synopsys
  - Design Compiler FPGA
  - FPGA Compiler II
- Synplicity
  - Synplify
  - Synplify Pro

Exercise 2 Demonstration

Design Entry Summary

- Multiple design entry methods supported
  - Text (Verilog, VHDL, AHDL)
  - 3rd-party netlist (VQM, EDIF)
  - Schematic
- MegaWizard Plug-In Manager configures megafunctions & IP
- Memory Editor allows generation of memory initialization files
- 3rd-party EDA tools supported for design entry & synthesis

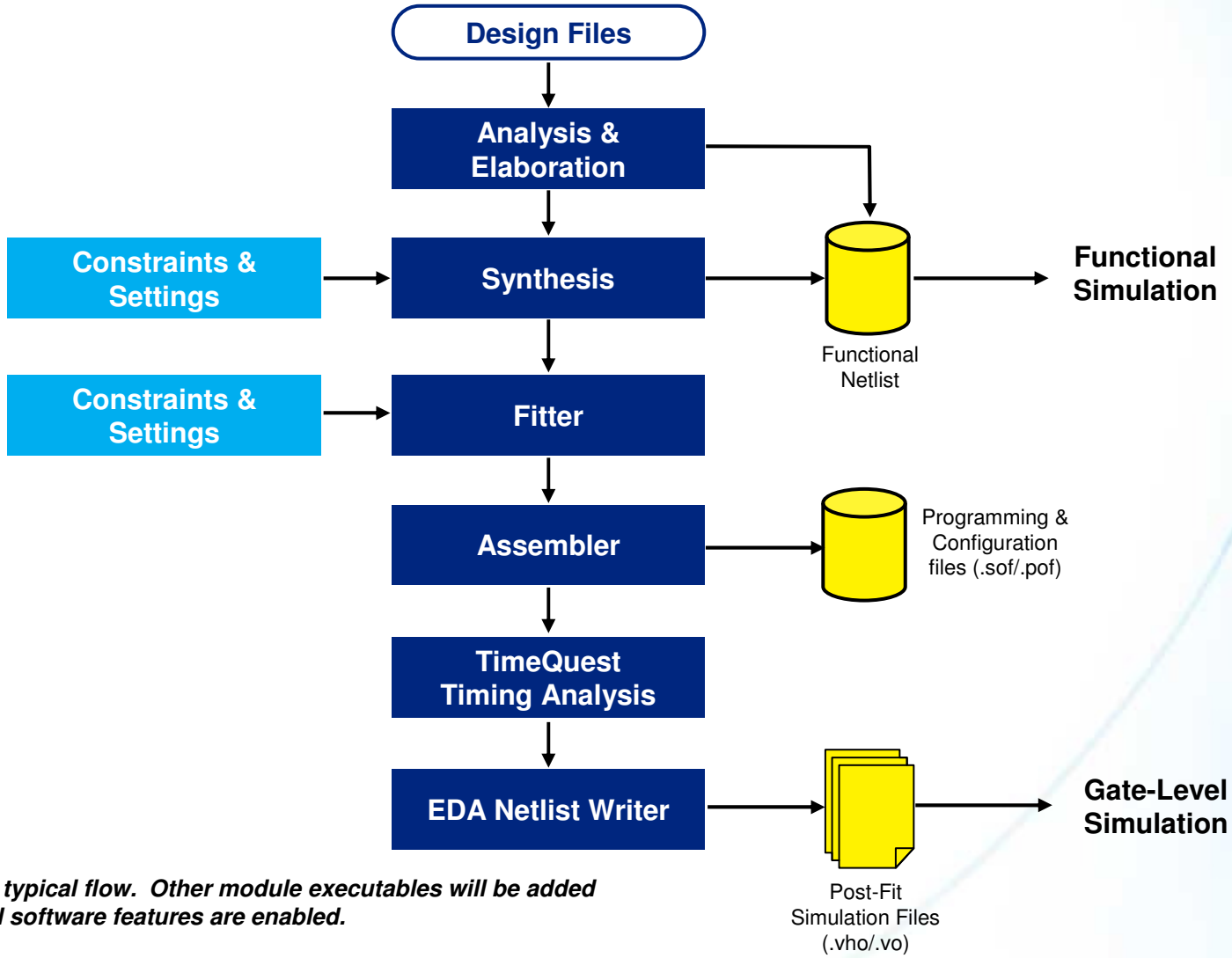
Design Entry Support Resources

- Quartus II Handbook chapters
  - "Design Recommendations for Altera Devices" (Volume 1)
  - "Recommended HDL Coding Styles" (Volume 1)
  - 3rd-Party EDA tool chapters (Volume 1, Section 3)
- Training courses & demonstrations
  - VHDL & Verilog Basics (online courses)
  - Introduction & Advanced HDL courses

Quartus II Software Design Series: Foundation

Quartus II Compilation

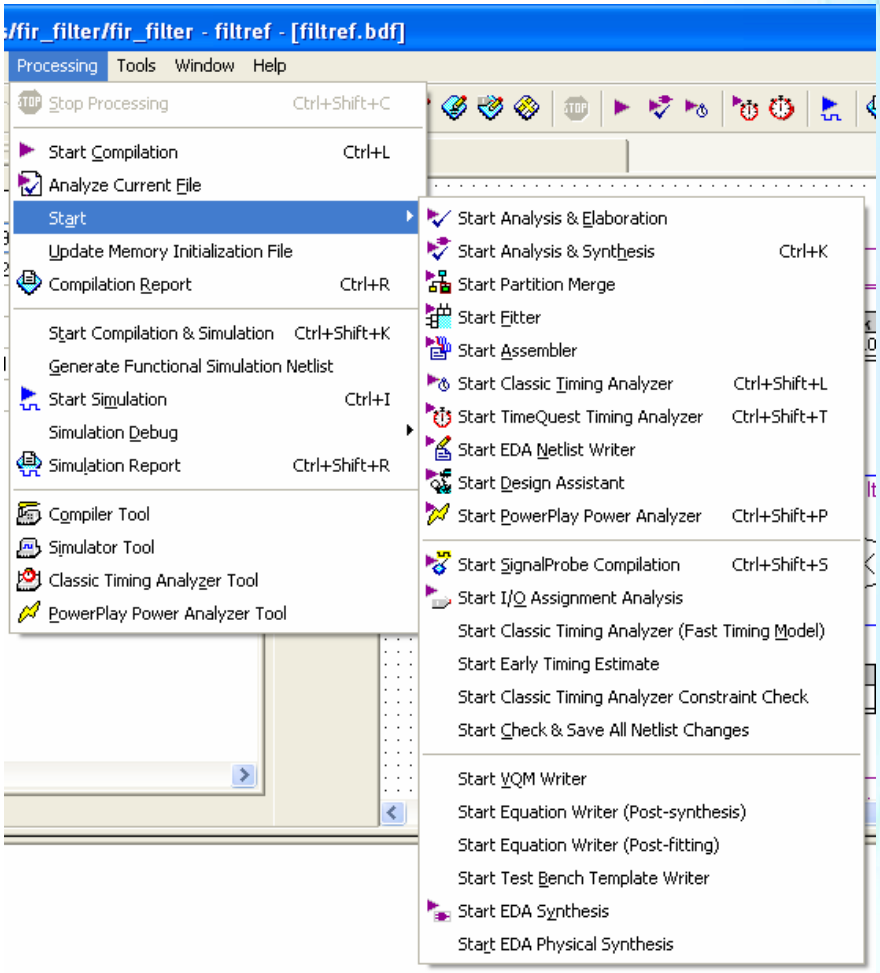
Quartus II Full Compilation Flow*



This is the typical flow. Other module executables will be added if additional software features are enabled.

Processing Options

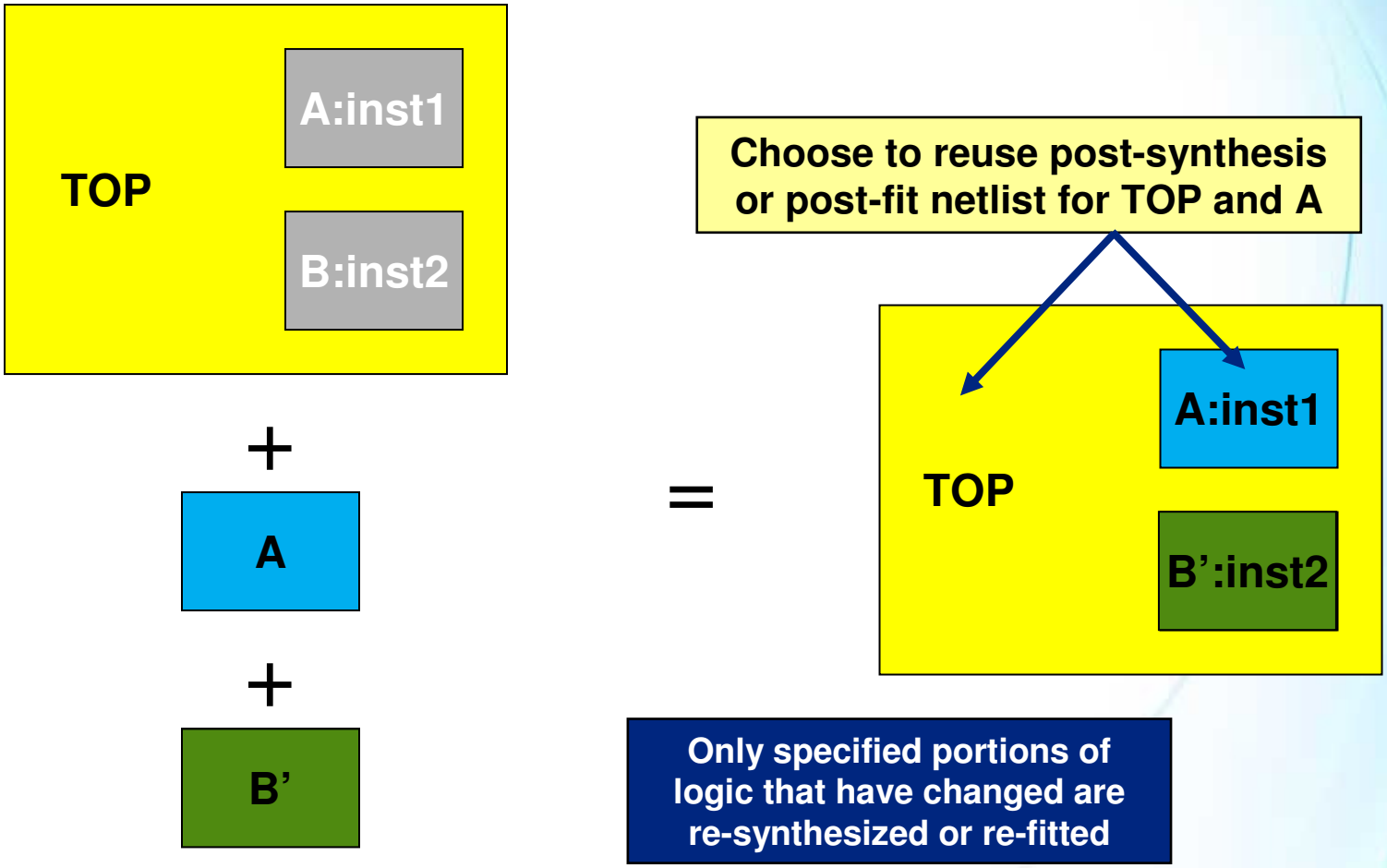
- Start Compilation
  - Performs full compilation
- Start Analysis & Elaboration
  - Checks syntax & builds database only
  - Performs initial synthesis
- Start Analysis & Synthesis
  - Synthesizes & optimizes code
- Start Fitter
  - Places & routes design
  - Generates output netlists
- Start Assembler
  - Generate programming files
- Start TimeQuest Timing Analyzer
- Start I/O Assignment Analysis
- Start Design Assistant



Compilation Design Flows

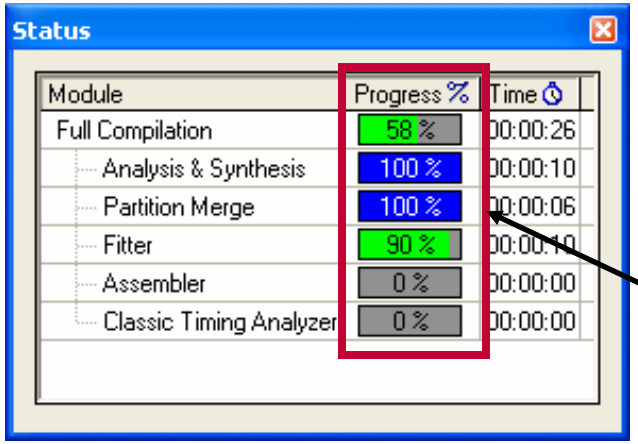
- Default “flat” compilation flow
  - Design compiled as a whole
  - Global optimizations performed
- Incremental flow (on by default for new projects)
  - User assigns design partitions
  - Each partition processed separately & results merged to form complete design
  - Netlists for partitions reused from prior successful compilation or imported from another project
    - Top-down or bottom-up flow
  - Benefits
    - Decrease compilation time
    - Preserve compilation results and timing performance
    - Enable faster timing closure

Incremental Compilation Concept



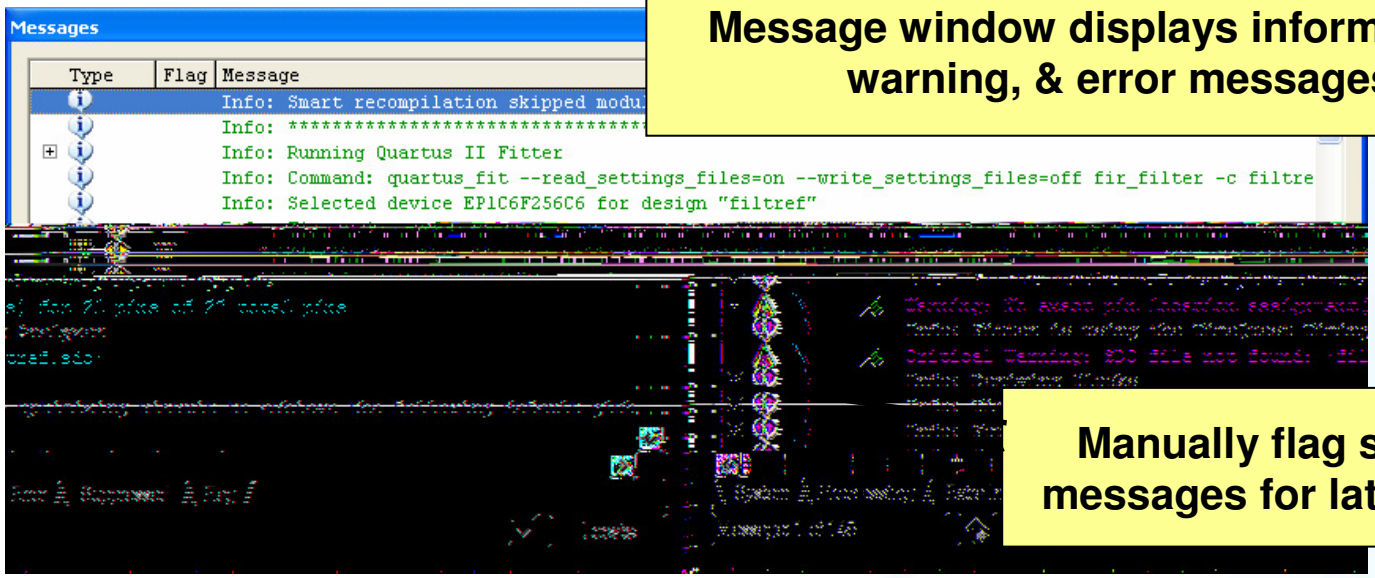
Note: For more details on using incremental compilation, please attend the course "Quartus II Software Design Series: Optimization" or watch the web-recording "Using Quartus II: Incremental Compilation"

Status & Message Windows



Menu Bar: View ⇒ Utility Windows

Status bars scroll to indicate progress in each stage of compilation

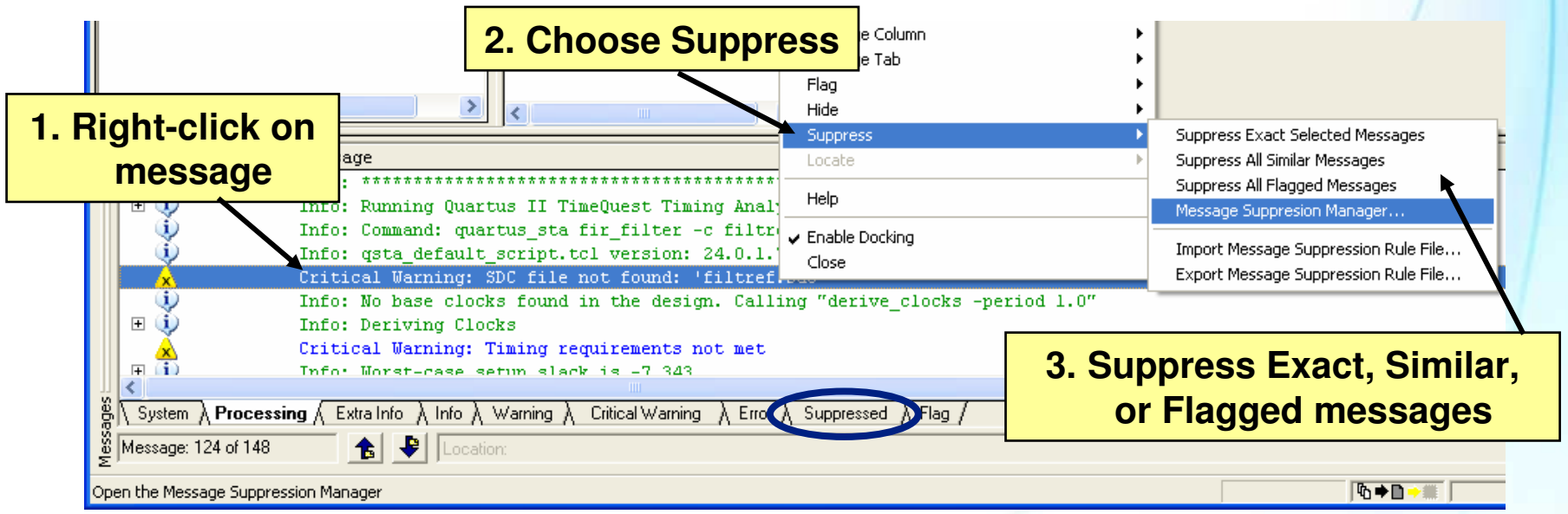


Message window displays informational, warning, & error messages

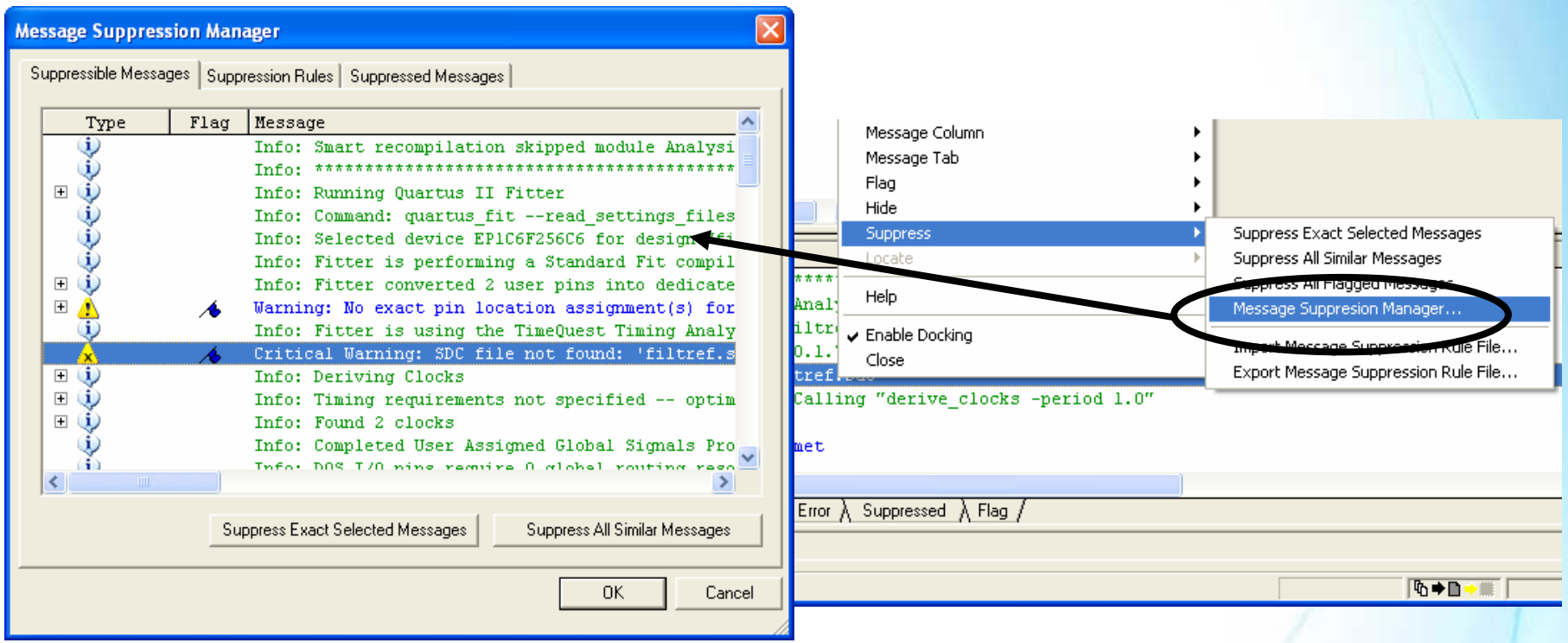
Manually flag selected messages for later review

Message Suppression

- Hides messages from current & future compiles
  - Ex. Known synthesis warning message already investigated
- Displays suppressed messages on different tab in message window
- Stores suppression rules in <revision_name>.SRF file



Message Suppression Manager Tool



- Use to
  - View all suppressible messages
  - View/add/remove suppression rules
  - View messages suppressed for current & future compiles

Viewing Compilation Results

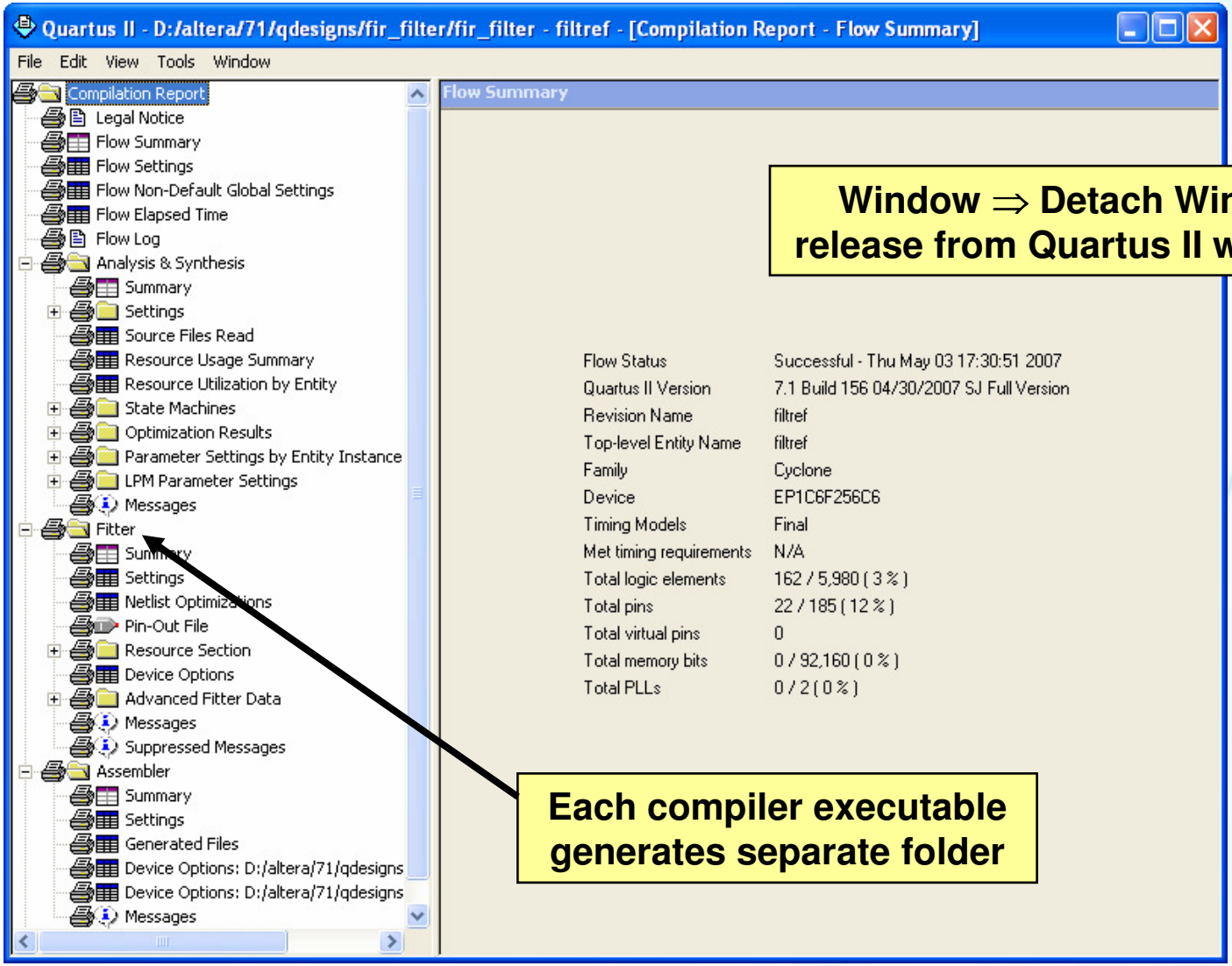
- Quartus II graphical tools available for
  - Understanding design processing
  - Verifying correct design results
  - Debugging incorrect results
- Compilation Report
- Viewers
  - RTL & Technology Map
  - State Machine
- Chip Planner
- Resource Property Editor

Compilation Report

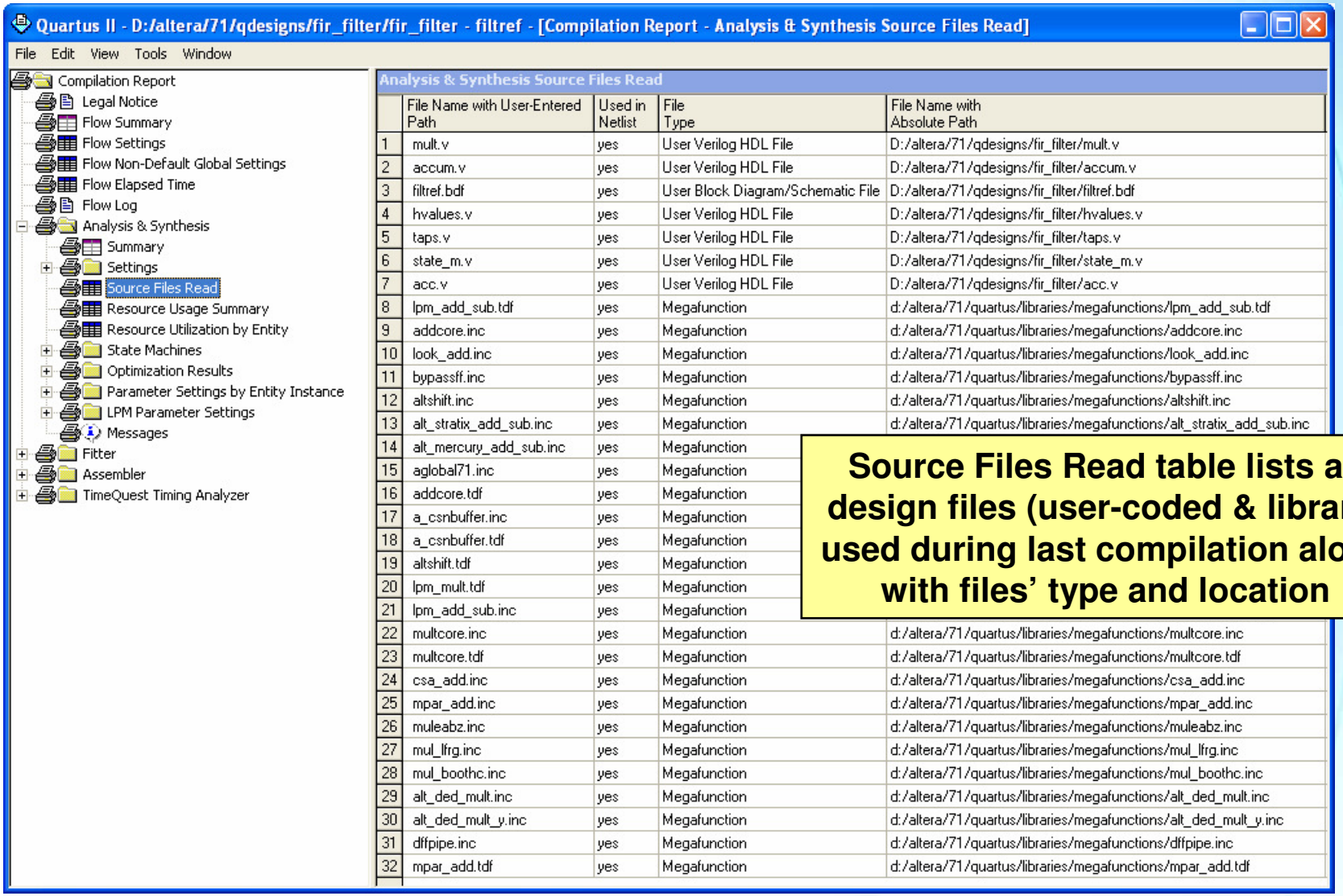
- Graphical window containing all compilation processing information
  - Resource Usage
  - Device pin-out
  - Settings and constraints applied
  - Messages
- Opens automatically when processing begins
- *Recommendation:* Go through report for a design to get sense of information being provided

- Information also available as text files in project directory
  - Ex. *<project_name>.fit.rpt* & *<project_name>.map.rpt*

Compilation Report



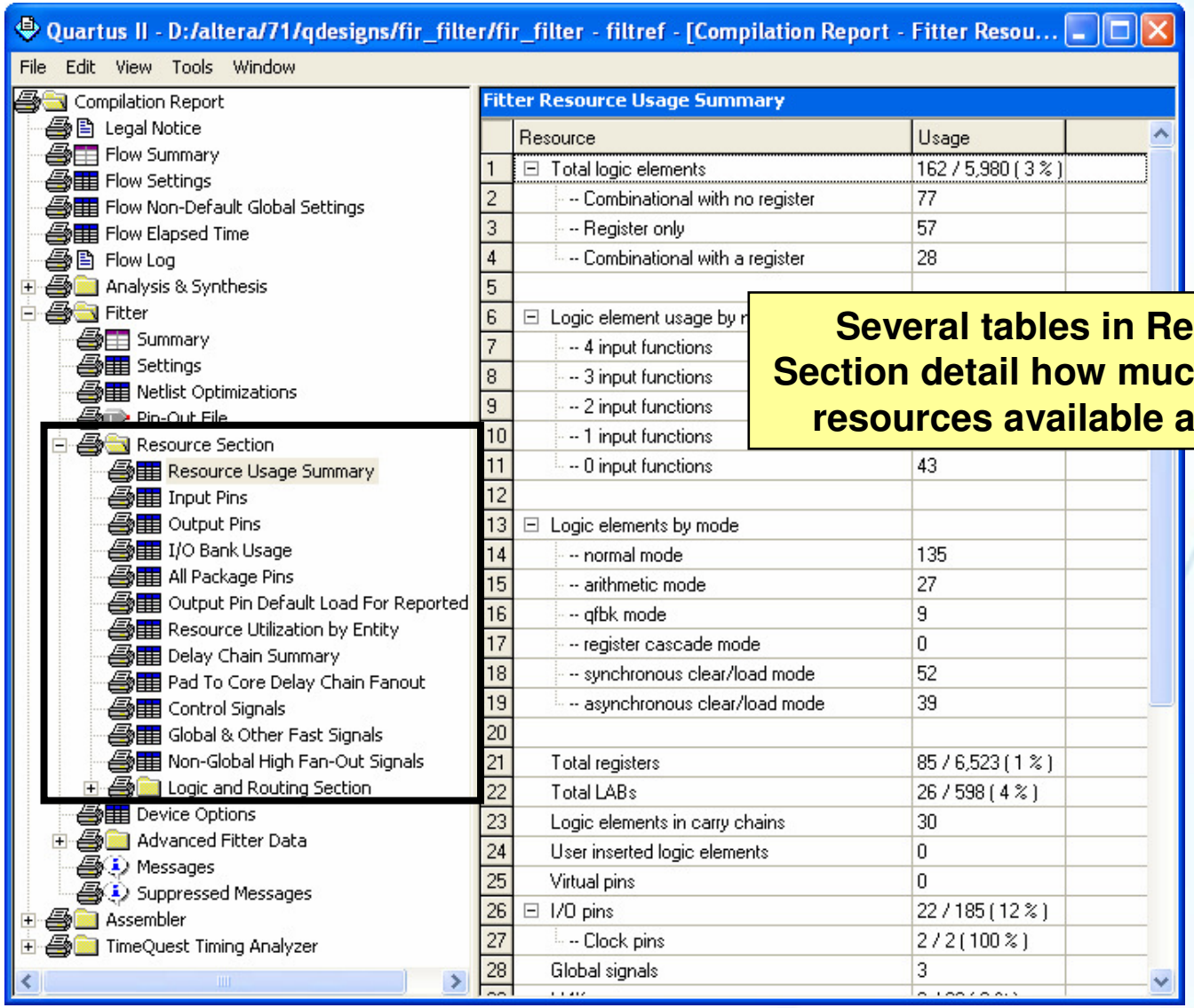
Example: Source Files Read



	File Name with User-Entered Path	Used in Netlist	File Type	File Name with Absolute Path
1	mult.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/mult.v
2	accum.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/accum.v
3	filtref.bdf	yes	User Block Diagram/Schematic File	D:/altera/71/qdesigns/fir_filter/filtref.bdf
4	hvalues.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/hvalues.v
5	taps.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/taps.v
6	state_m.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/state_m.v
7	acc.v	yes	User Verilog HDL File	D:/altera/71/qdesigns/fir_filter/acc.v
8	lpm_add_sub.tdf	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/lpm_add_sub.tdf
9	addcore.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/addcore.inc
10	look_add.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/look_add.inc
11	bypassff.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/bypassff.inc
12	altshift.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/altshift.inc
13	alt_stratix_add_sub.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/alt_stratix_add_sub.inc
14	alt_mercury_add_sub.inc	yes	Megafunction	
15	aglobal71.inc	yes	Megafunction	
16	addcore.tdf	yes	Megafunction	
17	a_csnbuffer.inc	yes	Megafunction	
18	a_csnbuffer.tdf	yes	Megafunction	
19	altshift.tdf	yes	Megafunction	
20	lpm_mult.tdf	yes	Megafunction	
21	lpm_add_sub.inc	yes	Megafunction	
22	multcore.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/multcore.inc
23	multcore.tdf	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/multcore.tdf
24	csa_add.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/csa_add.inc
25	mpar_add.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/mpar_add.inc
26	muleabz.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/muleabz.inc
27	mul_lfrg.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/mul_lfrg.inc
28	mul_boothc.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/mul_boothc.inc
29	alt_ded_mult.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/alt_ded_mult.inc
30	alt_ded_mult_y.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/alt_ded_mult_y.inc
31	dffpipe.inc	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/dffpipe.inc
32	mpar_add.tdf	yes	Megafunction	d:/altera/71/quartus/libraries/megafunctions/mpar_add.tdf

Source Files Read table lists all design files (user-coded & library) used during last compilation along with files' type and location

Example: Resource Usage

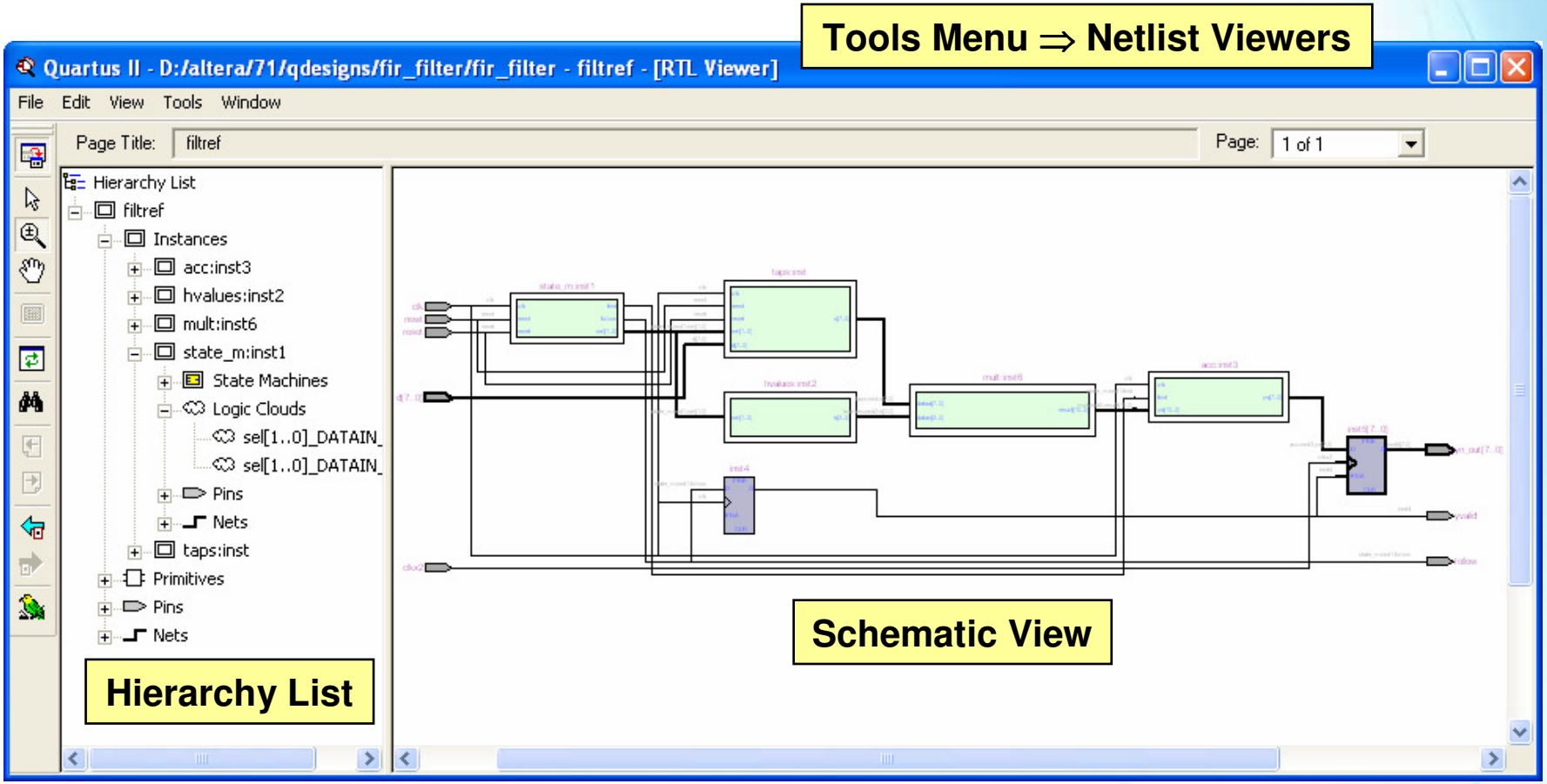


Several tables in Resource Section detail how much of FPGA resources available and used

Netlist Viewers

- RTL Viewer
  - Graphically represents results of synthesis
  - Visually check initial HDL synthesis results
    - Before any Quartus II optimizations
  - Locate synthesized nodes for assigning constraints
  - Debug verification issues
- Technology Map Viewers (Post-Mapping & regular)
  - Graphically represents results of mapping (post-synthesis) & fitting
  - Analyze critical timing paths graphically
    - Delay values displayed if timing
  - Locate nodes & node names after optimizations
    - Assigning constraints
    - Debugging

RTL Viewer

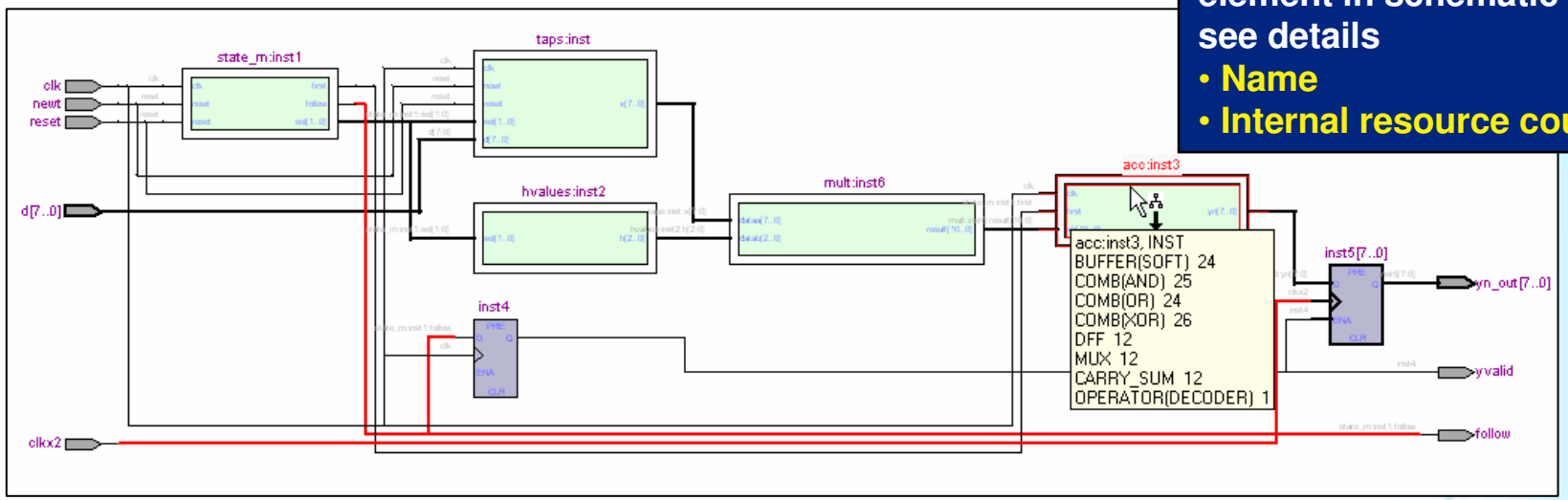


Note:

1) Must Perform Elaboration First (e.g. Analysis & Elaboration OR Analysis & Synthesis)

Schematic View (RTL Viewer)

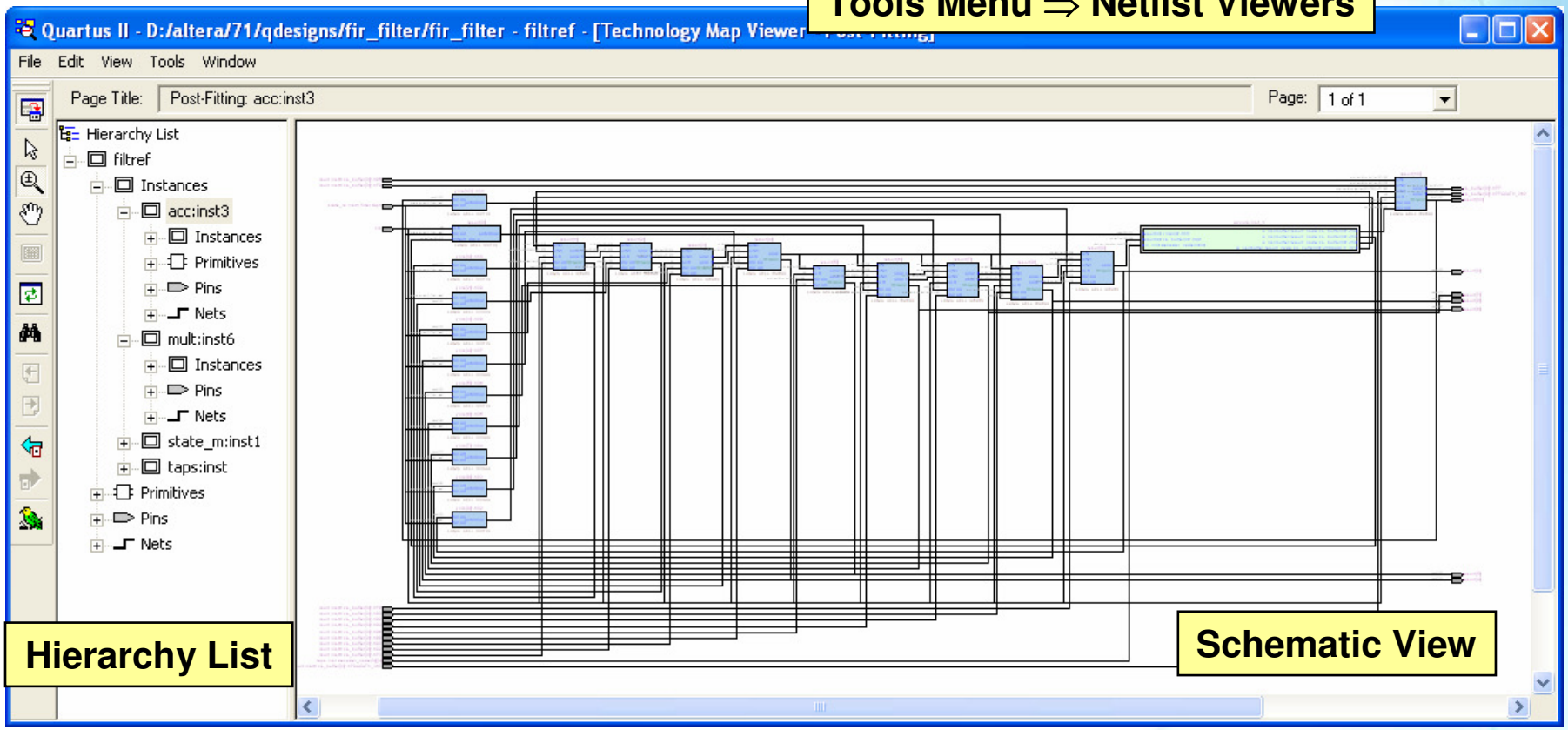
Place pointer over any element in schematic to see details
- **Name**
- **Internal resource count**



- Represents design using logic blocks & nets
  - I/O pins
  - Registers
  - Muxes
  - Gates (e.g. AND, OR, etc.)
  - Operators (e.g. adders, multipliers, etc.)

Technology Map Viewers

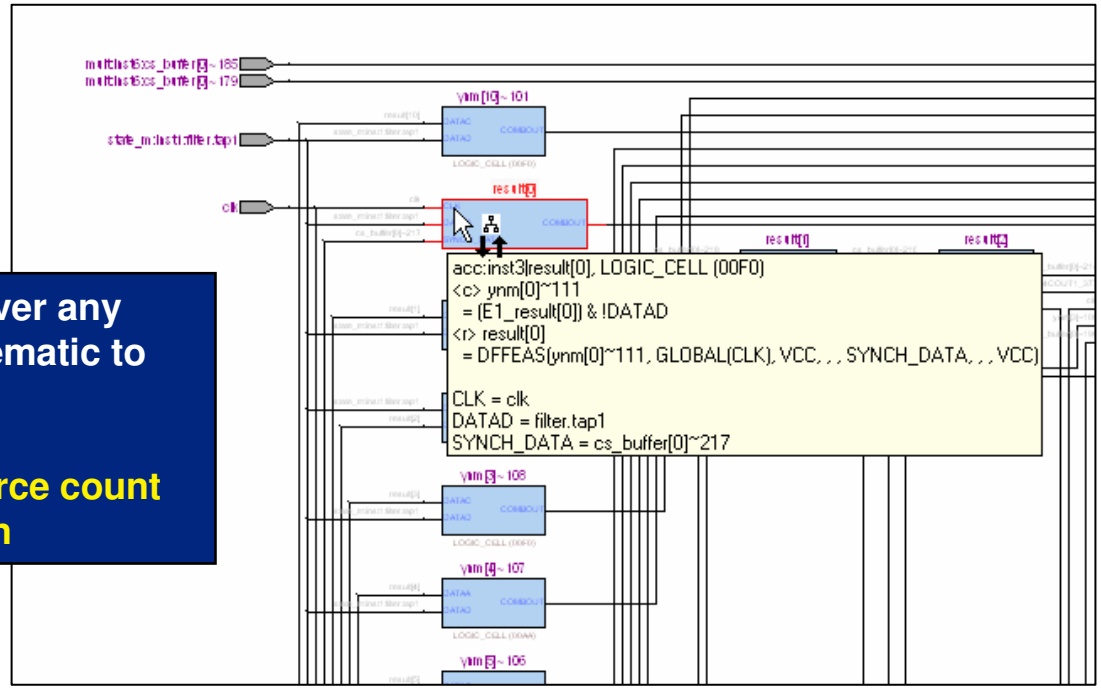
Tools Menu ⇒ Netlist Viewers



Note:

1) Must Run Synthesis and/or Fitting First

Schematic View (Technology Viewer)

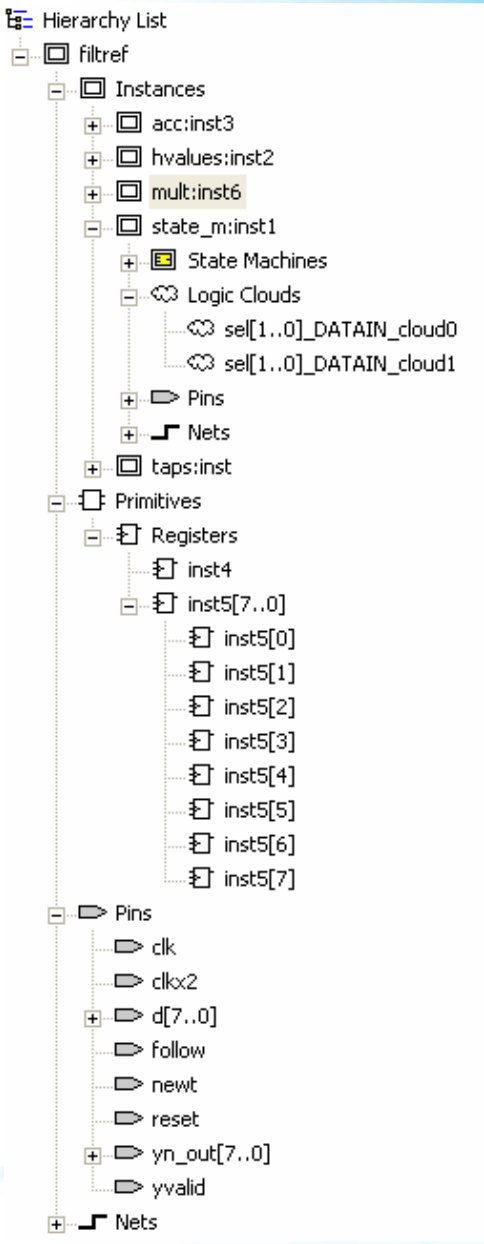


Place pointer over any element in schematic to see details
- **Name**
- **Internal resource count**
- **Logic equation**

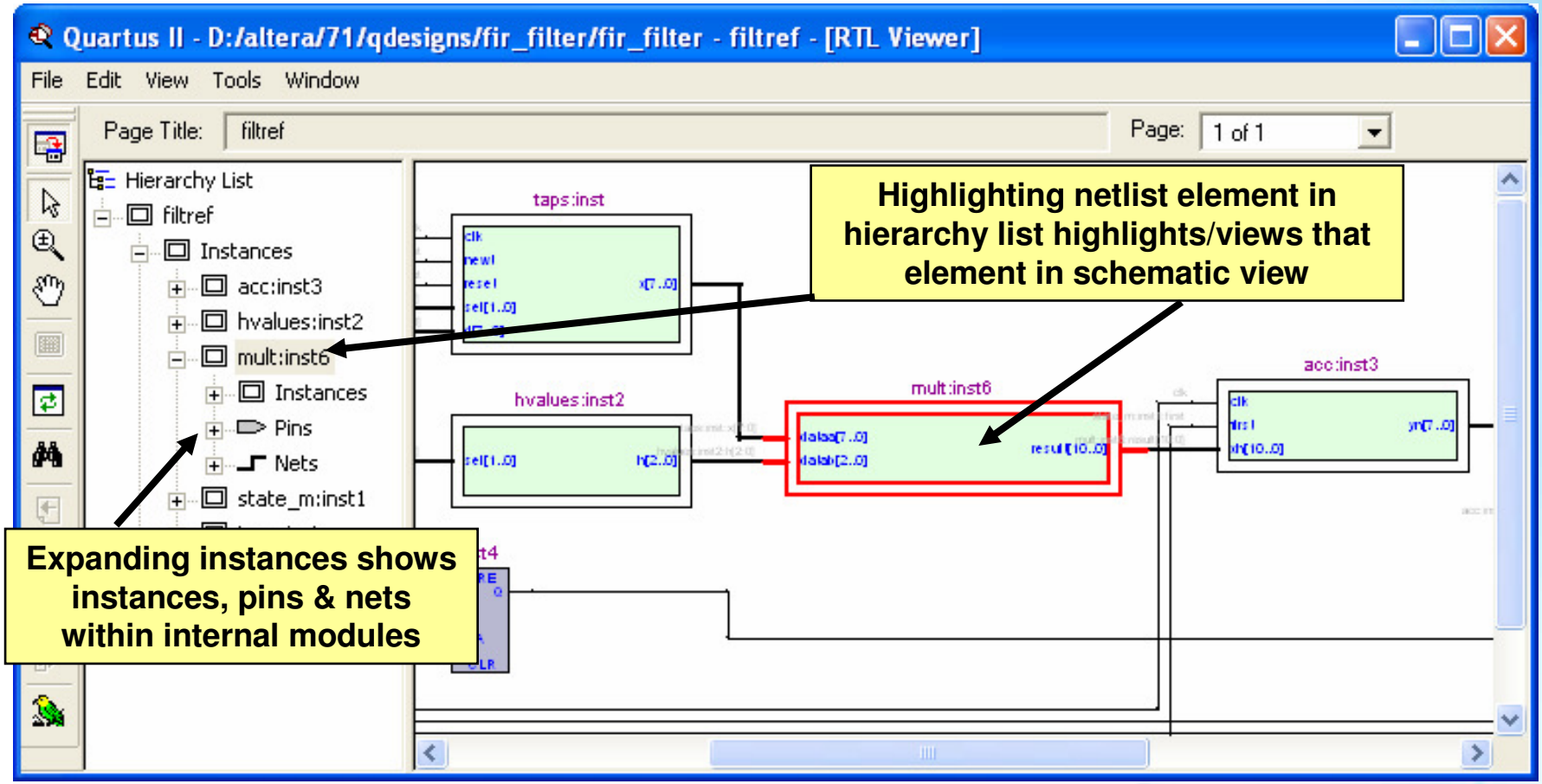
- Represents design using atoms
  - I/O pins & cells
  - Lcells
  - Memory blocks
  - MAC (DSP blocks)

Hierarchy List

- Traverse between levels of design hierarchy
- View logic schematic for each hierarchical level
- Break down each hierarchical level into netlist elements or atoms
  - Instances
  - Primitives
  - Pins
  - Nets
  - State machines
  - Logic clouds (if enabled)

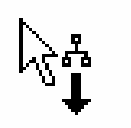


Using Hierarchy List

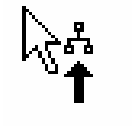


Schematic Hierarchy Navigation

- Mouse pointer indicates action

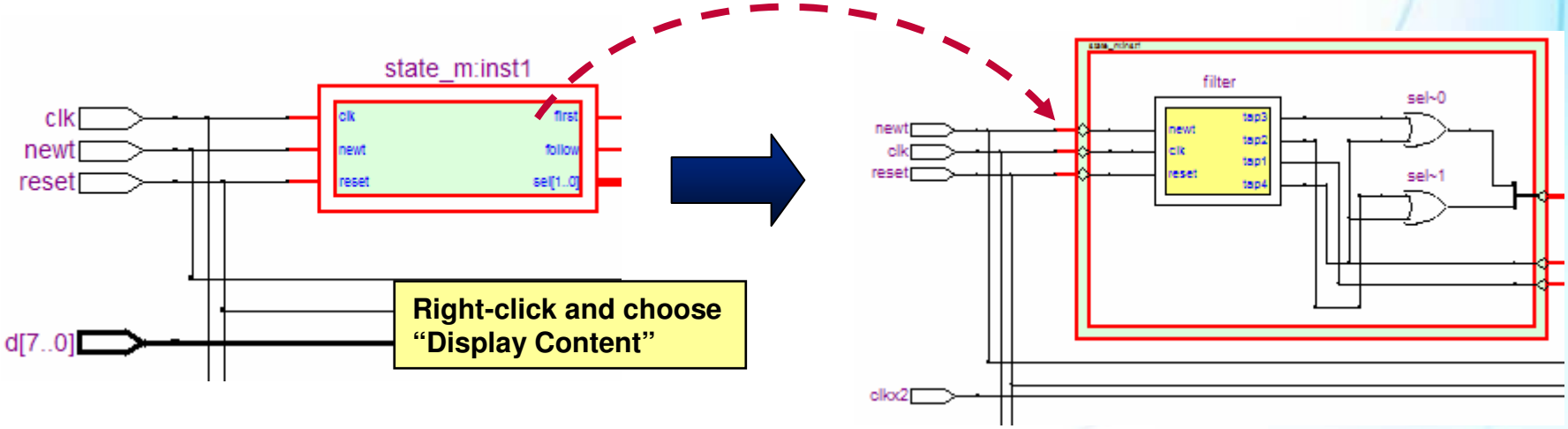


← Descending hierarchy
  - Double-click on instance
  - Right-click & select **hierarchy down**

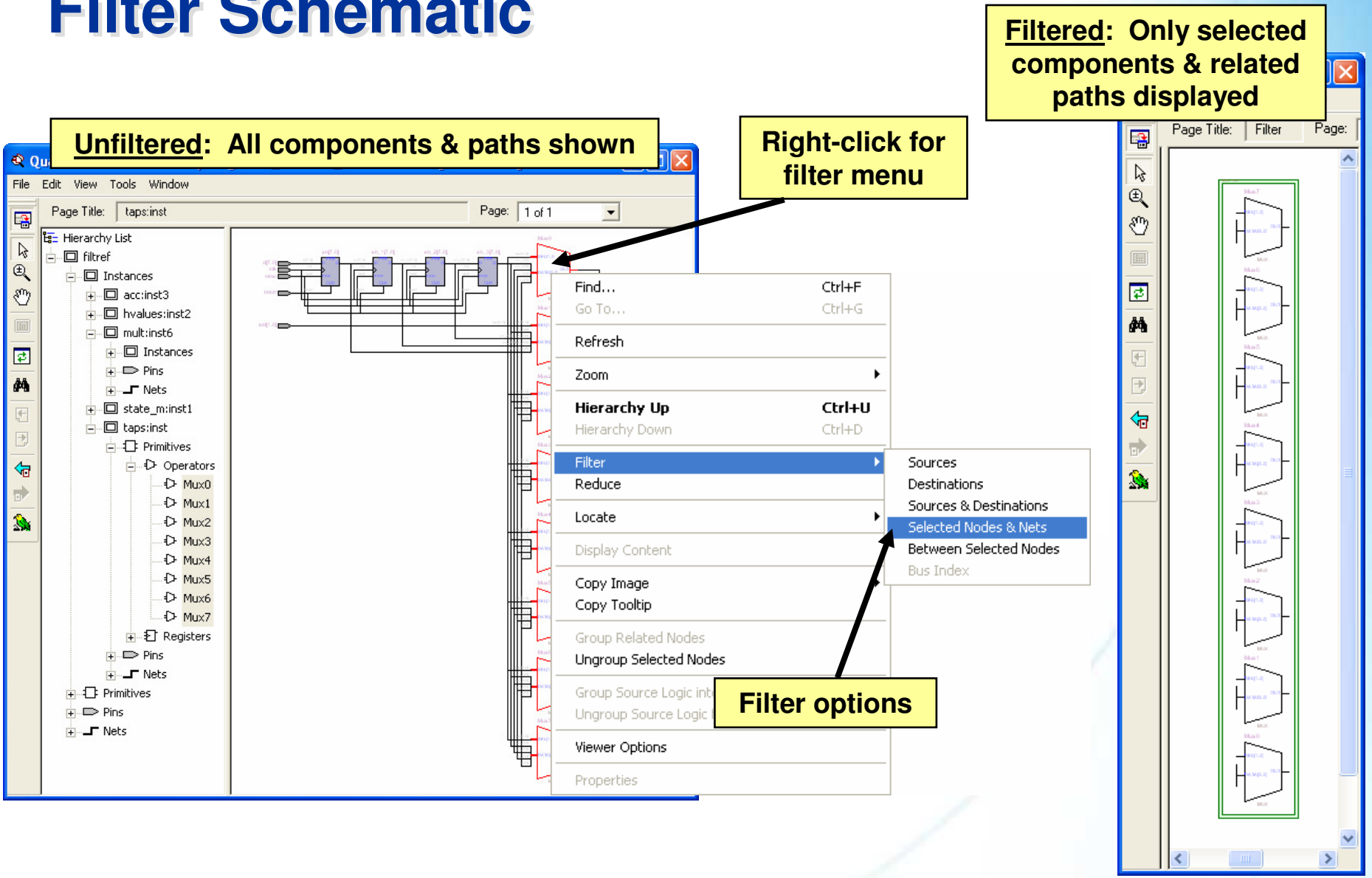


← Ascending hierarchy
  - Double-click in empty space
  - Right-click & select **hierarchy up**

- Expand instances within schematic



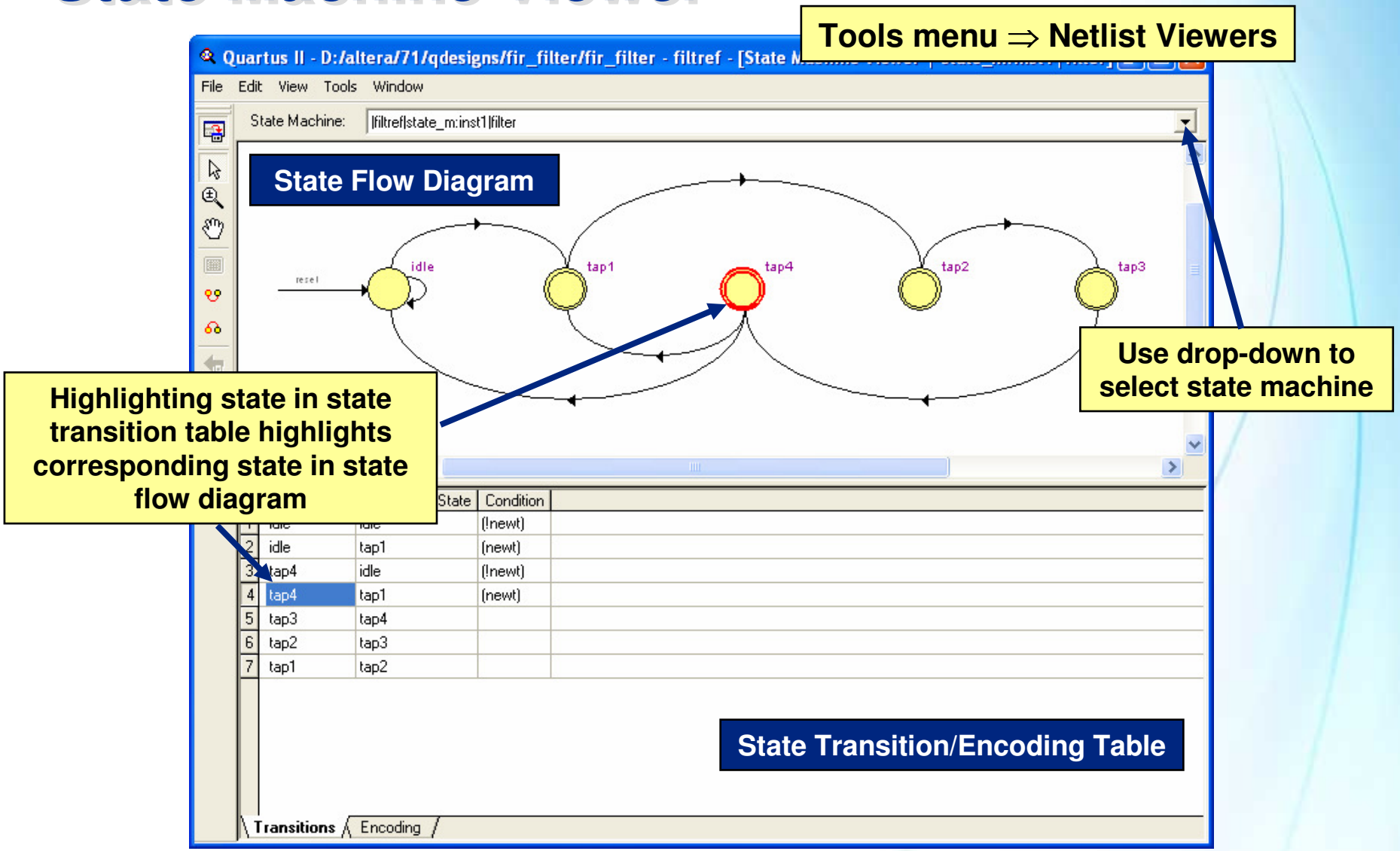
Filter Schematic



Other Features

- Bird's Eye View
  - Displays overall view of design
- Page control
  - Hierarchical levels automatically partitioned
    - Control design size per page (tools ⇒ customize ⇒ options)
  - Navigate nets between page
- Go to net driver
  - Traces net back to source driver
- LUT internal detail
  - View LUT truth table, Karnaugh map, and expanded to gate logic
- Cross-probing : locate nodes from/to
  - Design files
  - Assignment Editor
  - Chip Planner
  - Chip Editor
  - Resource Property Editor
  - RTL/Technology Map Viewers

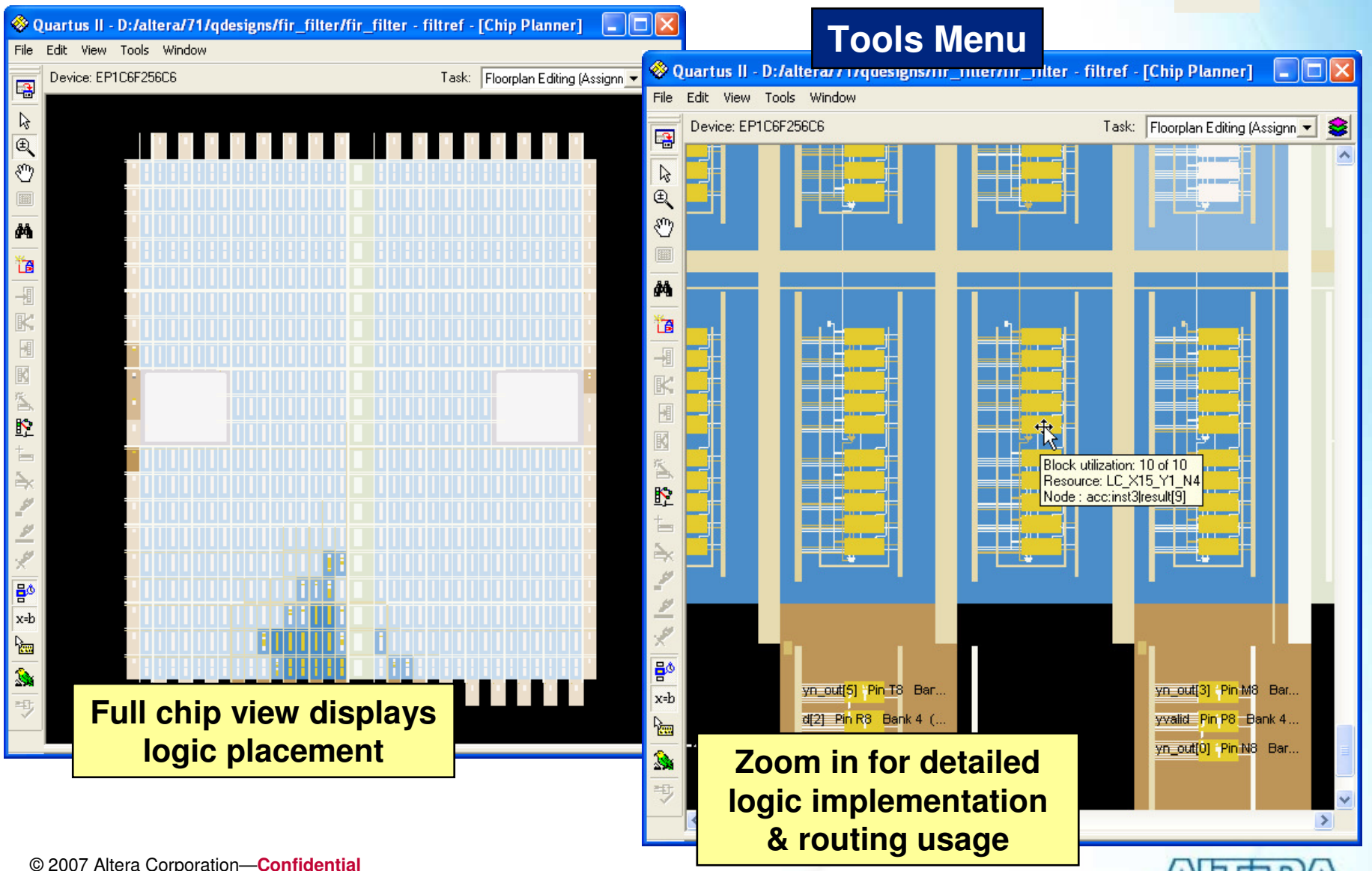
State Machine Viewer



Chip Planner

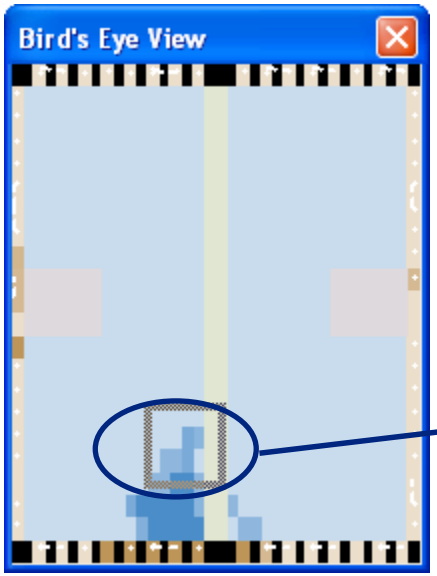
- Editable graphical view of target device
- Displays
  - Graphical layout of device resources
  - Routing channels between device resources
    - Internal routing channels within LABs
- Uses
  - View placement of design logic
  - View connectivity between resources used in design
  - Make placement assignments
  - Debugging placement related issues

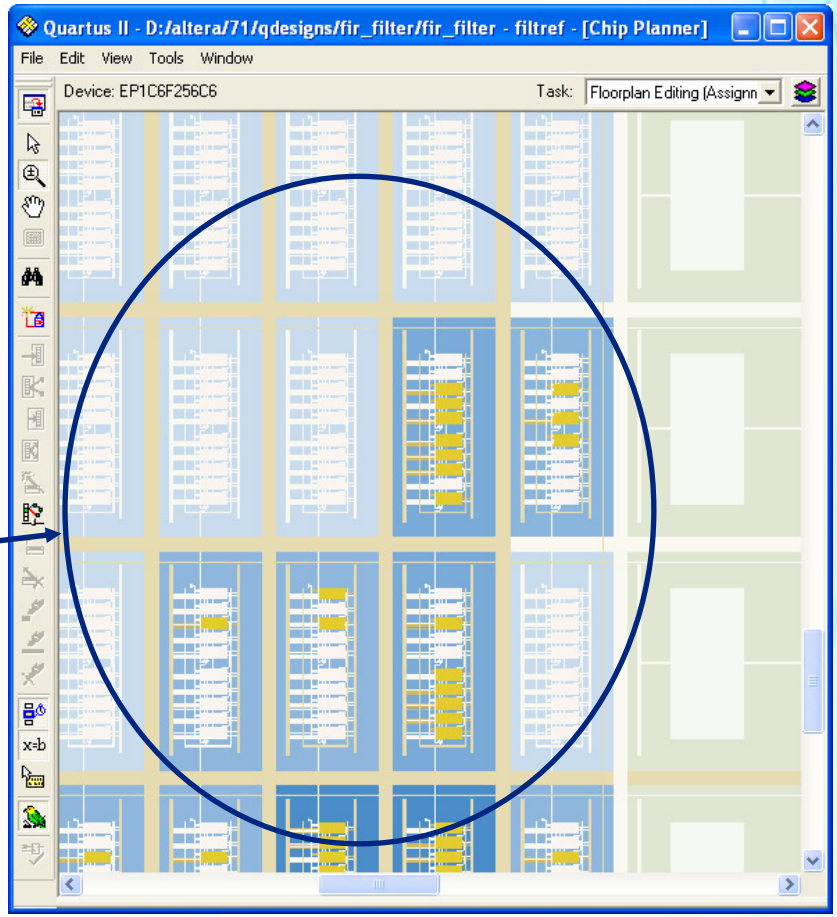
Chip Planner



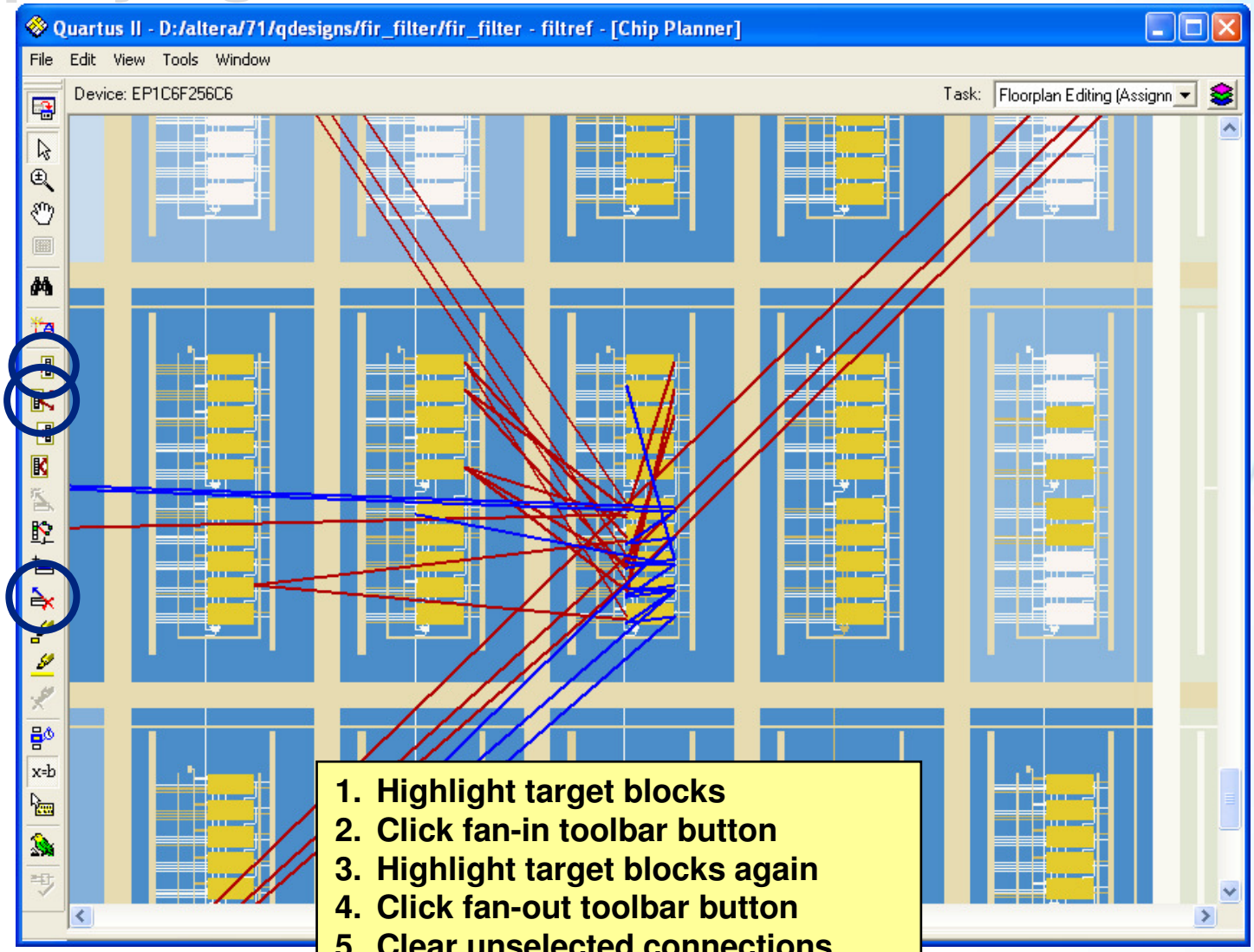
Bird's Eye View

- Provides overall view of the entire device
- Use to navigate through the Chip Planner Floorplan





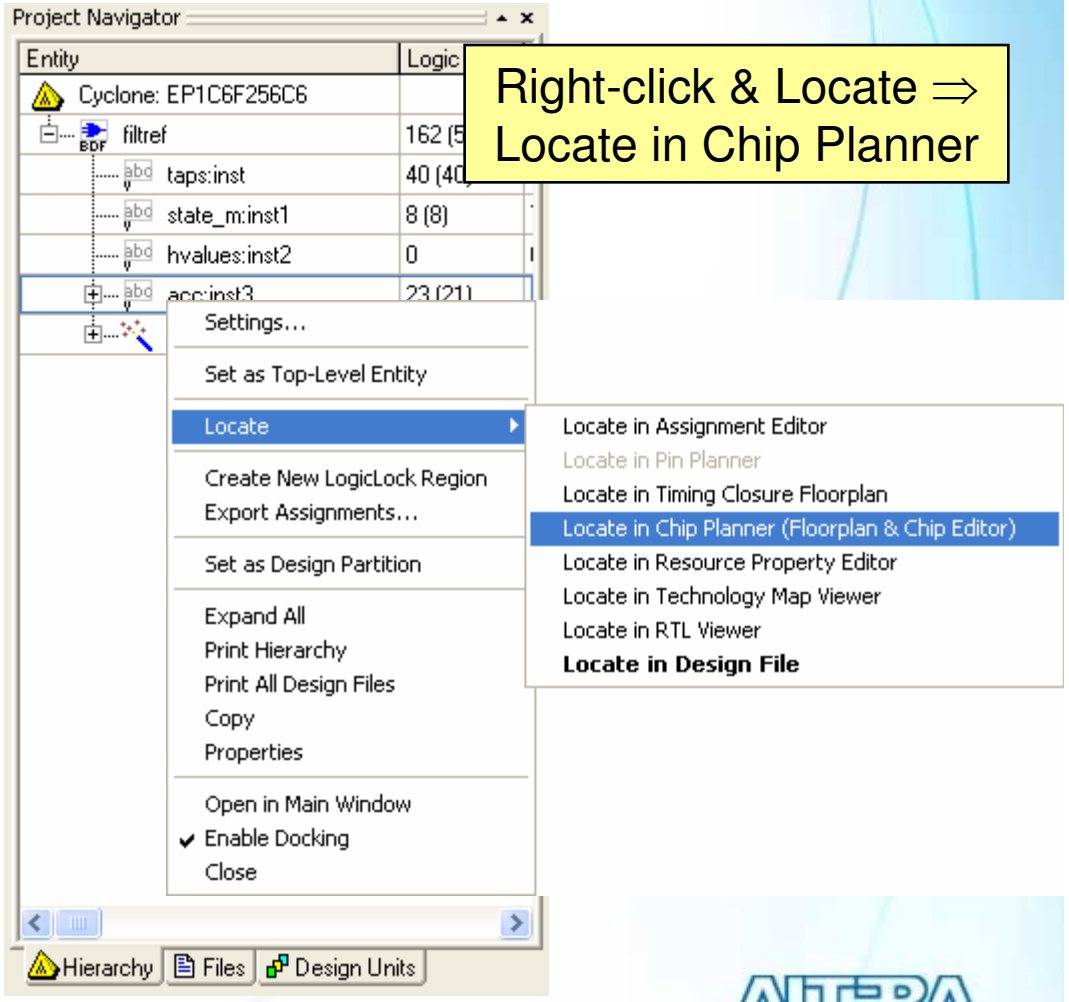
Displaying Fan-In & Fan-Out



Cross-Probing from/to Chip Planner

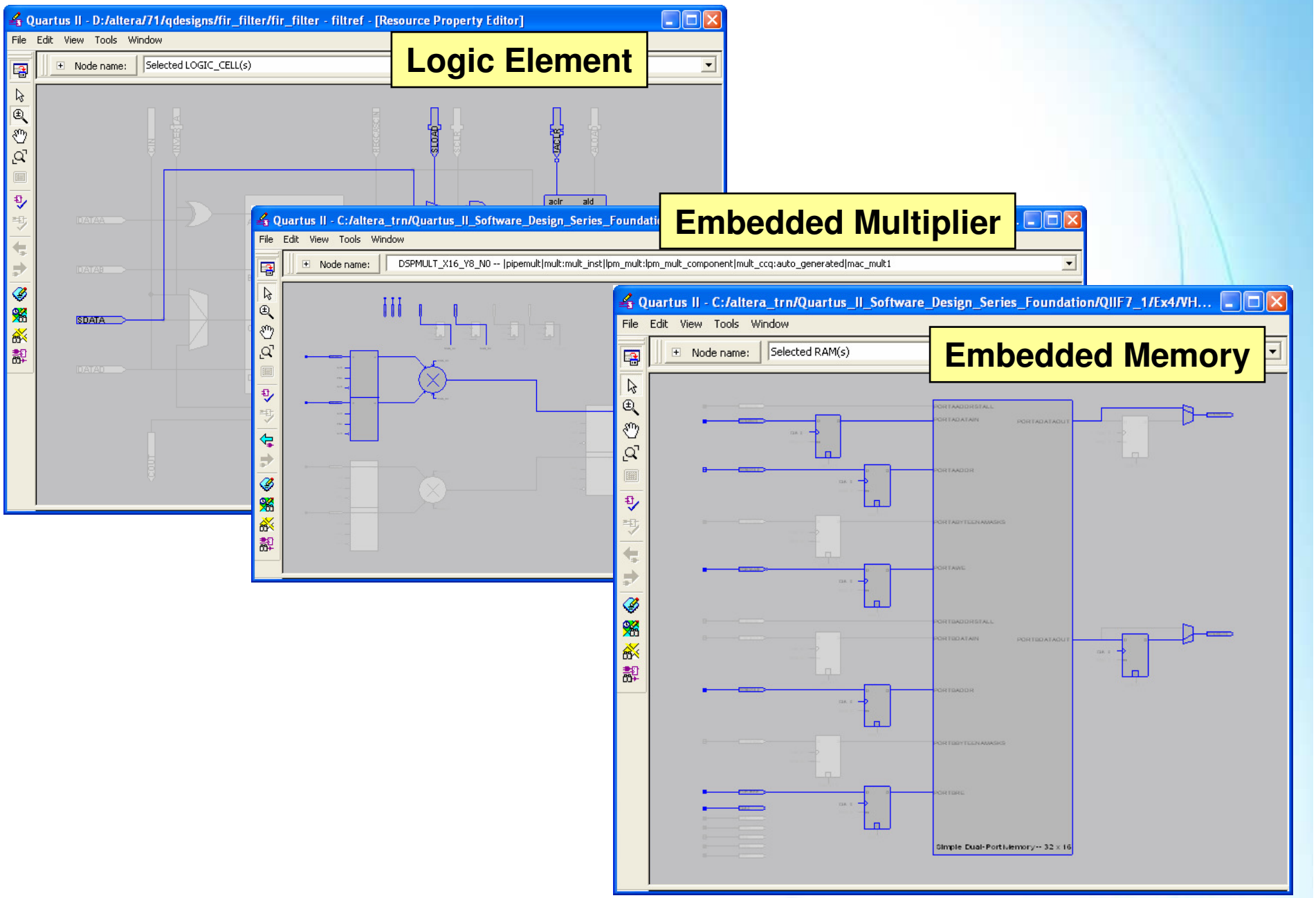
- Locate hierarchy blocks or specific logic from other Quartus II windows

- Project Navigator
- Compilation Report
- Design files
- RTL Viewer
- Technology Viewer
- Message window
- Pin Planner



Resource Property Editor

- Use to view detailed logic implementation & connections
  - Cross-probe from other Quartus II windows
- Views
  - Logic cells (look-up tables & registers)
  - Embedded memory
  - Embedded multipliers
  - I/O cells
  - PLLs



Undesired Compilation Results?

- Incorrect coding
- Non-recommended coding style
- Suboptimal synthesis & fitting settings/constraints

Note: For more details on optimizing designs based on undesired results, please attend the course "Quartus II Software Design Series: Optimization"

Exercise 3 Demonstration

Compilation Summary

- Compilation includes synthesis & fitting
- Compilation Report contains detailed information on compilation results
- Use Quartus II software features to understand how design was processed
  - RTL Viewer
  - Technology Map Viewers
  - State Machine Viewer
  - Chip Planner
  - Resource Property Editors

Compilation Support Resources

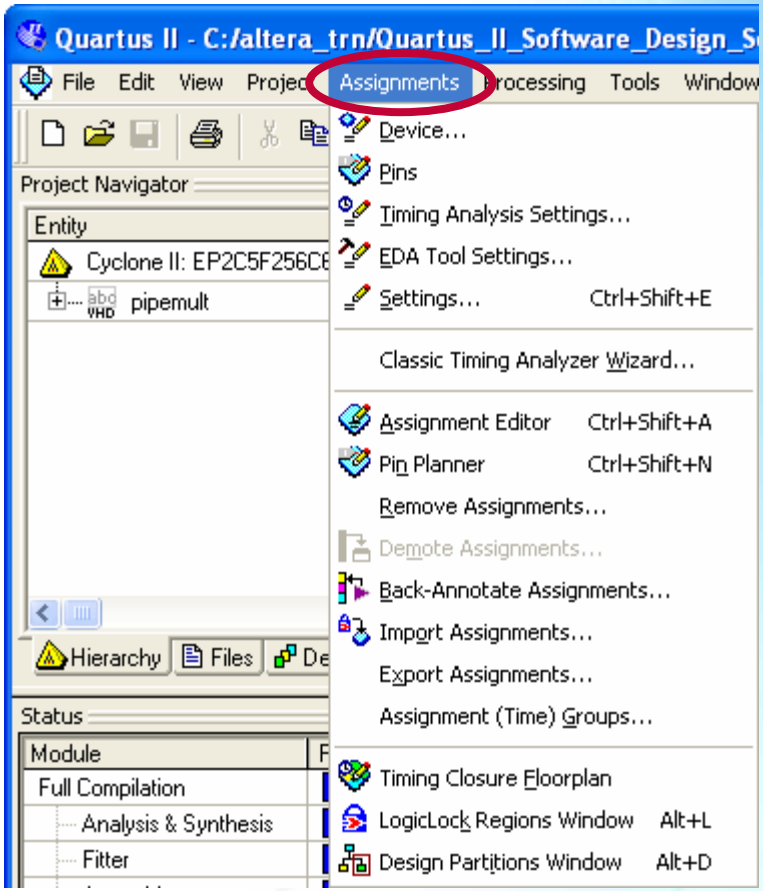
- Quartus II Handbook chapters
  - "Quartus II Incremental Compilation for Hierarchical & Team-Based Design" (Volume 1)
  - "Design Analysis & Engineering Change Management with Chip Planner" (Volume 3)

Quartus II Software Design Series: Foundation

Settings & Assignments

Synthesis & Fitting Control

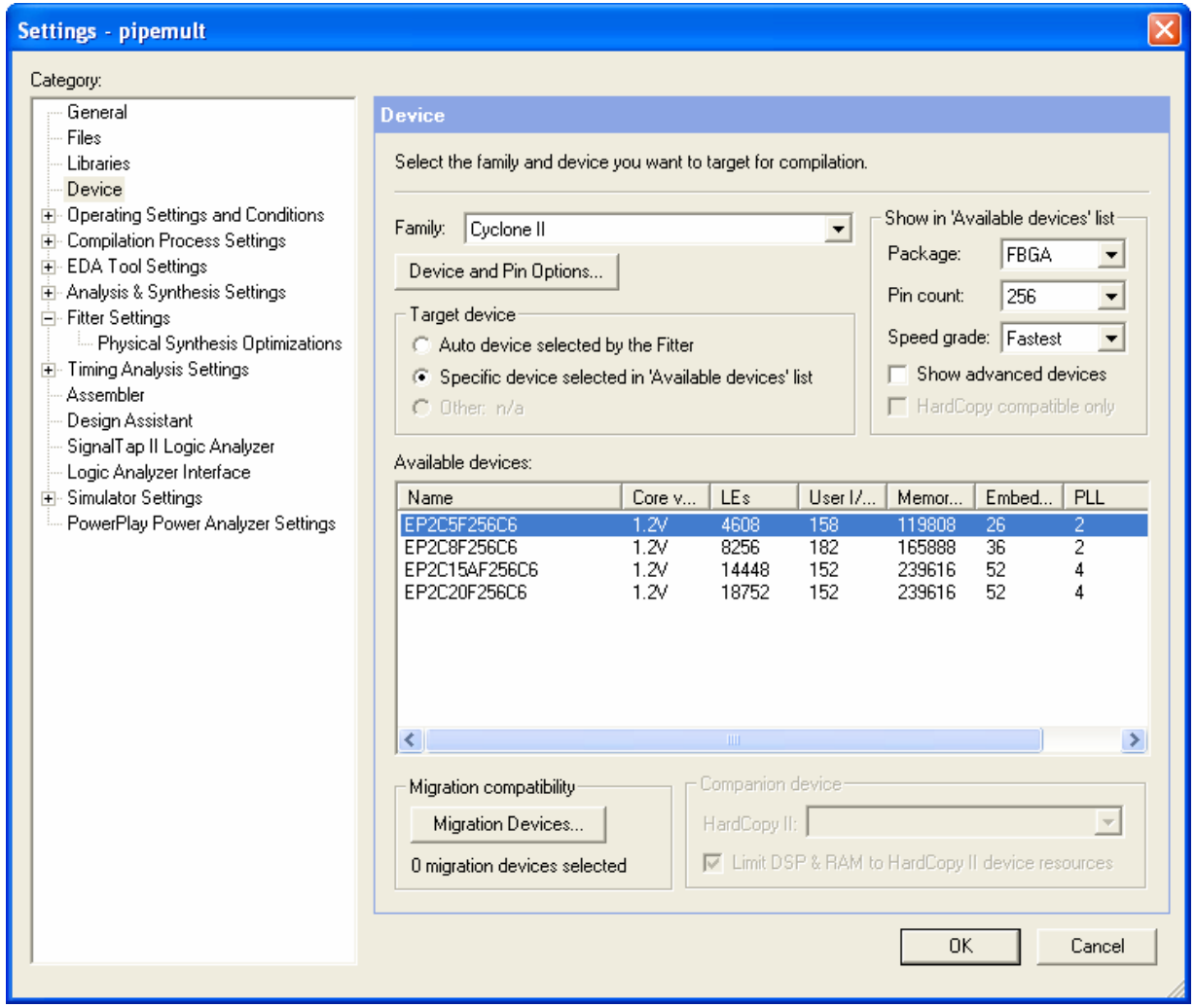
- Controlled using two methods
  - Settings
    - Project-wide switches
  - Assignments (i.e. logic options; constraints)
    - Individual entity/node controls
- Accessed using Assignments menu
- Stored in QSF file for project/revision



Settings

- Project-wide switches
- Examples
  - Device selection
  - Synthesis optimization
  - Fitter settings
  - Physical synthesis
  - Design Assistant
- Located in Settings dialog box (Assignments menu)

Settings Dialog Box

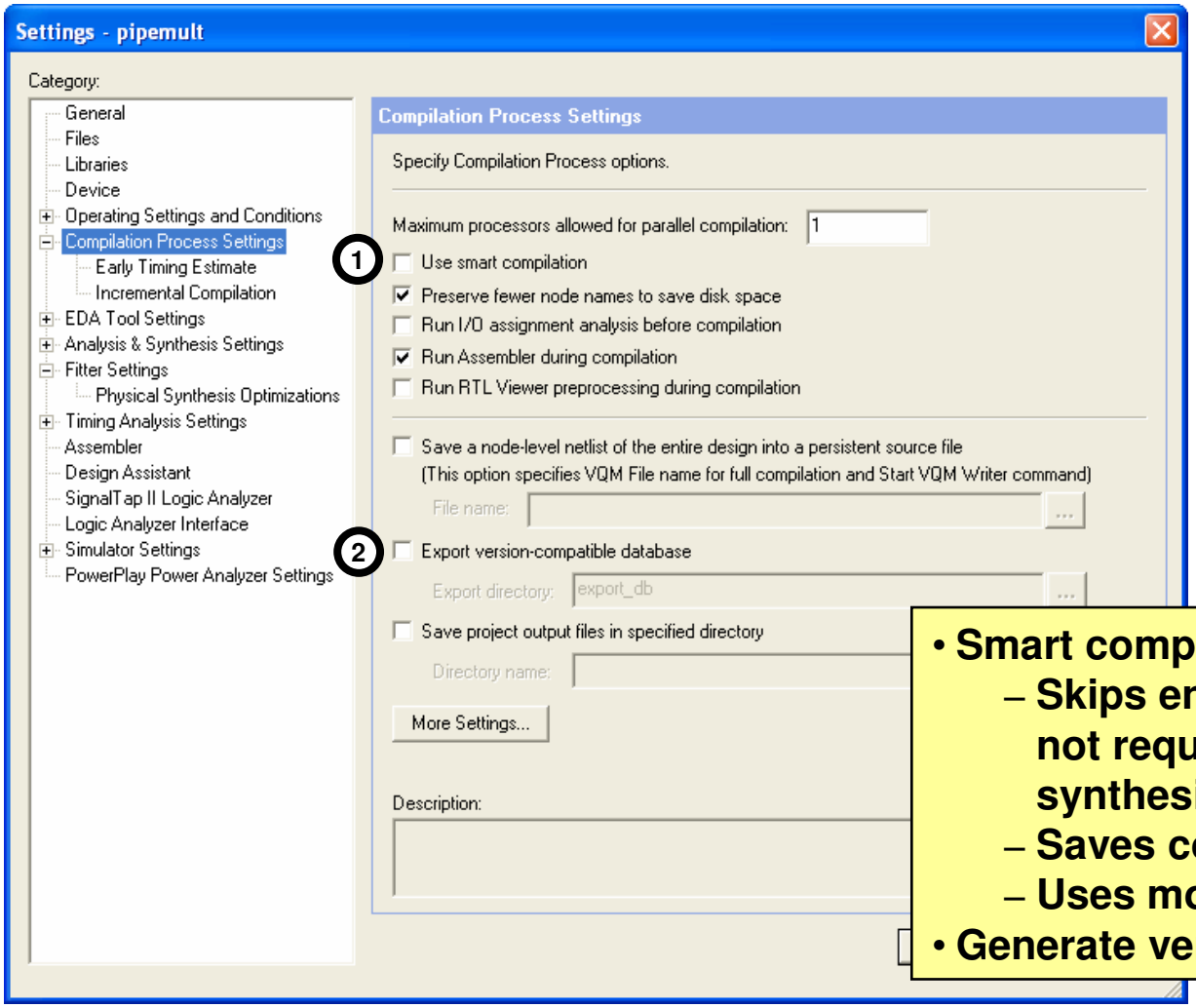


Change settings

- Top-level entity
- Target device
- Add/remove files
- Libraries
- VHDL ’87 or ‘93?
- Verilog ‘95, ’01 or SystemVerilog?
- EDA tool settings
- Timing settings
- Compiler settings
- Synthesis settings
- Fitter settings
- Simulator settings

Tcl: set_global_assignment –name <assignment_name*> <value>

Compilation Process



- **Smart compilation(1)**
  - **Skips entire compiler modules when not required (i.e. elaboration, synthesis, etc.)**
  - **Saves compiler time**
  - **Uses more disk space**
- **Generate version-compatible database(2)**

Tcl: set_global_assignment –name SMART_RECOMPILE ON

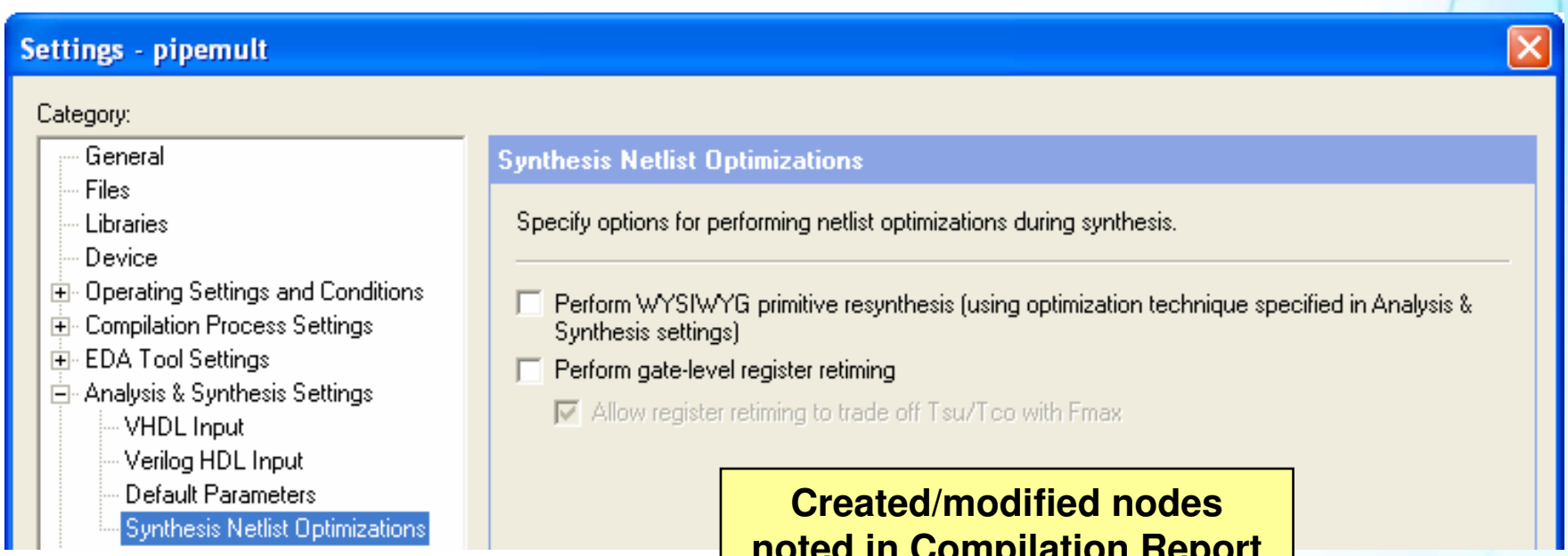
Version-Compatible Database

- Recommended if migrating design between versions of Quartus II software
- Exports a database from one version that can be imported directly into another version
- Use to preserve compilation results between Quartus II software versions
  - Re-running timing analysis or simulation with updated timing models
- Two methods to create
  - Settings dialog box
  - Project menu

Tcl: set_global_assignment –name AUTO_EXPORT_VER_COMPATIBLE ON
Tcl: set_global_assignment –name VER_COMPATIBLE_DB_DIR <directory_name>

Synthesis Netlist Optimizations

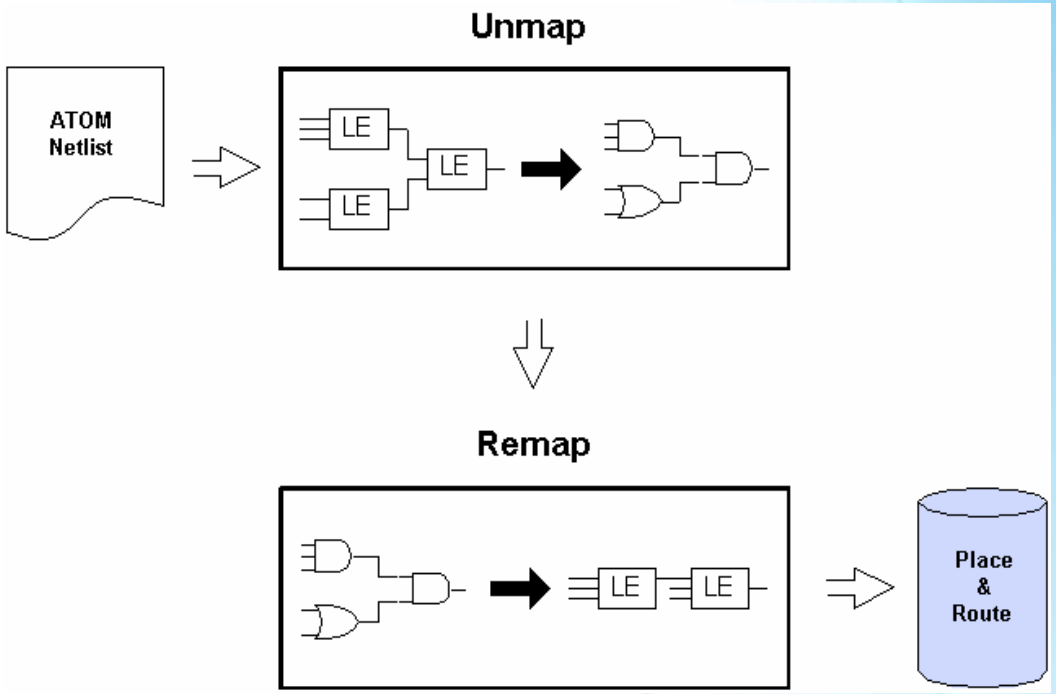
- Further optimize netlists during synthesis
- Types
  - WYSIWYG primitive resynthesis
  - Gate-level register retiming



Created/modified nodes noted in Compilation Report

WYSIWYG Primitive Resynthesis

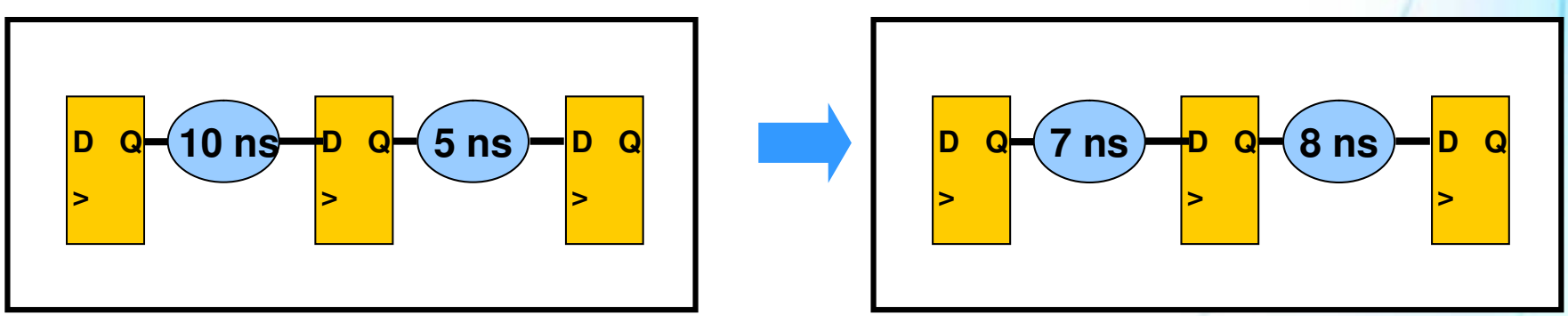
- Unmaps 3rd-party atom netlist back to gates & then remaps to Altera primitives
  - Unnecessary when using integrated synthesis
- Considerations
  - Node names may change
  - 3rd-party synthesis attributes may be lost
    - Preserve/keep
  - Some registers may be synthesized away



Tcl: set_global_assignment –name ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP ON

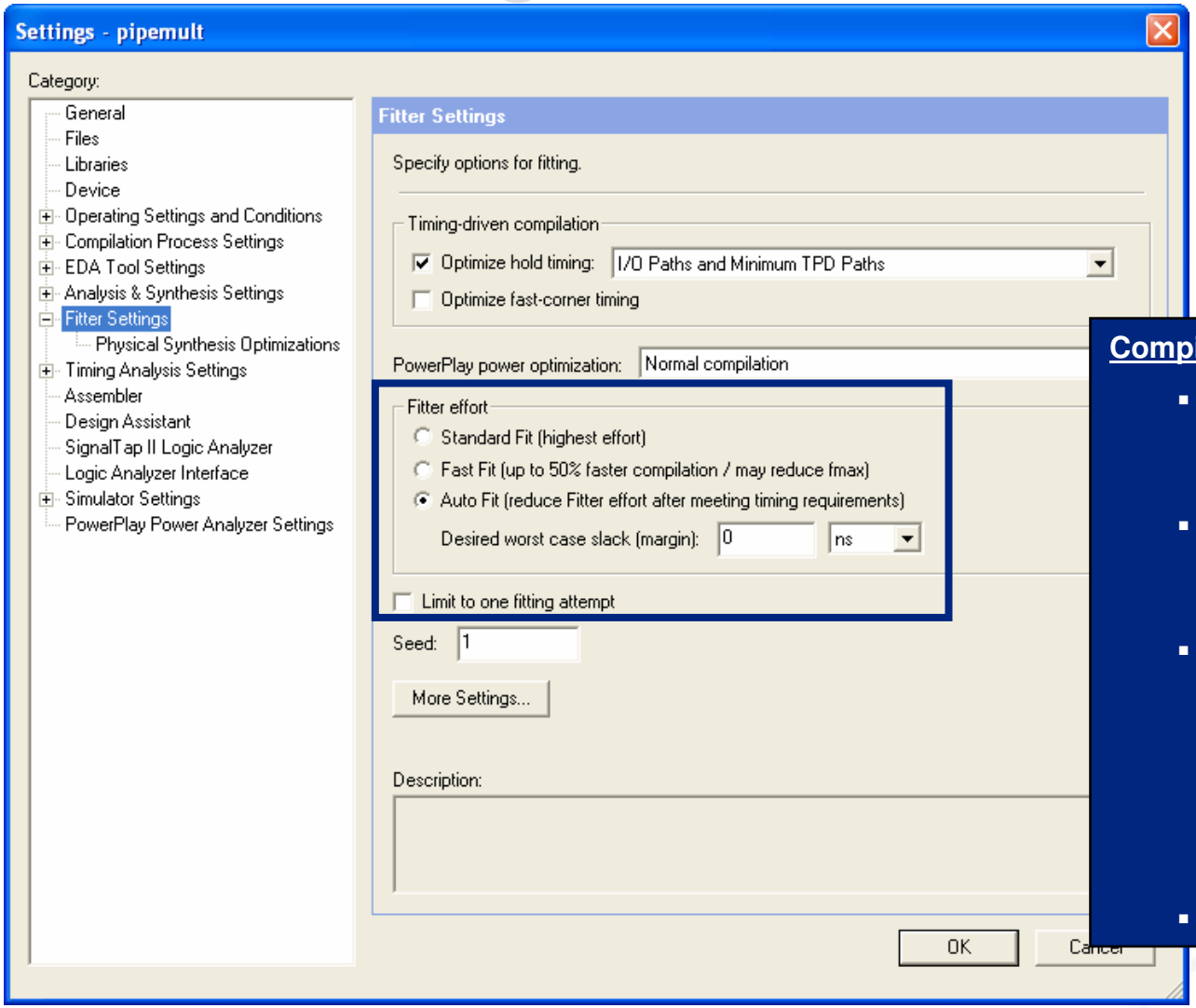
Gate-Level Register Retiming

- Moves registers across combinatorial logic to balance timing
- Trades between critical & non-critical paths
- Makes changes at gate level



Tcl: set_global_assignment –name ADV_NETLIST_OPT_SYNTH_GATE_RETIME ON

Fitter Settings



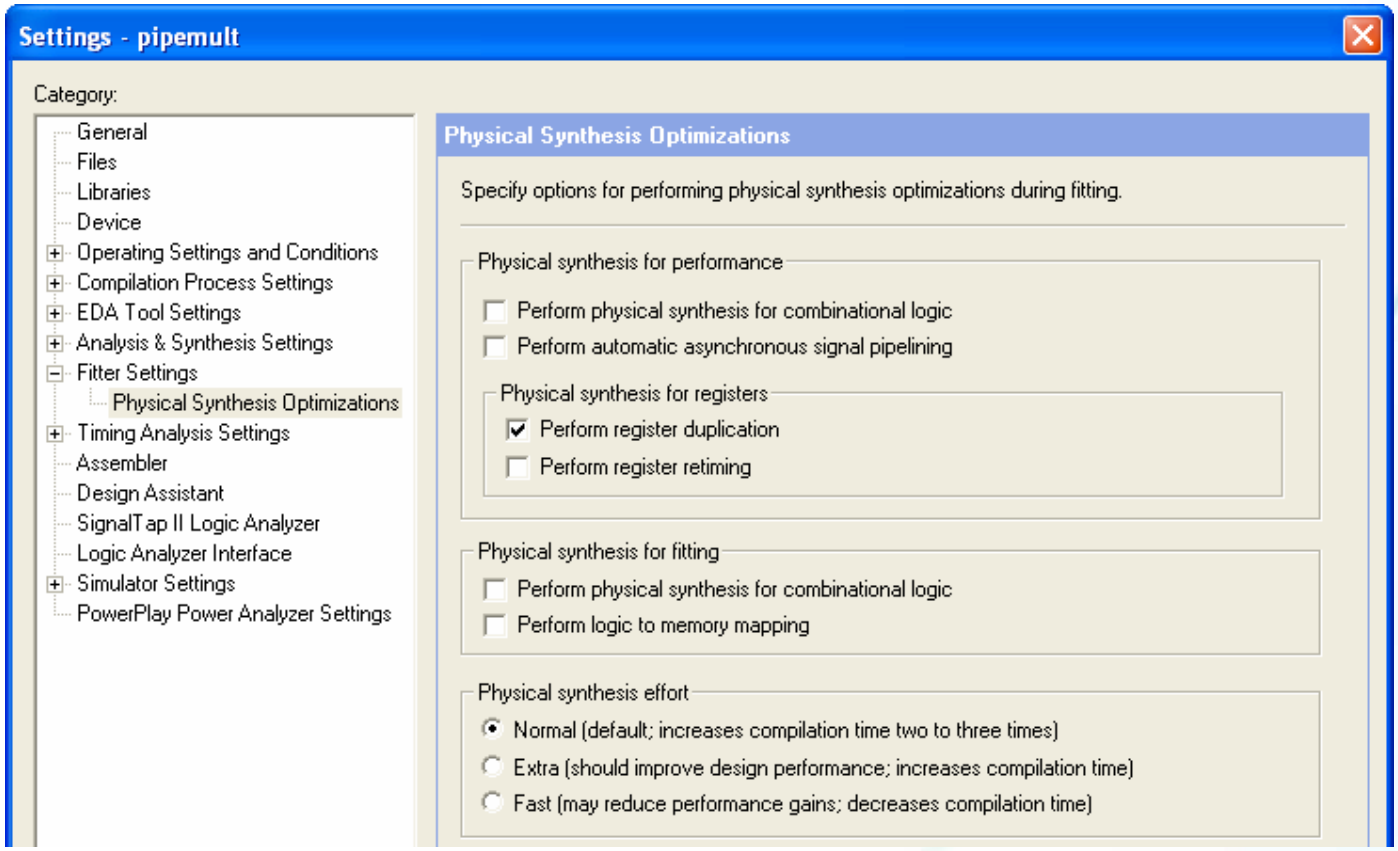
Compilation speed/fitter effort

- **Standard fit**
  - **Highest effort**
  - **Longest compile time**
- **Fast fit**
  - **Faster compile but possibly lesser design performance**
- **Auto fit**
  - **Compile stops after meeting timing**
  - **Conserves CPU time**
  - **Will mimic standard fit for hard-to-fit designs**
  - **Default for new designs**
- **One fitting attempt**

Tcl: set_global_assignment –name FITTER_EFFORT "<Effort Level>"

Physical Synthesis

- Re-synthesis based on fitter output
  - Makes incremental changes that improve results for a given placement
  - Compensates for routing delays from fitter



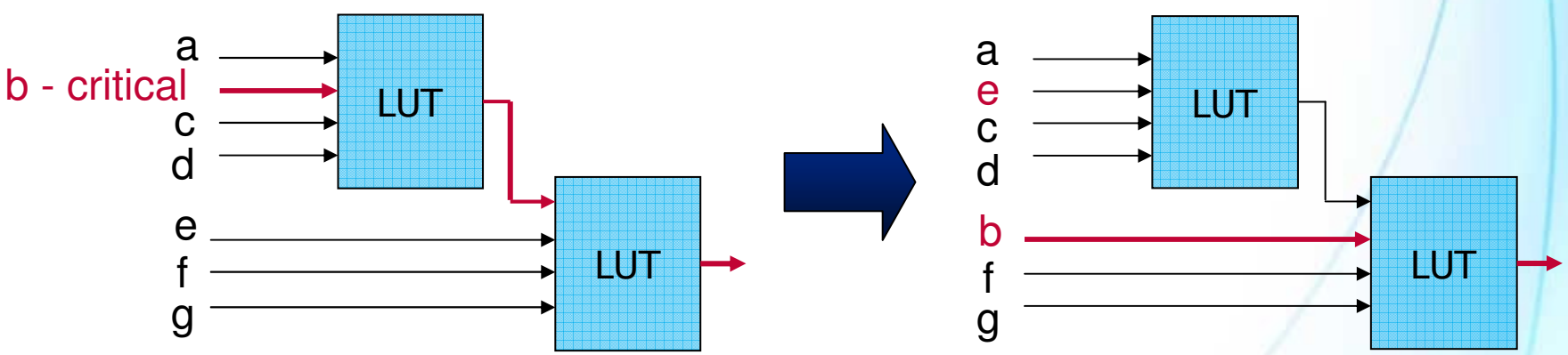
Physical Synthesis

- Types
  - Performance optimization
    - Combinational logic
    - Asynchronous signal pipelining
    - Register duplication
    - Register retiming
  - Area optimization
    - Combinational logic
    - Logic to memory mapping
- Effort
  - Trades performance vs. compile time
  - Normal, extra, or fast
- New or modified nodes appear in Compilation Report

Tcl: set_global_assignment –name PHYSICAL_SYNTHESIS_EFFORT <Effort Level>

Combinational Logic

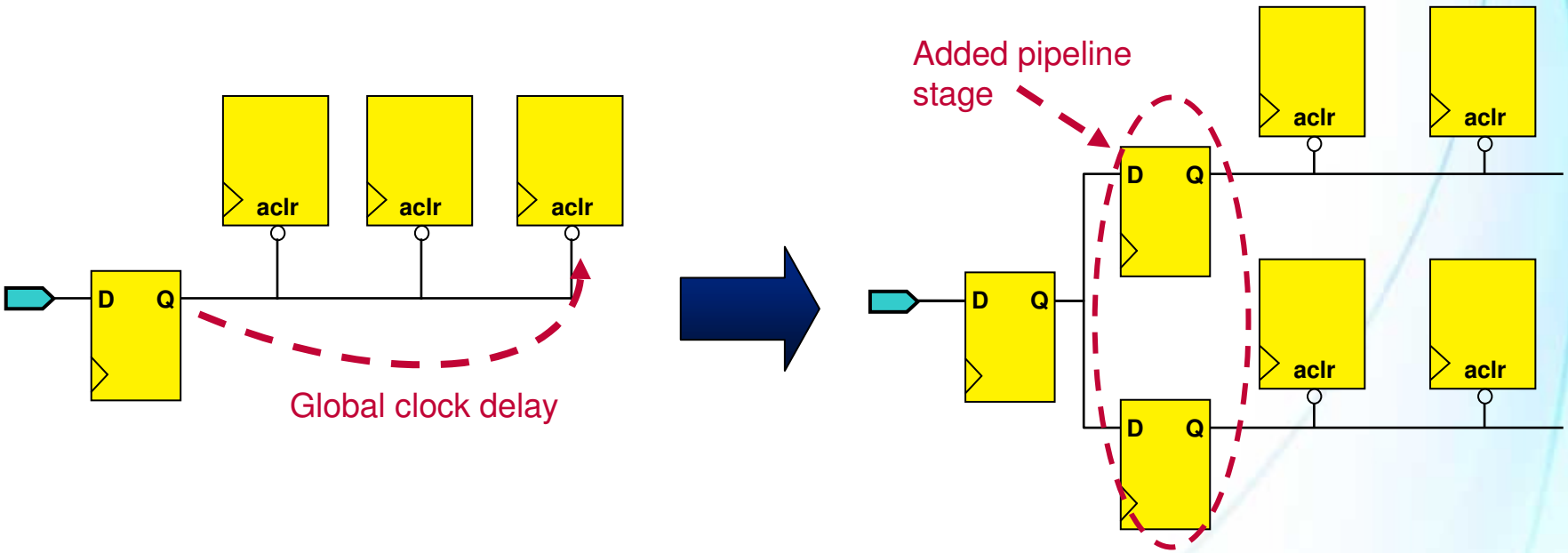
- Swaps look-up table (LUT) ports within LEs to reduce critical path LEs



Tcl: set_global_assignment –name PHYSICAL_SYNTHESIS_COMBO_LOGIC ON

Asynchronous Signal Pipelining

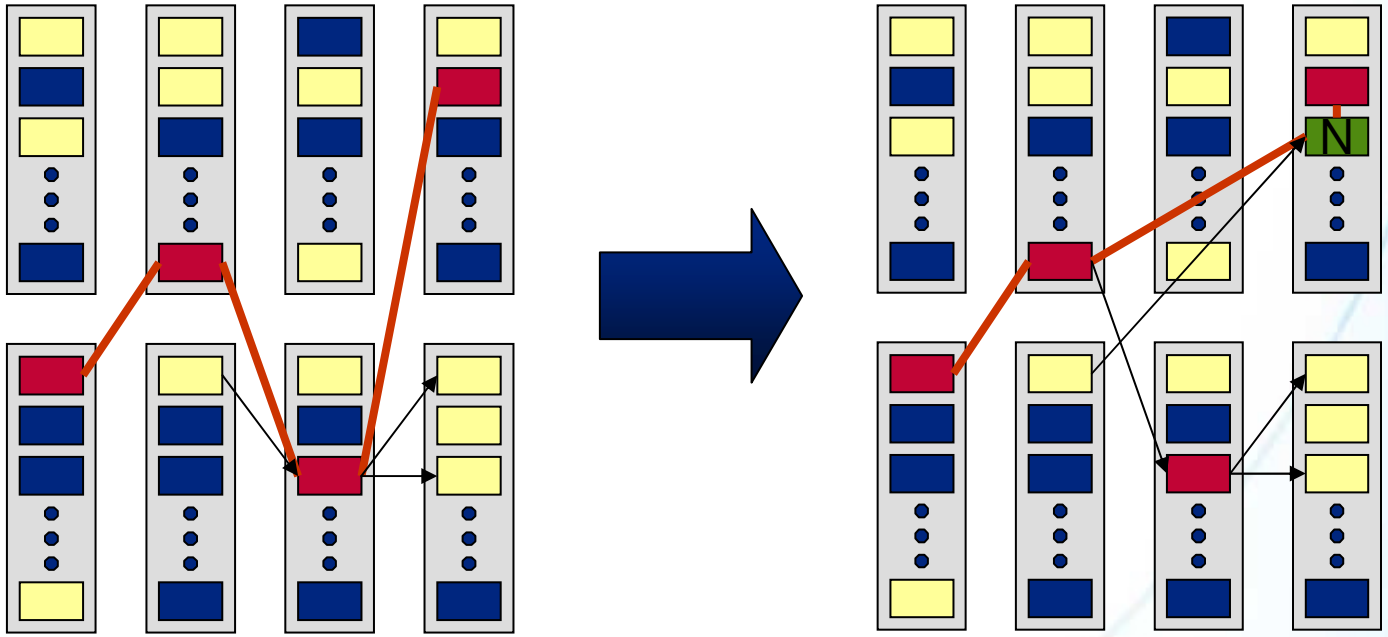
- Adds pipeline registers to asynchronous clear or load signals in very fast clock domains



Tcl: set_global_assignment –name PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING ON

Duplication

- High fan-out registers or combinatorial logic duplicated & placed to reduce delay



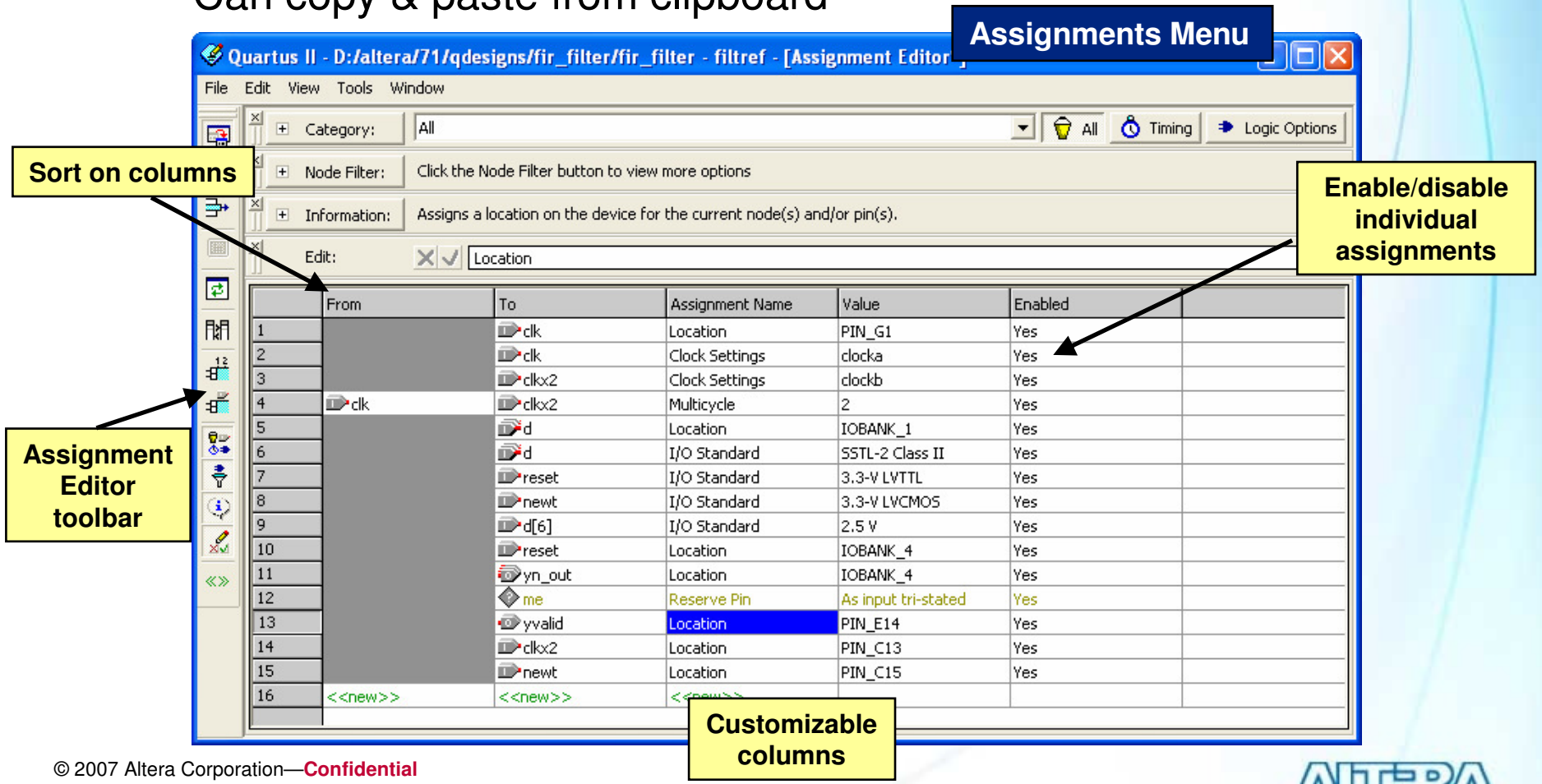
Tcl: set_global_assignment –name PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION ON

Assignments (Logic Options)

- Individual switches applied to I/O, internal nodes or hierarchy blocks
- Use Assignment Editor to manage assignments
- Example assignments
  - Optimization Technique
  - PCI I/O
- Must perform at least analysis & elaboration to obtain hierarchy & node information

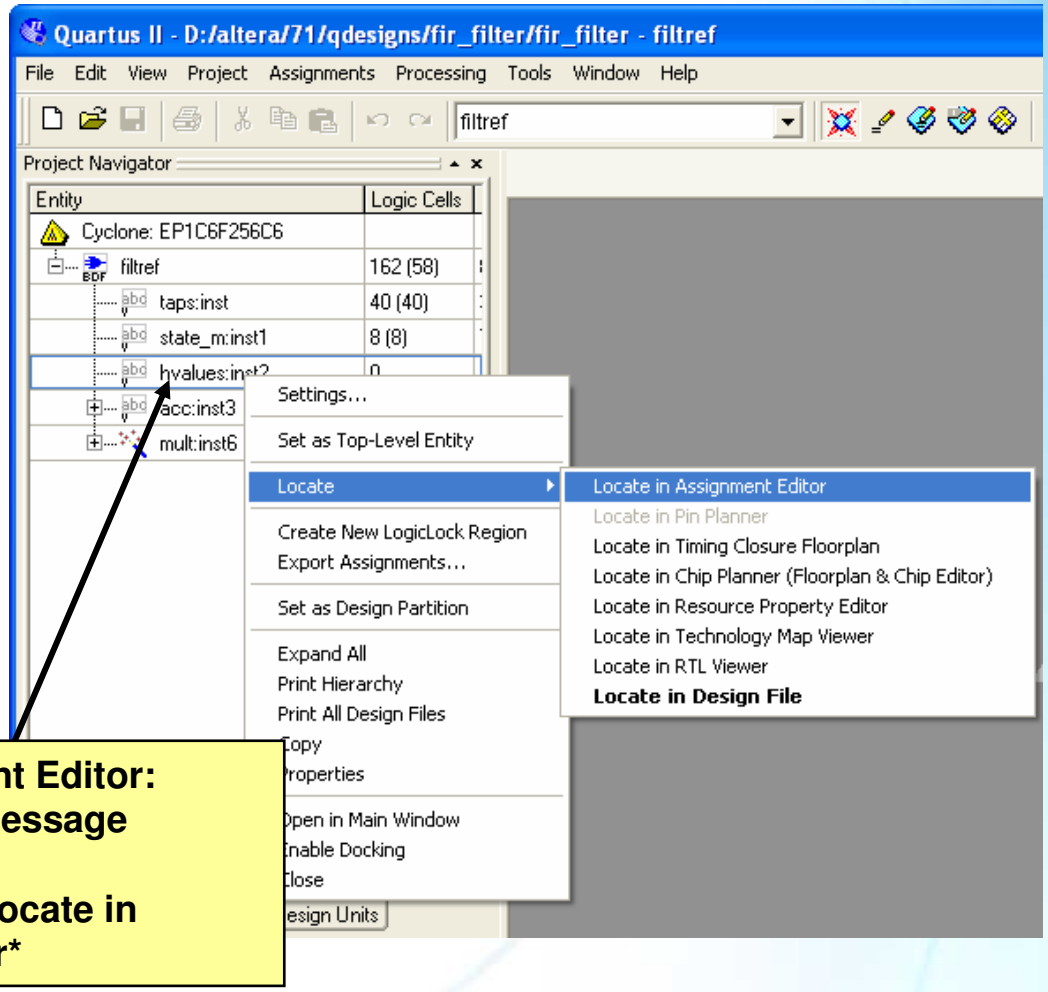
Assignment Editor (AE)

- Provides spreadsheet assignment entry & display
  - Can copy & paste from clipboard



Cross-Probing to Assignment Editor

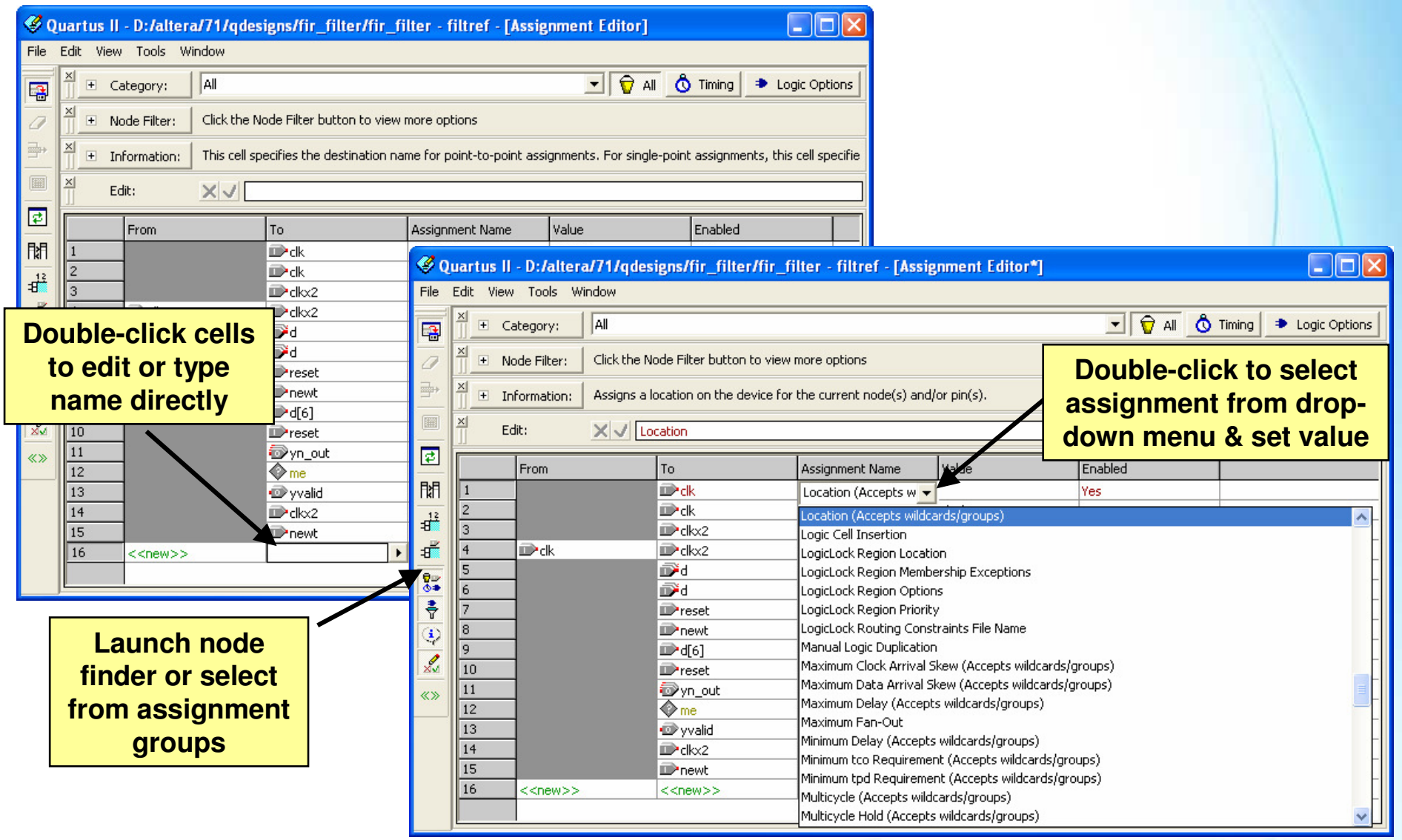
- Virtually all windows & tools cross-probe (locate) to Assignment Editor
- Examples
  - Project Navigator
  - Message window
  - Compilation Report
  - Design files



To invoke Assignment Editor:
1) Highlight object/message
2) Right-click
3) Select Locate ⇒ Locate in Assignment Editor*

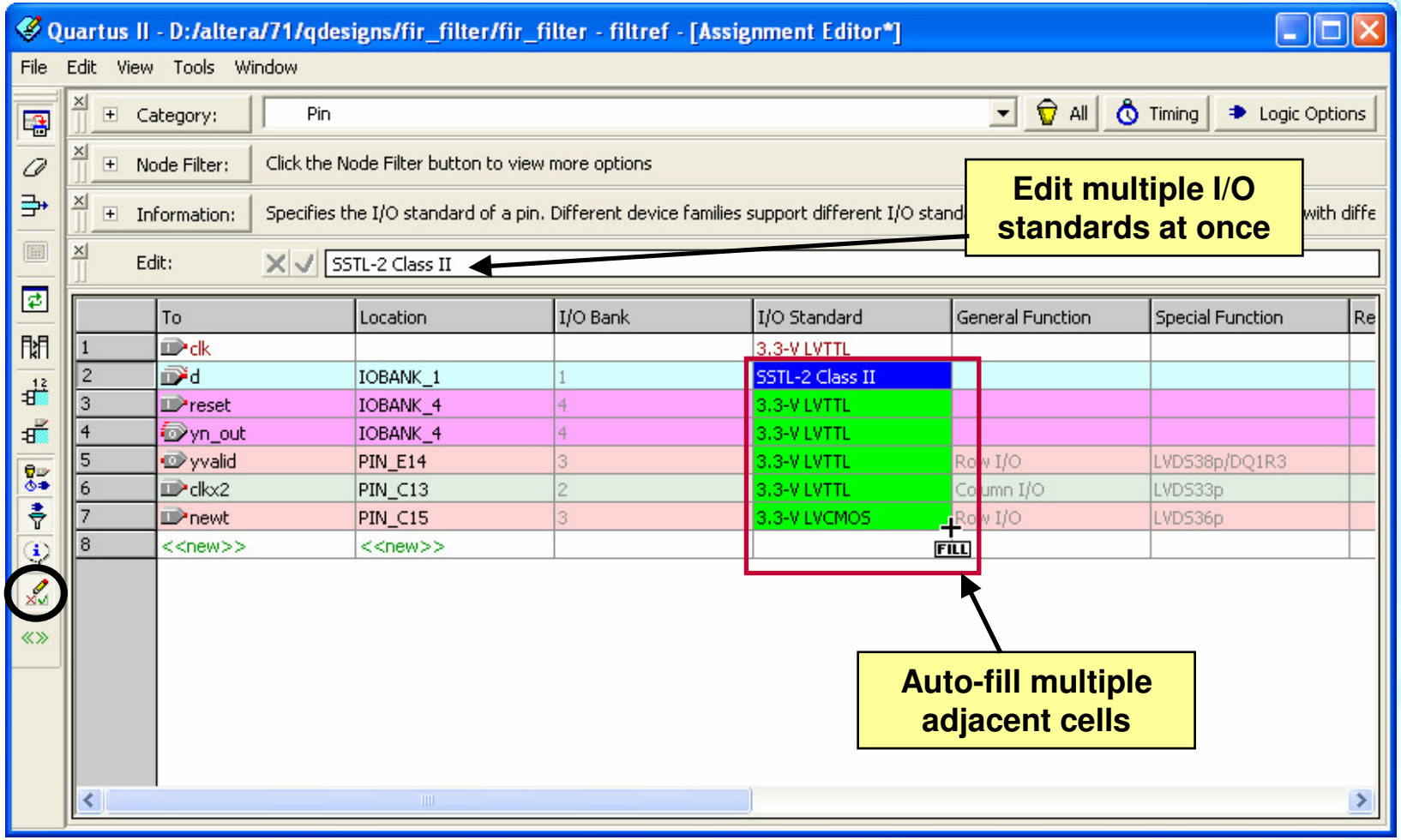
****Note:* Assignment Editor pre-filled with target node/pin name**

Using Assignment Editor



Editing Multiple Assignments

- Use Edit bar, auto-fill, copy & paste



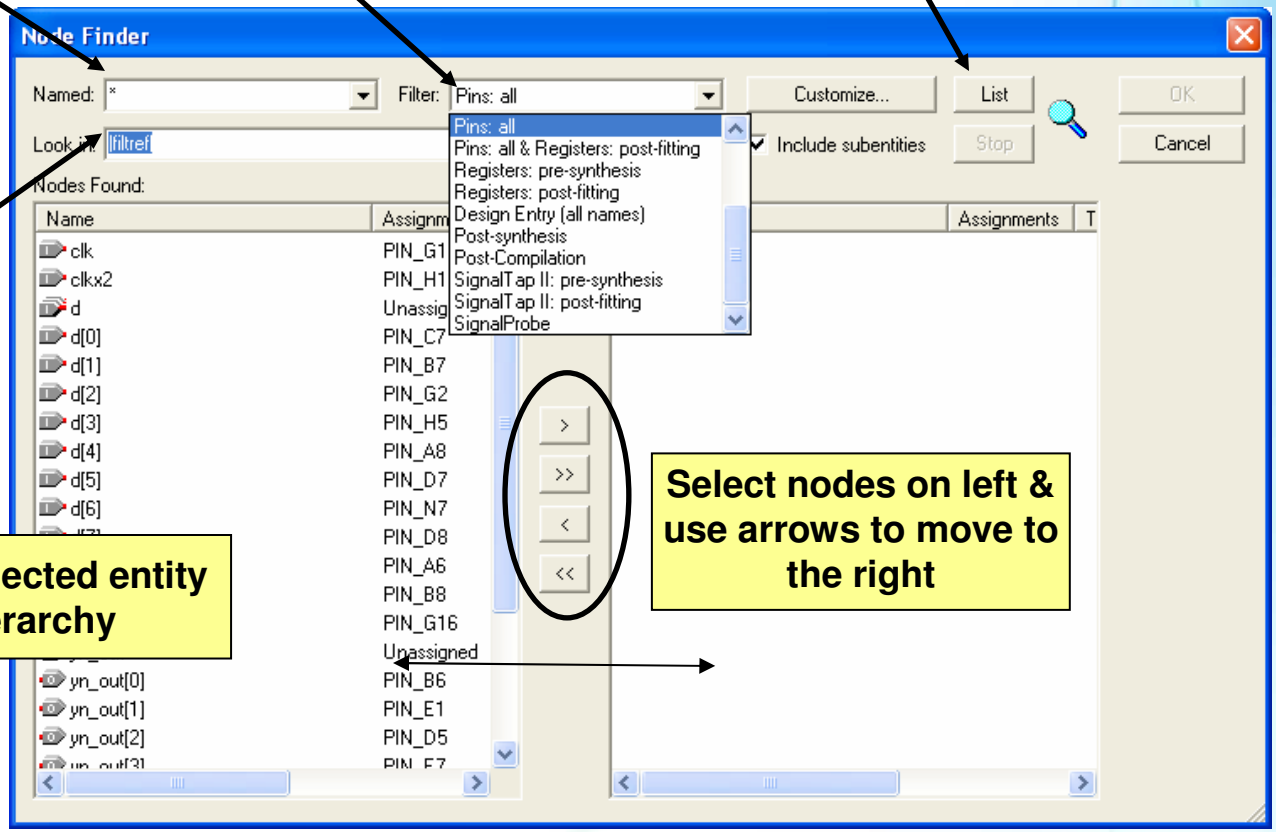
Node Finder

Search by name using wildcards (? or *)

Use filter to select the type of nodes to be displayed

Displays nodes meeting search criteria

Locate nodes in a certain level of hierarchy

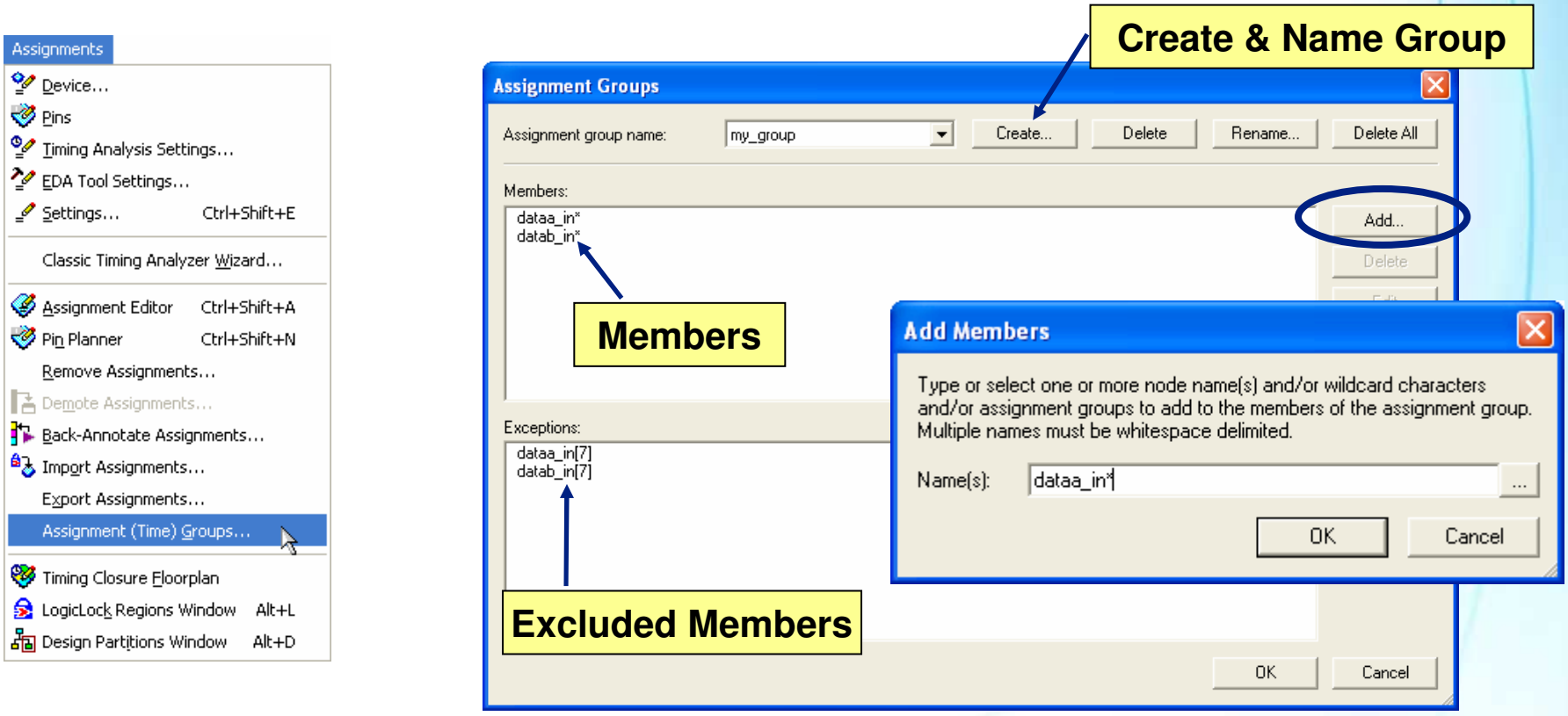


List of found nodes in selected entity & lower levels of hierarchy

Select nodes on left & use arrows to move to the right

Assignment (Time) Groups

- Assigns named to user-defined group of nodes
- Allows single assignment to constrain entire group



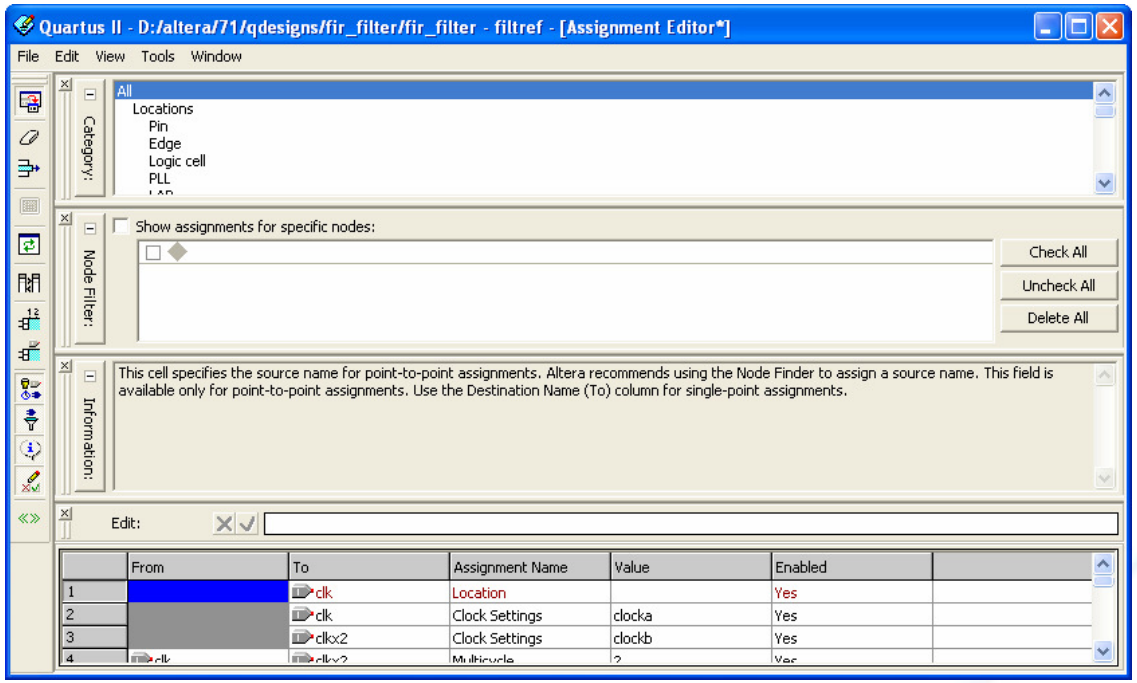
AE Dynamic Checking

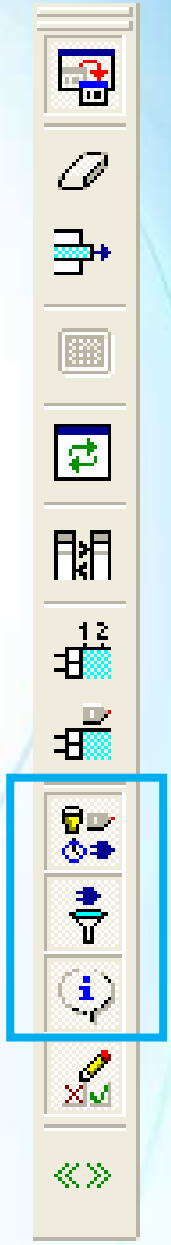
- Validity of constraint checked during entry
- Color-coded to display status
  - Grey – disabled
  - Black – applied
  - Yellow – assignment warning
  - Dark red – incomplete
  - Bright red – error/illegal value
  - Green – enter new assignment

	From	To	Assignment Name	Value	Enabled
1		yn_out	Location	IOBANK_2	Yes
2		yvalid	Location	PIN_75	Yes
3		d	Location	IOBANK_1	No
4		clk	Clock Settings	clk	Yes
5		unknown_clock	Clock Settings	clk2	Yes
6		clkx2	Clock Settings	clk2	Yes
7	clk	clkx2	Multicycle	2	Yes
8		d	DQS Frequency	1MHz	Yes
9		yvalid		Minimum Current	Yes
10		d	I/O Standard	LVCMOS	Yes
11		yn_out	I/O Standard	LVCMOS	Yes
12		yvalid	I/O Standard	LVCMOS	Yes
13	<<new>>	<<new>>	<<new>>		

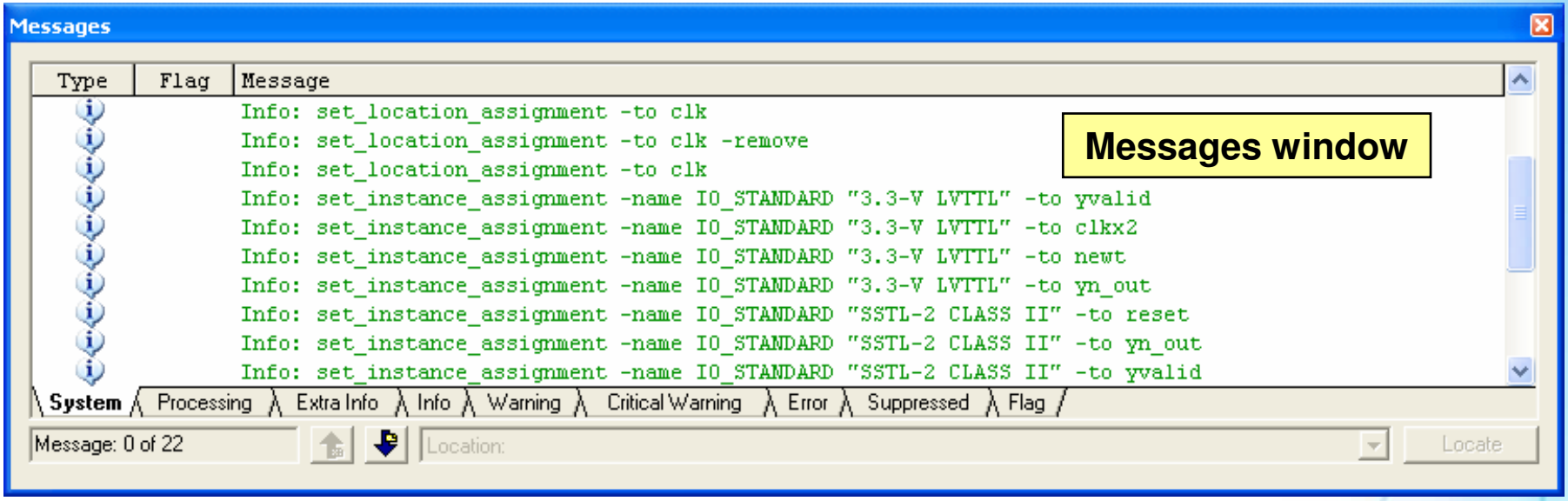
Assignment Editor Features

- Category bar
  - Filters displayed constraints based on category
    - Ex. Pin assignments, timing assignments
- Node Filter bar
  - Filters displayed constraints based on node name
- Information bar
  - Displays description of selected cell or assignment





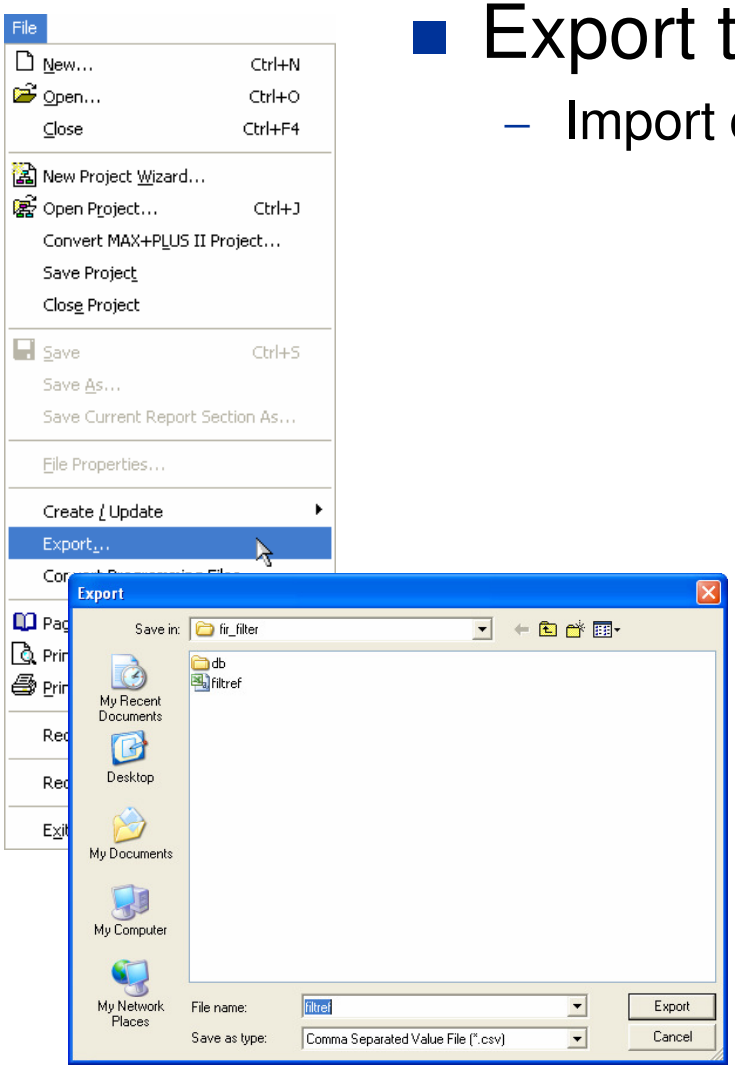
AE Tcl Commands



- Equivalent Tcl commands displayed as assignments are entered
  - Manually copy to create Tcl scripts
  - Export command (File menu) writes all assignments to a Tcl file

Export CSV File Assignments (Excel)

- Export to CSV file (File menu)
  - Import data into Excel



From	To	Assignment Name	Value
	~DATA0~	Location	PIN_L8
	d[7]	Location	PIN_J4
	d[6]	Location	PIN_H4
	d[5]	Location	PIN_E6
	d[4]	Location	PIN_F1
	d[3]	Location	PIN_H3
	d[2]	Location	PIN_J6
	d[1]	Location	PIN_G4
	d[0]	Location	PIN_F2
	yn[7]	Location	PIN_H2
	yn[6]	Location	PIN_L6
	yn[5]	Location	PIN_G2
	yn[4]	Location	PIN_J2
	yn[3]	Location	PIN_G1
	yn[2]	Location	PIN_J3
	yn[1]	Location	PIN_H1
	yn[0]	Location	PIN_K2
	clk	Location	PIN_L2
	reset	Location	PIN_L3
	newt	Location	PIN_K6
	yvalid	Location	PIN_E5
	nxt	Location	PIN_A6
	yn	Output Maximum Delay	4ns
clk	input data	Input Maximum Delay	7ns
mult_mega:u4|*	acc:u5|*	Cut Timing Path	On
	clk	Clock Settings	clk

Example Assignments

- Optimization Technique
- PCI I/O

Optimization Technique

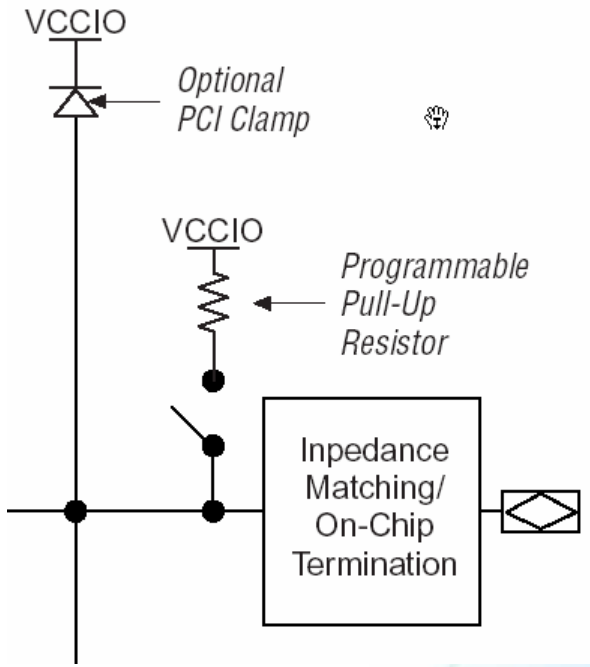
- Selects synthesis optimization goal
  - Speed
  - Balanced (default)
  - Area
- Applies only to hierarchical entities
  - Locate from Project Navigator
  - Drag and drop into Assignment Editor
- Effects synthesis & logic mapping
- Only applies to Quartus II integrated synthesis

	From	To	Assignment Name	Value	Enabled	
1		acc:b2v_inst3	Optimization Technique -- Stratix II	Speed	Yes	
2	<<new>>	<<new>>	<<new>>			

Tcl: set_instance_assignment –name STRATIXII_OPTIMIZATION_TECHNIQUE SPEED –to <node name>

PCI I/O

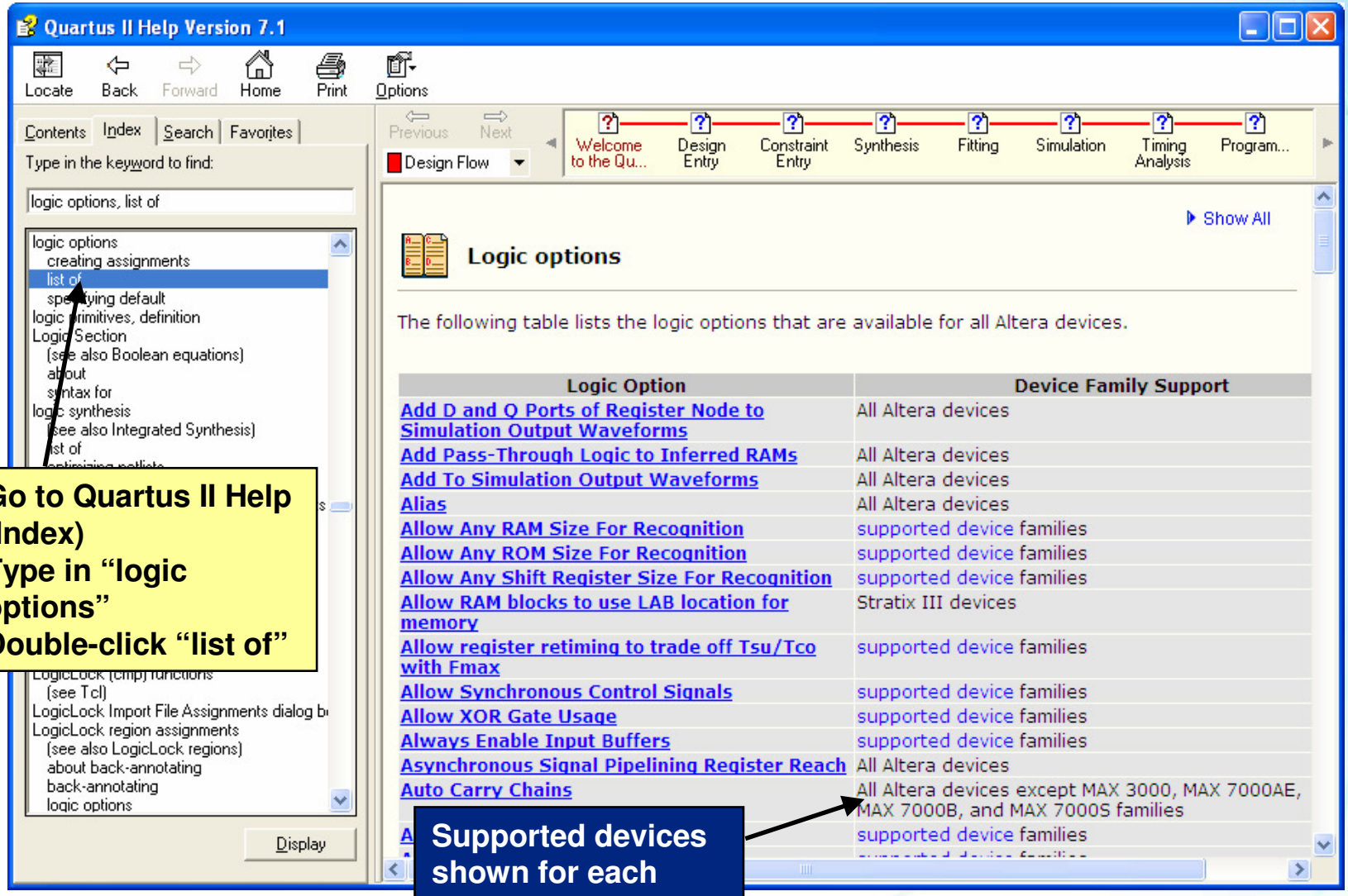
- Turns on PCI compatibility for pins
  - Ignored if applied to anything other than a pin or a top-level design entity
- Controls clamping diode located in the I/O elements



	From	To	Assignment Name	Value	Enabled	
1		yn_out[0]	PCI I/O	On	Yes	
2		yn_out[1]	PCI I/O	On	Yes	
3		yn_out[2]	PCI I/O	On	Yes	
4		yn_out[3]	PCI I/O	On	Yes	
5	<<new>>	<<new>>	<<new>>			

Tcl: set_instance_assignment –name PCI_IO ON –to <pin name>

Available Logic Options (Assignments)



1) Go to Quartus II Help (Index)
2) Type in "logic options"
3) Double-click "list of"

Supported devices shown for each assignment

Updating QSF File

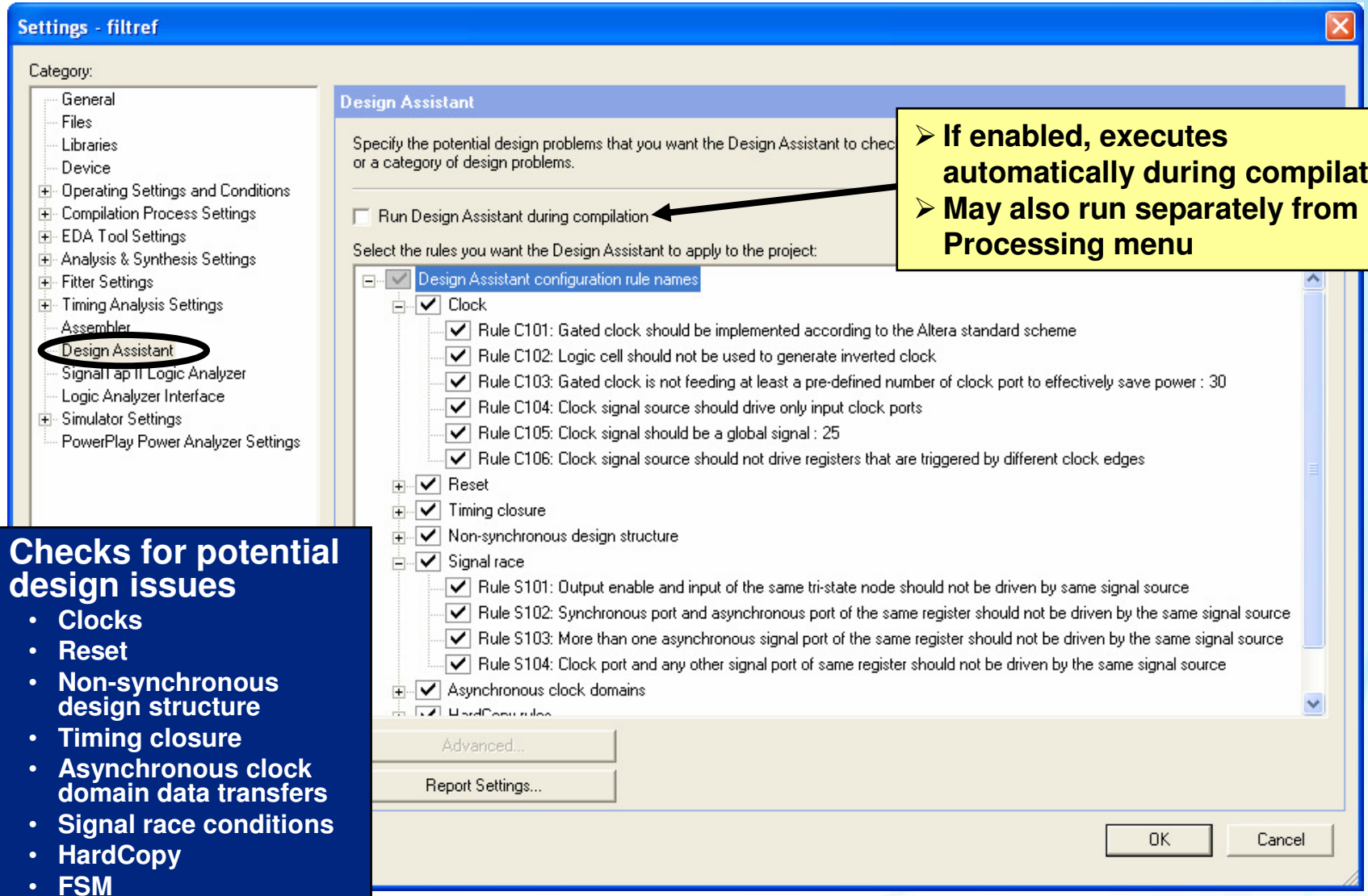
- QSF not updated automatically when constraint entered or Assignment Editor is saved
- QSF updated when
  - Project is saved (File menu)
  - Beginning of compilation
- Change behavior to updating assignments immediately (Tools menu ⇒ Options ⇒ General ⇒ Processing)
  - Will impact software performance due to file accesses

Design Assistance

- Quartus II tools available to help improve designs

- Design Assistant
- Optimization Advisors
  - Resource
  - Timing
  - Power
  - Incremental Compilation
  - Compilation Time

Design Assistant

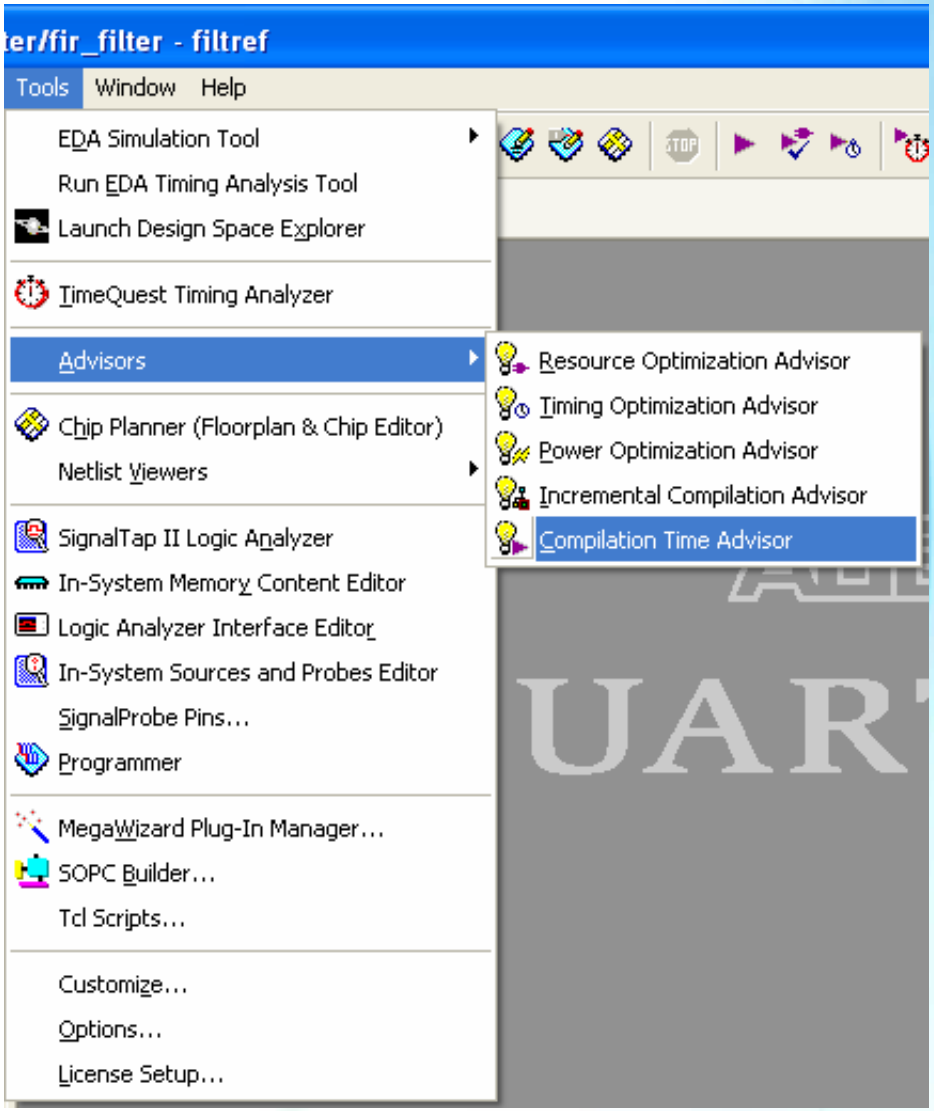


- If enabled, executes automatically during compilation
- May also run separately from Processing menu

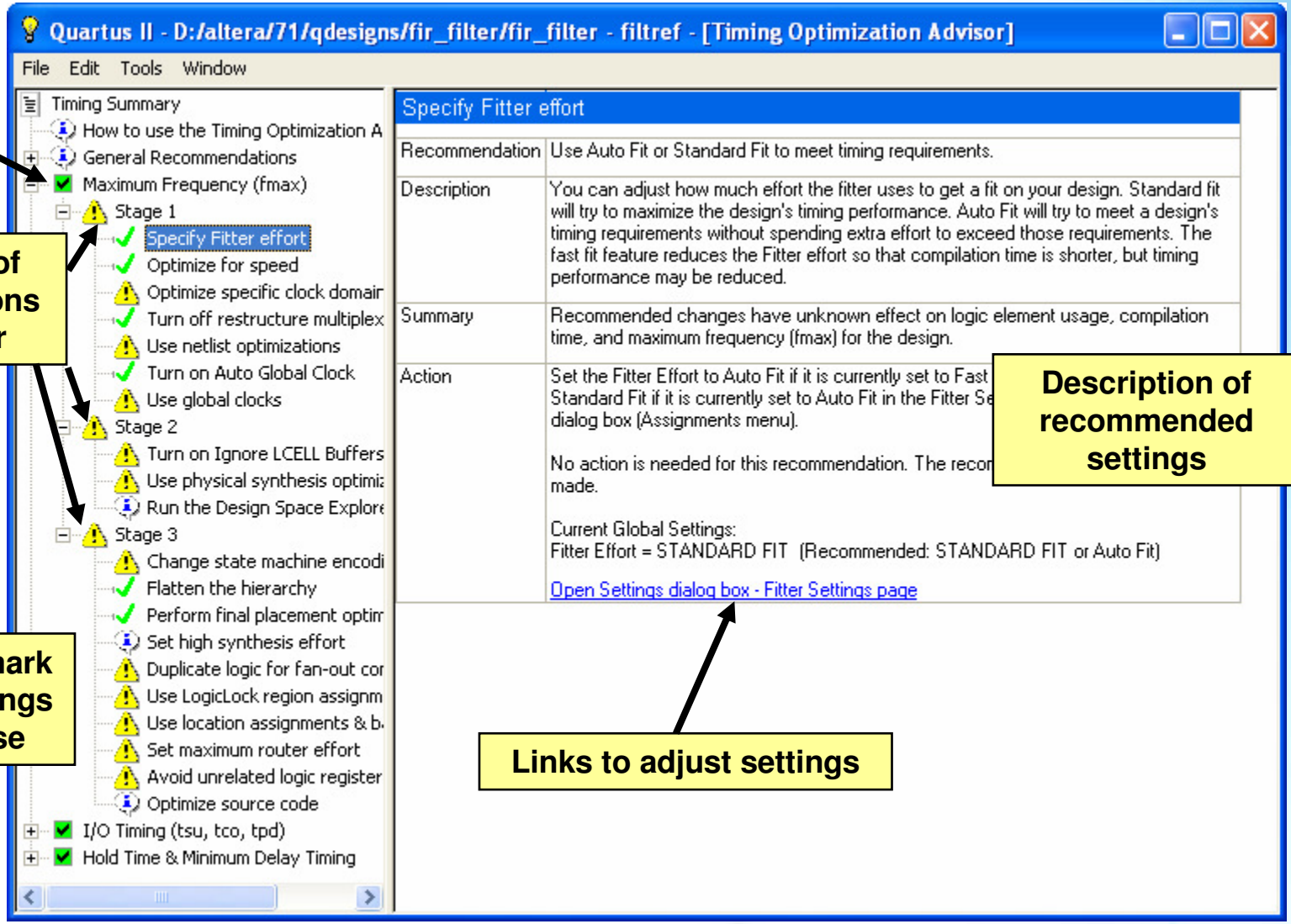
- **Checks for potential design issues**
  - Clocks
  - Reset
  - Non-synchronous design structure
  - Timing closure
  - Asynchronous clock domain data transfers
  - Signal race conditions
  - HardCopy
  - FSM

Optimization Advisors

- Provide design-specific recommendations (feedback) on optimizing designs
- Five types
  - Resource usage optimization
  - Timing (performance) optimization
  - Power optimization
  - Incremental compilation suggestions
  - Compilation time reduction



Example Optimization Advisor



Exercise 4 Demonstration

Settings & Assignments Summary

- Settings & assignments allow a designer to control how a design is synthesized, placed, & routed
- Use the Settings dialog box to adjust project-wide settings
- Use the Assignment Editor to enable/disable individual assignments targeting hierarchy blocks, internal nodes, or I/O
- The Quartus II software provides features such as the Design Assistant & Optimization Advisors to help improve design results

Design Analysis Support Resources

- Quartus II Handbook chapters (all Volume 2)
  - "Area & Timing Optimization"
  - "Power Optimization"
  - "Assignment Editor"
  - "Netlist Optimizations & Physical Synthesis"
- Training Courses and Demonstrations
  - Optimization Advisor demonstration

Quartus II Software Design Series: Foundation

I/O Planning

I/O Planning Need

- I/O standards increasing in complexity
- FPGA/CPLD I/O structure increasing in complexity
  - Results in increased pin placement guidelines
- PCB development performed simultaneously with FPGA design
  - Sometimes before!
- Pin assignments need to be verified earlier in design cycle
- Designers need easy way to transfer pin assignments into board tools

I/O Planning Agenda

- Assigning Device I/O Locations
- I/O Assignment Analysis
- Early I/O Planning Methodology
- PCB Tool Support

Assigning Device I/O Locations

- Pin Planner
- Import from spreadsheet in CSV format
- Type directly into QSF file
- Scripting
- Using synthesis attributes in HDL

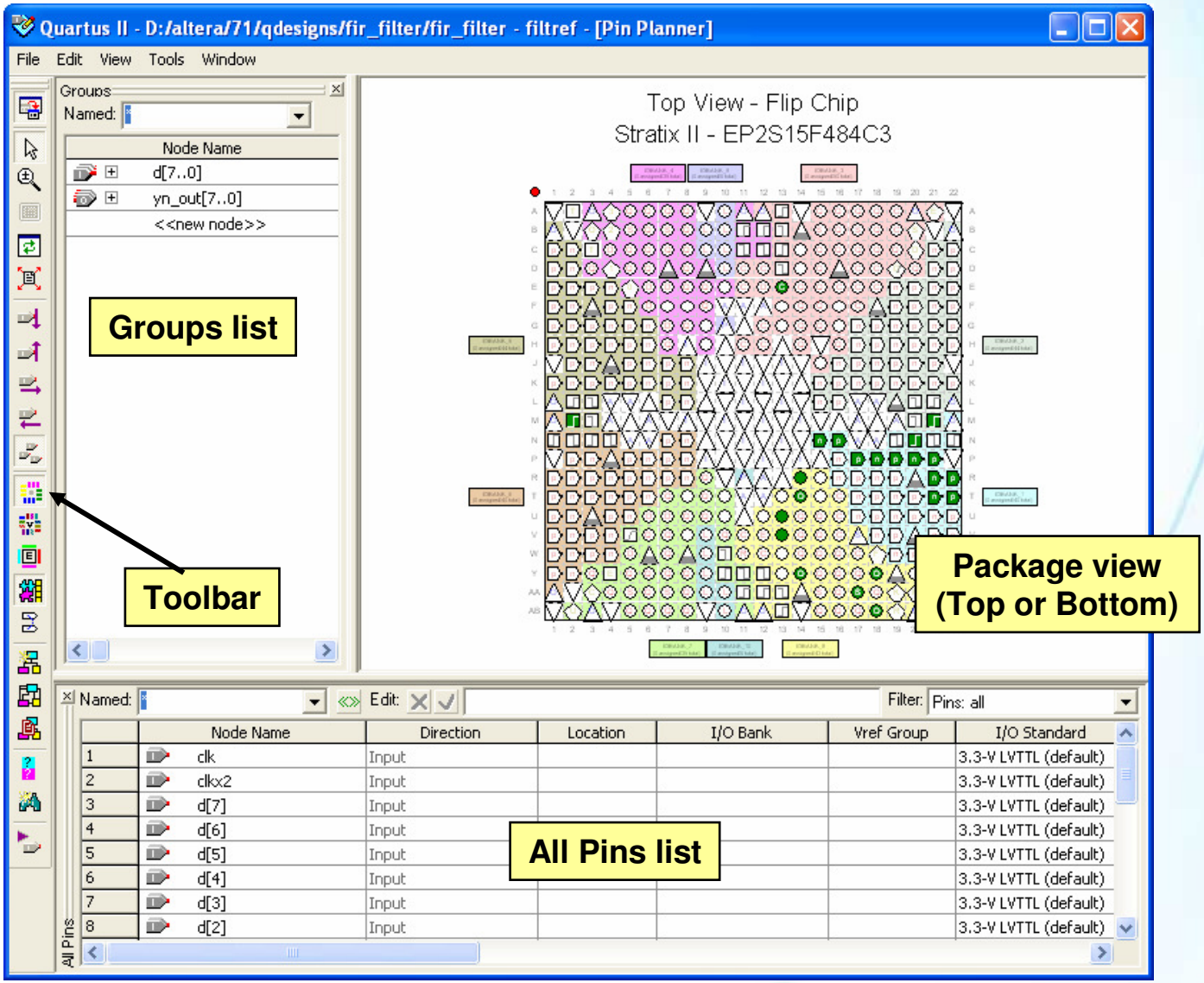
Note: Other methods/tools are available in Quartus II to make I/O assignments. The above are the most common or recommended.

Pin Planner

- Interactive graphical tool for assigning pins
  - Drag & drop pin assignments
  - Set pin I/O standards
  - Reserve future I/O locations
- Three main sections
  - Package view
  - All Pins list
  - Groups list

Assignments Menu ⇒ Pin Planner

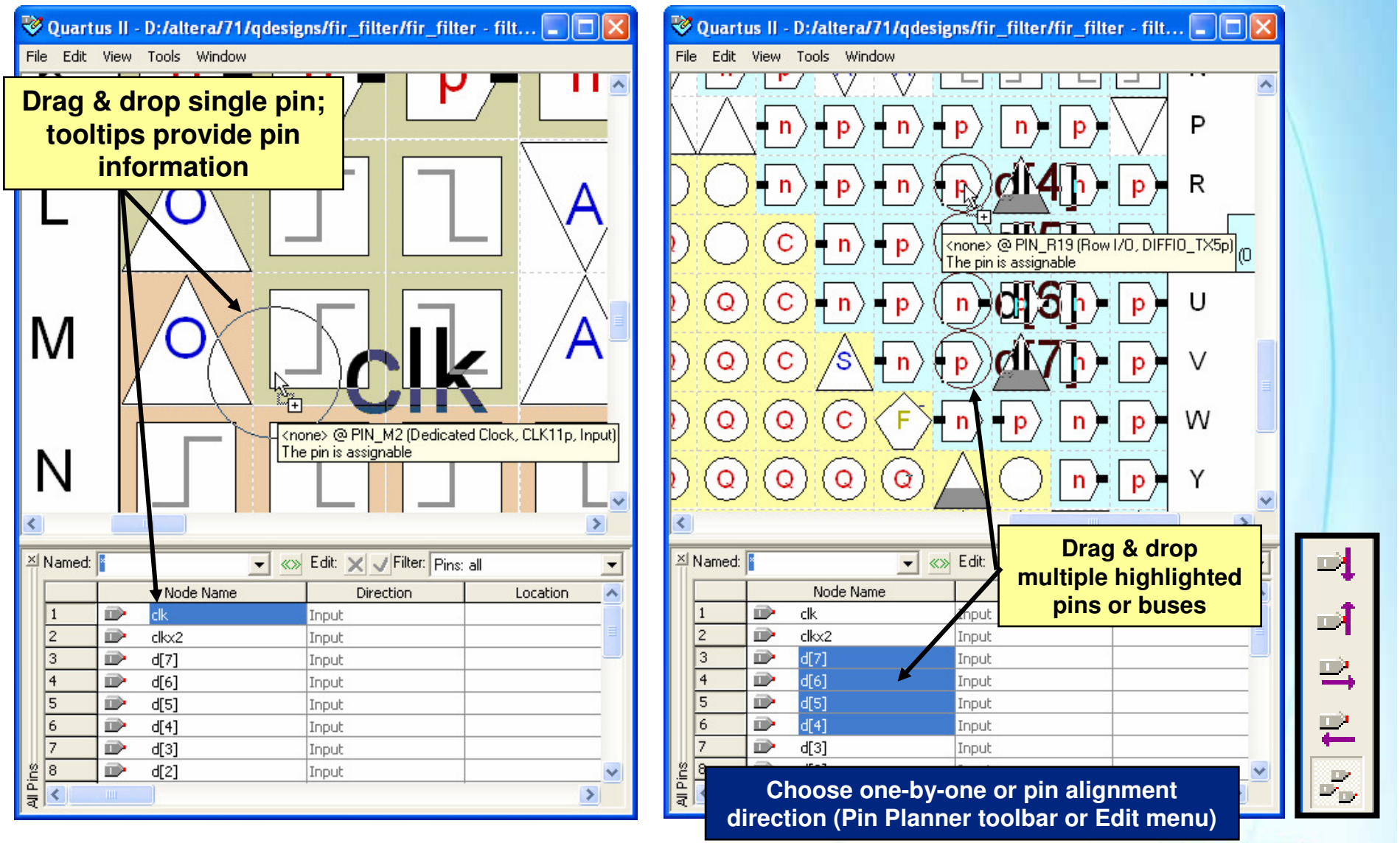
Pin Planner Window



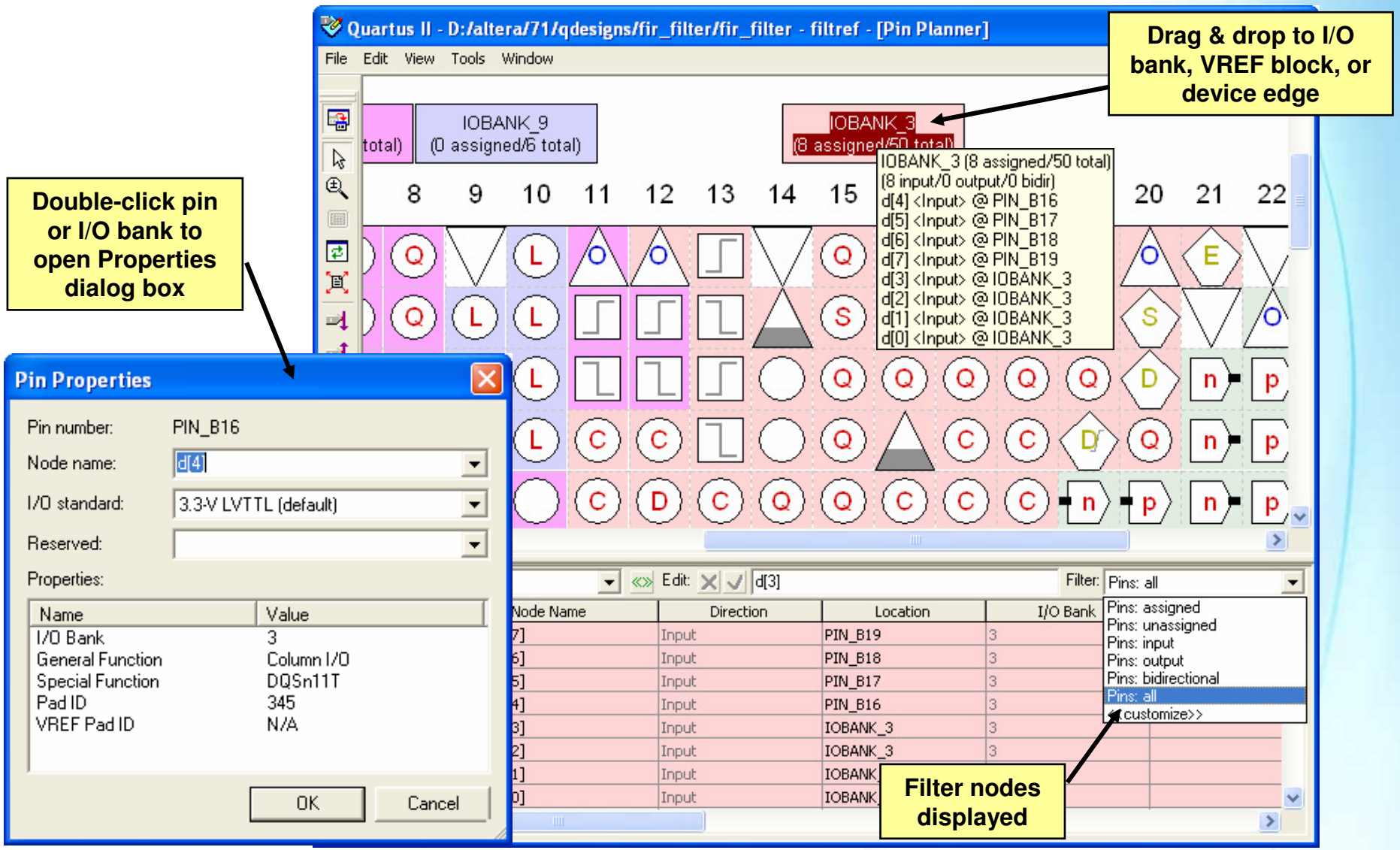
Pin Planner Window (cont.)

- Package view
  - Displays graphical representation of chip package
  - Use to make or edit design I/O assignments
  - Use to locate other package pins (i.e. power & configuration pins)
- All pins list
  - Displays pins as indicated by filter
    - Buses are auto-expanded
  - Use to edit pin settings/properties directly
- Groups list
  - Similar to All Pins list except displays only groups & buses
  - Use to make bus and group assignments
  - Use to create new user-defined groups

Assigning Pins Using Pin Planner

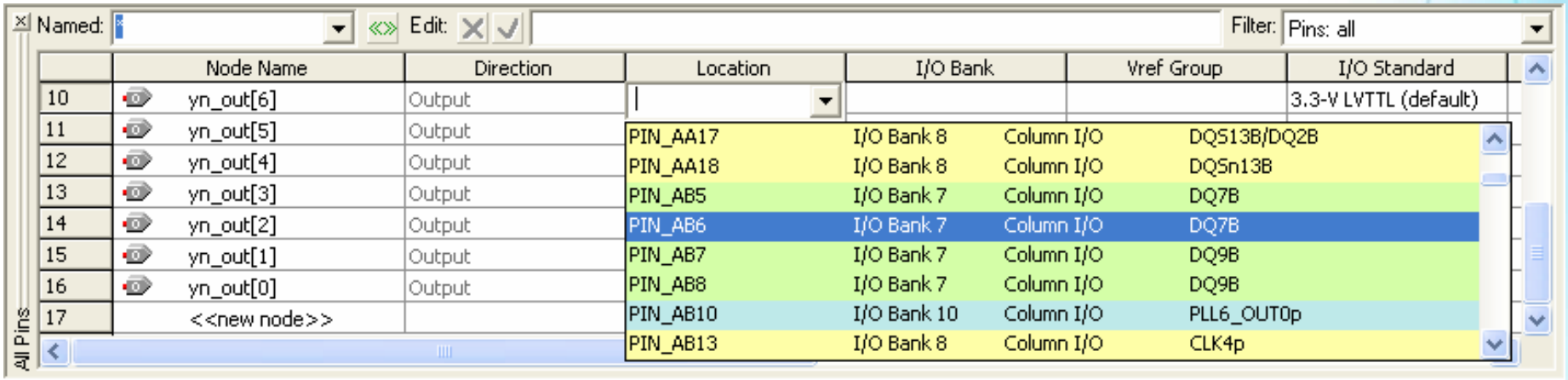


Assigning Pins Using Pin Planner (2)



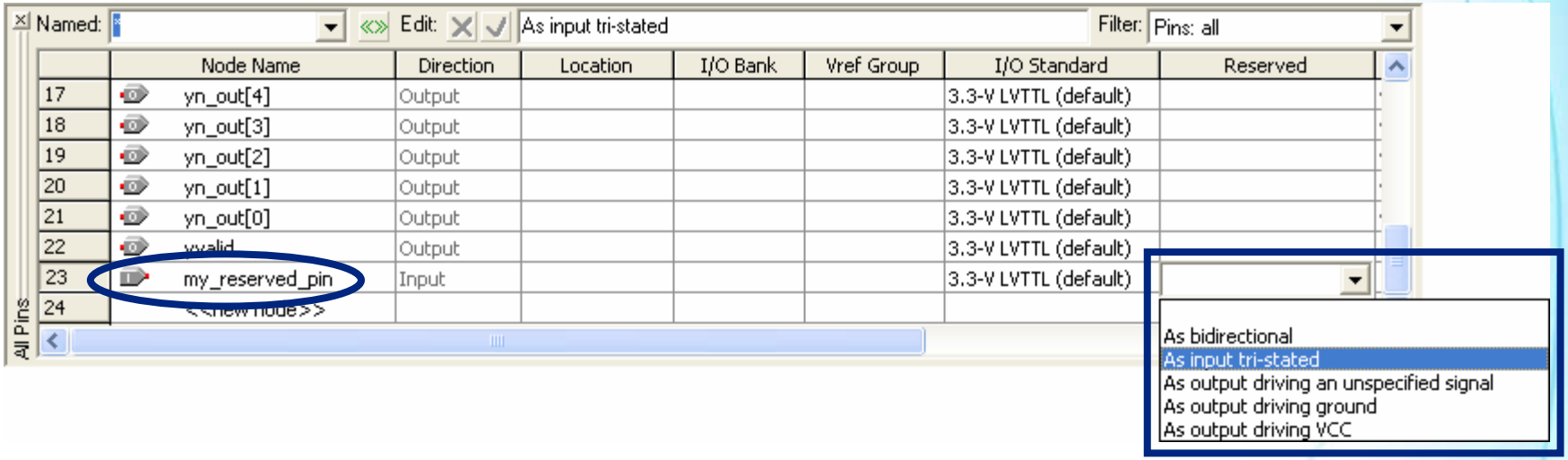
Assigning Pins Using Pin Planner (3)

- Select available locations from list of pins color-coded by I/O bank



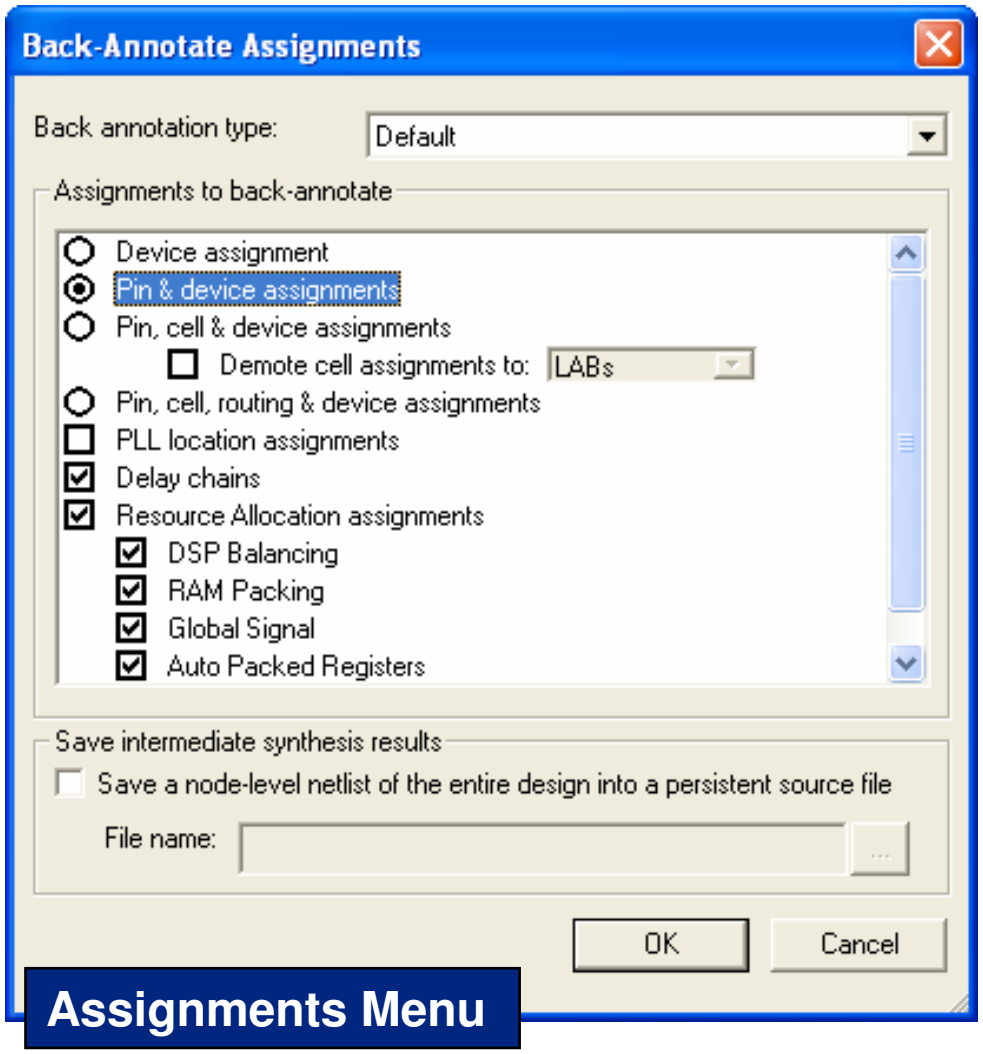
Reserving I/O Pins

- Type reserved I/O name directly into Pins List & select reserve configuration



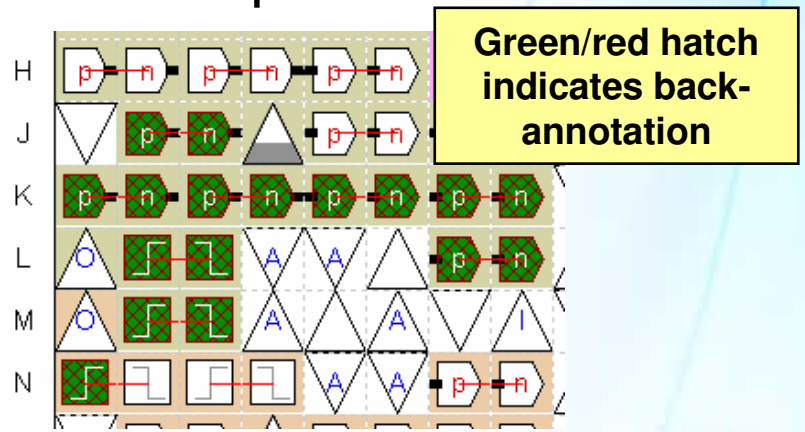
- Or right-click on pin in Package View and choose Reserve ⇒ As…
  - Pin name set to *user_reserve_<pin_number>*

Back-Annotation



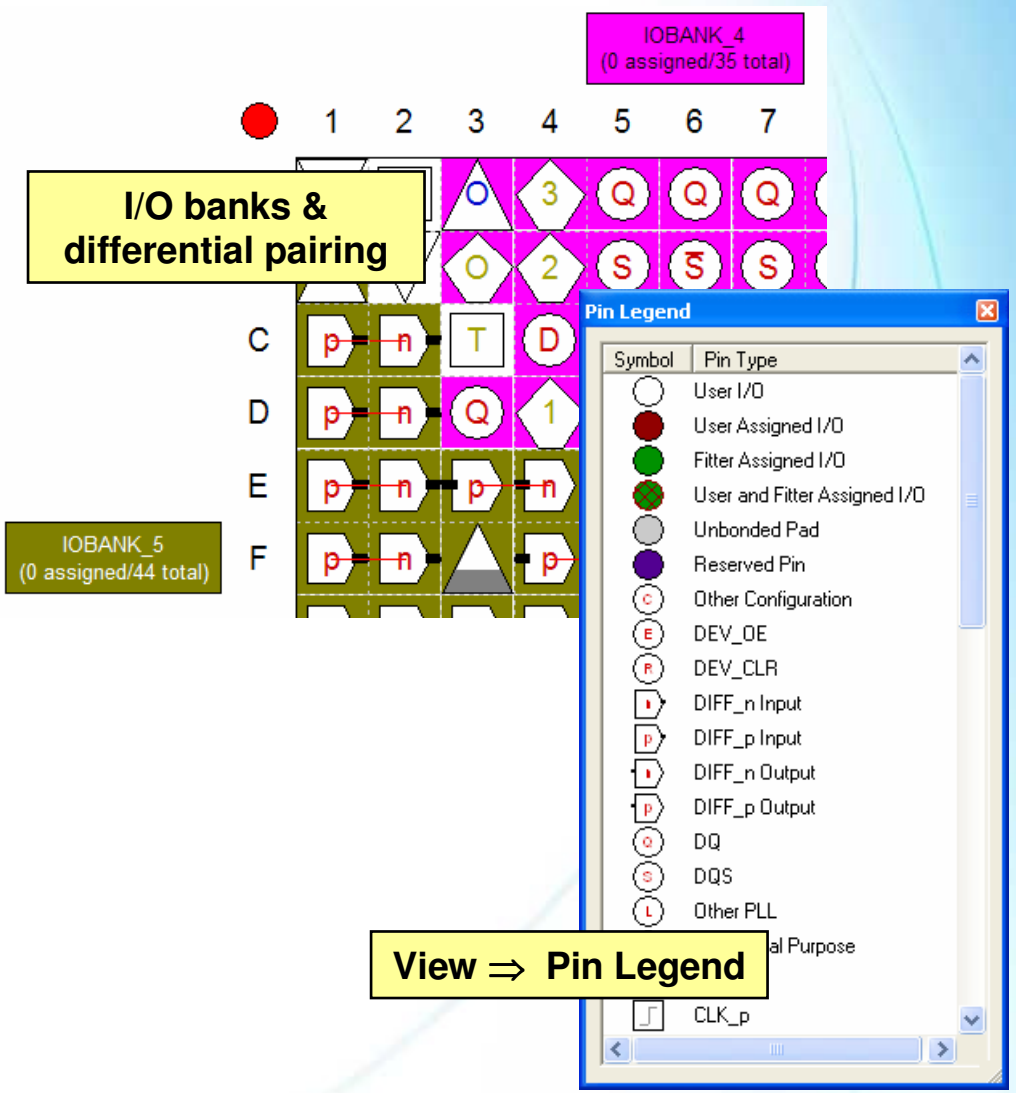
Assignments Menu

- Use to lock fitter-chosen (green) pin assignments for future compilations
  - Copies device & resource locations chosen by fitter into QSF file
    - Pins
    - Logic
    - Routing
- "Locks down" locations in floorplan



Pin Planner Features

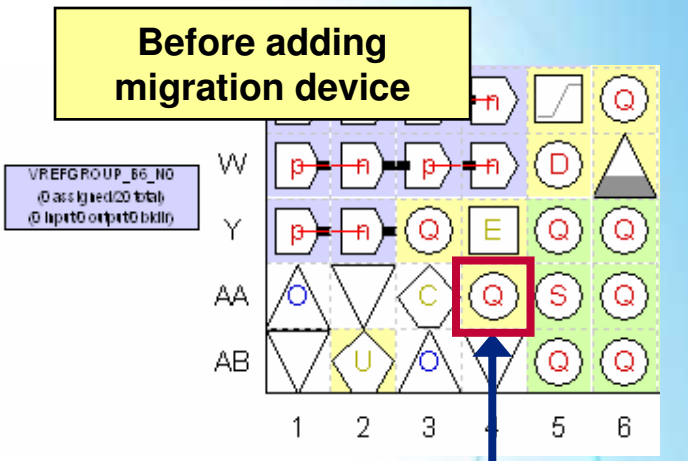
- Displays
  - Device edges
  - I/O banks
  - VREF groups
  - Differential pin pairing
- Easy-to-read pin legend



Pin Migration View

- Select migration devices in Device Settings
- View & compare pin function differences between migration devices
- Package View adjusts to prevent non-migratable assignments

Before adding migration device



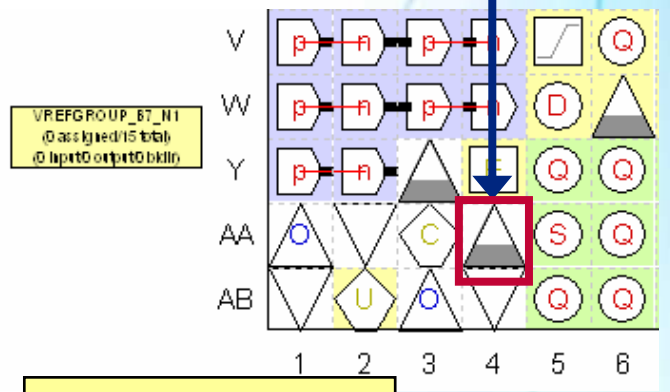
View ⇒ Pin Migration View

Pin Migration View

Current Device: EP2S15F484C3

	Pin Number	Migration Result			Migration Devices					
					EP2S15F484C3			EP2S60F484I4		
		Pin Function	I/O Bank	VREF Group	Pin Function	I/O Bank	VREF Group	Pin Function	I/O Bank	VREF Group
1	PIN_A6	Column I/O	4	B4_N1	Column I/O	4	B4_N1	Column I/O	4	B4_N2
2	PIN_A7	Column I/O	4	B4_N1	Column I/O	4	B4_N1	Column I/O	4	B4_N2
3	PIN_A8	Column I/O	4	B4_N1	Column I/O	4	B4_N1	Column I/O	4	B4_N2
4	PIN_A10	Column I/O	9	B4_N1	Column I/O	9	B4_N1	Column I/O	9	B4_N2
5	PIN_A16	Column I/O	3	B3_N0	Column I/O	3	B3_N0	Column I/O	3	B3_N1
6	PIN_A17	Column I/O	3	B3_N0	Column I/O	3	B3_N0	Column I/O	3	B3_N1
7	PIN_A18	Column I/O	3	B3_N0	Column I/O	3	B3_N0	Column I/O	3	B3_N1
8	PIN_A19	Column I/O	3	B3_N1	Column I/O	3	B3_N1	Column I/O	3	B3_N2
10	PIN_AA4	VREFB7N2	7		Column I/O	7	B7_N1	VREFB7N2	7	B7_N2
12	PIN_AA12	Column I/O	8	B8_N1	Column I/O	8	B8_N1	Column I/O	8	B8_N2

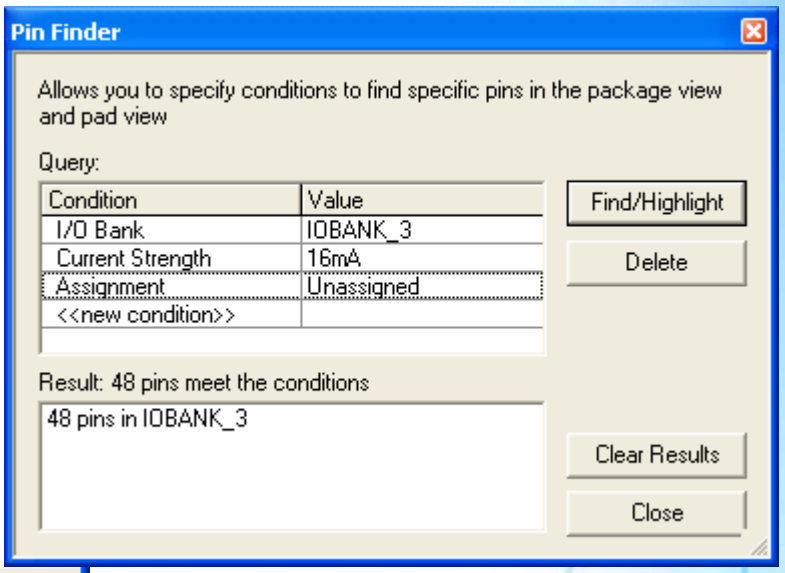
Device... Pin Finder... Show only highlighted pins Show migration differences

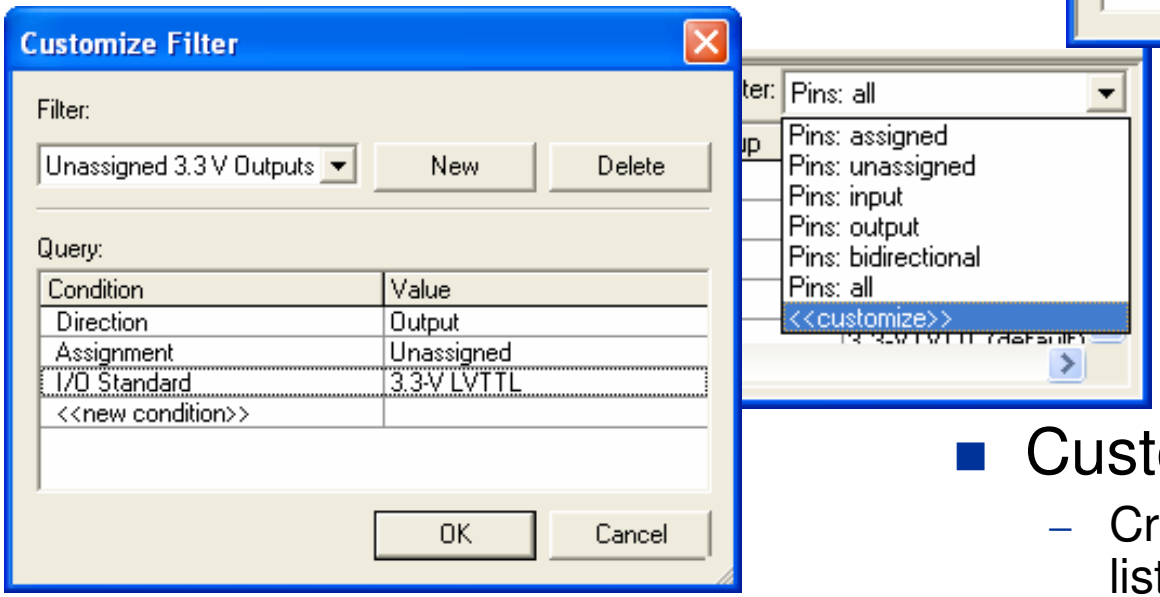


After adding migration device

Additional Pin Planner Features

- Pin Finder
  - Locate pins meeting user-defined criteria with Pin Finder (Edit menu or right-click)
  - Use to find compatible pin locations
  - Found pins highlighted in Package View

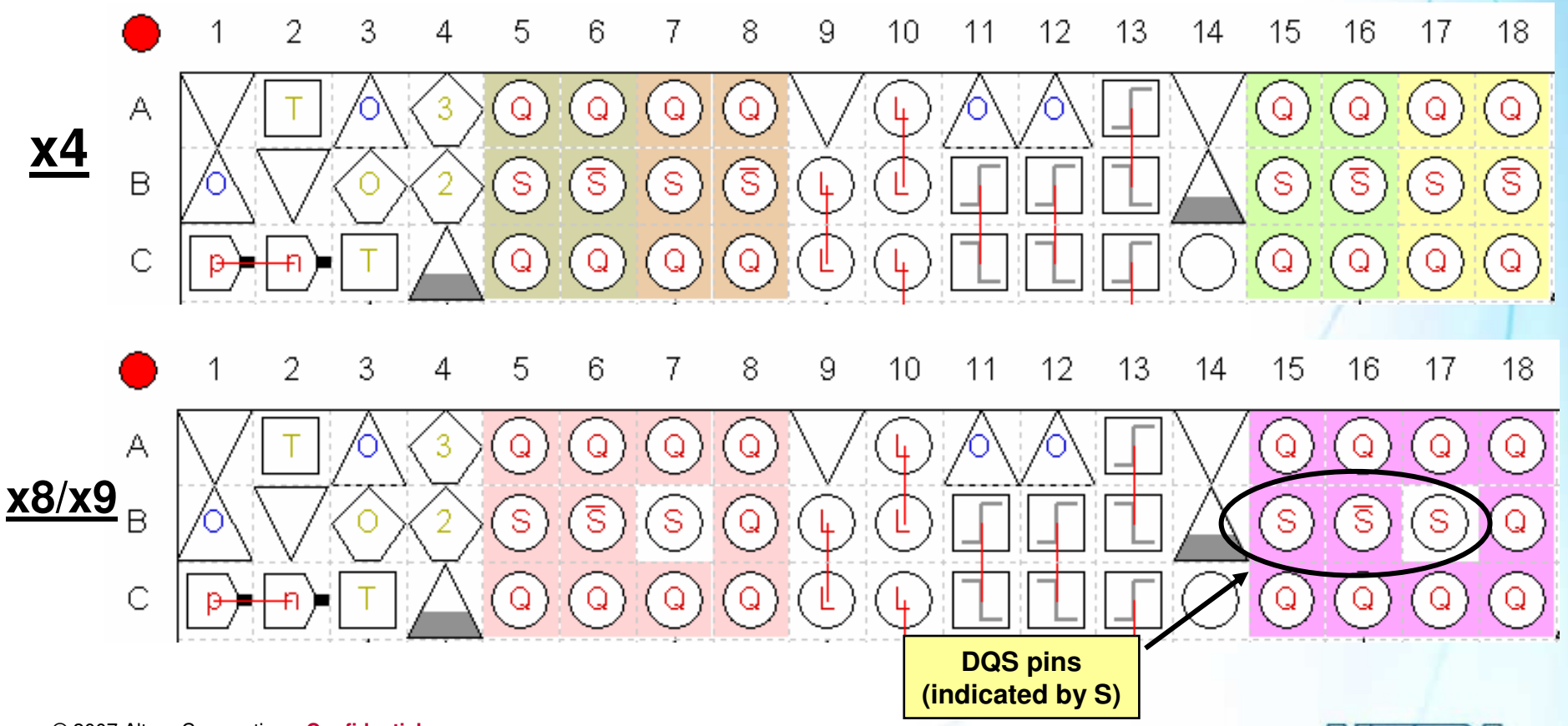




- Custom Filters
  - Create custom filters for All Pins list

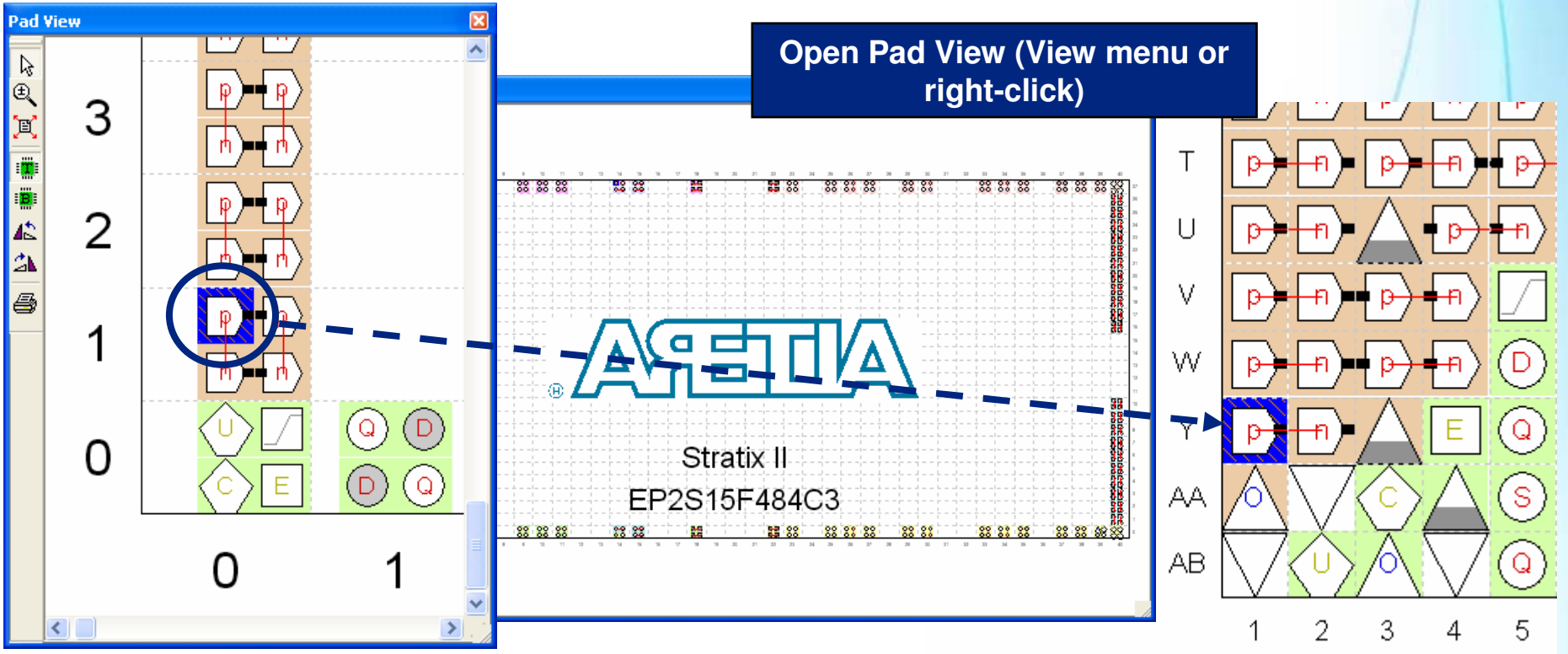
Show DQ/DQS Pins (View Menu)

- Show color-coded DQ/DQS sets in x4, x8/x9, x16/x18, or x32/x36 modes in the Package View



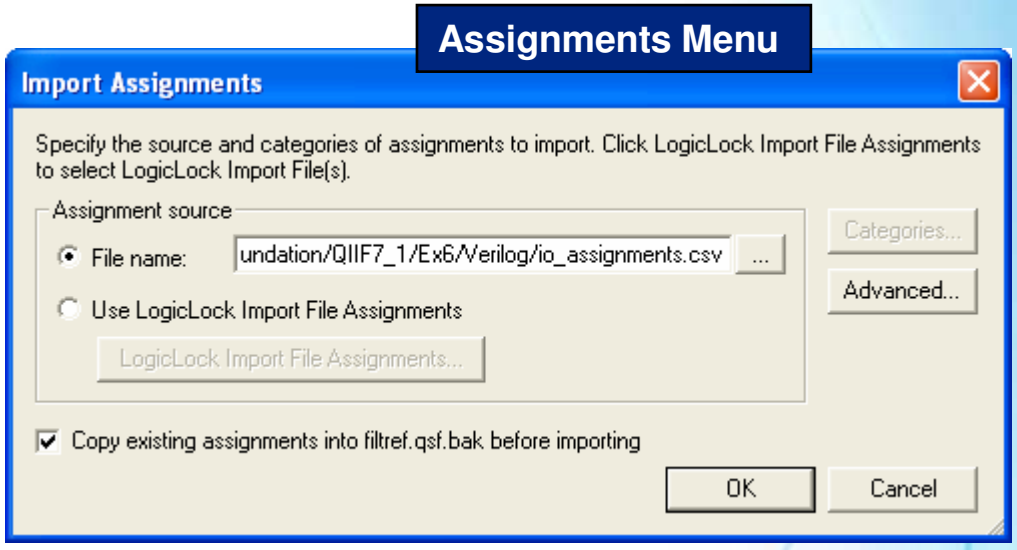
Pad View

- Floating window to cross-reference package pin location to silicon pad location
  - Assign pins in Pad View based on pad location
- Reversed “Altera” indicates flip-chip die



Import/Export via CSV

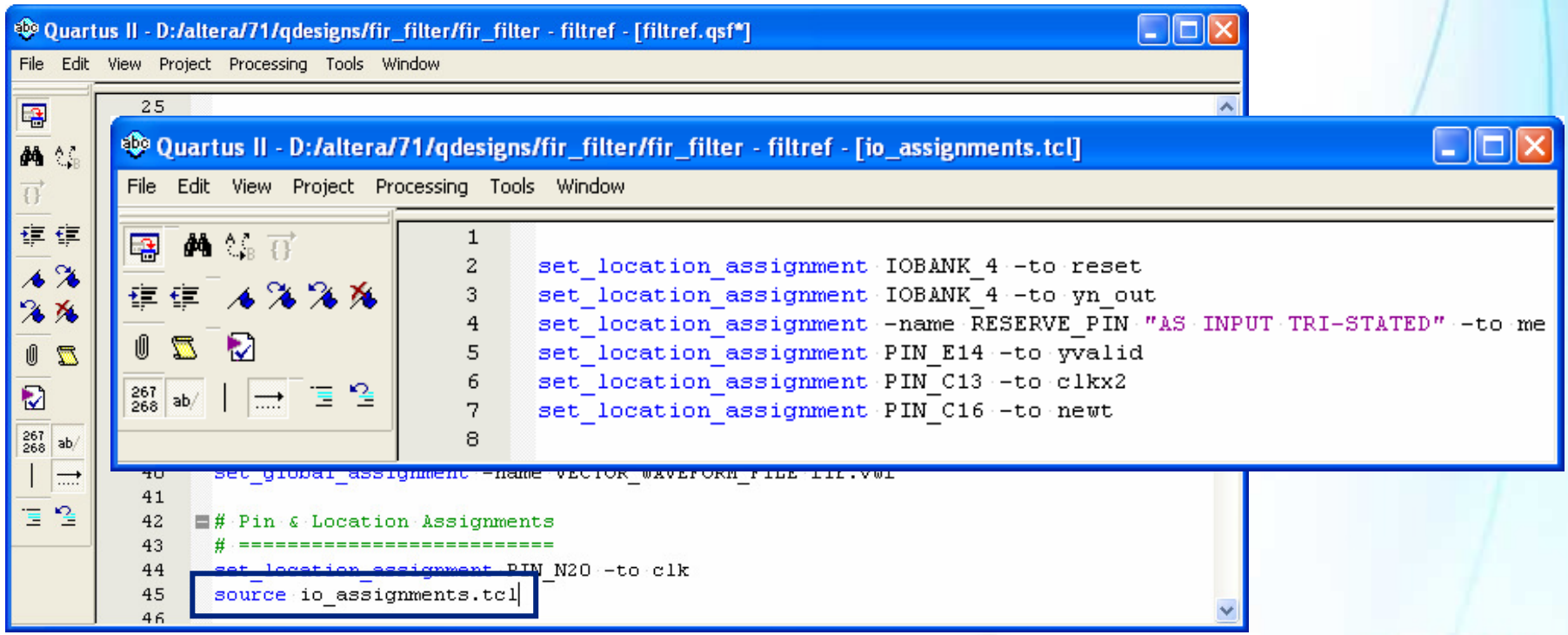
- Use spreadsheet Comma Separated Value (.CSV) file to enter or edit I/O locations
- CSV column names must match Pin Planner column headings
  - To
    - Pin name
  - Assignment Name
    - Location
  - Value
    - PIN_<*pin_number*>
  - I/O standard



	A	B	C
1	To	Assignment Name	Value
2	d[7]	Location	PIN_J4
3	d[6]	Location	PIN_H4

Type I/O Assignments & Scripting

- Type pin-related assignments directly into QSF
- Type pin-related assignments into separate Tcl
  - Source Tcl file in project QSF
  - Execute Tcl file to write assignments into QSF



Using Synthesis Attributes in HDL

- chip_pin
- altera_chip_pin_lc
  - For compatibility with other synthesis tools

VHDL Example

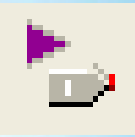
```
entity io_ex_vhdl is
  port (
    data_in : std_logic_vector (3 downto 0);
    data_out : std_logic);
end entity io_ex_vhdl;

attribute chip_pin :  string;
attribute chip_pin of data_in : signal is “D4, D5, D6, D7”;
attribute chip_pin of data_out : signal is “E9”;
```

Verilog Example

```
module io_ex_ver (data_in, data_out);
  input [3:0] data_in /* synthesis chip_pin = “D4, D5, D6, D7” */;
  output data_out /* synthesis chip_pin = “E9”;
```

Note: Other I/O-related features can be specified in HDL. See Quartus II Handbook chapter “Quartus II Integrated Synthesis” for more details.

I/O Assignment Analysis Command



Processing menu ⇒ Start ⇒ Start I/O Assignment Analysis

Run from Pin Planner toolbar

- Use to check legality of all I/O assignments without full compilation
- Requirements
  - I/O declaration
    - HDL port declaration
    - Reserved pin
  - Pin-related assignments
    - I/O standard
    - Current strength
    - Pin location (pin, bank, edge)
    - PCI clamping diode
    - Toggle rate

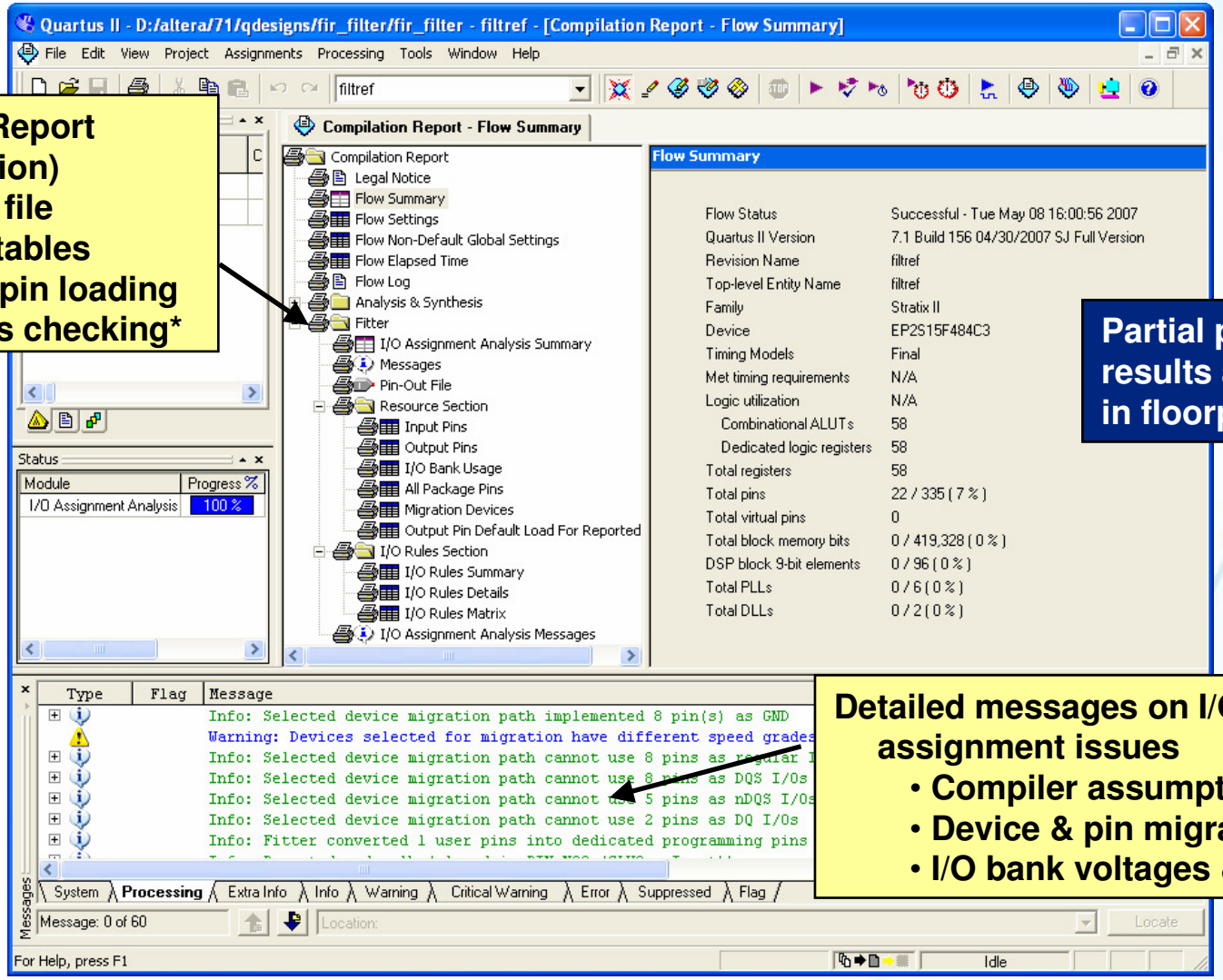
I/O Rules Checked

- No internal logic
  - Checks I/O locations & constraints with respect to other I/O & I/O banks
  - e.g. Each I/O bank supports a single V_{CCIO}
- I/O connected to logic
  - Checks I/O locations & constraints with respect to other I/O, I/O banks & internal resources
  - e.g. A PLL that must be driven by dedicated clock input pin

Note: When working with design files, synthesize design before running I/O Assignment Analysis

I/O Assignment Analysis Output

Compilation Report (Fitter section)
- Pin-out file
- I/O pin tables
- Output pin loading
- I/O rules checking*



Partial placement results also shown in floorplan

Detailed messages on I/O assignment issues
- Compiler assumptions
- Device & pin migration issues
- I/O bank voltages & standards

Note: See Appendix for special reports and information generated only for Stratix II, II GX, and HardCopy II devices

Question:

How do I validate an I/O pin-out for board development?

- Completed design
  - Run full compilation
  - Enable option to run I/O analysis before fitting
    - Checks for I/O layout problems before starting a possible time-consuming fit
    - Settings ⇒ Compilation Process Settings ⇒ Run I/O assignment analysis before compilation
- Incomplete design with completed top-level design file
  - Run I/O Assignment Analysis on design
- Incomplete or no design files
  - Use early I/O planning methodology (discussed next)

Early I/O Planning Methodology

- Process to produce validated pin layout without needing design files
- Steps
  1. Create I/O assignments using Pin Planner only
  2. Implement I/O in HDL
  3. Add all I/O timing constraints
  4. Continue design flow

1) I/O Assignments Using Pin Planner Only

- Purpose: Verify I/O-I/O relationships
- Tasks
  - Enter pin names
  - Enter all pin-related assignments
    - I/O voltage standard
    - Current drive strength
    - PCI I/O support
    - On-chip termination
  - Reserve I/O as inputs or outputs based on direction
    - Reserve bidirectionals as outputs as typically more restrictive
  - Run I/O Assignment Analysis to check

Notes on Step 1

- Place most restrictive pins first
  - E.g. differential pins, double-data rate, etc.
- Assign to I/O blocks or I/O region for more flexibility
  - Let fitter choose exact locations
- As alternative, use spreadsheet or script to enter I/O information

2) Implement I/O in HDL

- Purpose: Verify I/O in relation to core
- Tasks
  - Identify clocks & other global signals
    - Use GLOBAL SIGNAL assignment
  - Add I/O-related megafunctions & IP and make I/O assignments
    - PLLs (ALTPLL)
    - SERDES (ALTLVDS - with or without Dynamic Phase Alignment)
    - High-speed IP (e.g. RapidIO, HyperTransport)
    - DDR/QDR IP Cores
  - Set up and create top-level HDL file with all design I/O
    - Create file manually or use Quartus II commands to configure and create automatically
- Run I/O Assignment Analysis to check

Adding I/O-Related Functions

- Import Pin Planner File (.PPF)
  - Used to import I/O names and settings
  - Created automatically by MegaWizard when creating I/O-related megafunctions
  - Creates new group in Groups List of Pin Planner
- Two methods
  - Import PPF from previously created custom megafunction
  - Create new megafunction directly from Pin Planner
- Set up and create top-level HDL file
  - Based on only I/O megafunctions and I/O assignments

***Note*: See Appendix, Quartus II Handbook "I/O Management" (Volume 2), and online training "I/O Management" for information about importing/creating megafunctions through the Pin Planner and creating a top-level design file.**

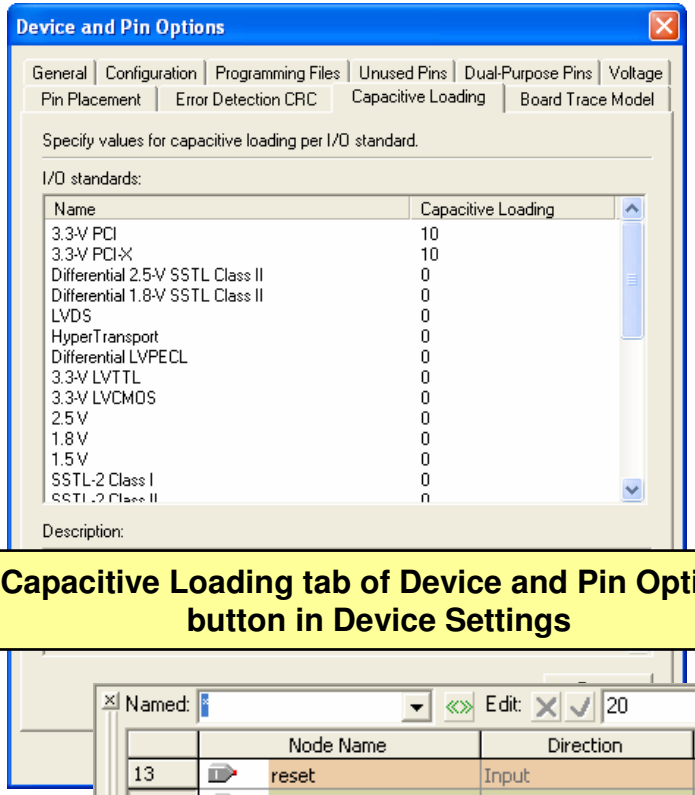
3) Add All I/O Timing Constraints

- Fully constrain I/O for timing analysis
- Discussed in timing analysis section

4) Continue design flow

- Merge validated pin assignments into another design project or revision
  - Use one project or revision to do I/O verification and another to start development of internal logic
- Use generated top-level HDL file as basis for design project
  - Remove virtual pins
  - Connect internal megafunction ports to internal logic

"Board-Aware" Settings: Output Pin Load



Capacitive Loading tab of Device and Pin Options button in Device Settings

- Specifies output pin loading in picofarads (pf)
  - Changes default loading value of I/O standard
  - Changes t_{co} of output pins
- Allows designer to accurately model board conditions
- Specify for entire I/O standard in Device Settings
- Apply to individual output or bidirectional pins in Assignment Editor or Pin Planner All Pins list

Named: Edit: 20 Filter: Pins: all

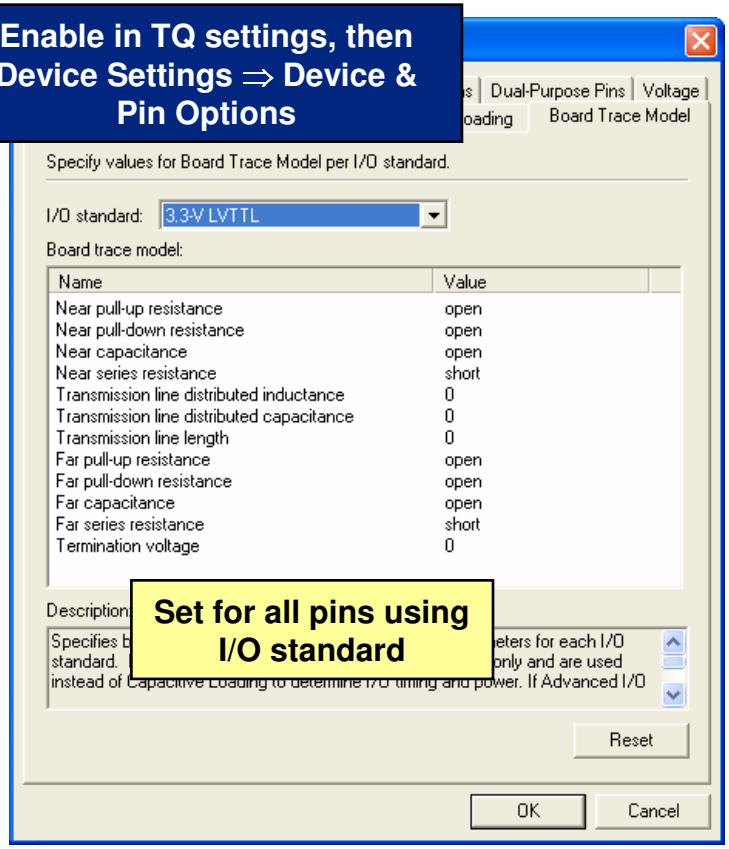
	Node Name	Direction	Location	I/O Standard	Output Pin Load	I/O Bank	
13	reset	Input	PIN_N3	3.3-V LVTTL (default)		6	B
14	yn_out[7]	Output	PIN_J6	3.3-V LVTTL (default)	20	5	B
15	yn_out[6]	Output	PIN_L8	3.3-V LVTTL (default)	20	5	B
16	yn_out[5]	Output	PIN_H1	3.3-V LVTTL (default)	20	5	B
17	yn_out[4]	Output	PIN_K2	3.3-V LVTTL (default)	20	5	B
18	yn_out[3]	Output	PIN_H2	3.3-V LVTTL (default)	20	5	B
19	yn_out[2]	Output	PIN_J5	3.3-V LVTTL (default)	20	5	B
20	yn_out[1]	Output	PIN_L2	3.3-V LVTTL (default)	20	5	B
21	yn_out[0]	Output	PIN_K5	3.3-V LVTTL (default)	20	5	B
22	yvalid	Output	PIN_L7	3.3-V LVTTL (default)		5	B
23	~DATA0~	Input	PIN_E13	3.3-V LVTTL (default)		3	B

Tcl: set_instance_assignment –name OUTPUT_PIN_LOAD <value> –to <pin name>

Advanced I/O Timing (Stratix II & III devices only)

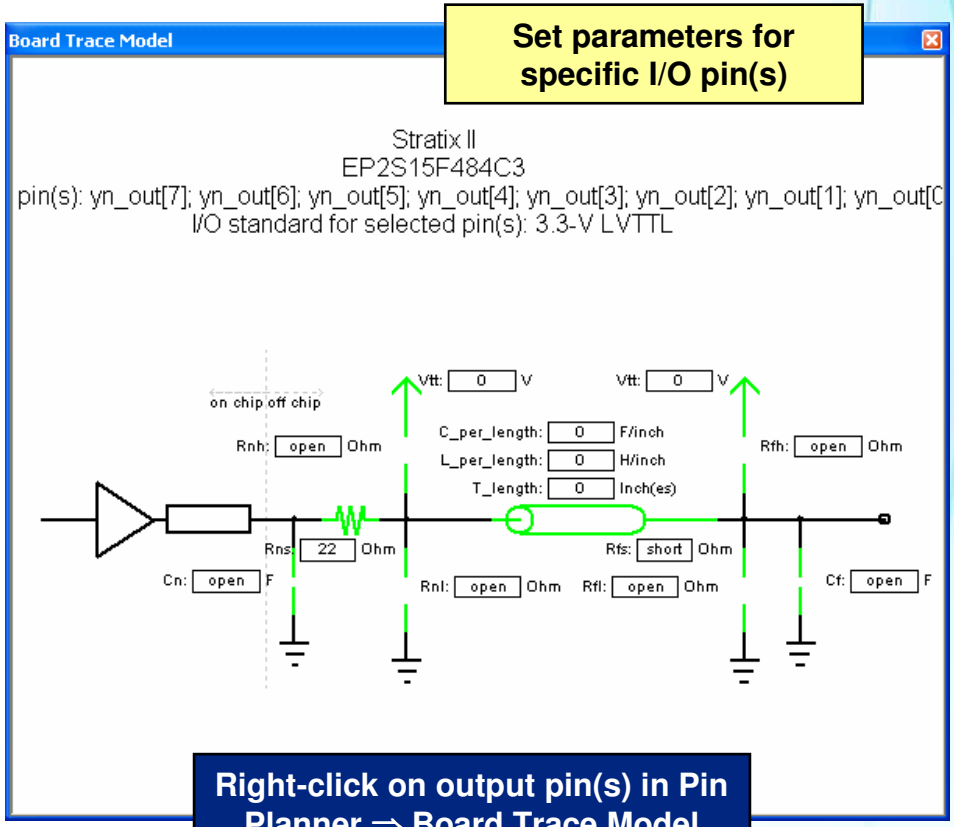
- Enhances analysis (over capacitive loading) by allowing user to enter board-level parameters
  - Use in lieu of or in addition to HSPICE & IBIS modeling
- View signal integrity metrics in Compilation Report (TimeQuest folder)

Enable in TQ settings, then Device Settings ⇒ Device & Pin Options



Set for all pins using I/O standard

Set parameters for specific I/O pin(s)



Right-click on output pin(s) in Pin Planner ⇒ Board Trace Model

PCB Tool Support

- Mentor Graphics DxDesigner & I/O Designer software
- Cadence Allegro Design Entry software
- OrCAD Capture

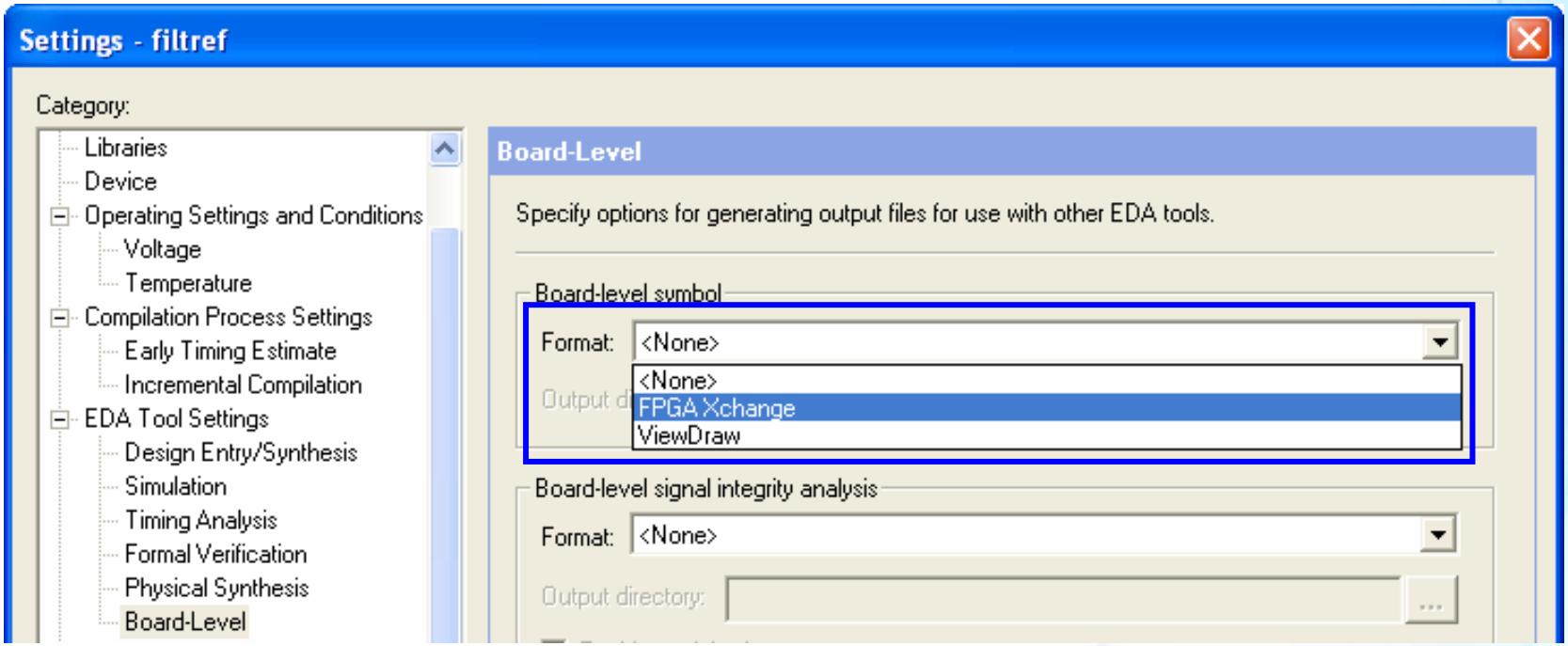
For more detail on interfacing, see Quartus II Handbook chapters *"Mentor Graphics PCB Design Tools Support"* & *"Cadence PCB Design Tools Support"* (Volume 2)

Transferring I/O Assignments

- FPGA Xchange file (.FX)
  - Use to transfer pin assignments between Quartus II software & I/O Designer software
  - Contains only assigned pins
  - Not created automatically; must manually turn on .fx file generation (EDA Netlist Writer)
- Pin-Out file (.PIN)
  - Uses
    - Transferring pin assignments from Quartus II software to PCB tools
      - Cannot be imported back into Quartus II software
    - Generating symbols in PCB tools
  - Contains all pins, included unused
  - Automatically generated during fitting or I/O assignment analysis

Generating .FX File

- Choose FPGA Xchange (Assignments ⇒ EDA Tools Settings ⇒ Board-Level)
- Compile or run EDA Netlist Writer (Processing ⇒ Start)



.PIN File

Pin-Out File

```
48   -- GND+          : Unused input pin. It can also be used to report unused dual-purpose pins.
49   --                 This pin should be connected to GND. It may also be connected  to a
50   --                 valid signal  on the board  (low, high, or toggling)  if that signal
51   --                 is required for a different revision of the design.
52   -- GND*           : Unused  I/O  pin.   This pin can either be left unconnected or
53   --                 connected to GND.  Connecting this pin to GND will improve the
54   --                 device's immunity to noise.
55   -- RESERVED       : Unused I/O pin, which MUST be left unconnected.
56   -- RESERVED_INPUT    : Pin is tri-stated and should be connected to the board.
57   -- RESERVED_INPUT_WITH_WEAK_PULLUP    : Pin is tri-stated with internal weak pull-up resistor.
58   -- RESERVED_INPUT_WITH_BUS_HOLD       : Pin is tri-stated with bus-hold circuitry.
59   -- RESERVED_OUTPUT_DRIVEN_HIGH        : Pin is output driven high.
60   ----------------------------------------------------------------------------
61
62   Quartus II Version 7.1 Build 156 04/30/2007 SJ Full Version
63   CHIP  "filtref"  ASSIGNED TO AN: EP2S15F484C3
64
65   Device Migration List: "EP2S60F484I4"
66   Pin Name/Usage               : Location  : Dir.    : I/O Standard      : Voltage : I/O Bank  : User Assignment
67   ------------------------------------------------------------------------------------------------------------
68   GND                          : A1        : gnd     :                   :         :           :
69   TEMPDIODEp                   : A2        :         :                   :         :           :
70   VCCIO4                       : A3        : power   :                   : 3.3V    : 4         :
71   MSEL3                        : A4        :         :                   :         : 4         :
72   GND*                         : A5        :         :                   :         : 4         :
73   GND*                         : A6        :         :                   :         : 4         :
74   GND*                         : A7        :         :                   :         : 4         :
75   GND*                         : A8        :         :                   :         : 4         :
76   GND                          : A9        : gnd     :                   :         :           :
77   GND*                         : A10       :         :                   :         : 9         :
78   VCCIO4                       : A11       : power   :                   : 3.3V    : 4         :
79   VCCIO3                       : A12       : power   :                   : 3.3V    : 3         :
80   GND*                         : A13       :         :                   :         : 3         :
```

Description section

I/O names & settings

Exercise 5 Demonstration

I/O Planning Summary

- Pin assignments can be performed in many ways, graphically & by means of text files
- The Pin Planner provides an easy-to-use graphical way or creating and managing pin assignments
- I/O Assignment Analysis helps validate a device-pin out without performing full compilations
- Pin validation can be completed during any point in design development

I/O Planning Support Resources

- Quartus II Handbook chapters
  - *"I/O Management"* (Volume 2)
  - *"Signal Integrity Analysis with Third-Party Tools"* (Volume 3)
  - *"Mentor Graphics PCB Design Tools Support"* (Volume 2)
  - *"Cadence PCB Design Tools Support"* (Volume 2)
- Training Courses and Demonstrations
  - Online tutorial: *"FPGA to Board Design Flow Using Mentor Graphics Tools"*

Quartus II Software Design Series: Foundation

Timing Analysis

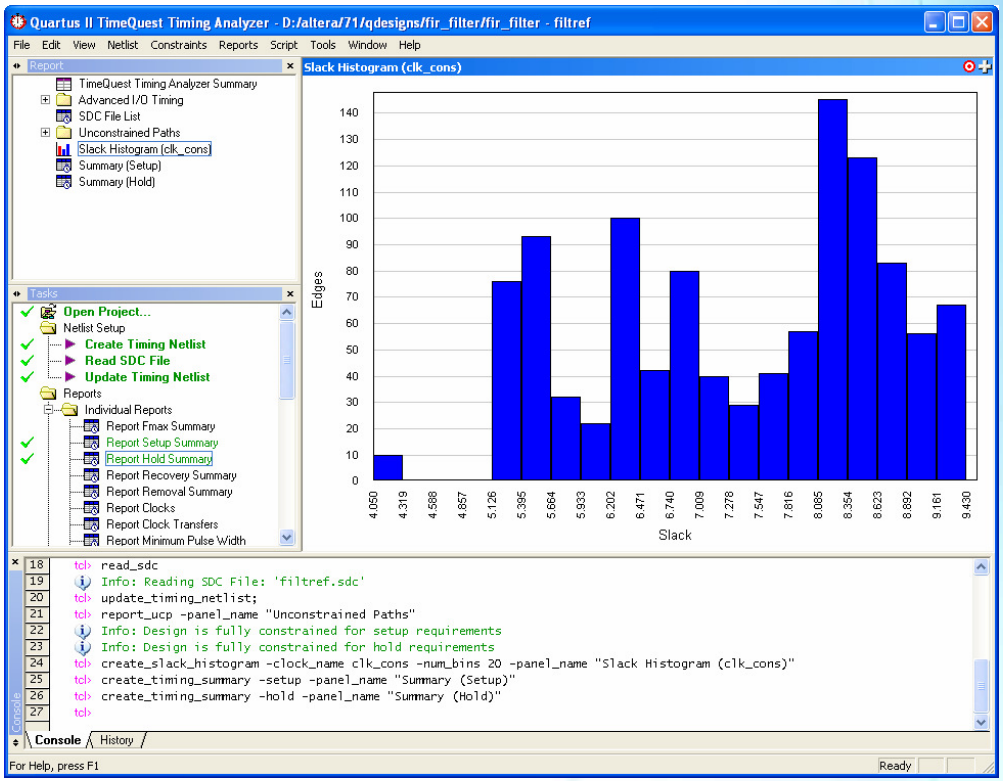
Timing Analysis Agenda

- TimeQuest GUI
- Using TimeQuest
- Using TimeQuest in the Quartus II flow

Note: For more details on verifying designs for timing, please attend the course "Quartus II Software Design Series: Verification"

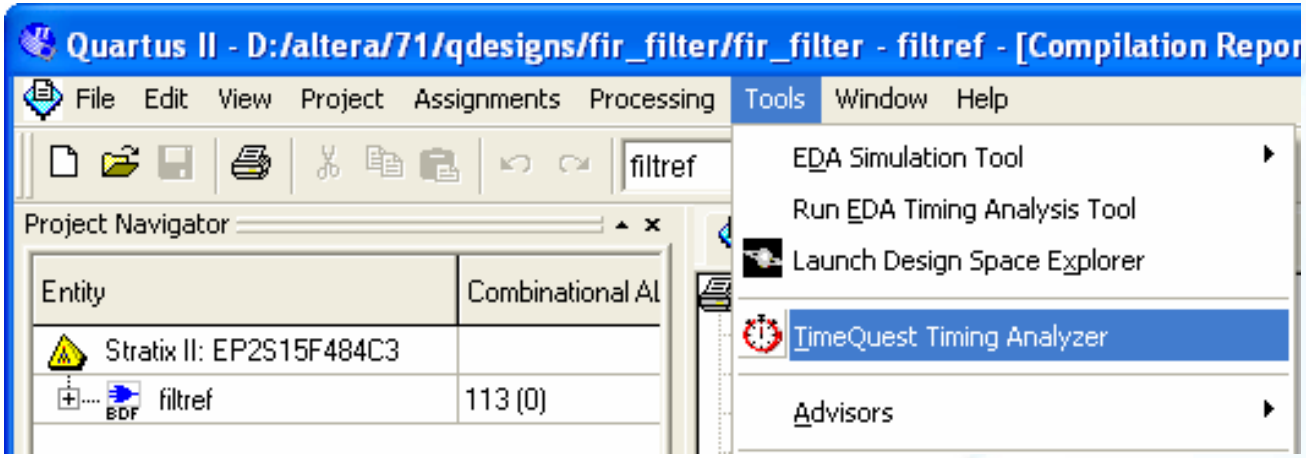
TimeQuest Timing Analyzer

- New timing engine in Quartus II software
- Provides timing analysis solution for all levels of experience
- Features
  - Synopsys Design Constraints (SDC) support
    - Standardized constraint methodology
  - Easy-to-use interface
    - Constraint entry
    - Standard reporting
  - Scripting emphasis
    - Presentation focuses on using GUI

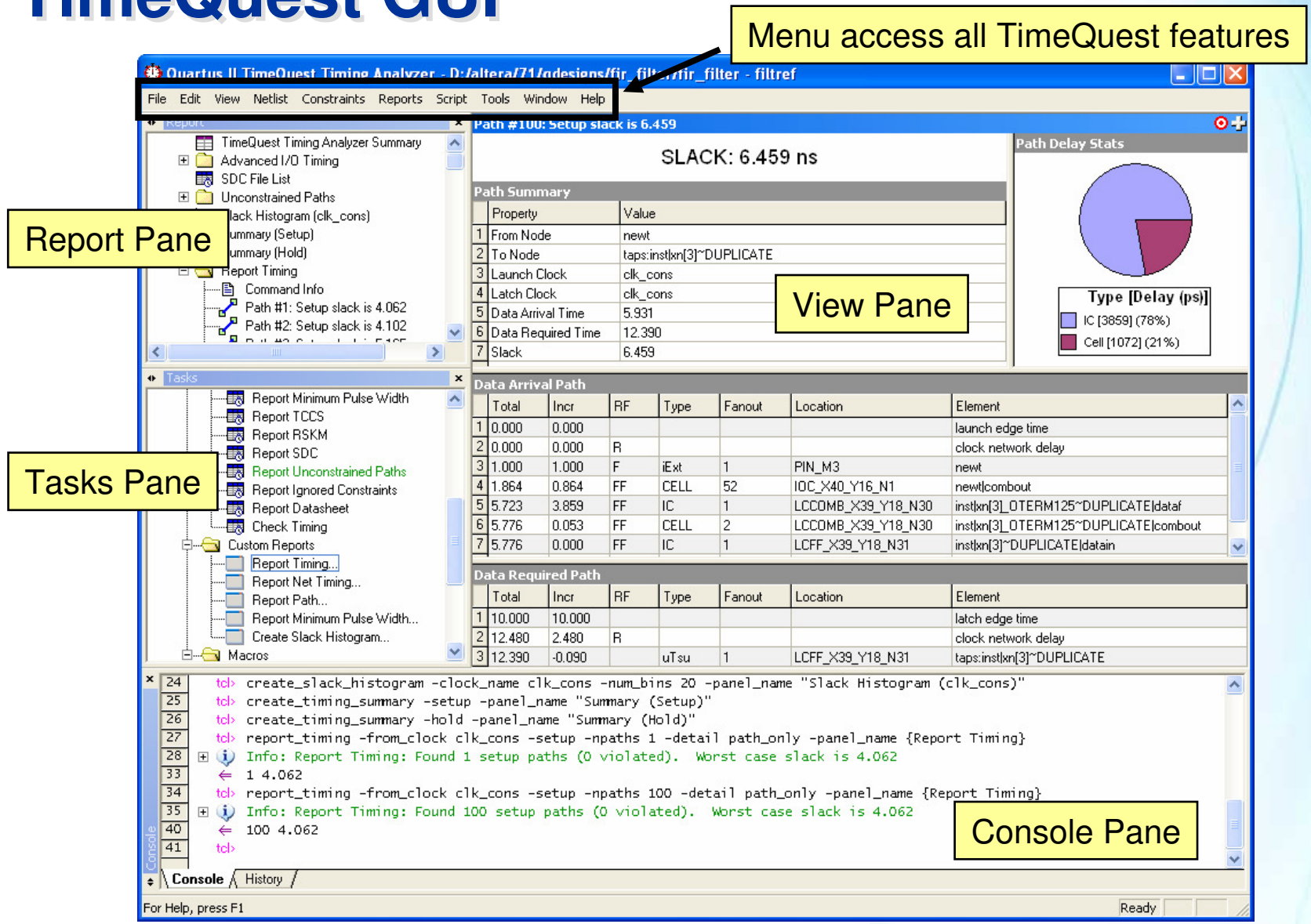


Opening TimeQuest

- Toolbar button 
- Tools menu
- Stand-alone mode
  - `quartus_staw`
- Command line

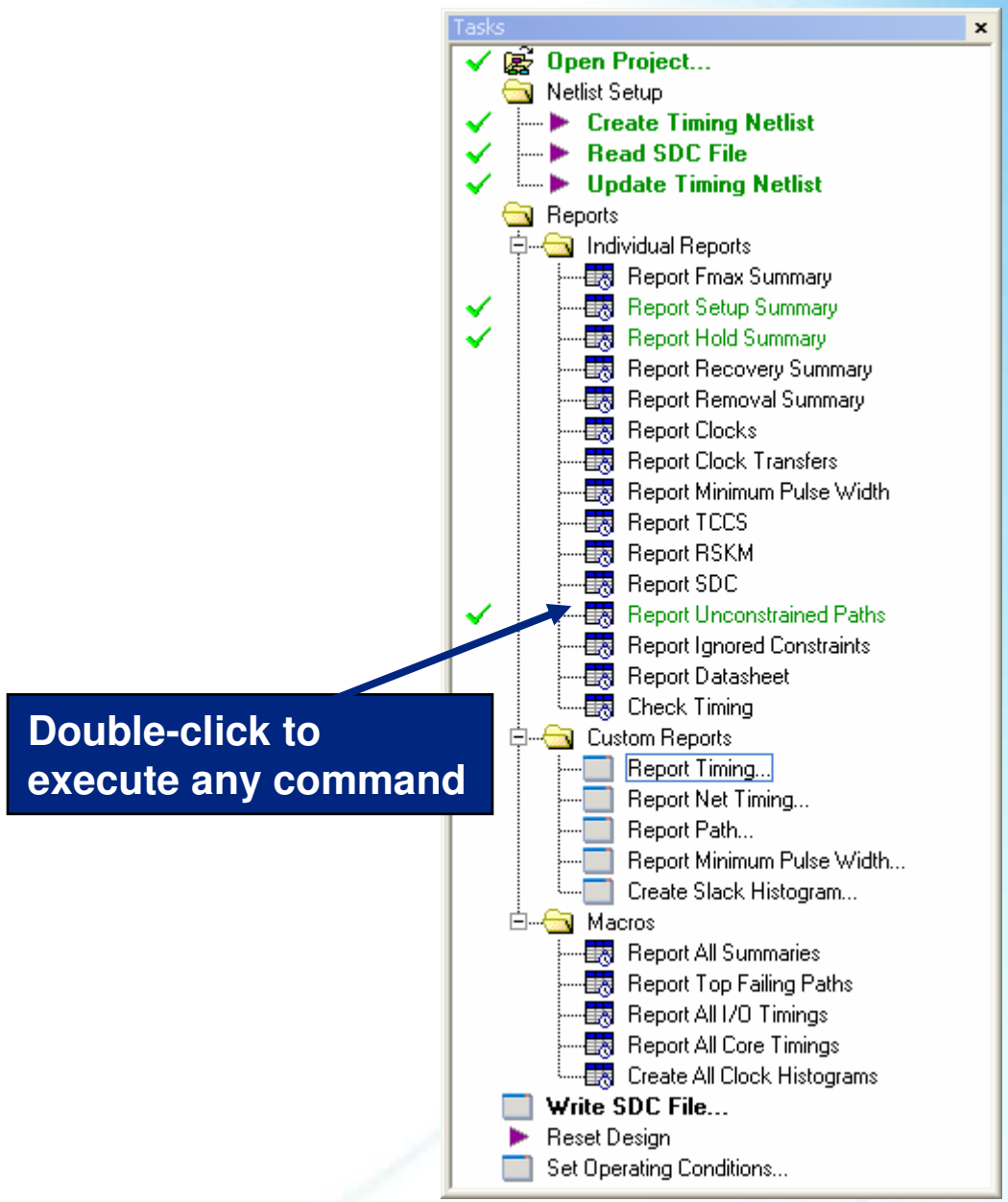


TimeQuest GUI

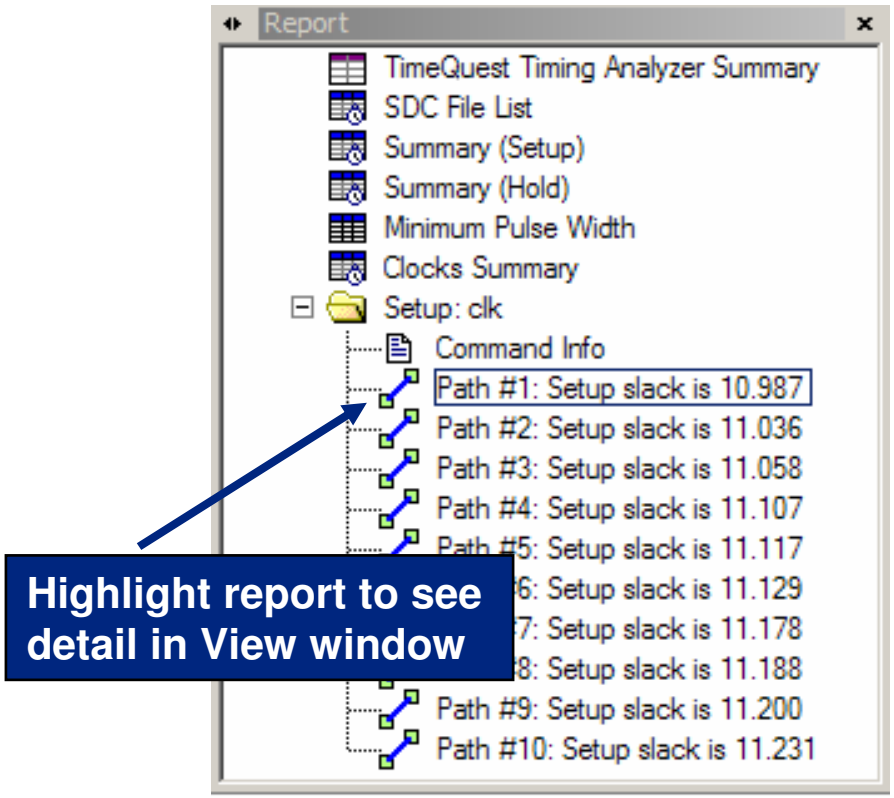


Tasks Pane

- Provides quick access to common operations
  - Command execution
  - Report generation
- Executes most commands with default settings



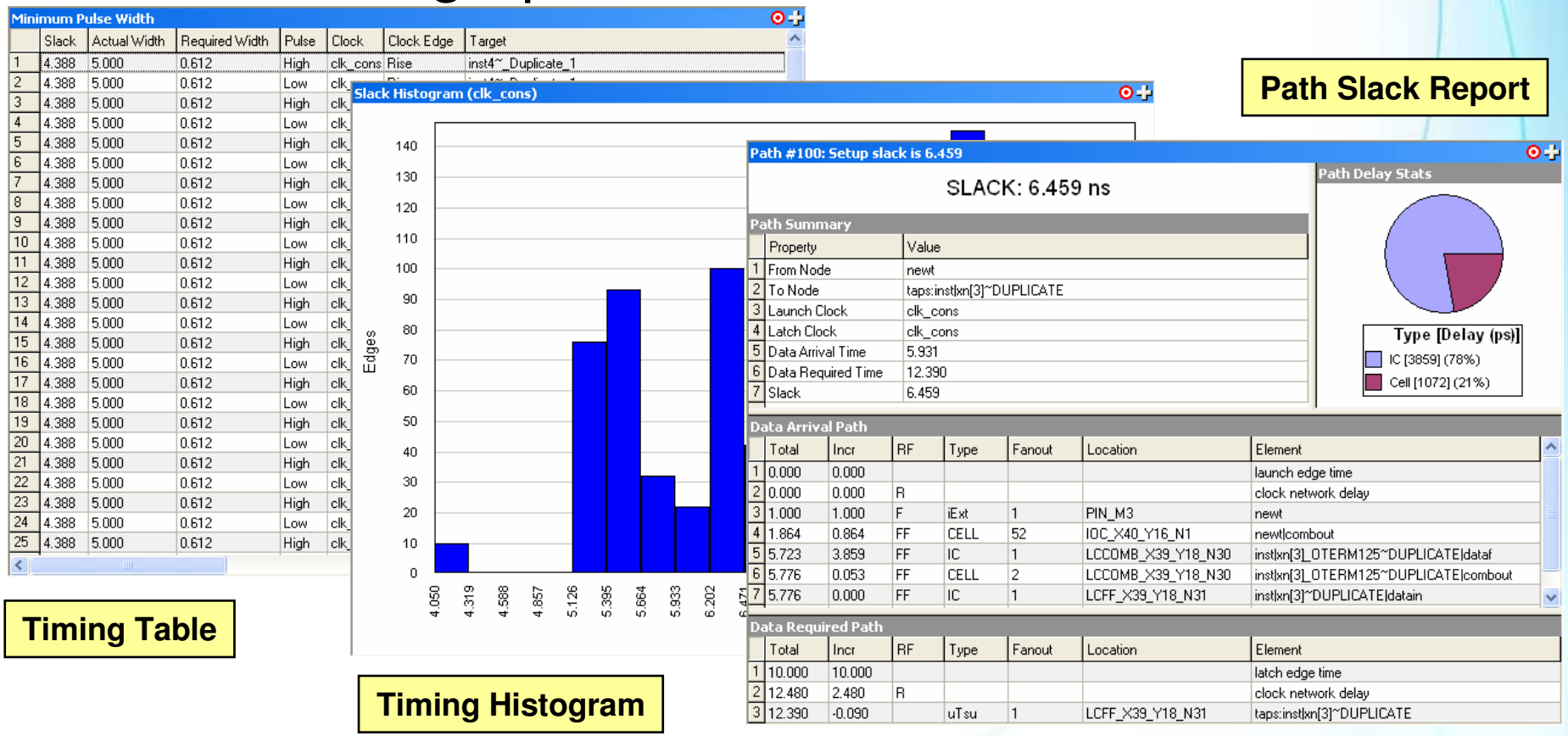
Report Pane



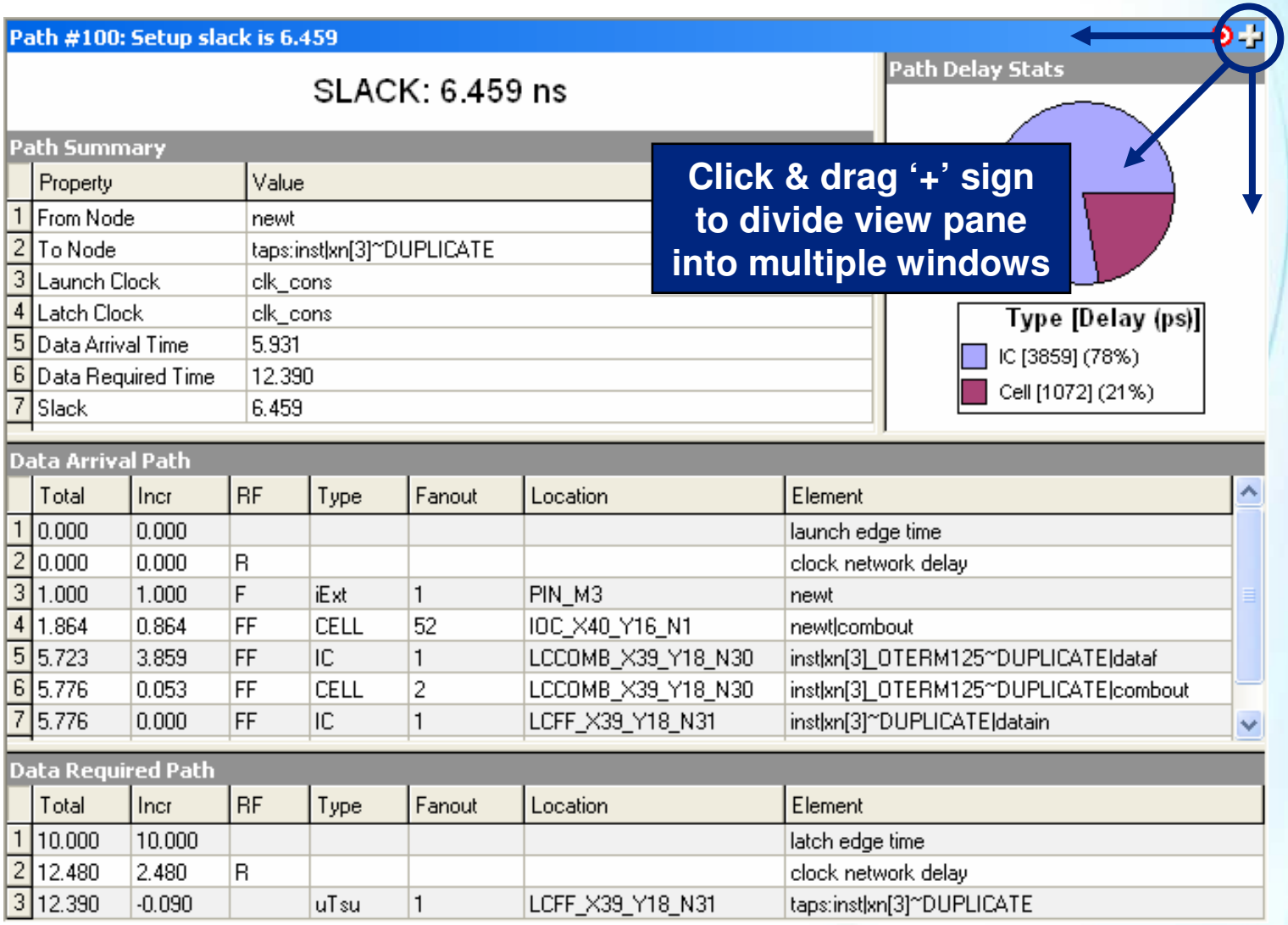
- Displays list of reports currently available for viewing
  - Reports generated by Tasks pane
  - Reports generated using report commands

View Pane

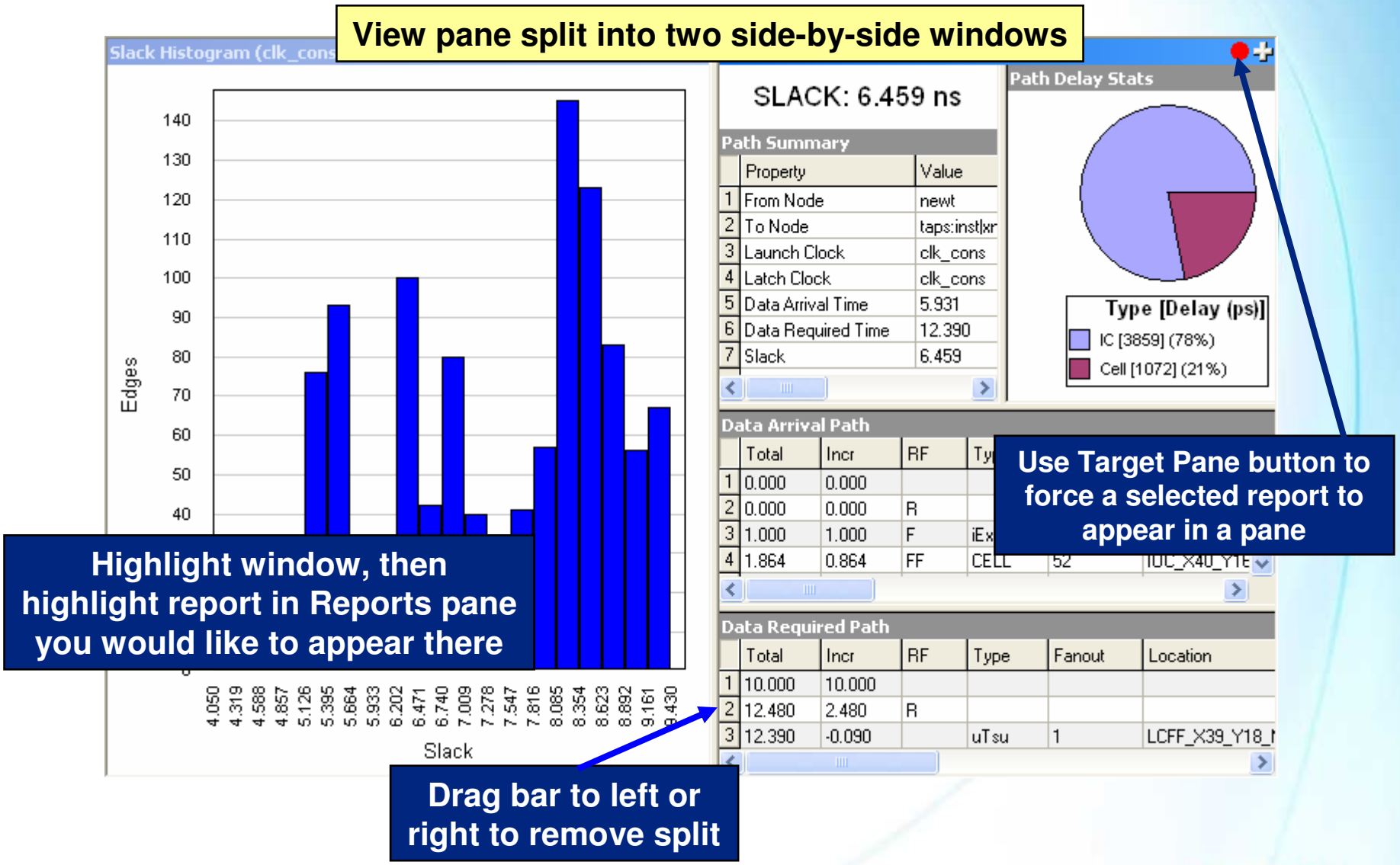
- Main viewing area that displays report table contents & graphical results



Viewing Multiple Reports

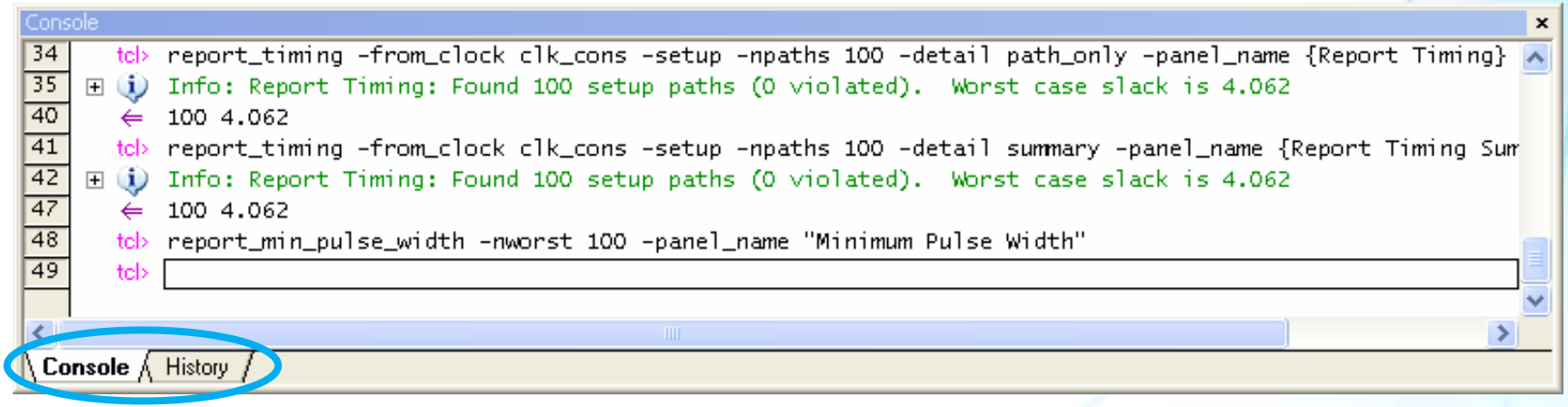


Viewing Multiple Reports Example



Console pane

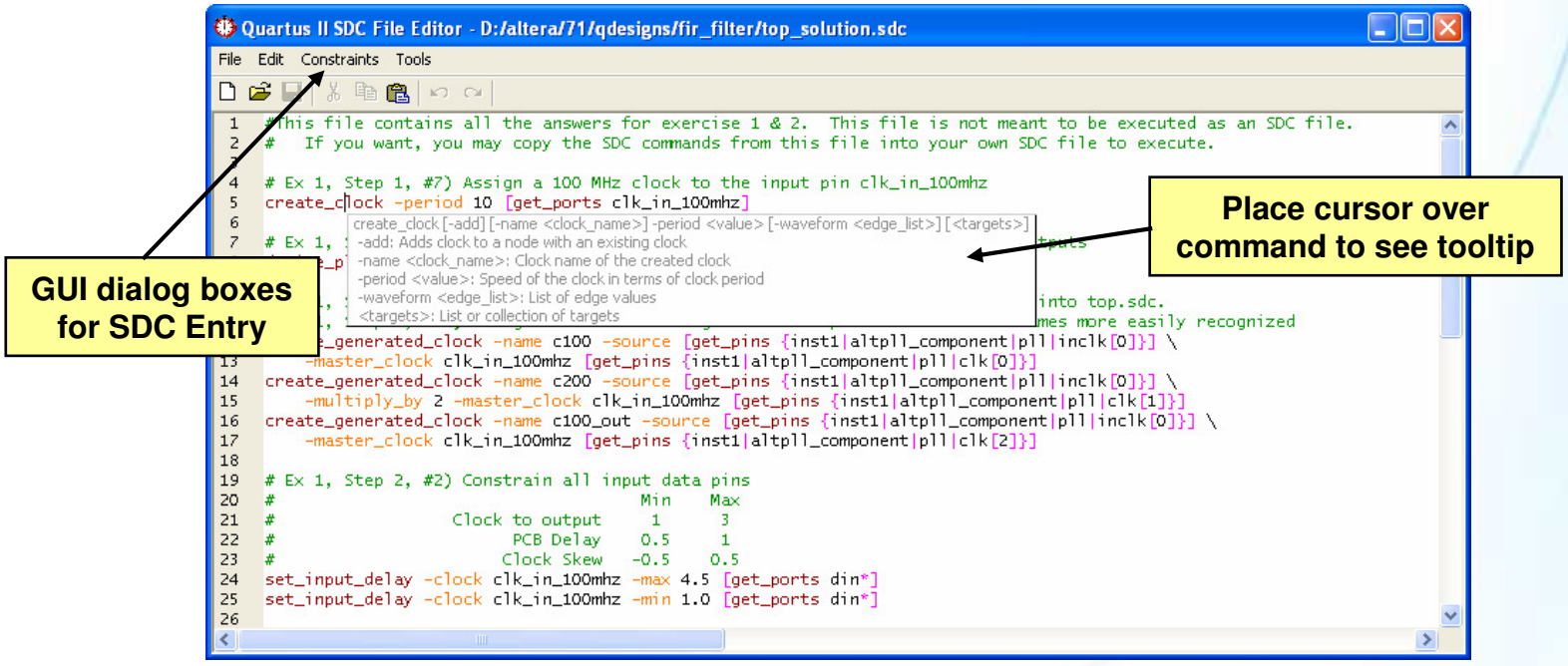
- Allows direct entry and execution of SDC & Tcl commands
  - Displays equivalent of command executed by GUI
- Displays TimeQuest output messages
- History tab records all executed SDC & Tcl commands



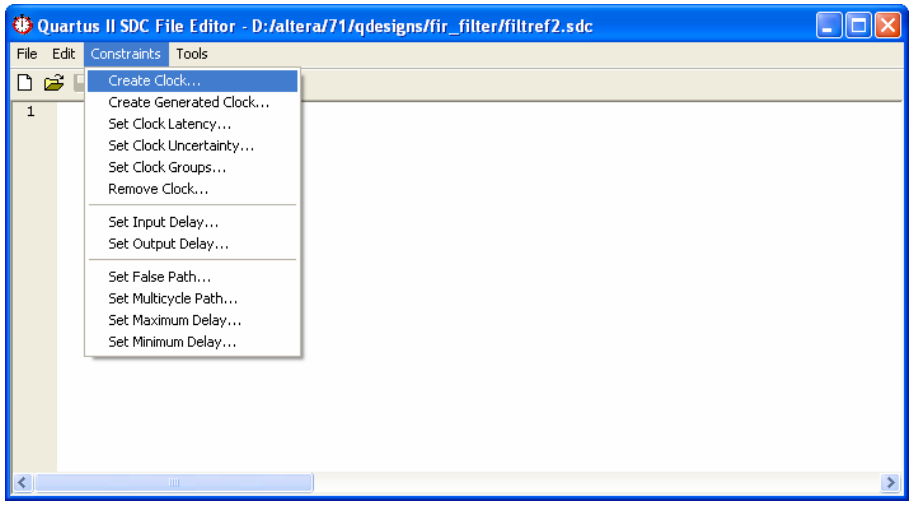
SDC File Editor

File menu ⇒ New/Open SDC File

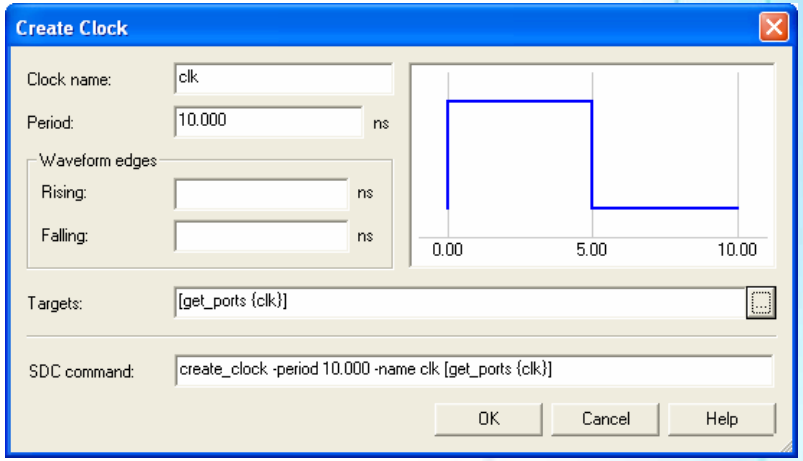
- Features
  - Access to GUI dialog boxes for constraint entry
  - Syntax coloring
  - Tooltip syntax help

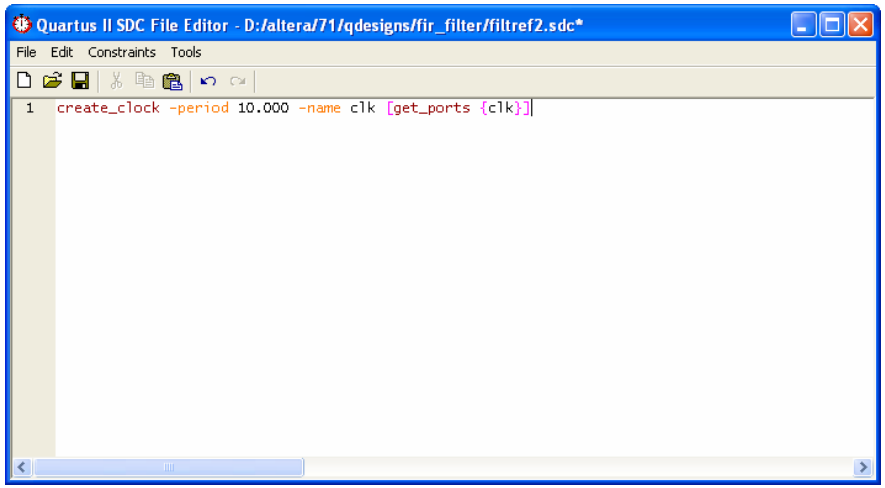


SDC File Editor (cont.)



Construct an SDC file using TimeQuest graphical file tools





Steps to Using TimeQuest

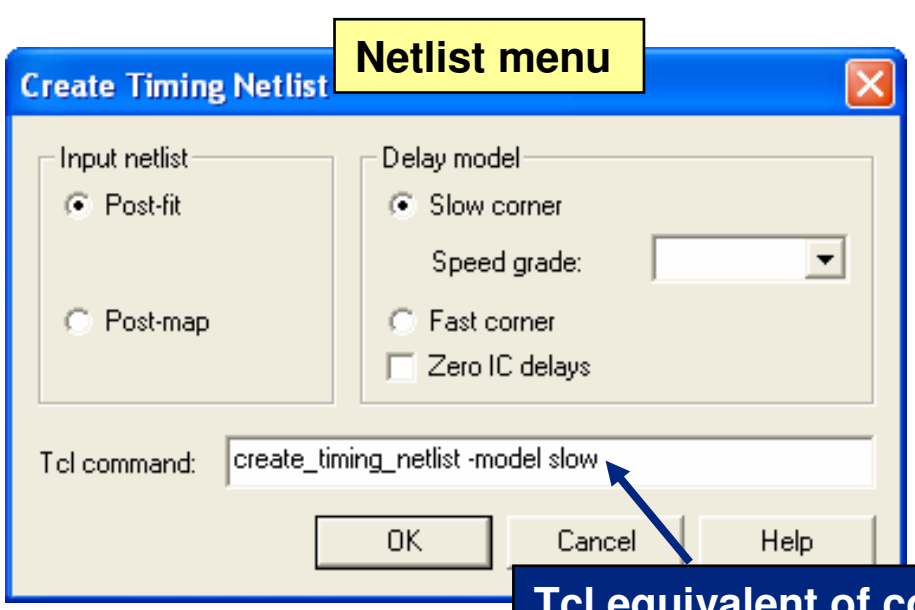
1. Generate timing netlist
2. Create and read SDC file (optional)

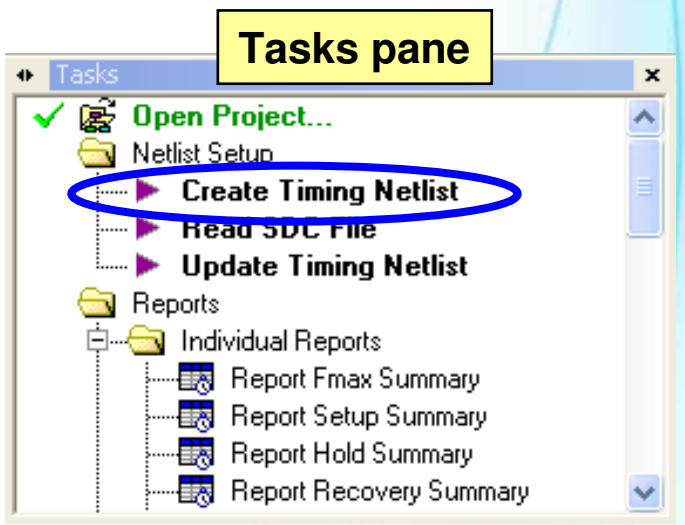
or

3. Constrain design (optional)
4. Update timing netlist
5. Generate timing reports
6. Save timing constraints (optional)

1) Generate Timing Netlist

- Creates timing netlist (i.e. database) based on compilation results
  - Post-synthesis (mapping) or post-fit
  - Worst-case (slow), best-case (fast) timing models, or set operating conditions (Stratix III and Cyclone III devices only)
- To execute:



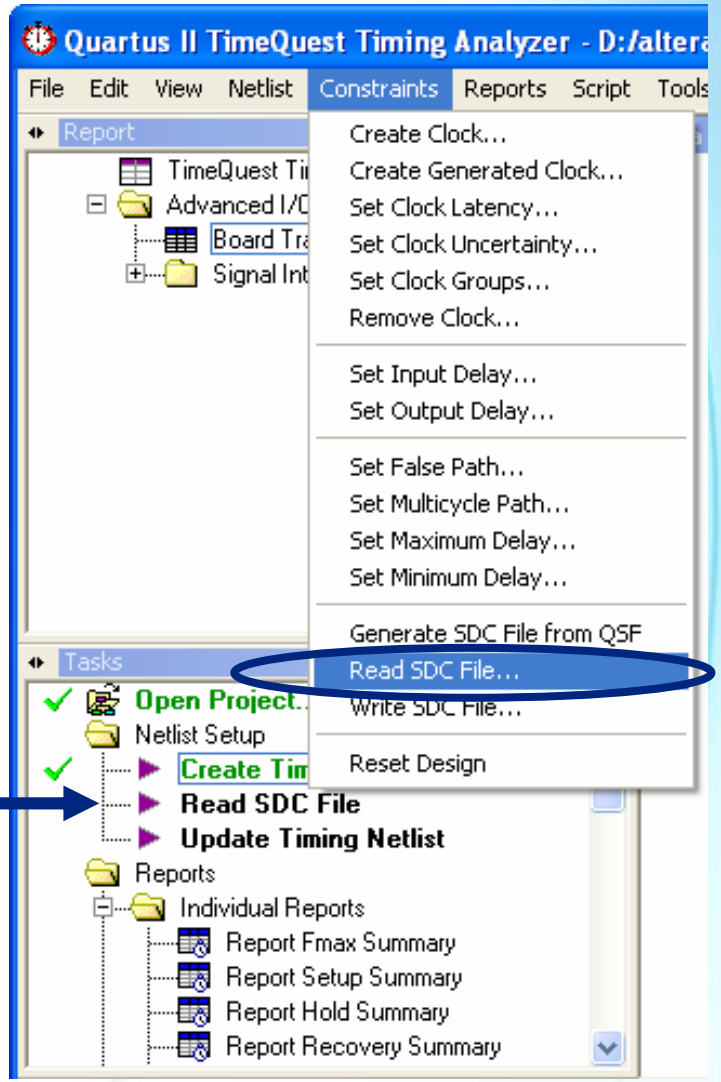


Tcl: create_timing_netlist

2) Create & Read SDC File (Optional)

- Create SDC file using SDC file editor
- Read in constraints & exceptions from SDC file
  - Skip if no SDC file
- Execution
  - Read SDC File (Tasks pane or Constraints menu)
- File Precedence (if no filename specified)
  - Files specifically added to Quartus II project
  - <current_revision>.sdc (if it exists)

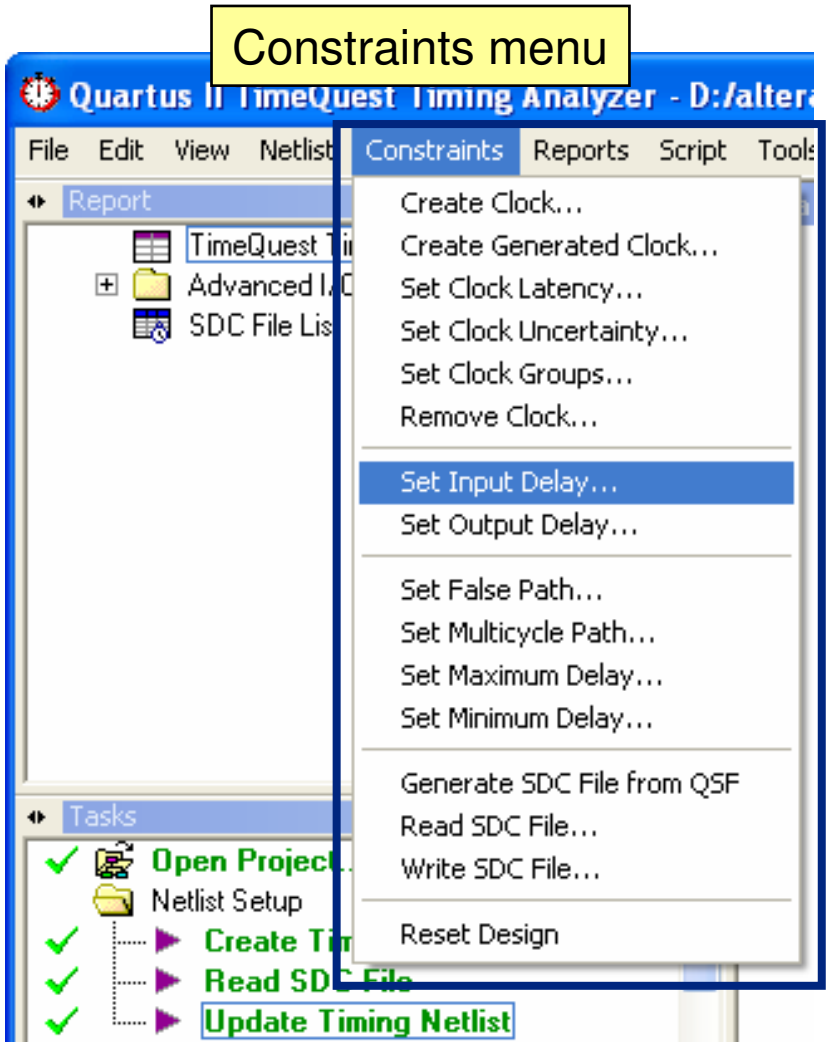
Tcl: read_sdc [<filename>]



3) Constrain Design (Optional)

- Add new constraints directly
  - Not added to SDC file
  - Use GUI or Console pane
  - Not needed if all constraints in SDC file
- Examples
  - `create_clock`
  - `create_generated_clock`
  - `set_input_delay`
  - `set_output_delay`

Using GUI to Enter Constraints



- Most common constraints can be accessed from the Constraints menu
- Use if unfamiliar with SDC syntax

Constraining

- User MUST enter constraints to fully analyze design
  - TimeQuest only performs slack analysis on constrained design paths
  - Recommendation: Constrain all paths (clocks & I/O)
- May create and read SDC file (Step 2), enter constraints directly in TimeQuest (Step 3) or both

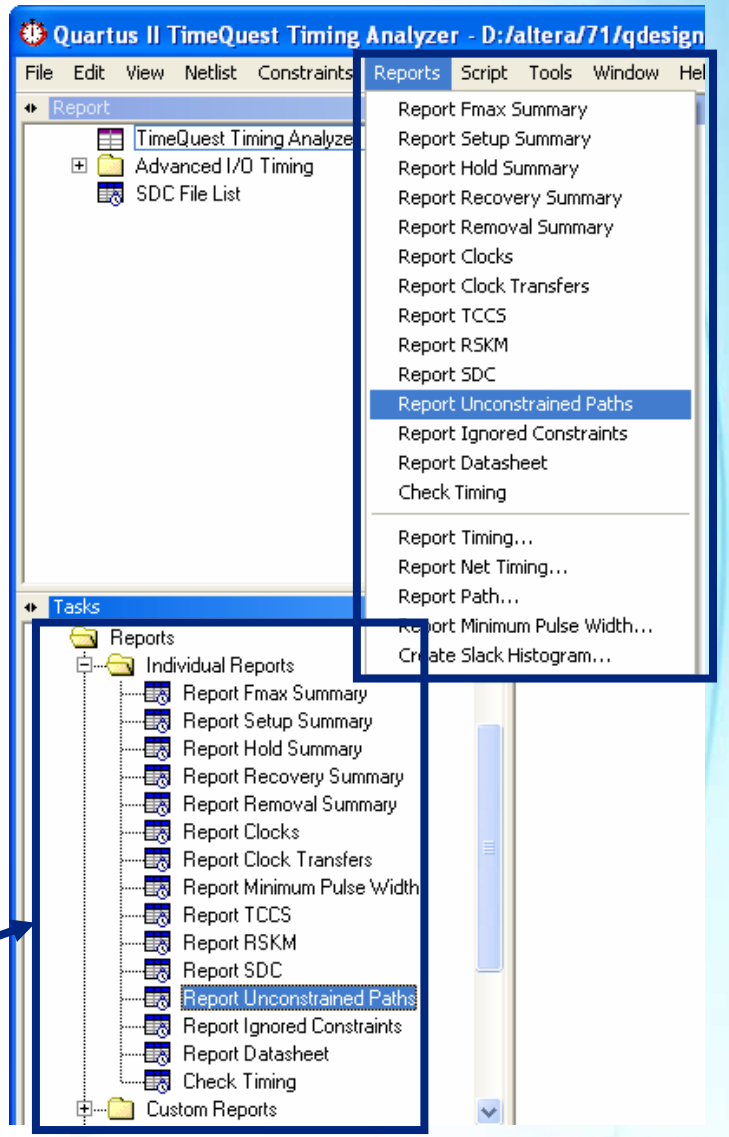
4) Update Timing Netlist

- Applies SDC constraints/exceptions to current timing netlist
- Generates warnings
  - Undefined clocks
  - Partially defined I/O delays
  - Combinatorial loops
- Update timing netlist after adding any new constraint
- Execution
  - Update Timing Netlist (Tasks pane or Netlist menu)

Tcl: update_timing_netlist

5) Generate Timing Reports

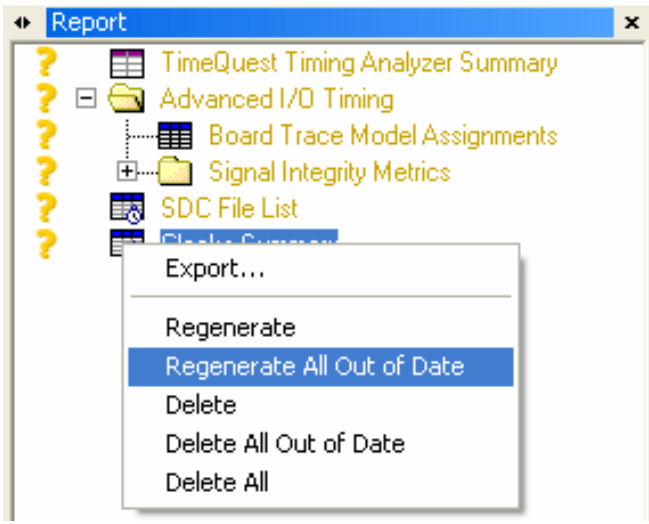
- Verify timing requirements and locate violations
- Check for fully constrained design or ignored timing constraints
- Two Methods
  - Tasks pane
    - Automatically creates/updates netlist & reads default SDC file if needed
  - Reports menu
    - Must have valid netlist to access
    - Tasks pane or Reports menu

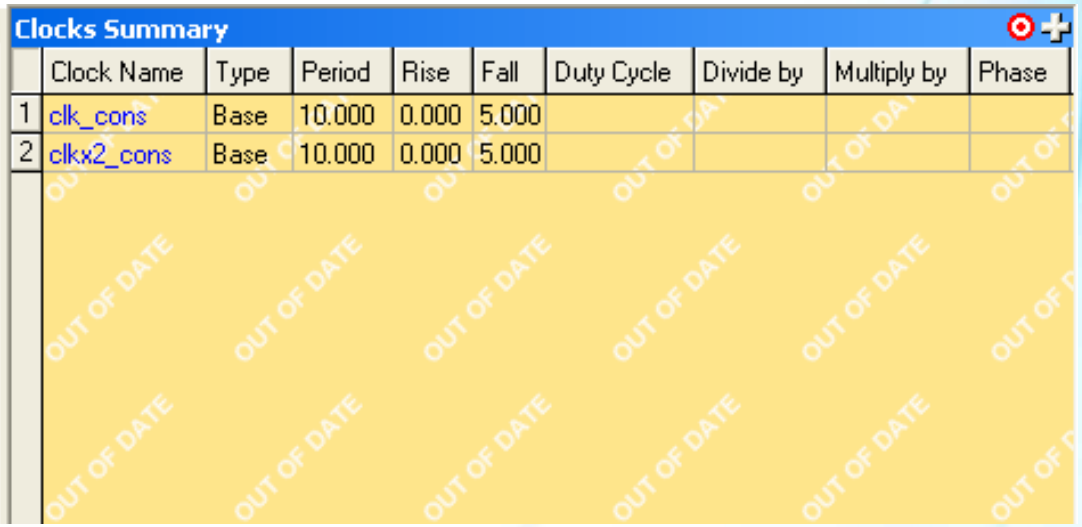


Double-click on individual report

"Out of Date" Reports

- Adding new constraints causes current reports to be "out of date"
- Update timing netlist & regenerate reports (report right-click menu)

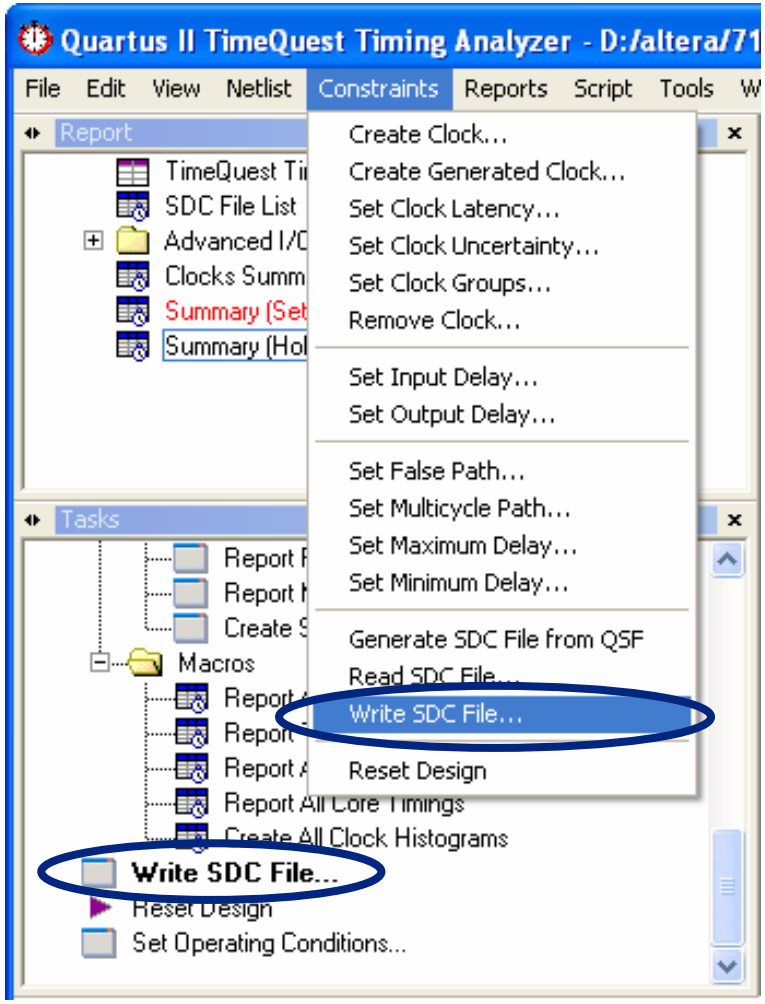




Reset Design Command

- Located in Tasks pane
- Flushes all timing constraints from current timing netlist
  - Functional Tcl equivalent: `delete_timing_netlist` command followed by `create_timing_netlist`
- Uses
  - "Re-starting" timing analysis on same timing netlist applying different constraints or SDC file
  - Starting analysis over if results seem to be unexpected

6) Save Timing Constraints (Optional)



- `write_sdc` command
  - Saves all constraints & exceptions applied to current netlist into SDC file
  - Use if constraints added during TimeQuest session outside of input SDC file
- Notes
  - SDC files generated by TimeQuest only if requested
  - Converts Altera-specific SDC commands into standard SDC
  - Run `report_sdc` command to see what will get written to SDC file

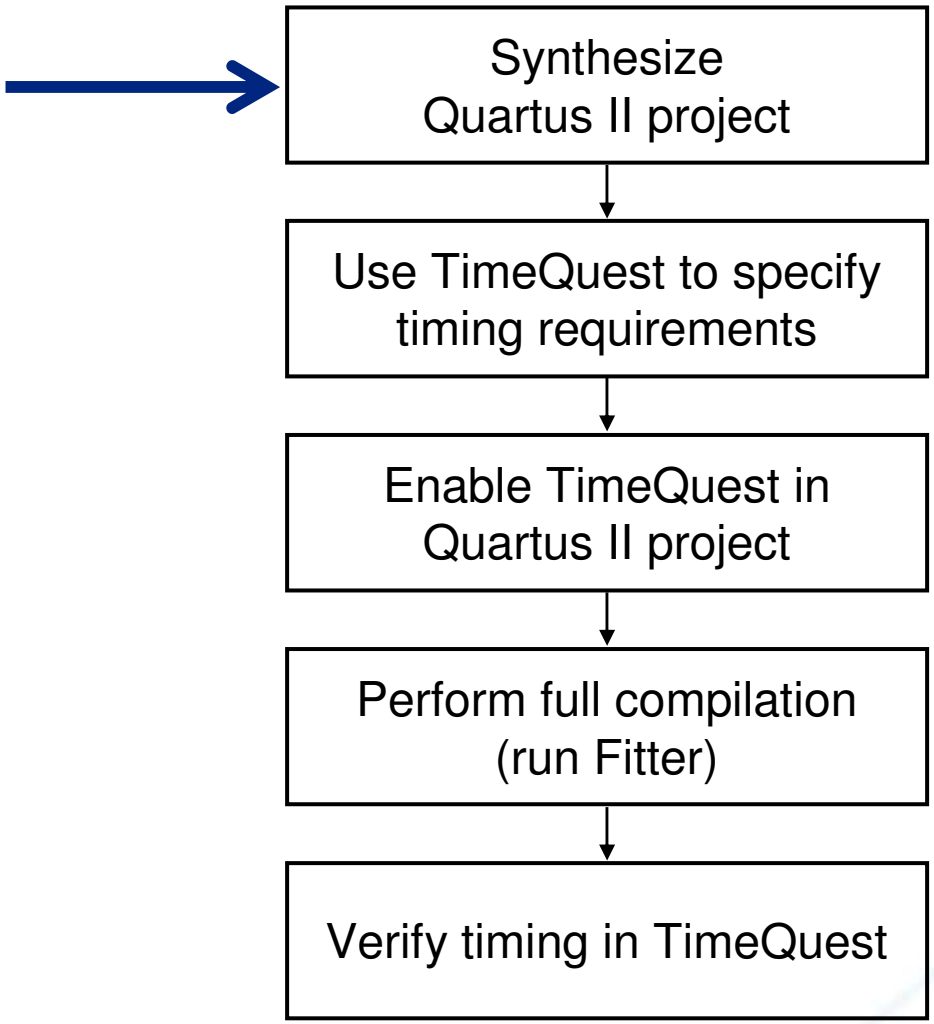
Steps to Using TimeQuest (Review)

1. Generate timing netlist
2. Read SDC file (optional)

or

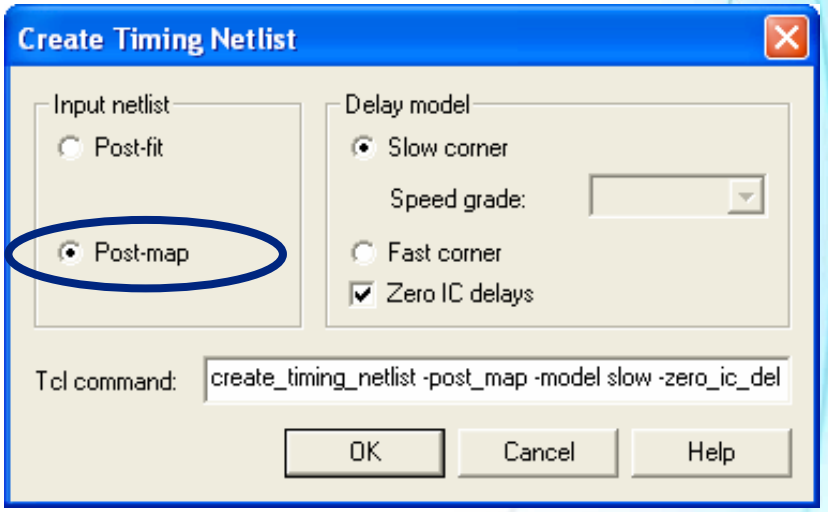
3. Constrain design (optional)
4. Update timing netlist
5. Generate timing reports
6. Save timing constraints (optional)

Using TimeQuest in Quartus II Flow



Specifying Timing Requirements

- Follow TimeQuest steps
- Use `-post_map` argument for synthesis (mapping) only netlist
  - If the design is already fully compiled, then choose `-post_fit` (default)

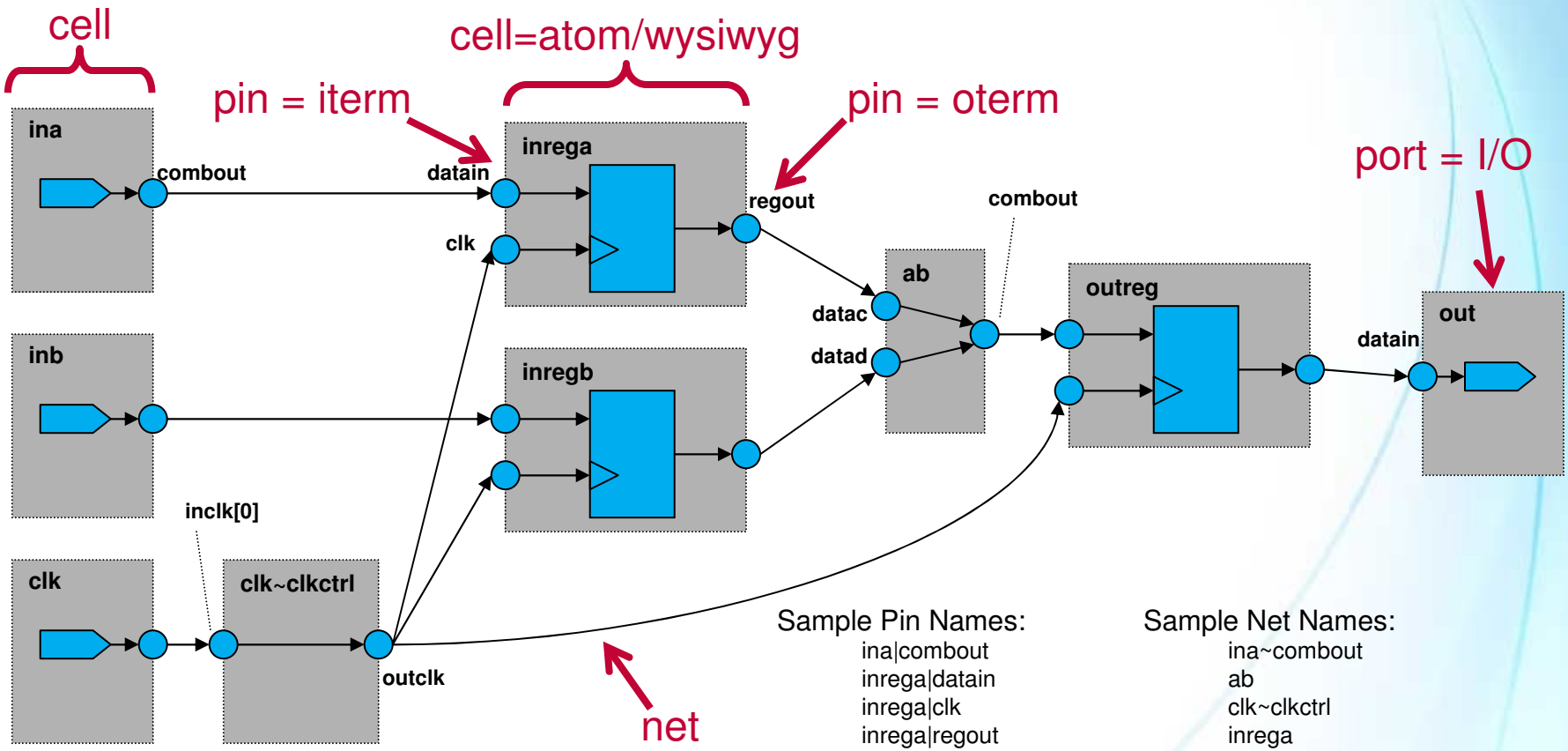


- Enter constraints via Constraints menu or Console pane

SDC Netlist Terminology

Term	Definition
Cell	Device building blocks (e.g. look-up tables, registers, embedded multipliers, memory blocks, I/O elements, PLLs, etc.)
Pin	Input or outputs of cells
Net	Connections between pins
Port	Top-level inputs and outputs (e.g. device pins)

SDC Netlist Example

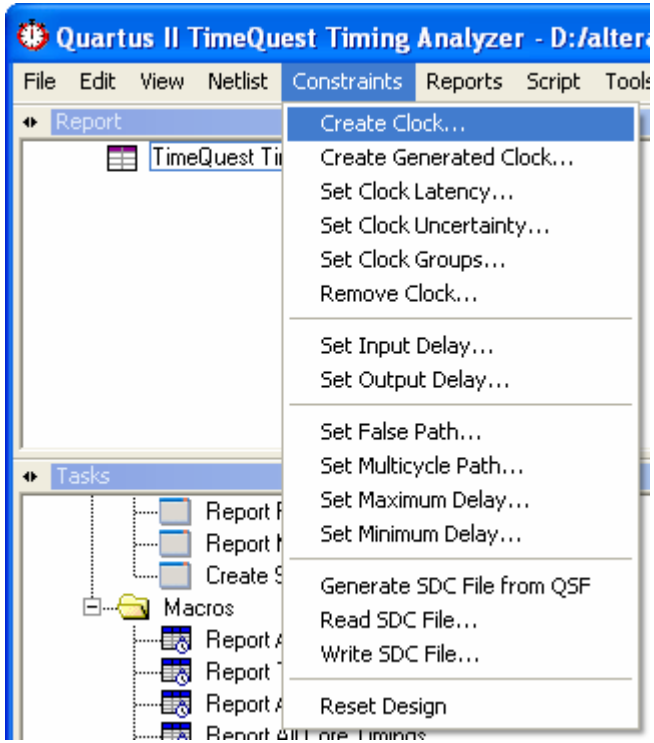


Collections

- Searches and returns from the design netlist with a list of names meeting criteria
- Used in SDC commands
  - Some collections searched automatically during commands usage and may not need to be specified
- Examples
  - `get_ports`
  - `get_pins`
  - `get_clocks`
  - `all_clocks`
  - `all_registers`
  - `all_inputs`
  - `all_outputs`

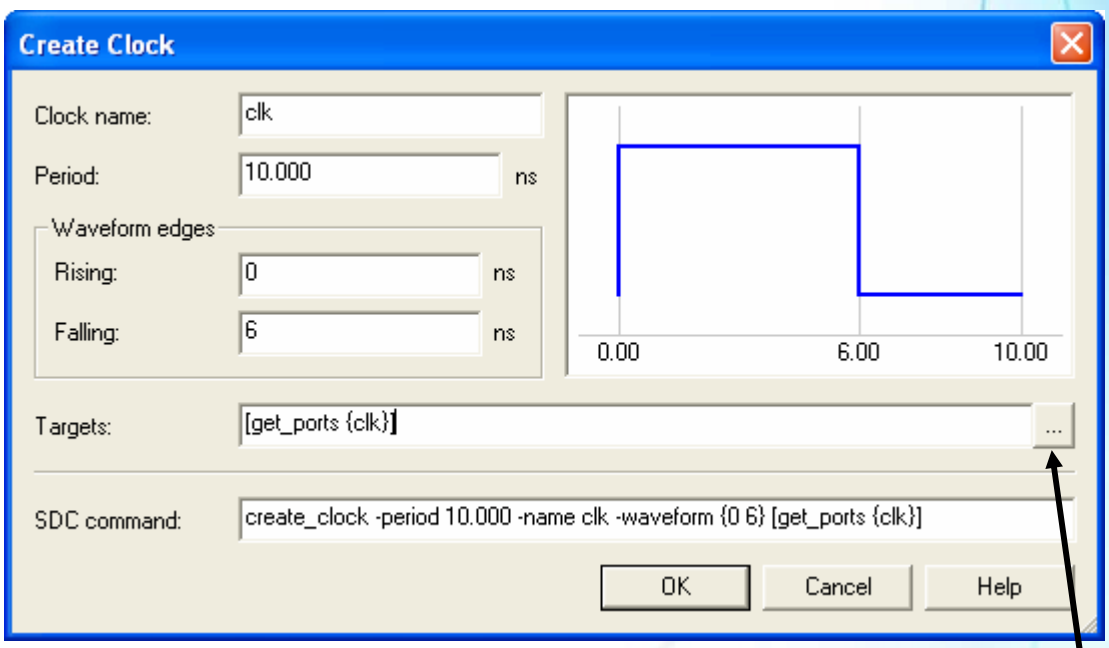
See "TimeQuest Timing Analyzer" chapter of the Quartus II Software Handbook for a complete list & description of each

Create Clock



Create Clock fields:

- **Clock Name – Assign name to clock setting; defaults to target node name**
- **Period – Clock period in nanoseconds**
- **Waveform edges – Use for non-50% duty cycle clocks**
- **Targets – Port or pin to which clock setting is being applied**

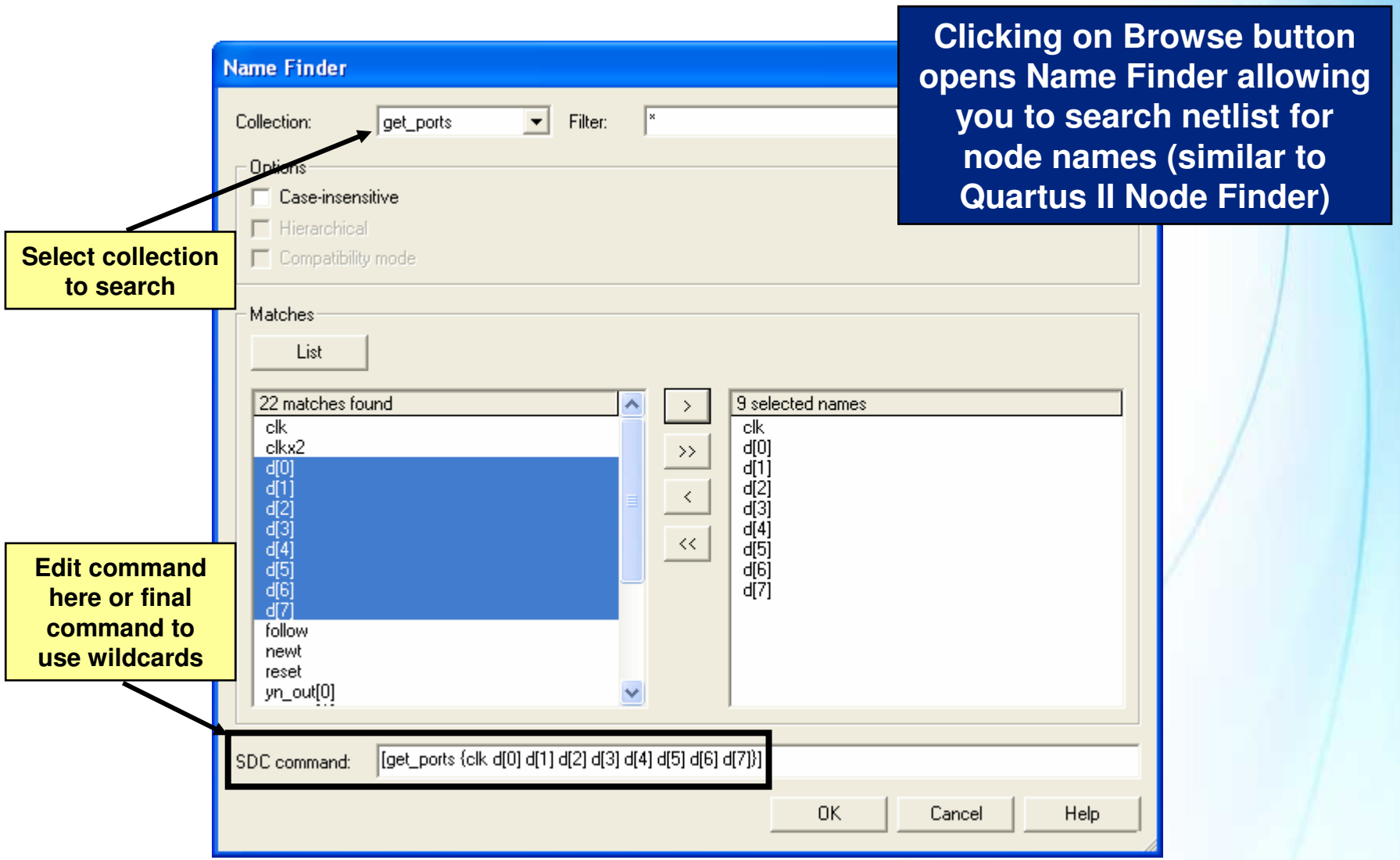


***Important Note*: Unlike the Classic Timing Analyzer, all design clocks are related by default. This means TimeQuest will analyzer paths between clock domains whether you have specifically related them or not.**

Name Finder (next slide)

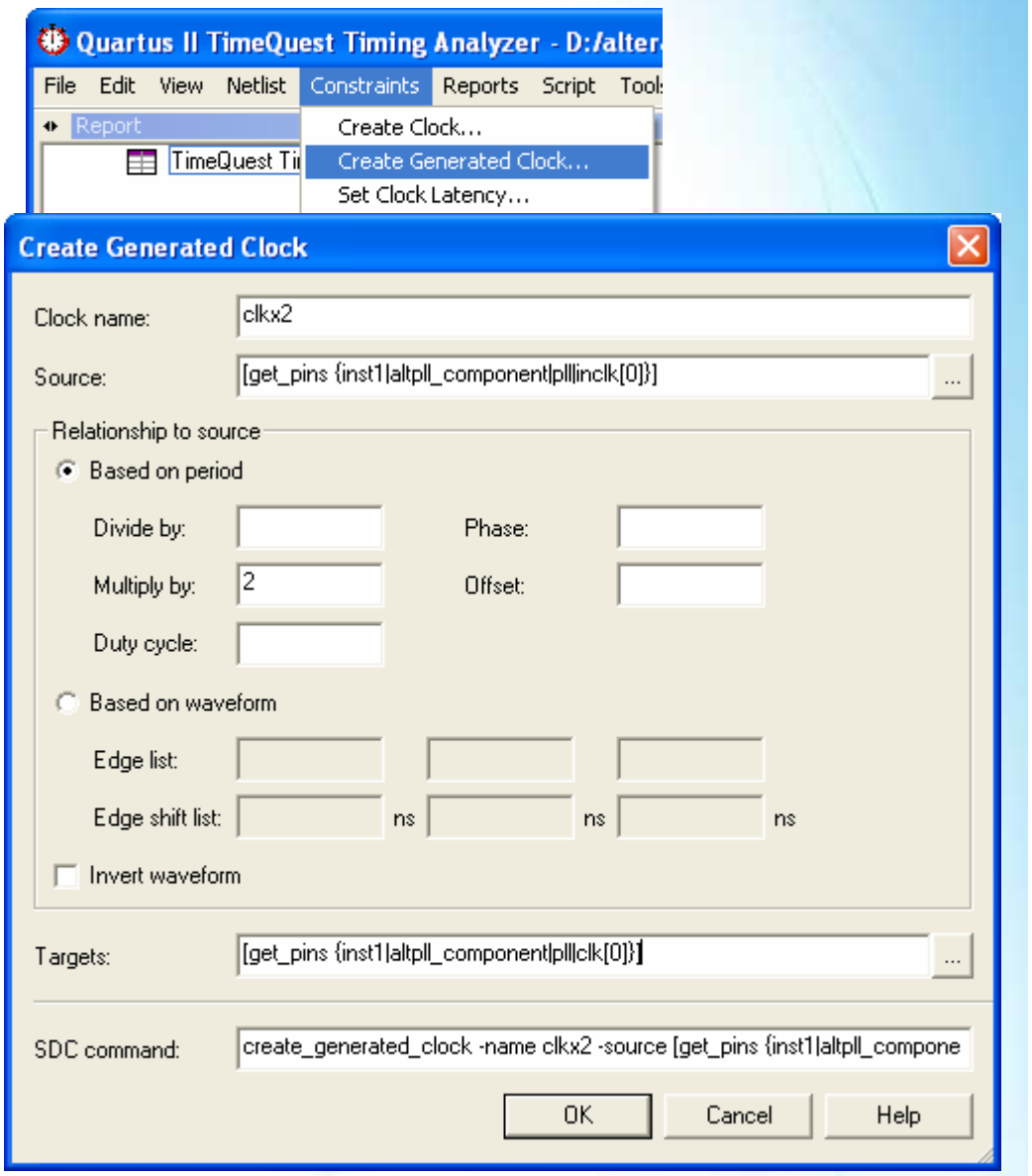
SDC: create_clock

Name Finder



Generated Clocks

- Generated clocks are clock signals derived from a previously created clock
  - E.g. clock dividers, ripple clocks, PLLs
  - Must be defined by a constraint



Create Generated Clock fields:

- **Clock Name – Assign name to clock setting**
- **Relationship to source – Specify how generated clock is related to base clock. The Based on waveform section allows for more complexity in the relationship to the base clock (not discussed)**
- **Targets – Port or pin to which clock setting is being applied**

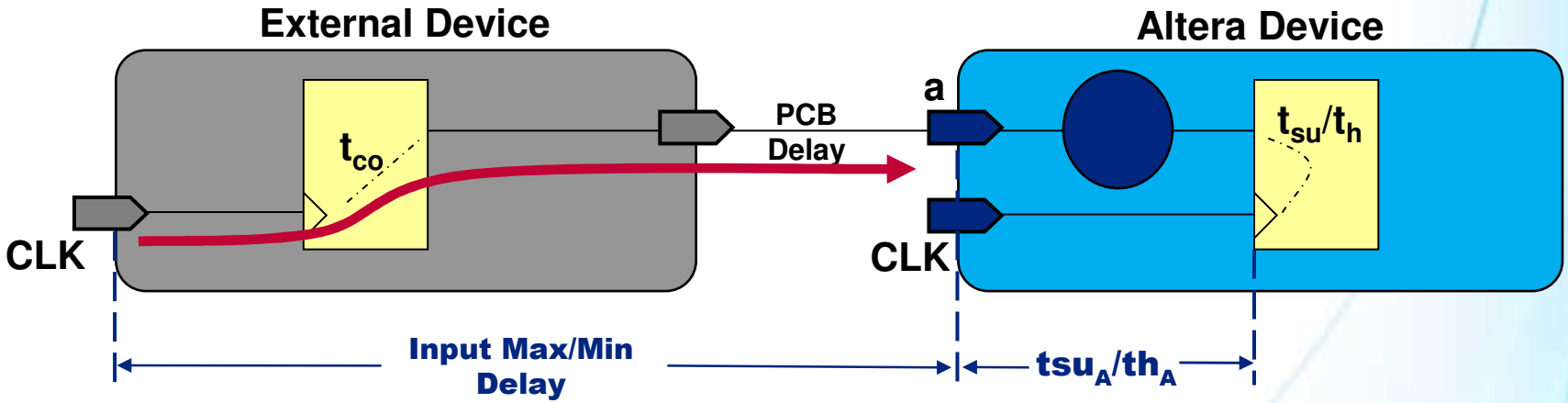
SDC: create_generated_clock

I/O Constraining

- Specify system-level timing constraints
- Settings
  - Input maximum delay
  - Input minimum delay
  - Output maximum delay
  - Output minimum delay

Input Maximum/Minimum Delay

- Maximum/minimum delay from external device to Altera I/O
  - Represents external device max & min t_{co} + PCB delay - PCB clock skew
- Constrains registered input path (t_{su}/t_h)

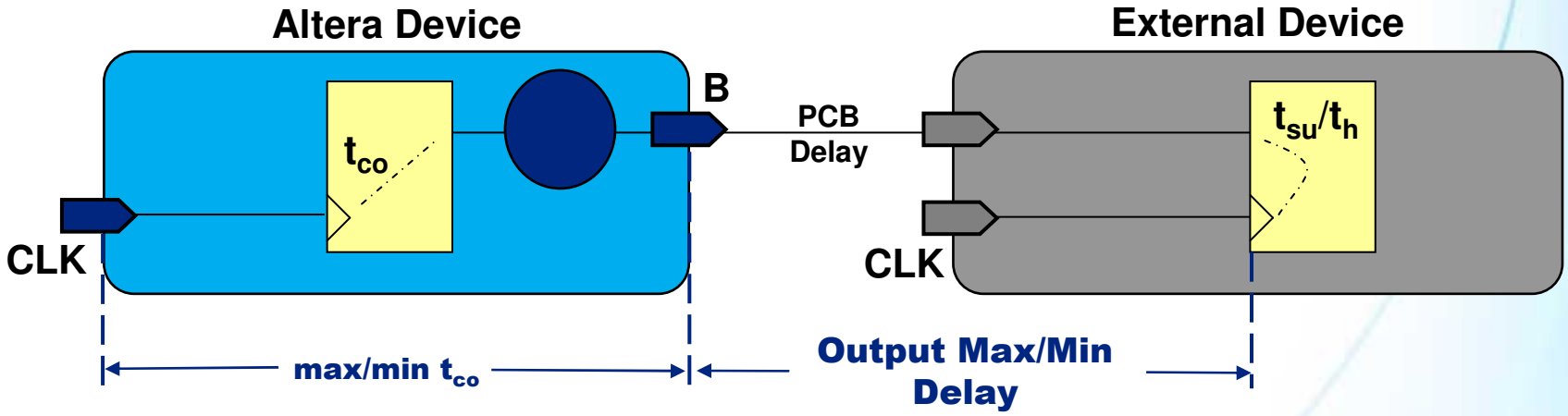


$tsu_A \leq t_{CLK}$ – *Input Maximum Delay*

$th_A \leq$ *Input Minimum Delay*

Output Maximum/Minimum Delay

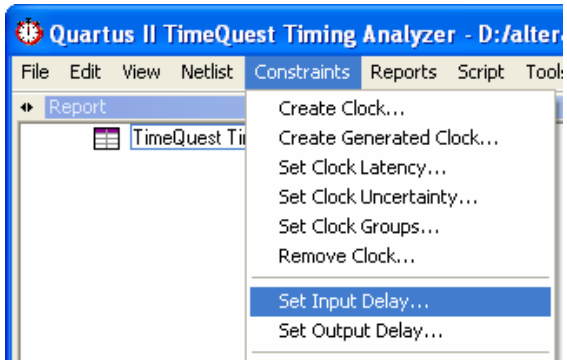
- Maximum/minimum delay from Altera I/O to external device
  - Represents external device t_{su}/t_h + PCB delay - PCB clock skew
- Constrains registered output path (max & min t_{co})



$tco_B \leq t_{CLK}$ - *Output Maximum Delay*

$tco_B \geq$ *Output Minimum Delay*

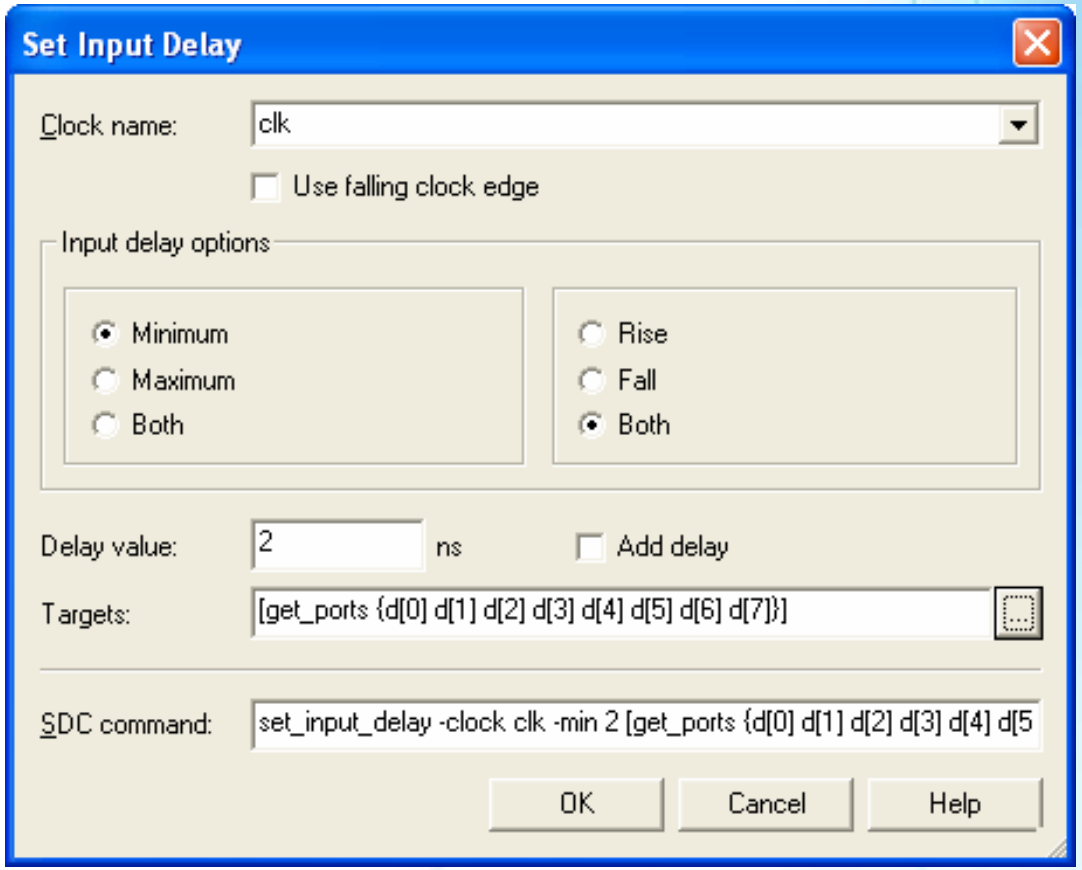
Set Input/Output Delay



Set Input/Output Delay fields:

- **Clock Name – Specify source clock**
- **Input delay options – Choose max or min constraint. Rise/Fall indicate if the constraint applies particularly to a rising or falling edge transition (advanced).**
- **Delay value – Total off chip delay**
- **Add delay – Must use if applying multiple sets of input/output delays to the same port (e.g. input ports feeding multiple internal registers)**
- **Targets – Port to which setting is being applied**

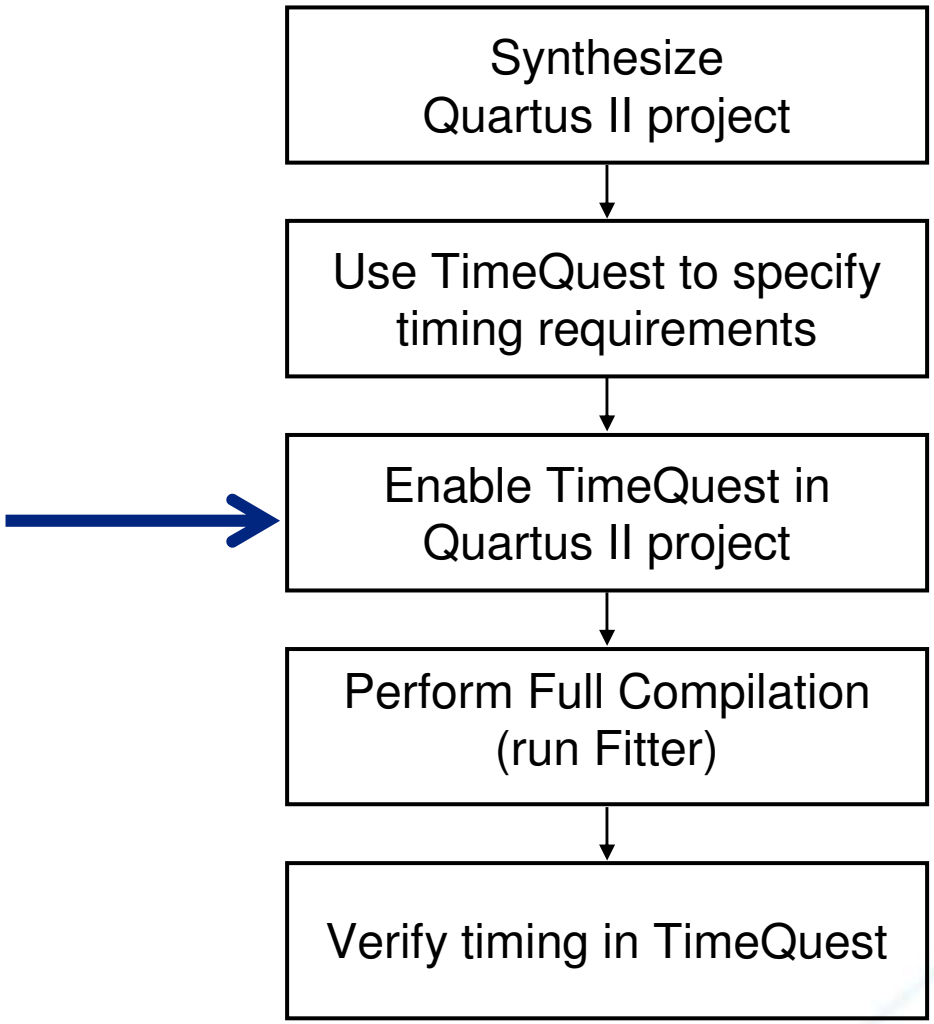
SDC: set_input_delay
SDC: set_output_delay



Useful Reports for Design Constraining

- Report Clocks
  - Use to ensure all clocks have been defined correctly
- Report Unconstrained Paths
  - Use to determine if any constraints are missing
- Report SDC
  - Use to review what constraints have currently been applied to the netlist
- Check Timing
  - Use to check issues with design or applied constraints
- Report Clock Transfers
  - Use to determines nets crossing clock domains
  - Remember, by default, all clock domains are related and analyzed with respect to one another
    - Paths between domains might need to be set as false paths
- Report Ignored Constraints
  - Use to determine if any constraints being ignored

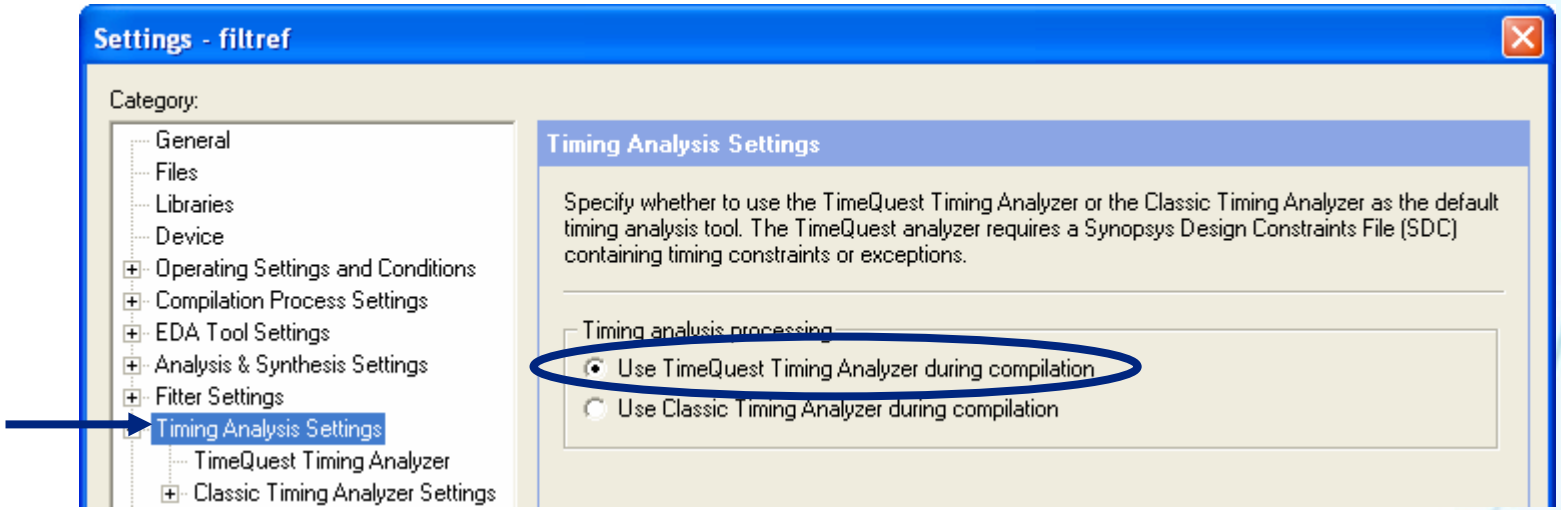
Using TimeQuest in Quartus II Flow



Enabling TimeQuest in Quartus II

- Tells Quartus II to use SDC constraints during fitting
- File order precedence
  1. Any SDC files added to Quartus II project (in order)
  2. <*current_revision*>.SDC

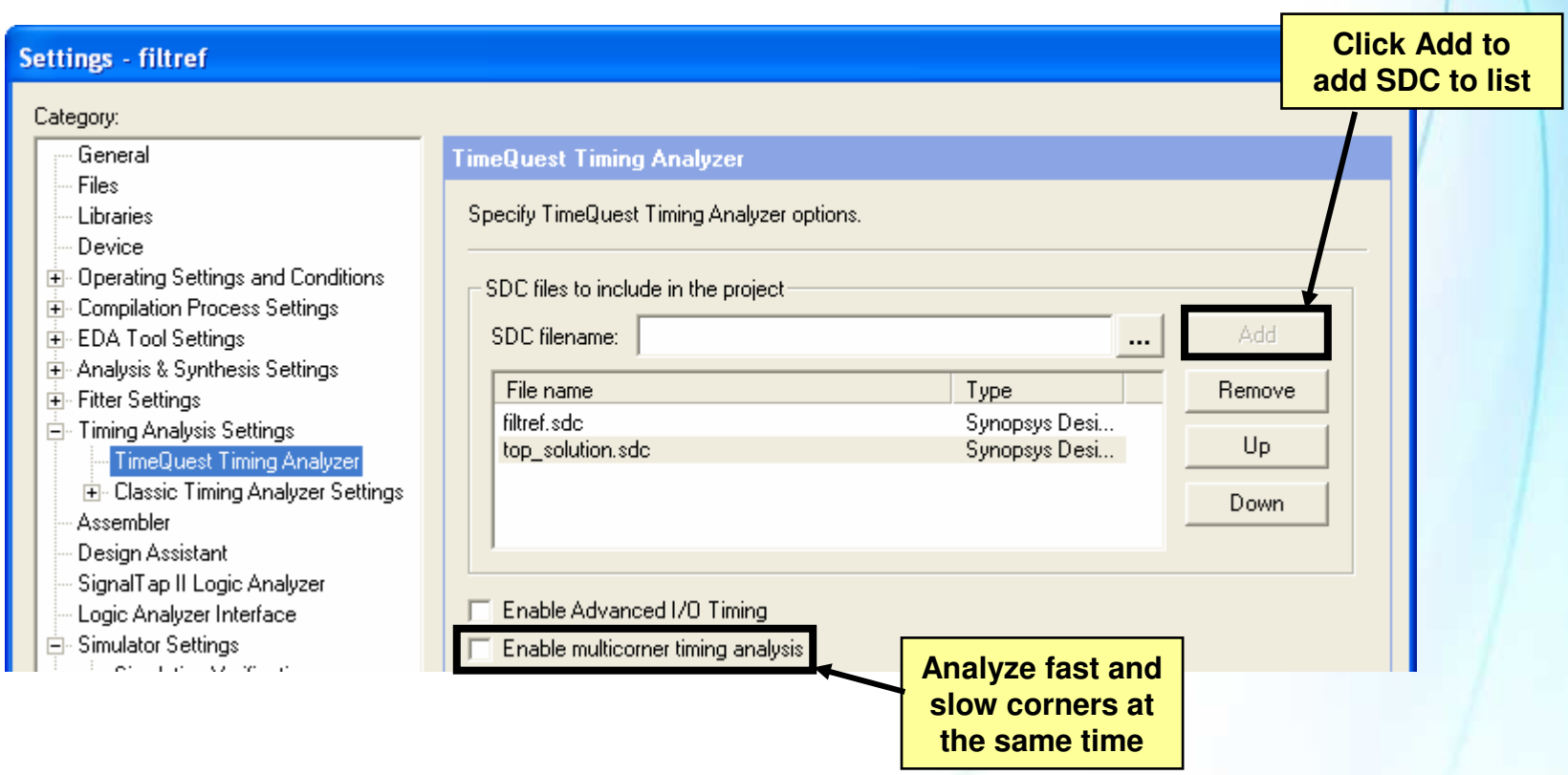
Enabling TimeQuest in Quartus II Software



Notes:
- **Arria GX only supports Timequest.**
- **TimeQuest is enabled by default for new Stratix III and Cyclone III designs.**

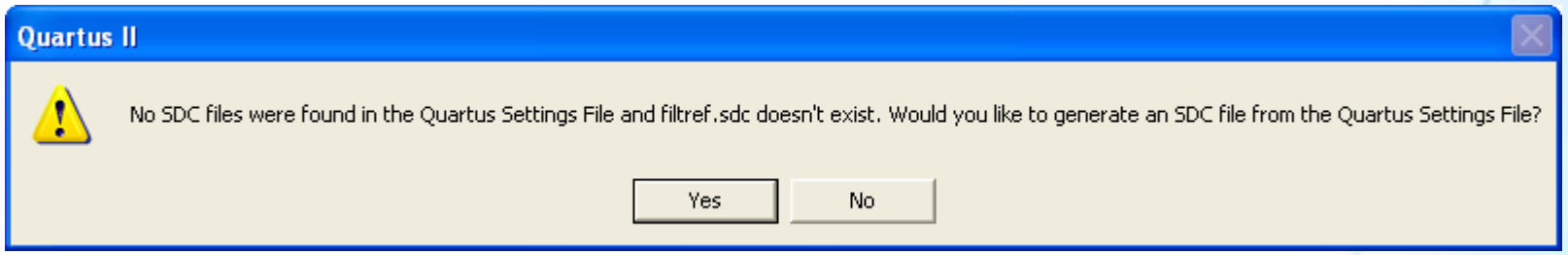
Adding SDC File to Quartus II Project

- Add SDC files to TimeQuest Timing Analyzer page of Settings dialog box



Quartus Settings File (QSF)

- SDC constraints are not stored in QSF
- TimeQuest uses script to convert QSF timing assignments to SDC
  - Constraints menu
  - Done automatically if no SDC file exists when first opening TimeQuest

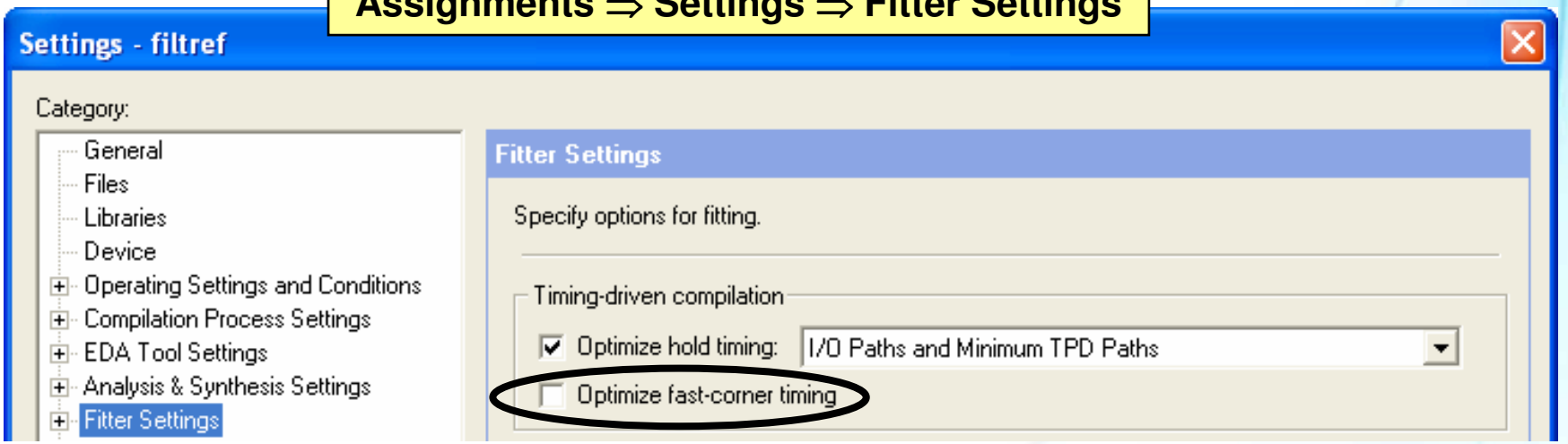


- See Quartus II Handbook Chapter, *"Switching to the TimeQuest Timing Analyzer"* for details
  - Differences between Classic Timing Analyzer and TimeQuest
  - Details on conversion utility

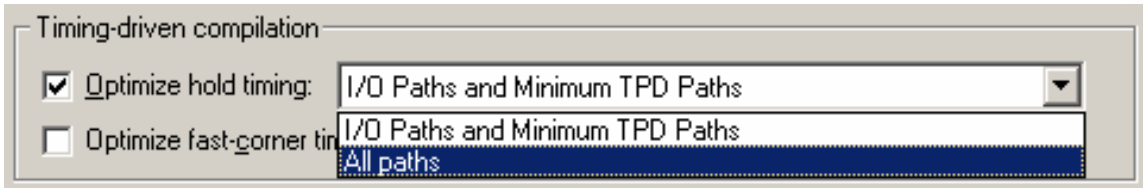
Timing Driven Compilation (TDC)

- Directs fitter to place & route logic to meet timing assignments
  - Optimize timing (on by default)
    - Placing nodes in critical paths closer together
    - Located in “more settings” box
  - Optimize fast-corner timing
    - Optimizing for fast process (minimum timing models)
    - Can add up to 10% to compile time

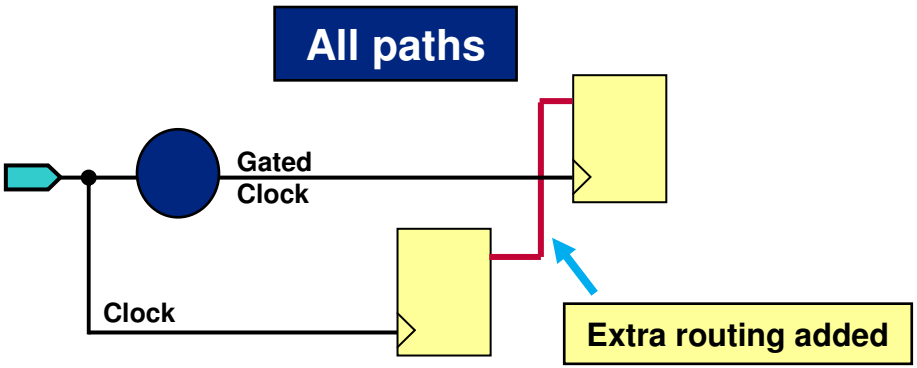
Assignments ⇒ Settings ⇒ Fitter Settings

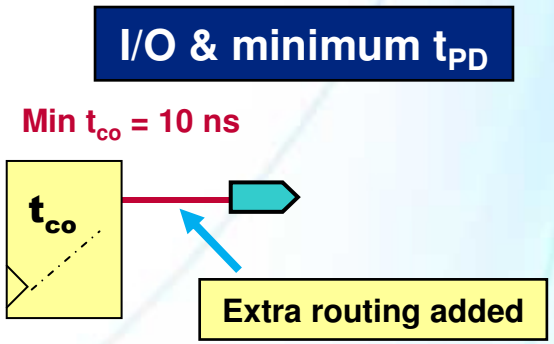


Optimize Hold Timing

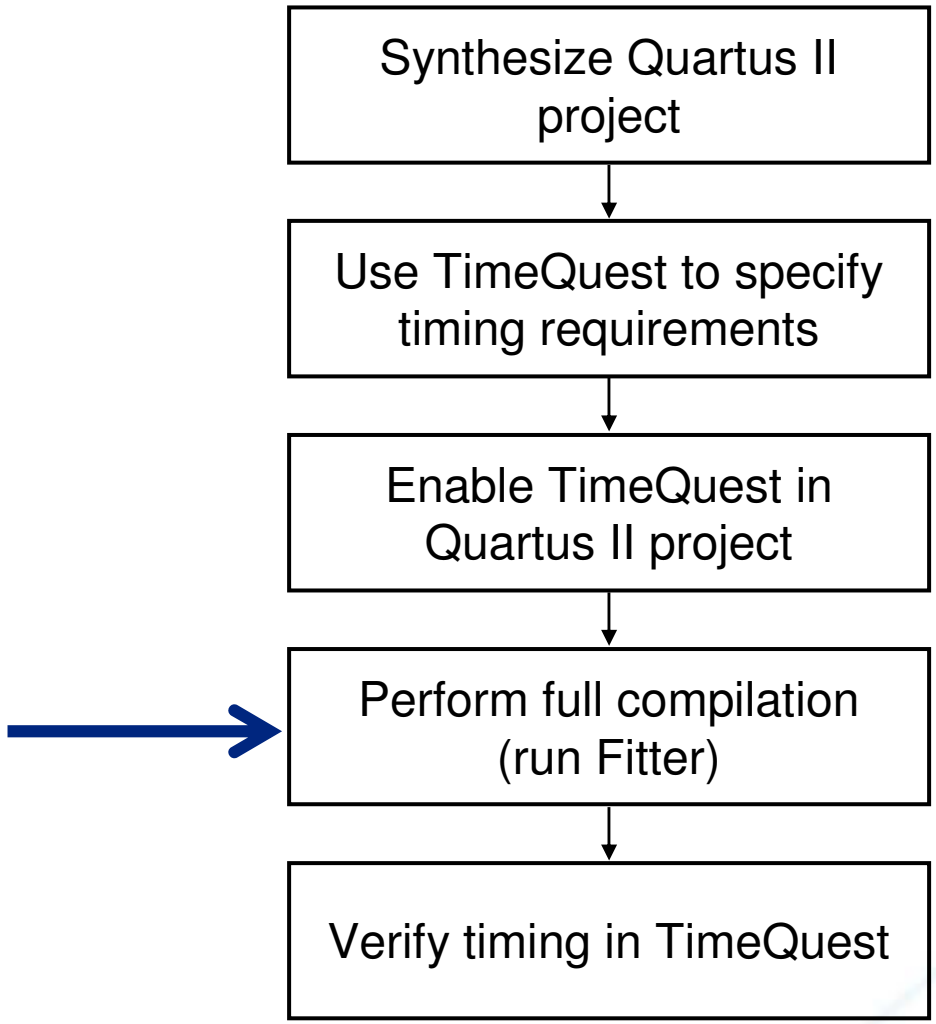


- Modifies place & route to meet hold or minimum timing requirements
  - May add additional routing in path
- Settings
  - Any I/O & minimum t_{pd} paths (default)
  - All paths (I/O & internal)





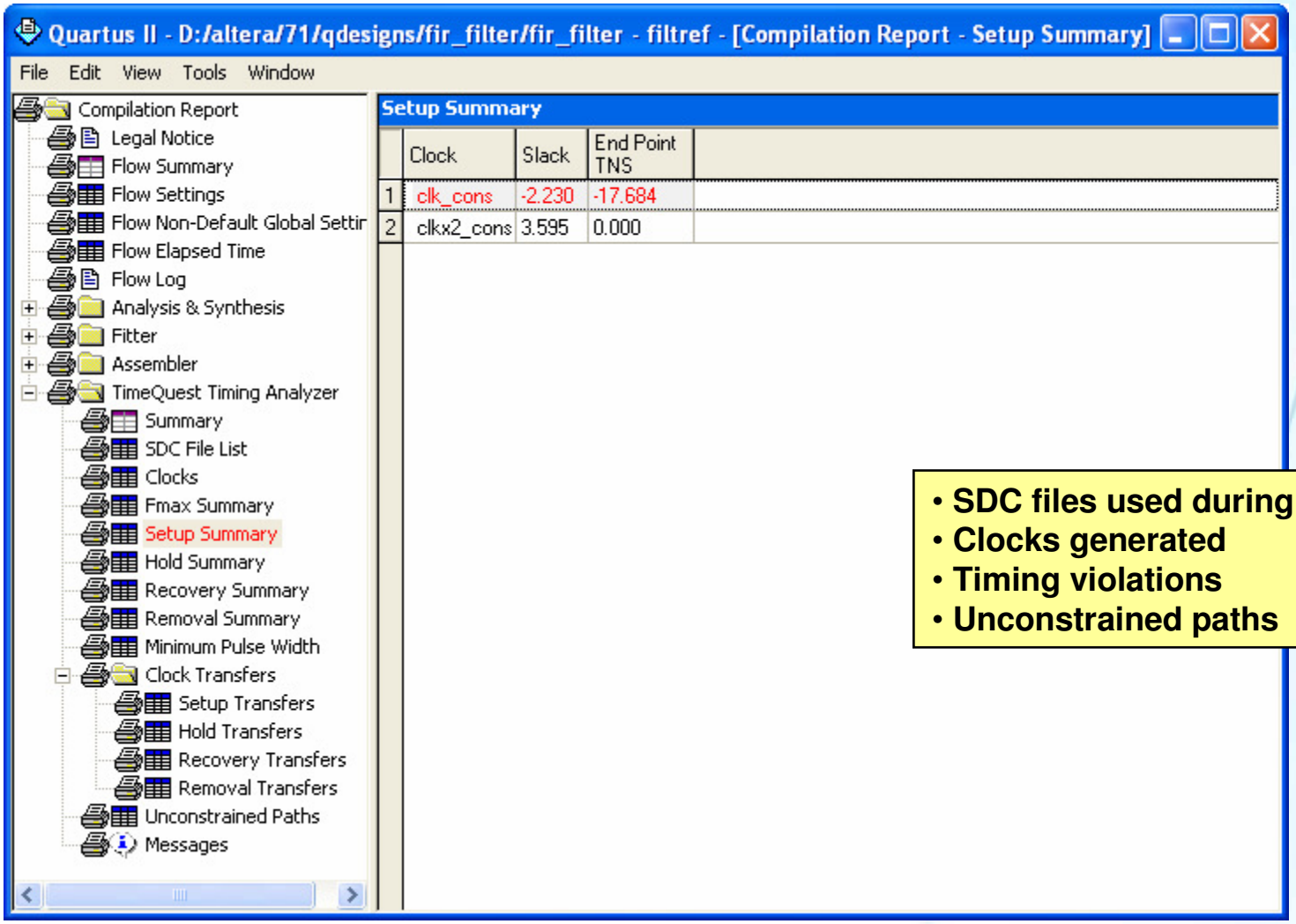
Using TimeQuest in Quartus II Flow



Verifying Timing Requirements

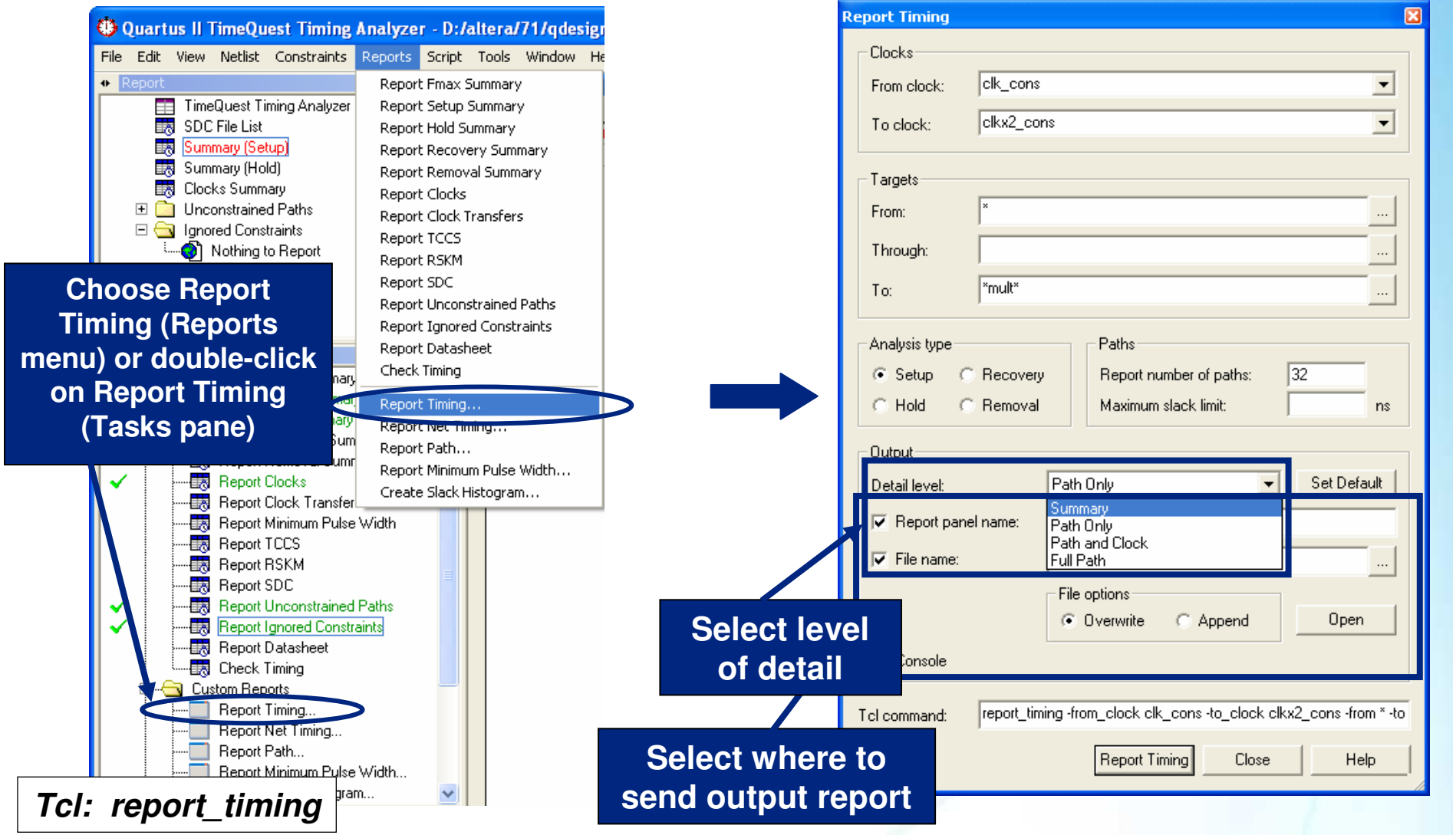
- View TimeQuest summary information directly in Quartus II Compilation Report
- Open TimeQuest for more thorough analysis
  - Follow TimeQuest flow
  - Run TimeQuest easy-to-use reporting capabilities (Tasks pane)
  - Place Tcl reporting commands into script file
    - Easy repetition

TimeQuest Summary Reports



- **SDC files used during fitting**
- **Clocks generated**
- **Timing violations**
- **Unconstrained paths**

Generating Detailed Reports



3rd-Party Timing Analysis Tool Support

- Synopsys
  - PrimeTime
- Mentor Graphics
  - TAU

Exercise 6 Demonstration

Summary

- TimeQuest provides an easy-to-use tool to verify timing
  - Entering timing constraints
  - Run various timing reports

TimeQuest Support Resources

- Quartus II Handbook Chapters
  - "The TimeQuest Timing Analyzer" (Volume 3)
  - "Switching to the TimeQuest Timing Analyzer" (Volume 3)
- Training & Demonstrations
  - "Validating Performance with the TimeQuest Static Timing Analyzer" (online recording)

Quartus II Software Design Series: Foundation

EDA Simulation

Simulation

- 3rd-party EDA tool simulation
  - RTL (functional)
  - Post-synthesis (functional)
    - Optional
  - Gate-level (timing)

- Quartus II simulation covered in Appendix

3rd-Party Simulation Support

- Mentor Graphics
  - ModelSim
- Cadence
  - NCSim
- Synopsys
  - VCS/MX
- Aldec
  - Active-HDL

- All support NativeLink tool flow

RTL Simulation Files (VHDL)

- Design files
- RTL (functional) models
  - LPM megafunction models
    - 220model.vhd & 220_pack.vhd
    - Compile into lpm library
  - Altera-specific megafunction models
    - altera_mf.vhd & altera_mf_components.vhd
    - Compile into altera_mf library
  - Altera primitive models (LCELL, OPNDRN, etc.)
    - altera_primitives.vhd & altera_primitives_components.vhd
    - Compile into altera library
- HEX files
  - Memory initialization

- **All model files located in "eda\sim_lib" directory in Quartus II installation path**
- **Pre-compiled in ModelSim-Altera**

RTL Simulation Files (Verilog)

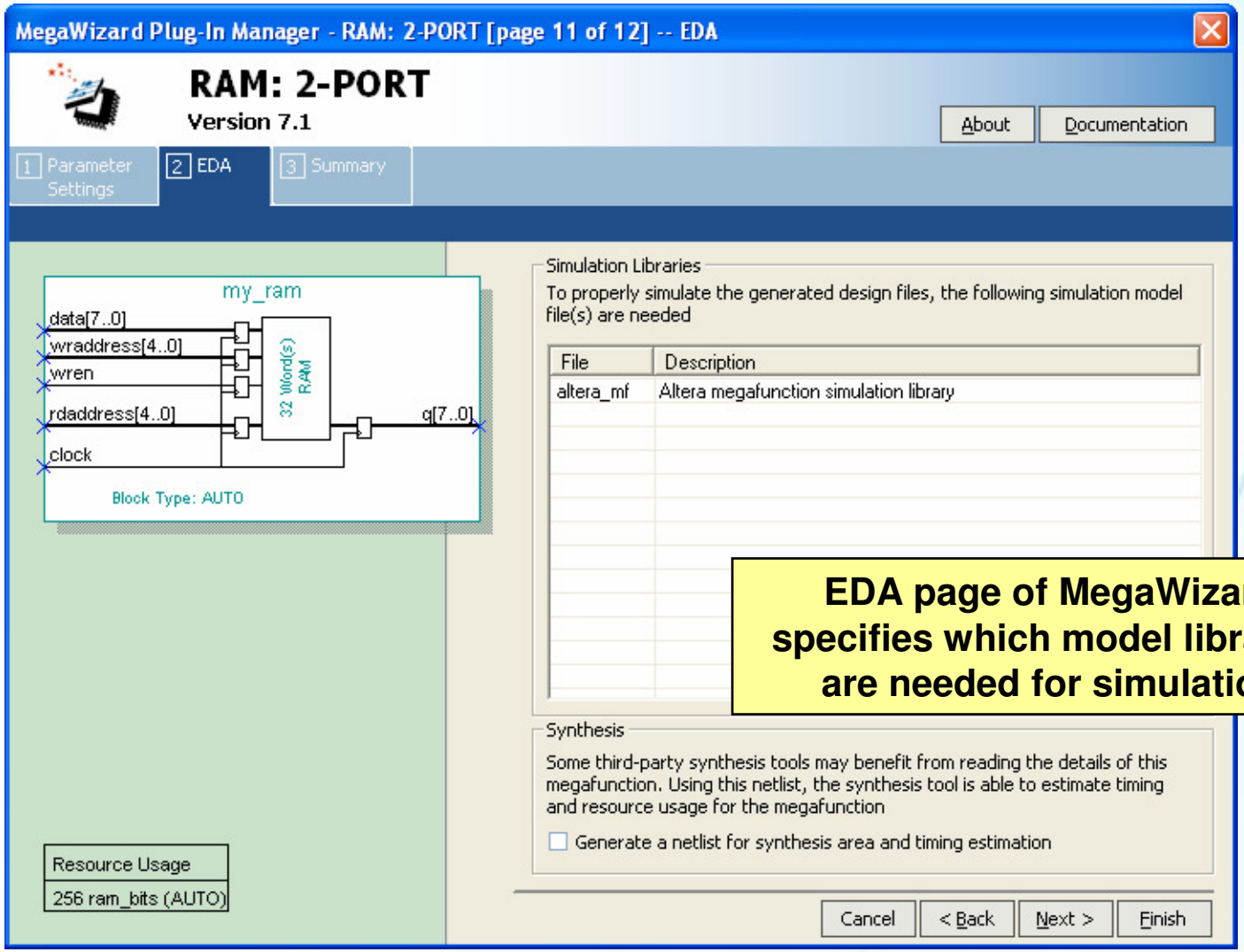
- Design Files
- RTL (functional) Models
  - LPM megafunction models
    - 220model.v
    - Compile into lpm library
  - Altera-specific megafunction models
    - altera_mf.v
    - Compile into altera_mf library
  - Altera primitive models (LCELL, OPNDRN, etc.)
    - altera_primitives.v
    - Compile into altera library
- HEX files
  - Memory initialization

- **All model files located in "eda\sim_lib" directory in Quartus II installation path**
- **Pre-compiled in ModelSim-Altera**

Megafunction Exceptions

- RTL simulation requires using device-specific simulation models (similar to gate-level)
  - altclkbuf
  - alkclkctrl
  - DDR megafunctions (not IP)
  - MAX II UFM megafunctions
  - altmemmult
  - altremote_update
- Stratix II GX and Stratix GX designs require additional libraries for RTL simulation
  - See Quartus II Handbook, Volume 3, Section 1: “Simulation”

MegaWizard Simulation Libraries



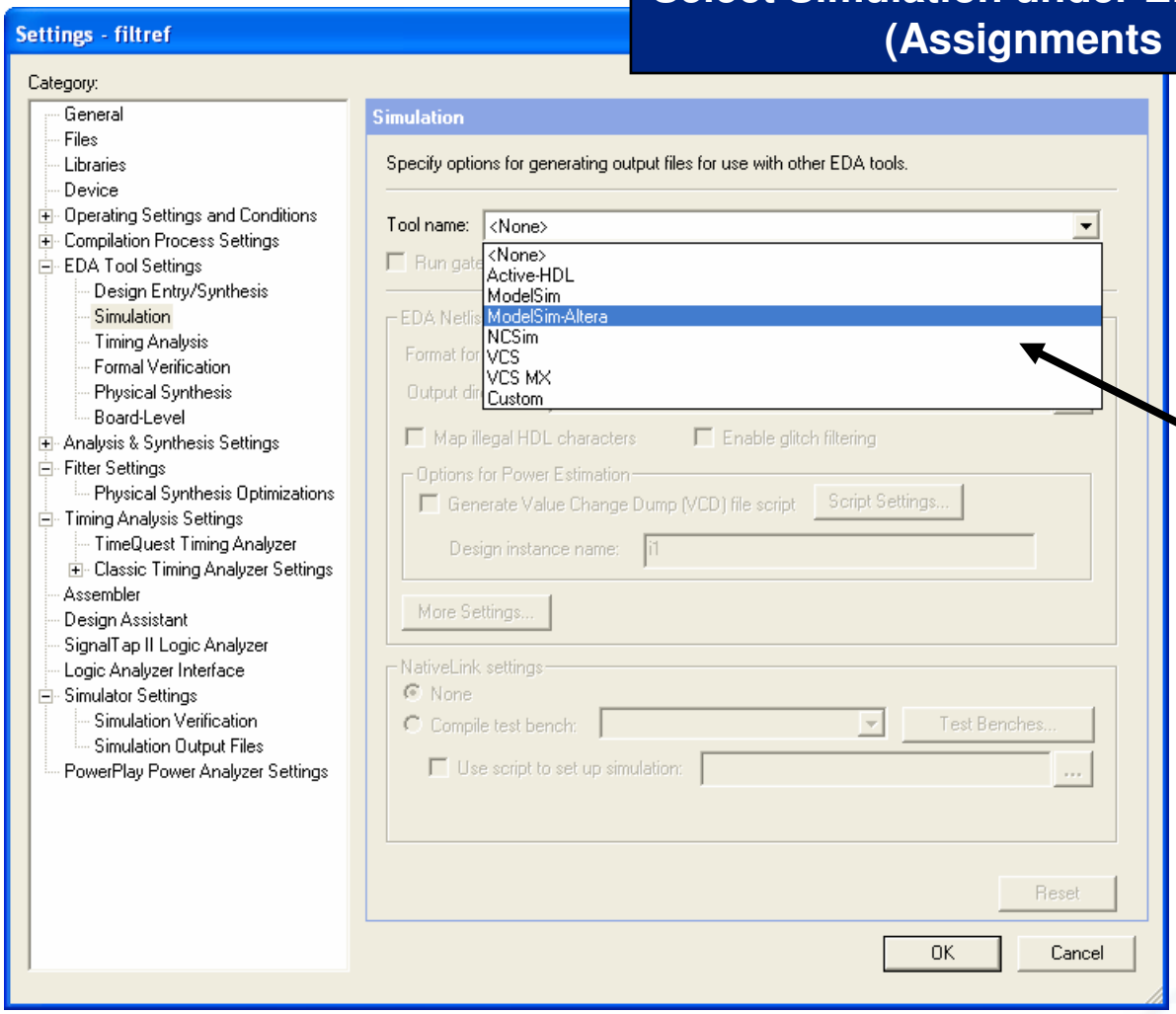
EDA page of MegaWizard specifies which model libraries are needed for simulation

Post-Processing Simulations

- Two types
  - Post-synthesis
  - Gate-level (timing)
- Quartus II software must generate output simulation netlist files for specific simulator
  - EDA Netlist Writer

Specify Simulator

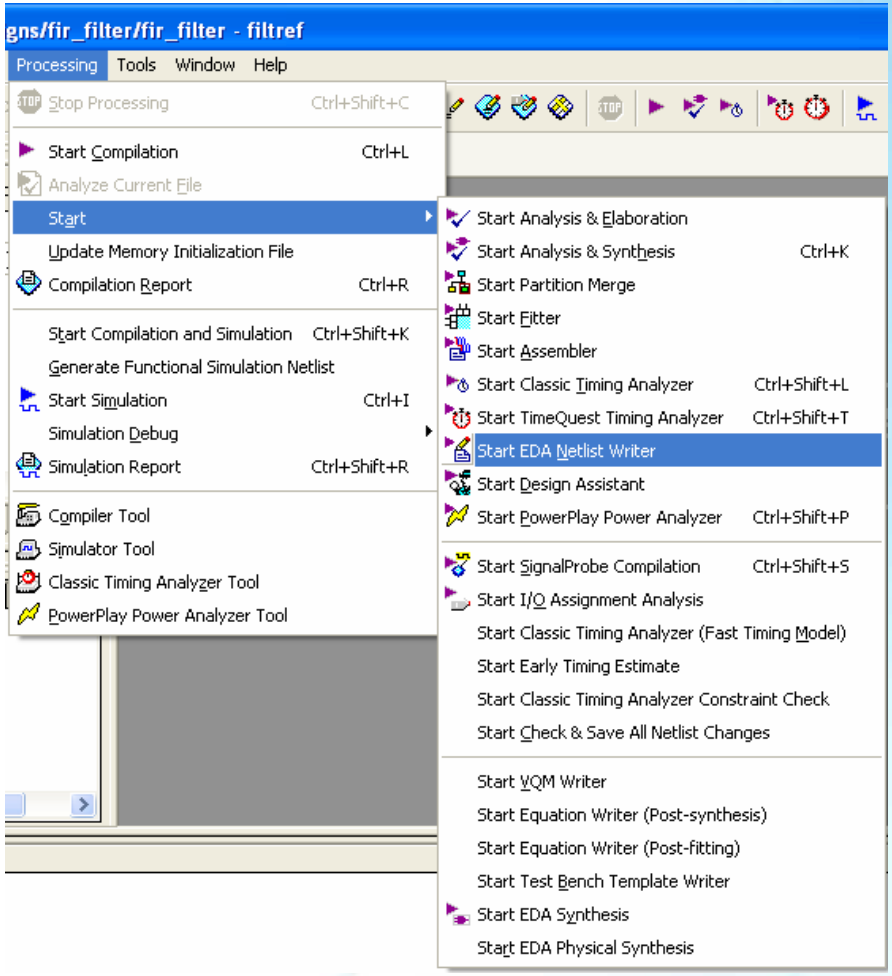
Select Simulation under EDA Tools Settings (Assignments menu)



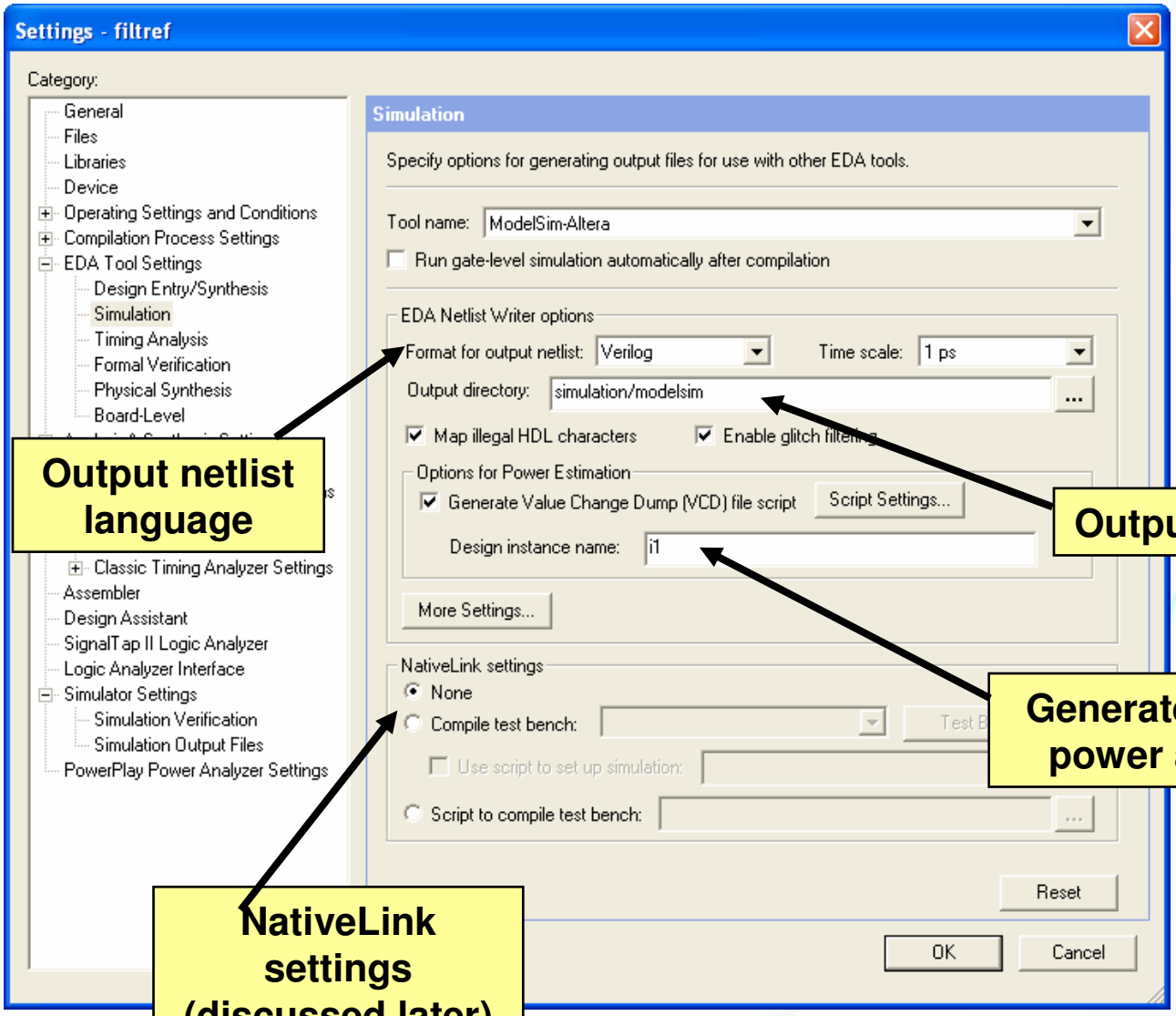
Select simulation tool

Generating 3rd-Party Netlists

- Full compilation
- Execute process individually
  - Processing menu ⇒ Start ⇒ Start EDA Netlist Writer
  - Generates files without full compilation
- Scripting



Specifying Simulation Options



Gate-Level Simulation Files (VHDL)

- All model files located in "eda\sim_lib" directory in Quartus II installation path
- Pre-compiled in ModelSim-Altera

- Use Quartus II-generated files
  - VHDL output file (.VHO)
  - Standard delay format output file (.SDO)
  - Located in "*simulation\<simulator_tool>"* subdirectory of project (default)
- Device-specific simulation models
  - Use <device_name>_atoms.vhd & <device_name>_atoms_components.vhd
  - Compile into <device_name> library
- HEX files for memory initialization

Gate-Level Simulation Files (Verilog)

- All model files located in "eda\sim_lib" directory in Quartus II installation path
- Pre-compiled in ModelSim-Altera

- Use Quartus II-generated files
  - Verilog output file (.VO)
  - Standard delay format output file (.SDO)
  - Located in "*simulation\<simulator_tool>*" subdirectory of project (default)
- Device-specific simulation models
  - <device_name>_atoms.vo
  - Compile into <device_name> library
- HEX files for memory initialization

Device Library Names

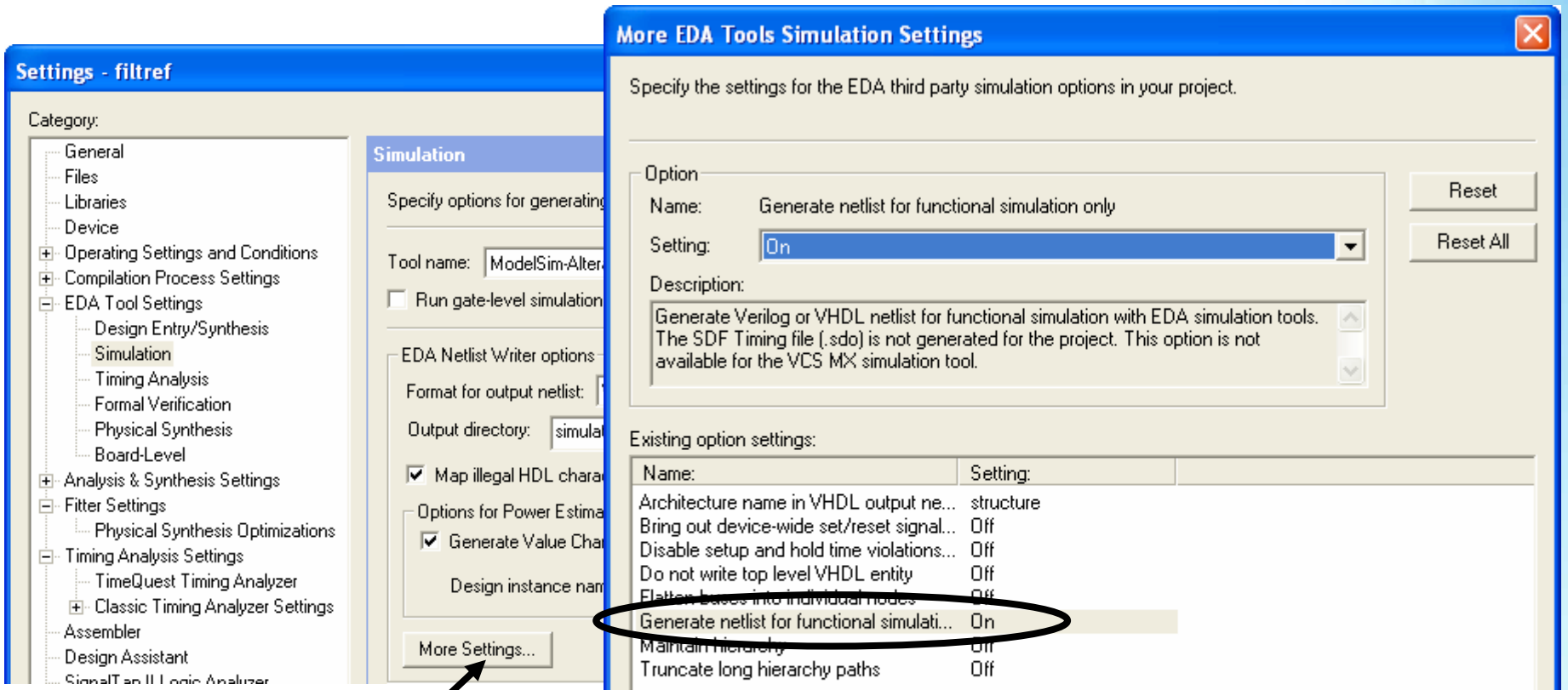
- Use these library names for compiling device-specific simulation models
  - arriagx
  - arriagx_hssi
  - stratixiii
  - stratixii
  - stratixiigx
  - stratixiigx_hssi
  - stratix
  - stratixgx
  - stratixgx_gxb
  - cycloneiii
  - cycloneii
  - cyclone
  - maxii

➢ **For older device families, see Quartus II Handbook, Volume 3, Section 1: "Simulation"**

Post-Synthesis Simulation

- Enable functional simulation netlist
  - No SDO is generated
- Synthesize design
- Run EDA Netlist Writer
- Use gate-level (device-specific) simulations models

Enabling Functional Simulation Netlist



Click on "More Settings" button in Simulation category

Additional Simulation Libary

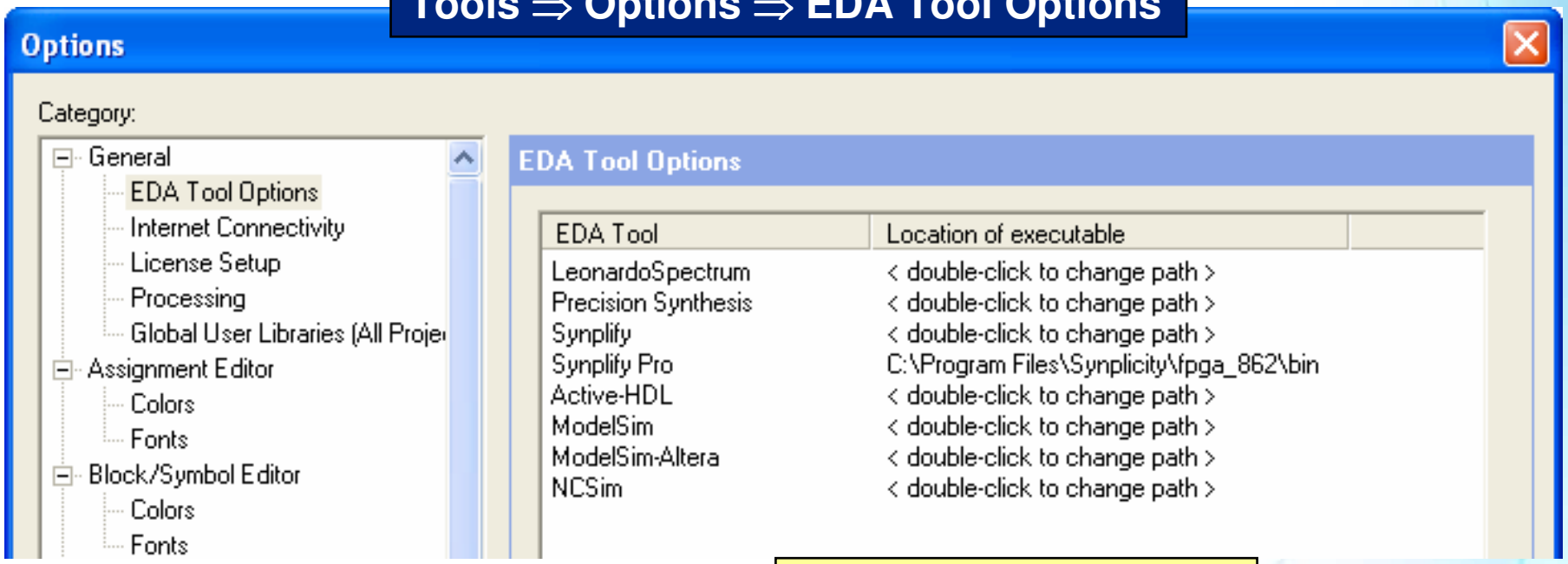
- sgate library
  - Needed for functional simulation of IP
  - Needed for any GX device simulation
  - VHDL
    - sgate.vhd & sgate_pack.vhd
  - Verilog
    - sgate.v

Using NativeLink for Simulation

- Specify path to simulation tool
- Enable NativeLink settings

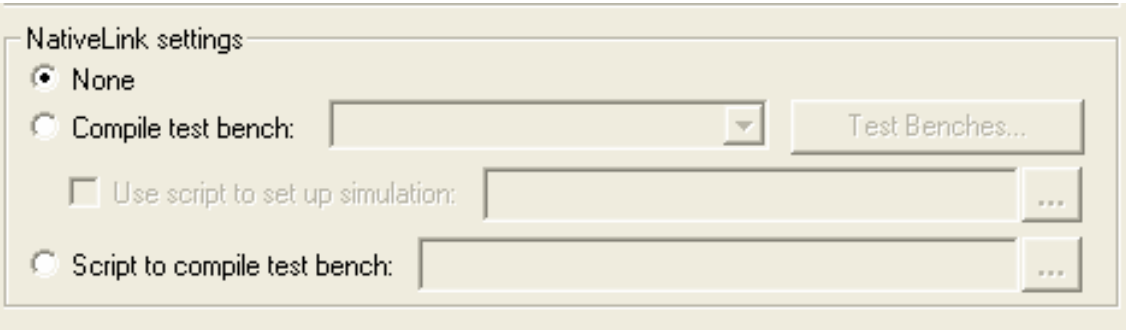
Specify Path to Simulator

Tools ⇒ Options ⇒ EDA Tool Options



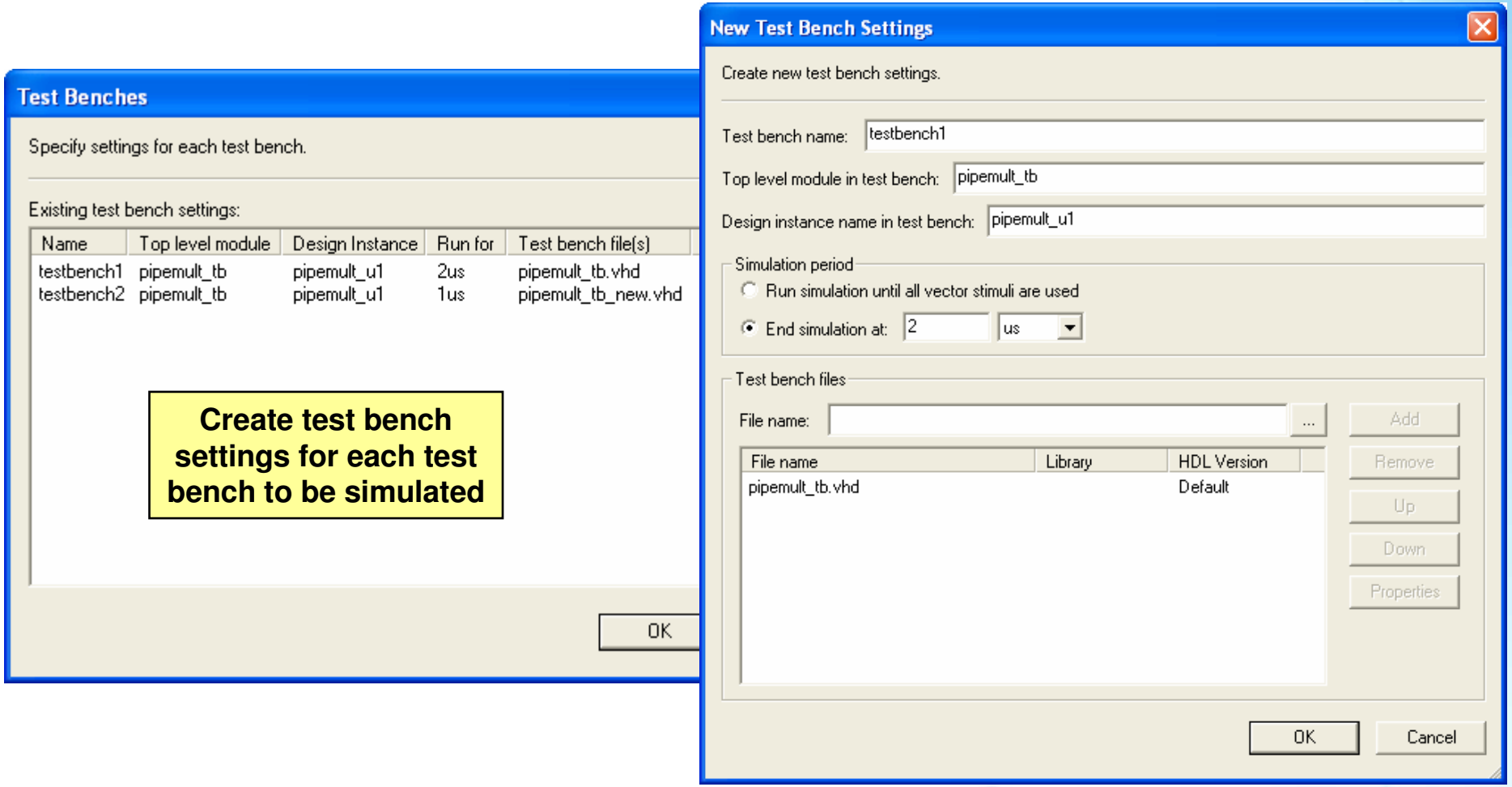
Double-click to specify path to simulation tool executable

Enable NativeLink Settings

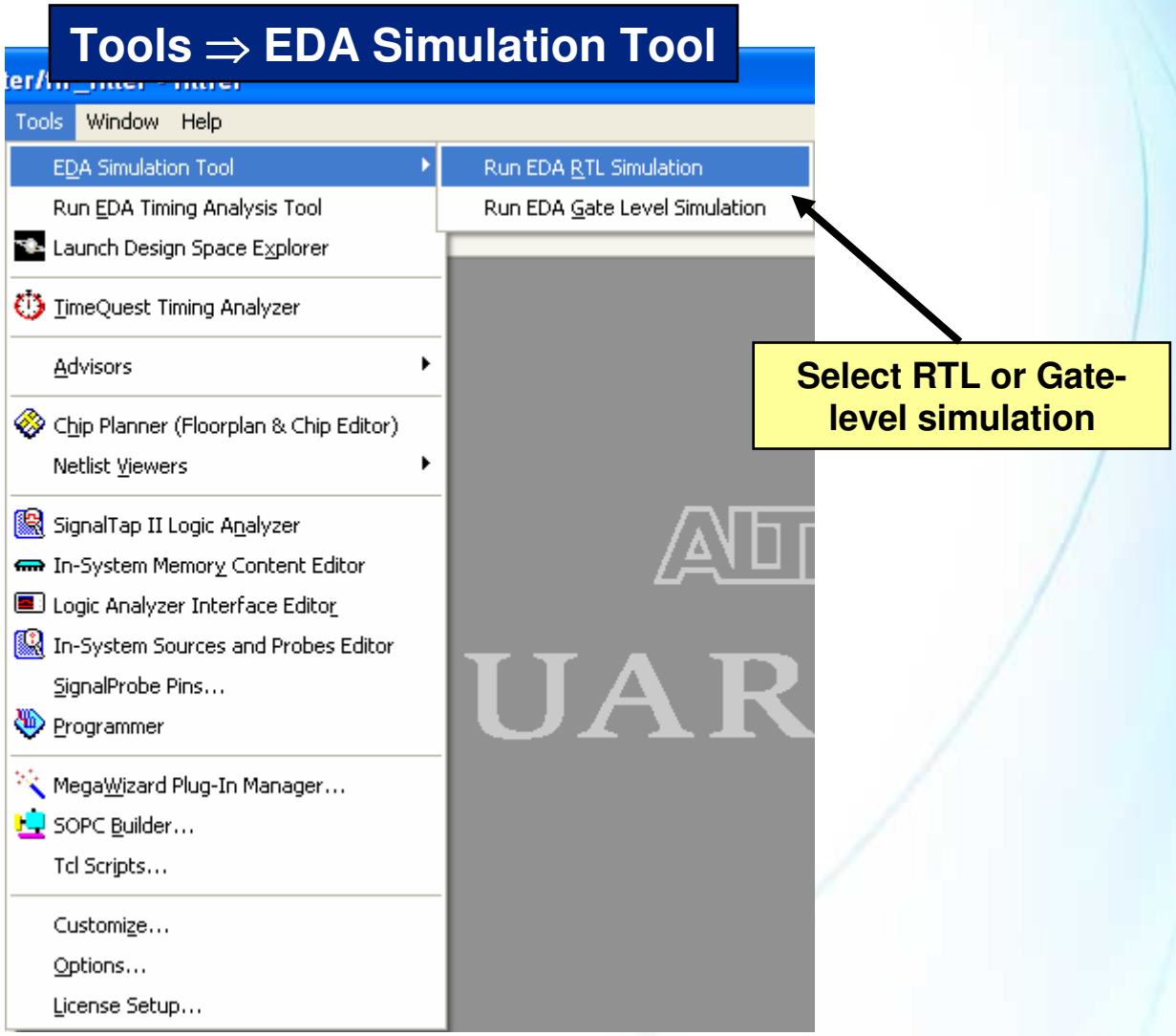


- None
  - NativeLink compiles simulation models & design files
- Compile test bench
  - NativeLink compiles all files (including test bench) and starts simulation
- Use script to compile test bench
  - NativeLink compiles simulation models & design files
  - User specifies script to compile test bench and start simulator

Setting Up Test Benches



Running NativeLink Simulation



Simulation Summary

- Simulating with 3rd-party EDA tools

Simulation Support Resources

- Quartus II Handbook chapters (Volume 3)
  - "Mentor Graphics ModelSim Support"
  - "Synopsys VCS Support"
  - "Cadence NC-Sim Support"
  - "Simulating Altera IP in Third-Party Simulator Tools"

Quartus II Software Design Series: Foundation

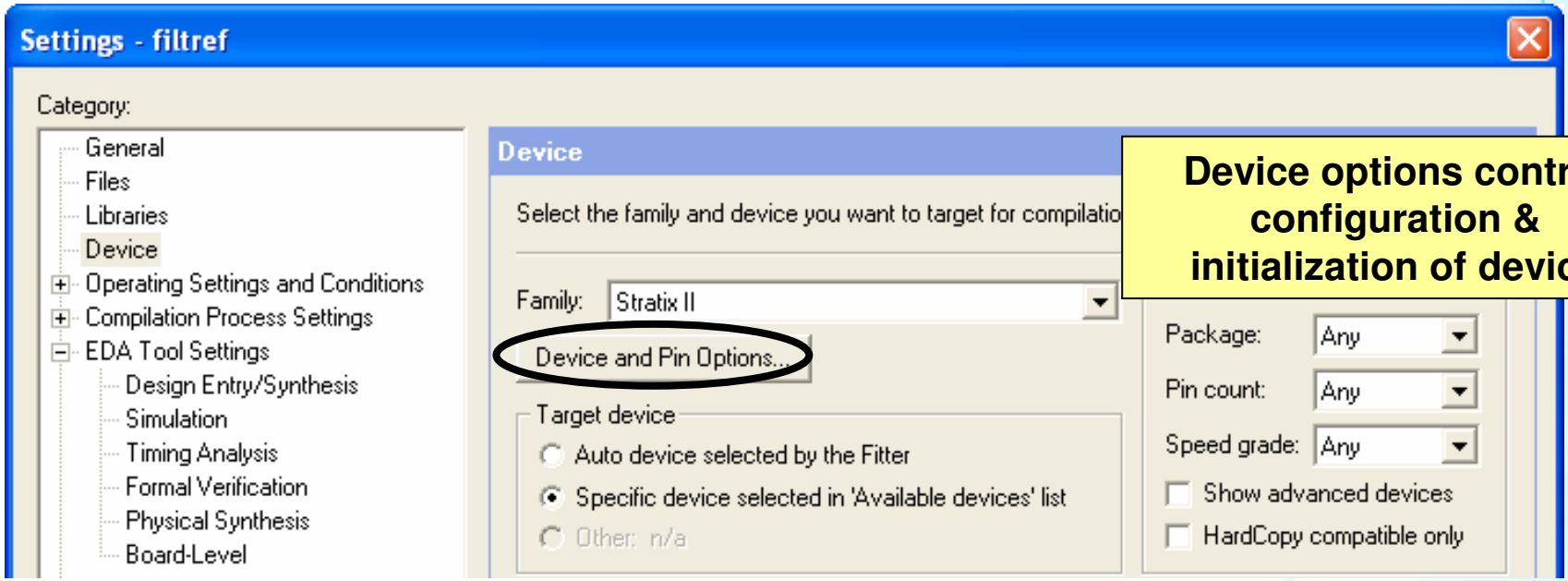
Programming/Configuration

Programming/Configuration

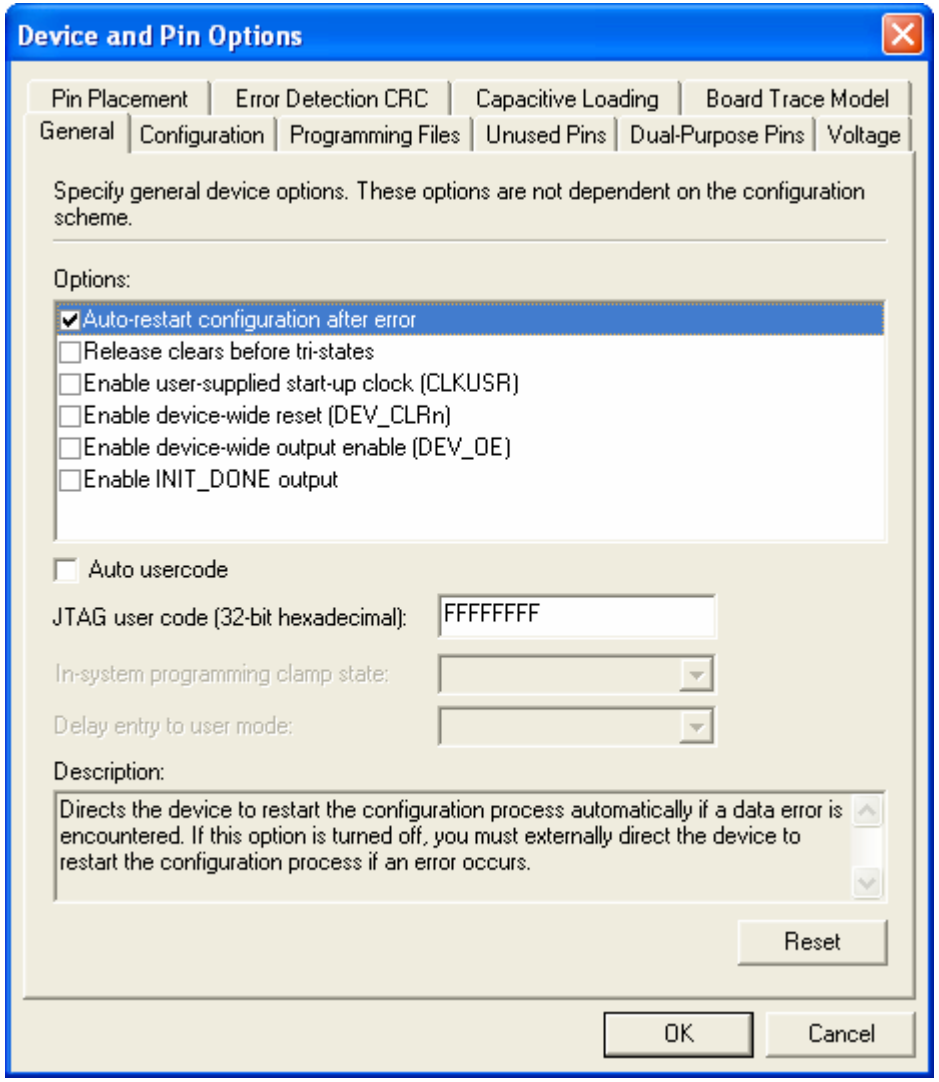
- Setting device options
- Assembler module
- Programmer & chain description file
  - Programming directly with the Quartus II Programmer
- File conversion
  - Creating multi-device programming files

Setting Device Options

- Assignments ⇒ Device ⇒ Device & Pin Options

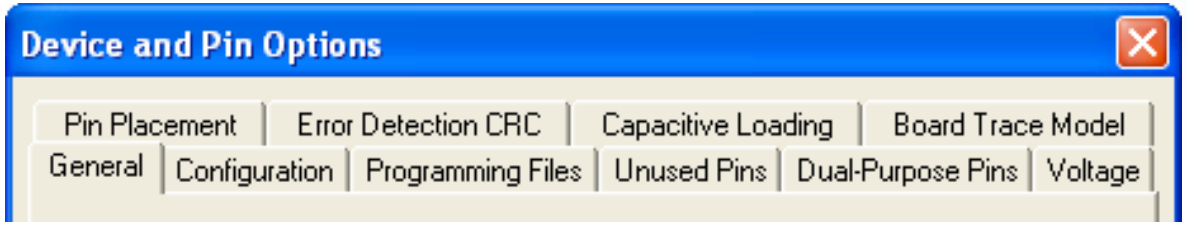


General Tab



- Device options not dependent on configuration scheme (off by default)
  - Enable device-wide clear
  - Enable device-wide output enable
  - Enable initialization done output pin

Other Device & Pin Option Tabs

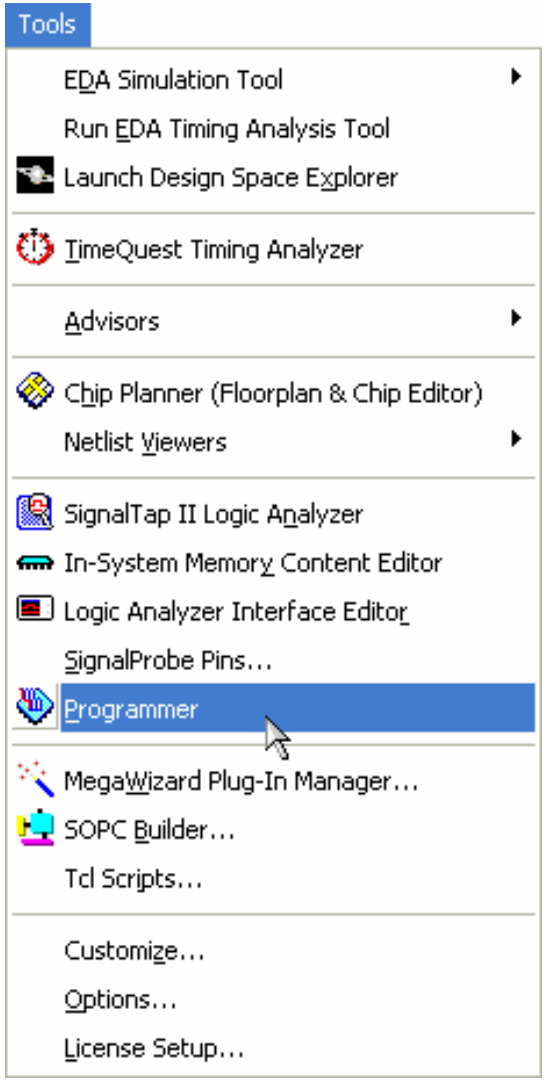


- Configuration
  - Generates correct configuration & programming files every compilation
  - Enables special features of configuration devices
    - Enable programming file compression
    - Set configuration clock frequency
- Programming Files
  - Output files always created
    - POF (programming object file)
    - SOF (SRAM object file)
  - Other selectable output files
    - Jam (jedec stapl)
    - JBC (JAM byte-code)
    - RBF (raw binary file)
    - HEXOUT (intel hex format)
- Unused pins
  - Indicates state of all unused I/O pins after configuration is complete
- Dual-purpose pins
  - Selects usage of dual-purpose pins after configuration is complete
- Error detection CRC
  - Enables internal CRC circuitry & frequency
- Capacitive Loading
  - Sets default capacitive loading for each I/O standard
- Board Trace Model
  - Sets default board trace model characteristics for each I/O standard (Stratix II & III devices only)

Quartus II Assembler Module

- Generates all configuration/programming files
  - As selected in device & pin options dialog box
- Ways to run assembler
  - Full compilation
  - Execute assembler individually
    - Processing menu ⇒ Start ⇒ Start Assembler
    - Generates files without full compilation
      - Switching configuration devices
      - Enabling/disabling configuration device feature
  - Scripting

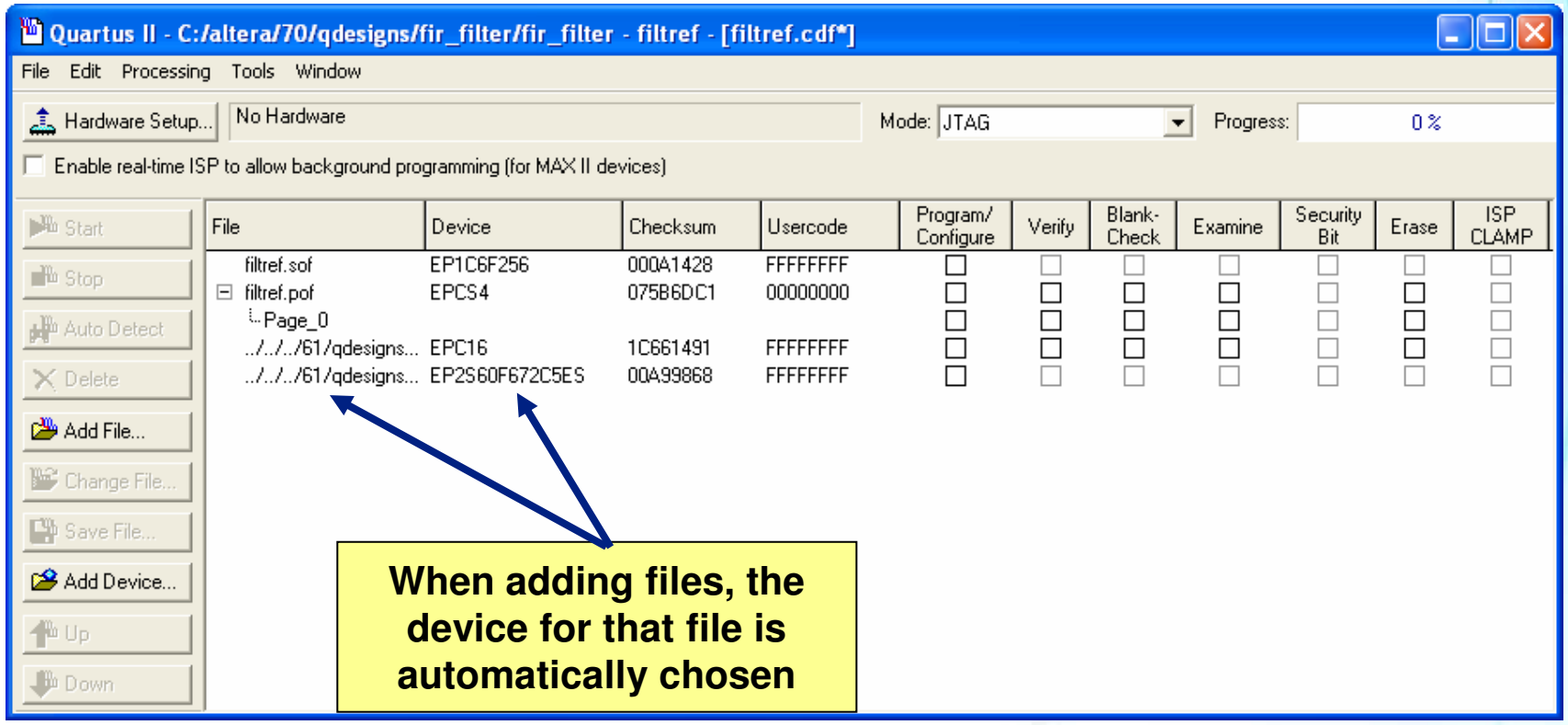
Open Programmer



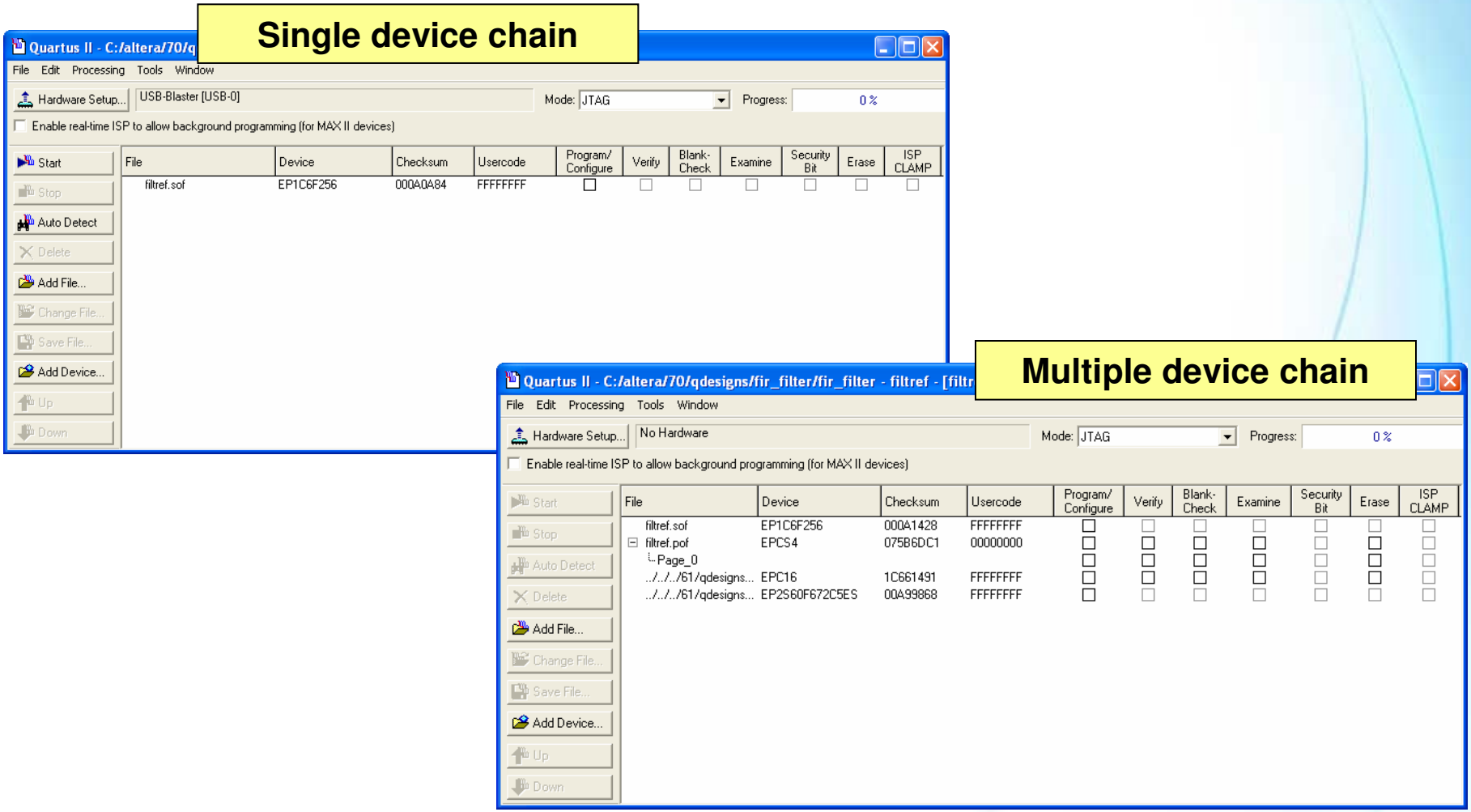
- Enables device programming
  - USB-Blaster™ cable
  - ByteBlaster™ II or ByteBlasterMV™ cables
  - Masterblaster™ cable
  - APU (Altera programming unit)
- Opens chain description file (.CDF)
  - Stores device programming chain information

CDF File

- Lists devices & files for programming or configuration
- Programs/configures in top-to-bottom order

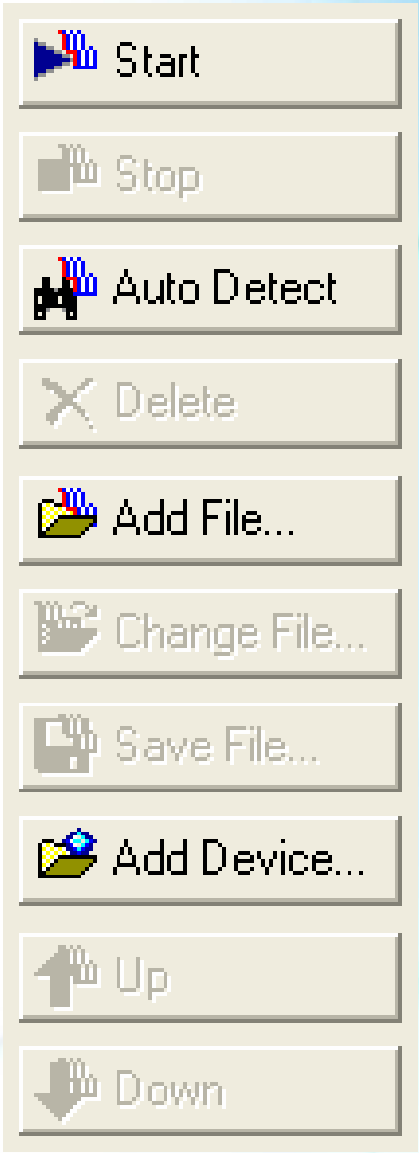


Example CDF Files



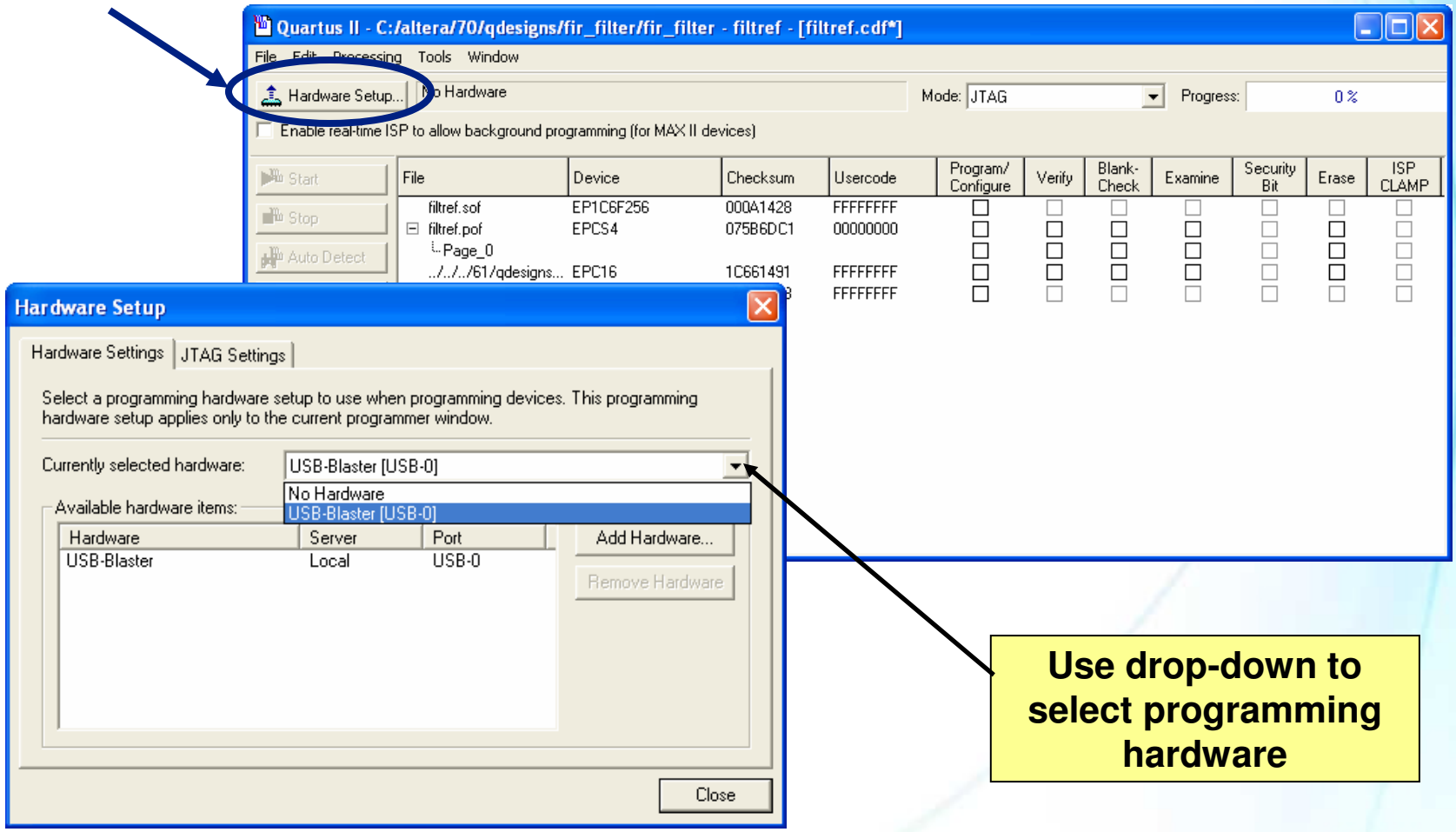
Programmer Toolbar

- Start programming
- Auto detect devices in JTAG chain
- Add/remove/change devices in chain
- Add/remove/changes files in chain
- Change order of files in chain
- Setup programming hardware



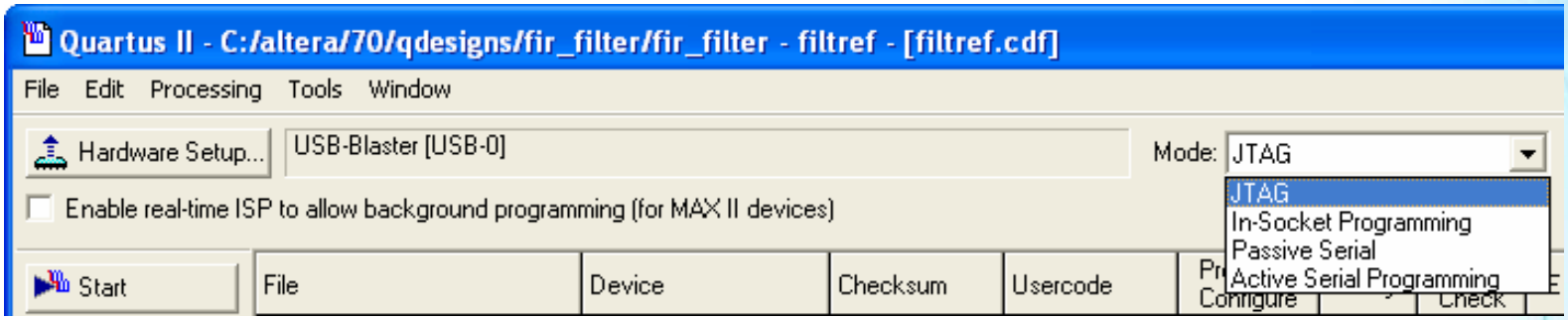
Setting Up Programming Hardware

Click on the Hardware Setup button



Use drop-down to select programming hardware

Chain Programming Modes

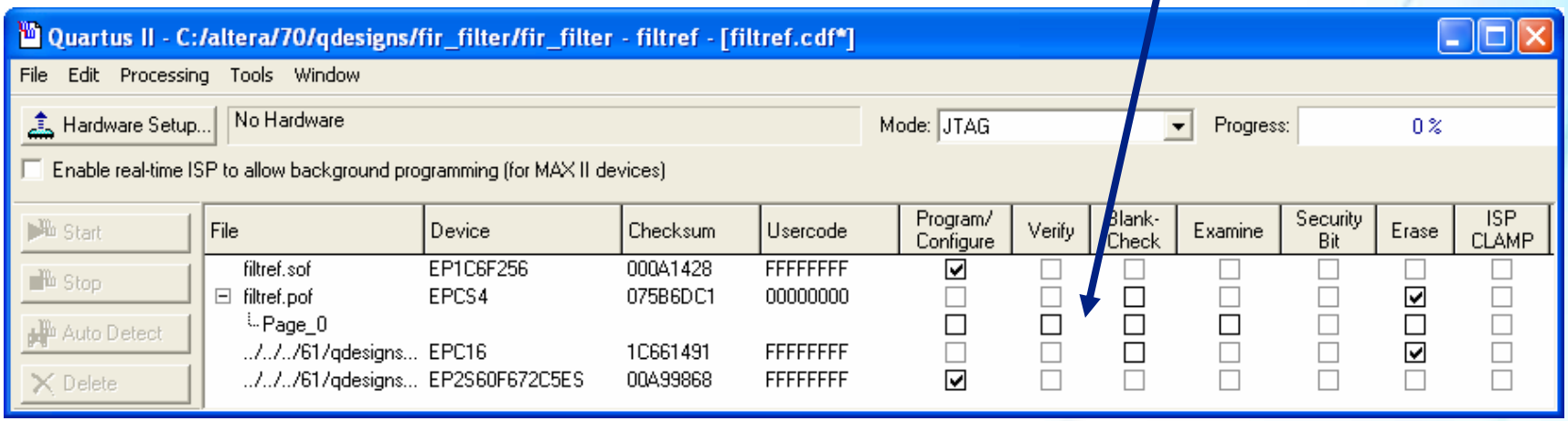


- JTAG
  - JTAG chain consisting of Altera & non-Altera devices
- Passive serial
  - Altera FPGAs only
- Active serial
  - Altera serial configuration devices
- In-socket programming
  - CPLDs & configuration devices in APU

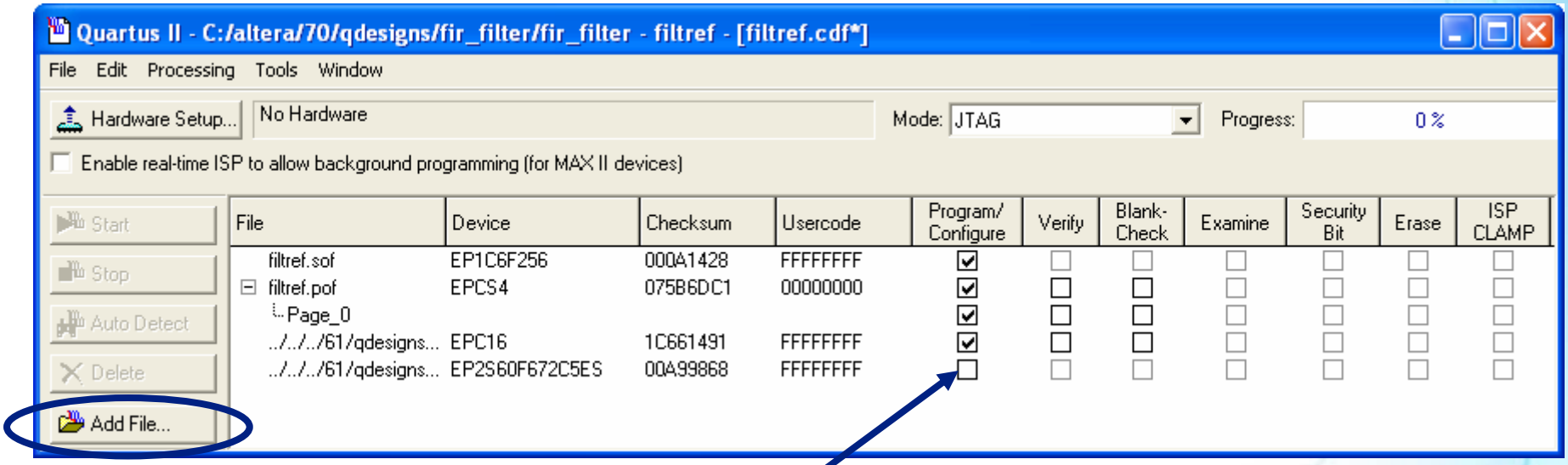
Programming Options

- Program/Configure
  - Applies to all devices
- Verify, Blank-Check, Examine & Erase
  - Configuration devices
  - MAX II, MAX 7000 & MAX 3000
- Security Bit & ISP Clamp
  - MAX II, MAX 7000 & MAX 3000

Check the appropriate boxes to perform actions when programming starts

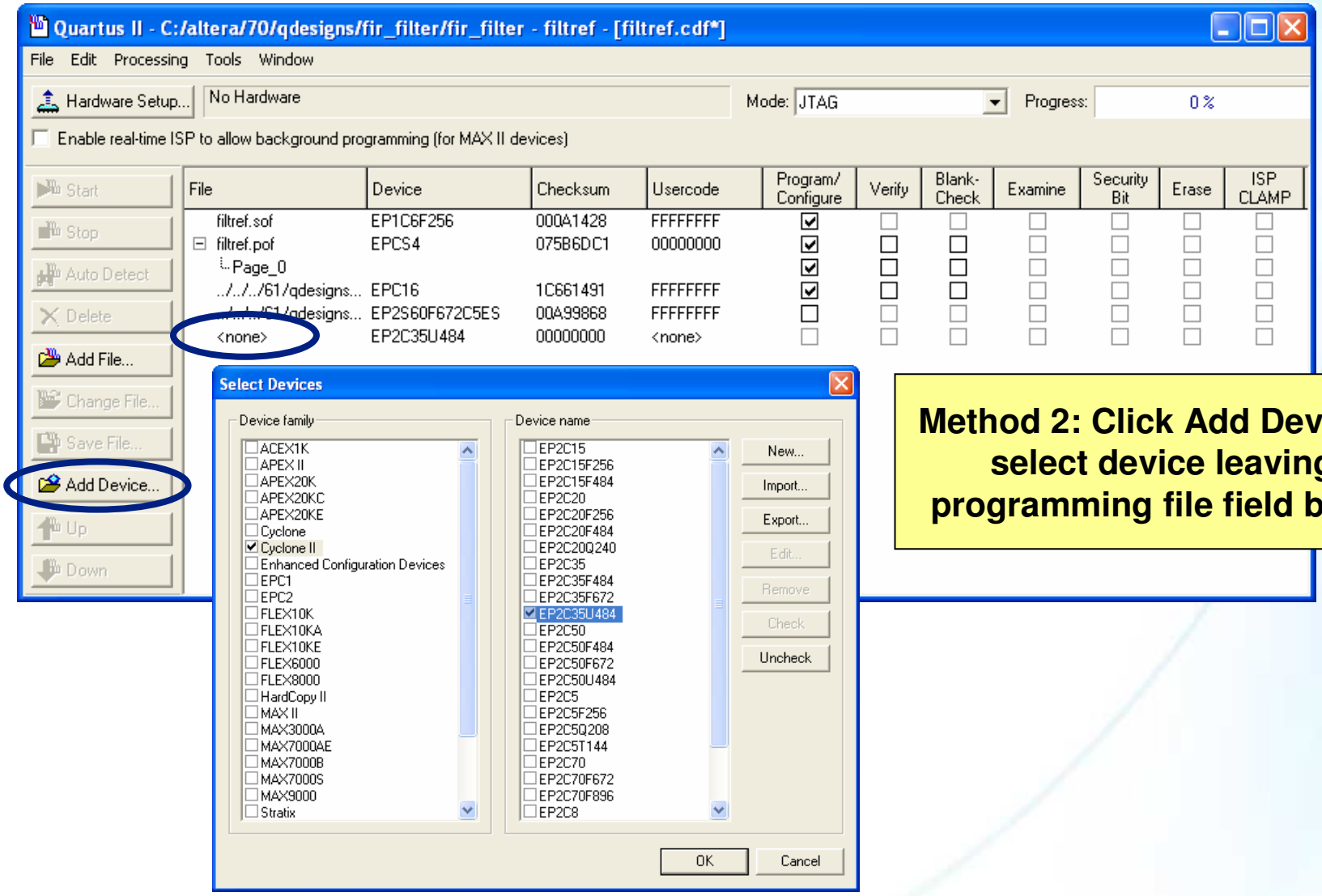


Bypassing Devices In JTAG Chain (1)



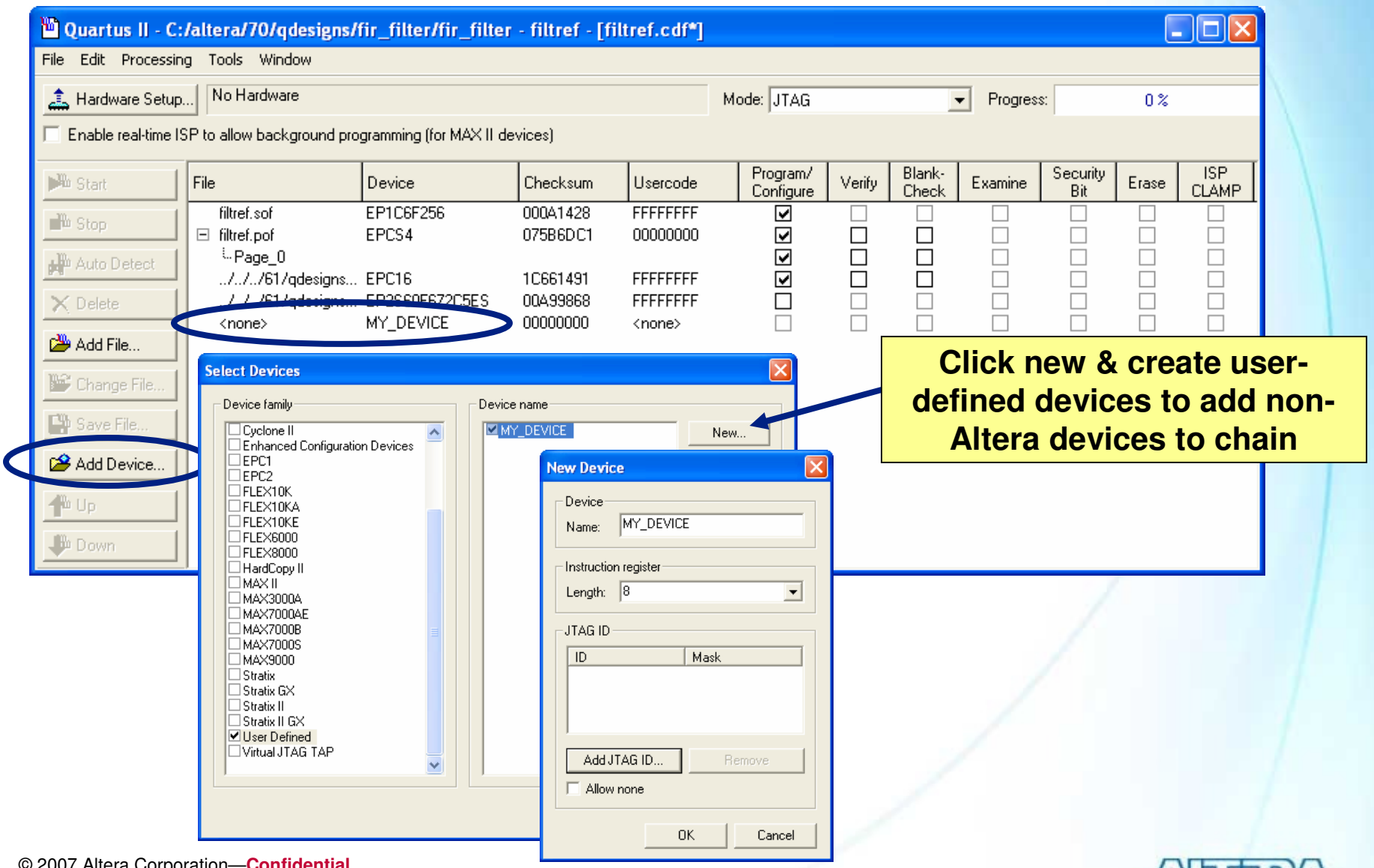
Method 1: Add programming file & leave Program/Configure box unchecked

Bypassing Devices In JTAG Chain (2)



Method 2: Click Add Device & select device leaving programming file field blank

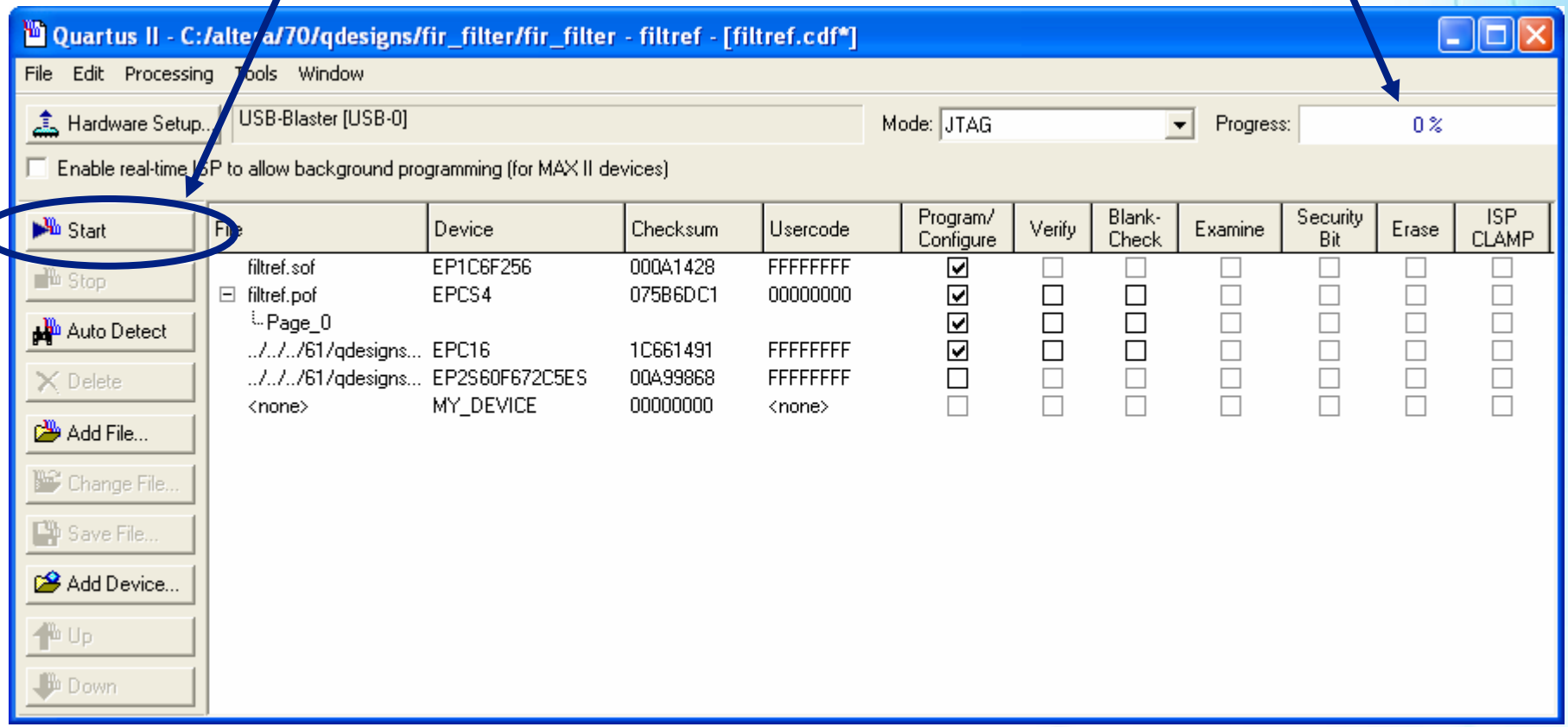
Adding Non-Altera Device to Chain



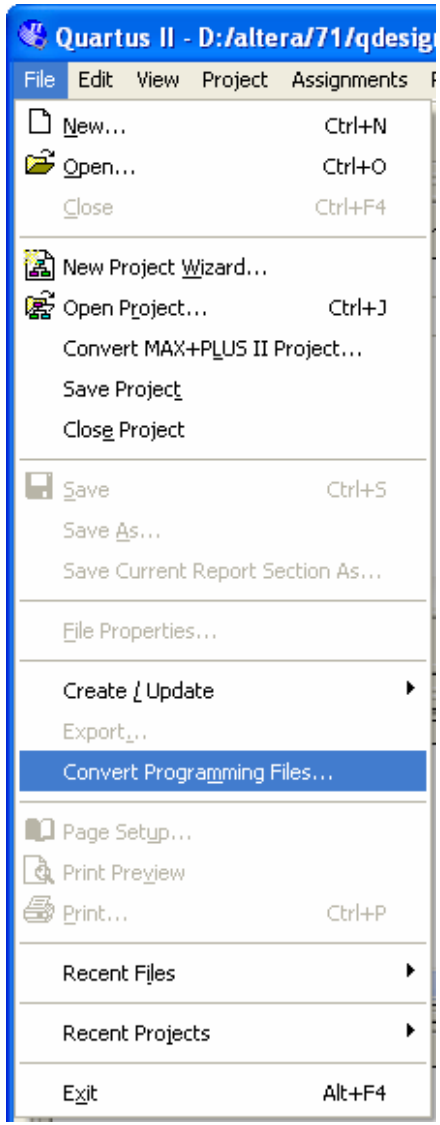
Starting the Programmer

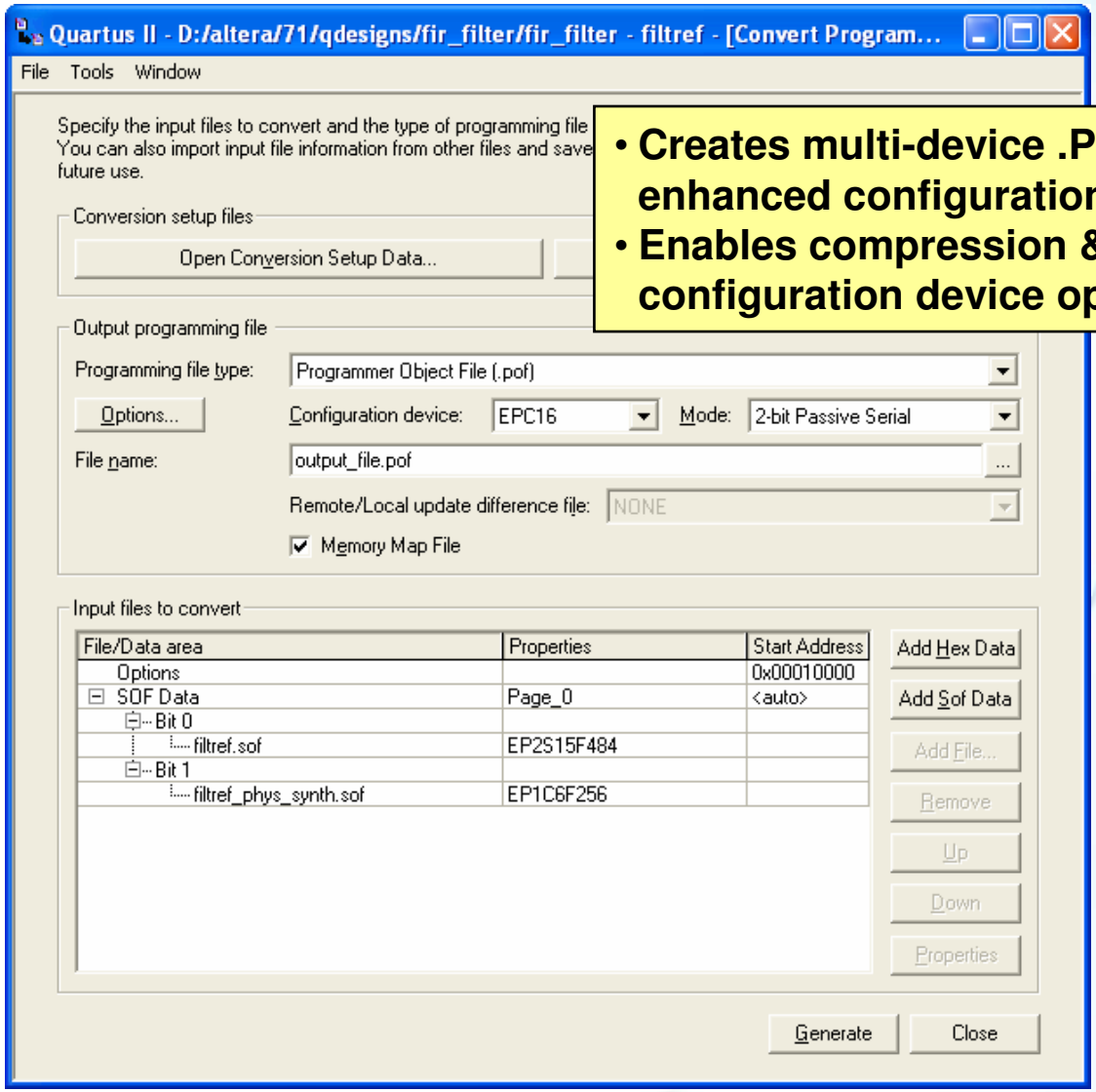
Click Start

Progress field shows percentage of programming completion



Converting SOF Programming Files





- **Creates multi-device .POF for enhanced configuration devices**
- **Enables compression & other configuration device options**

Exercise 7 Demonstration

Programming/Configuration Summary

- Setting device options
- Generating programming files
- Programming device or devices in chain
- Converting programming files

Programming Support Resources

- Programming Center
- Configuration Center

Class Summary

- Design entry techniques
- Project creation
- Compiler settings & assignment editor
- Timing analysis
- Simulation
- Programming/configuration

Learn More Through Technical Training

Instructor-Led Training

With Altera's instructor-led training courses, you can:

- Listen to a lecture from an Altera technical training engineer (instructor)
- Complete hands-on exercises with guidance from an Altera instructor
- Ask questions & receive real-time answers from an Altera instructor
- Each instructor-led class is one or two days in length (8 working hours per day).

Online Training

With Altera's online training courses, you can:

- Take a course at any time that is convenient for you
- Take a course from the comfort of your home or office (no need to travel as with instructor-led courses)

Each online course will take approximate one to three hours to complete.

http://www.altera.com/training

View training class schedule & register for a class

Advanced Quartus II Courses

- Quartus II Software Design Series: Verification
  - Timing analysis
    - Thorough investigation of performing timing analysis on an Altera device with TimeQuest
  - Power analysis
  - Debugging solutions
    - SignalProbe incremental routing
    - Logic Analyzer Interface
    - In-System Memory Content Editor
    - In-System Sources & Probes
    - Chip Planner & Resource Property Editor
    - SignalTap II Embedded Logic Analyzer
- Quartus II Software Design Series: Optimization
  - Incremental Compilation
  - Quartus II optimization features & techniques

Altera Technical Support

- Reference Quartus II software on-line help
- Quartus II Handbook
- Consult Altera applications (factory applications engineers)
  - MySupport: http://www.altera.com/mysupport
  - Hotline: (800) 800-EPLD (7:00 a.m. - 5:00 p.m. PST)
- Field applications engineers: contact your local Altera sales office
- Receive literature by mail: (888) 3-ALTERA
- FTP: ftp.altera.com
- World-wide web: http://www.altera.com
  - Use solutions to search for answers to technical problems
  - View design examples

Give us your feedback

- When you registered for this training you received a confirmation email
- Please click on the link in the email to complete a short survey
- Your feedback is important to help us improve future trainings!

Thank you!

Quartus II Software Design Series: Foundation

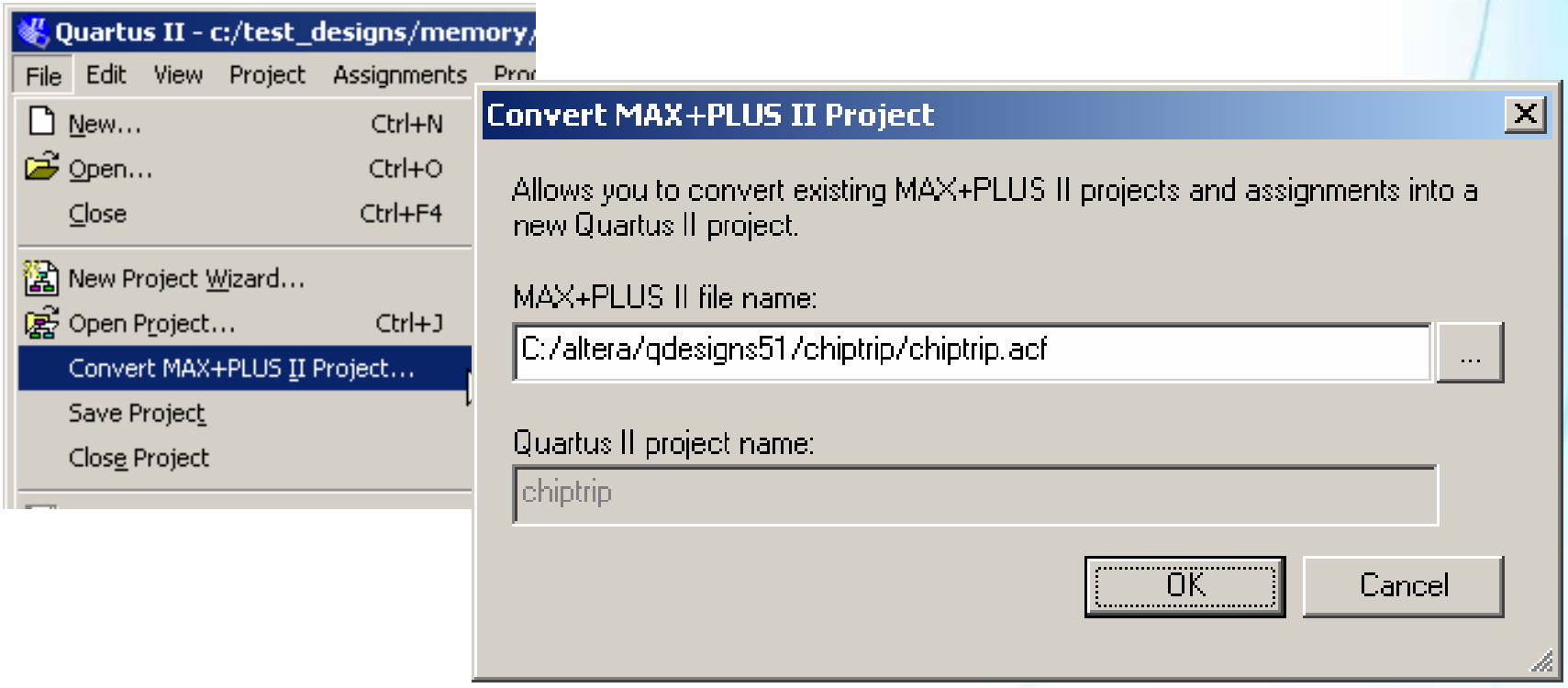
Appendix

Appendix

- Converting MAX+PLUS II designs to Quartus II
- Schematic design entry
- QSF Notes
- More fitter settings
- Power Optimizations
- Early I/O planning tasks
- TimeQuest “golden” SDC file tip
- Simulating in Quartus II

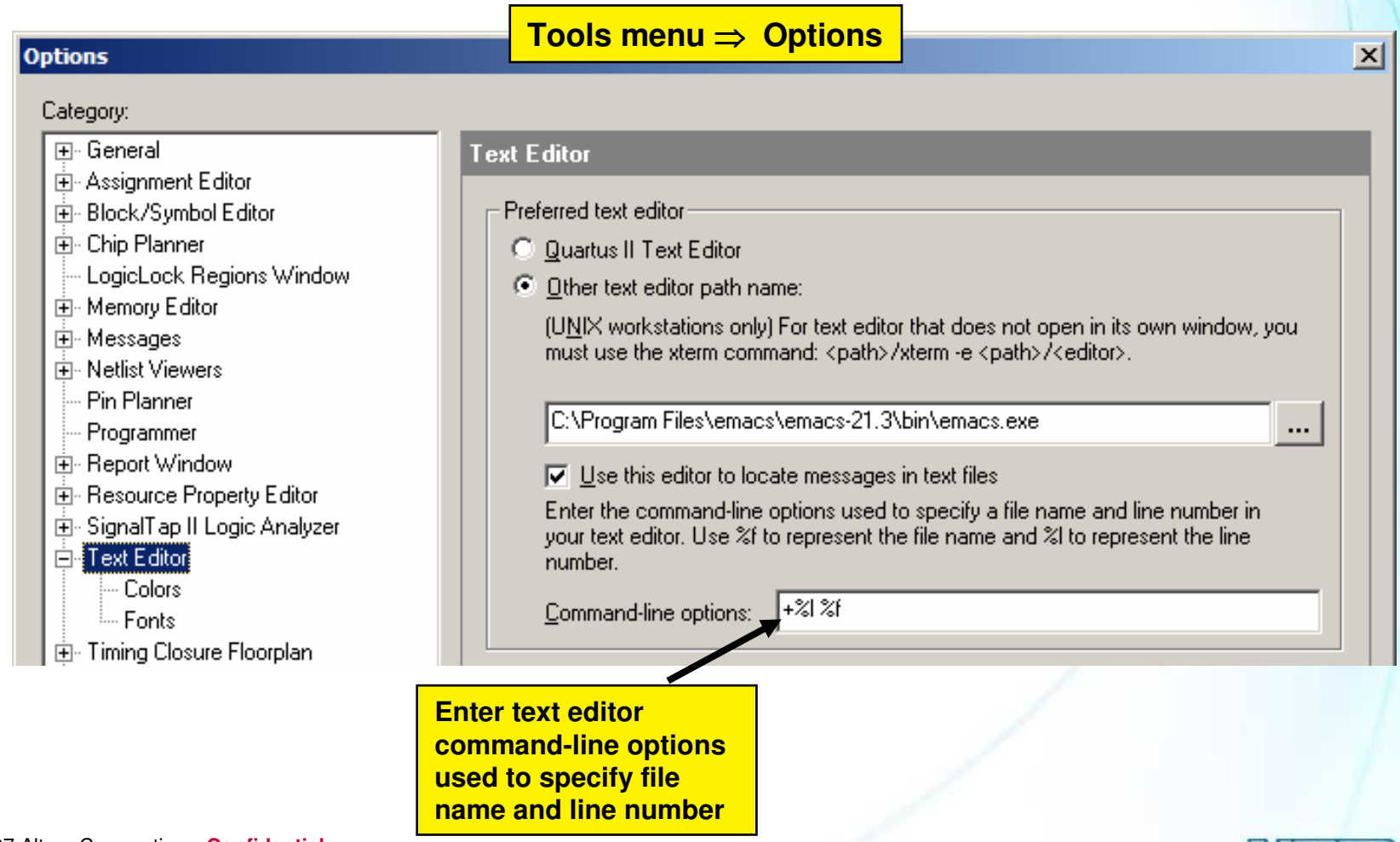
MAX+PLUS II to Quartus II

- Convert MAX+PLUS II projects into Quartus II projects
- Assignments automatically translated



Using Own Text Editor

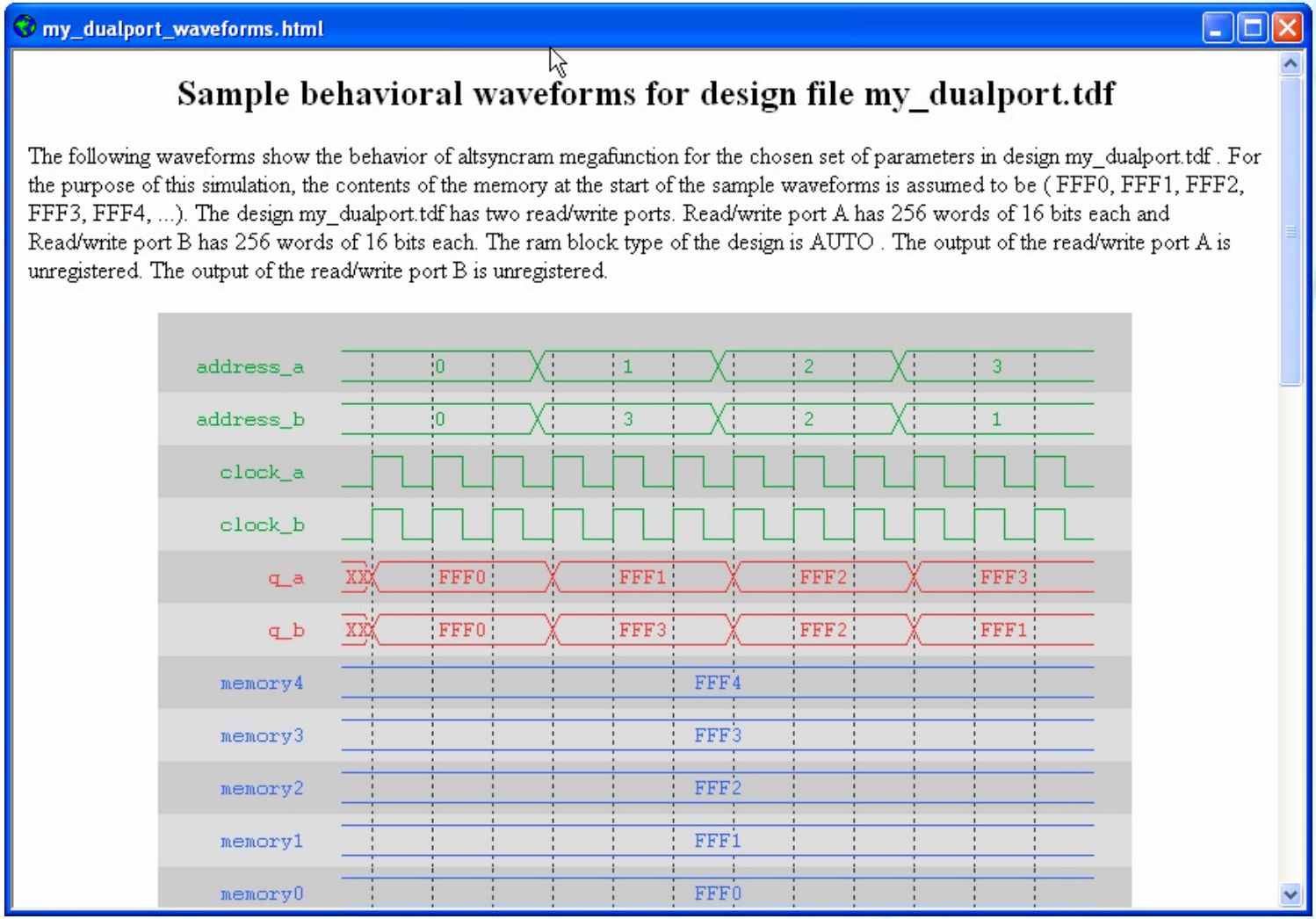
- Enter path to preferred text editor executable



Behavioral Waveforms

- HTML file generated by MegaWizard
- Description of megafunction functionality
  - Reviews selected parameters
  - Describes read & write operations
- Supported megafunctions
  - Subset of memory
  - Subset of arithmetic
  - PLL

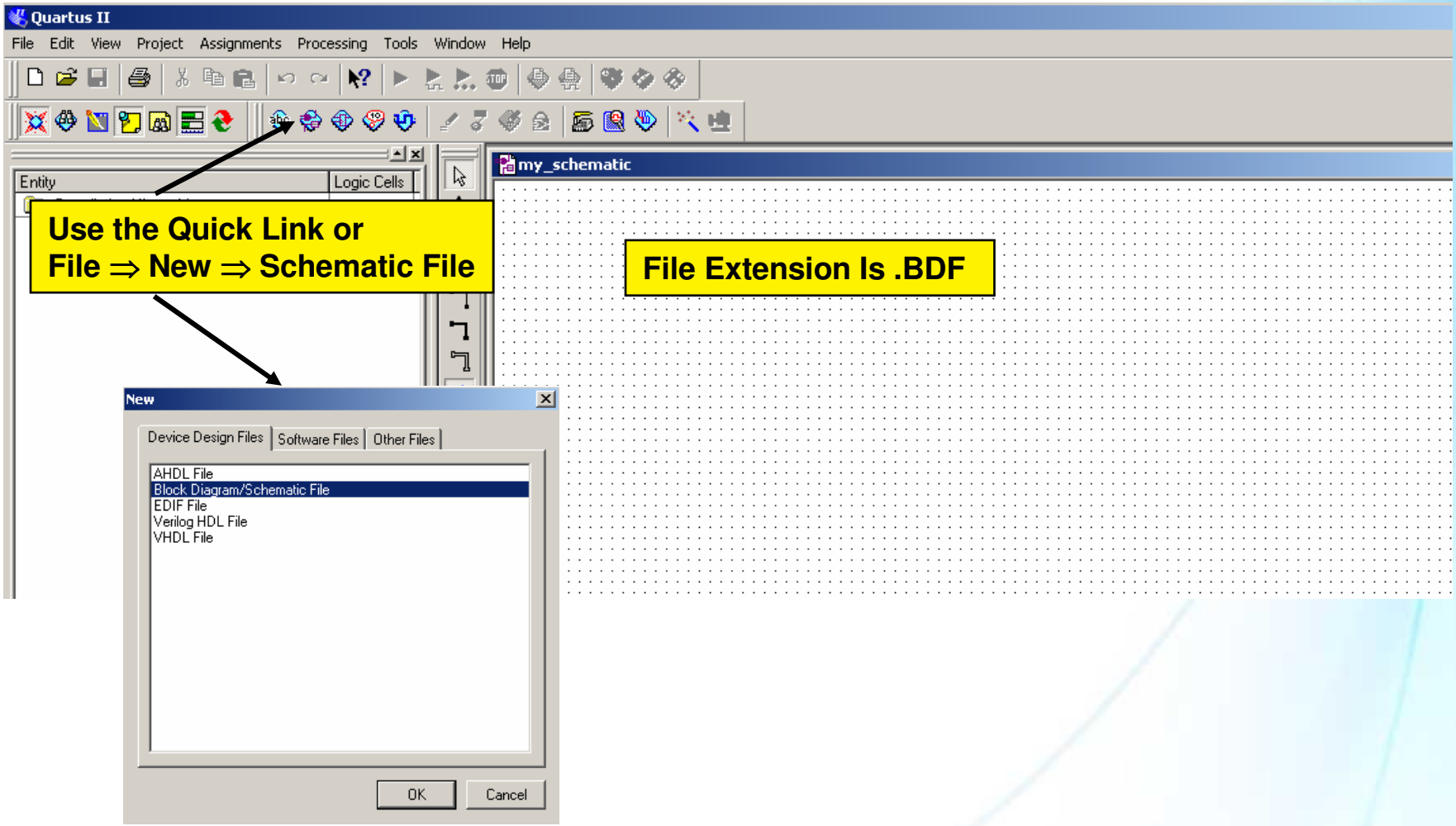
Example Waveform



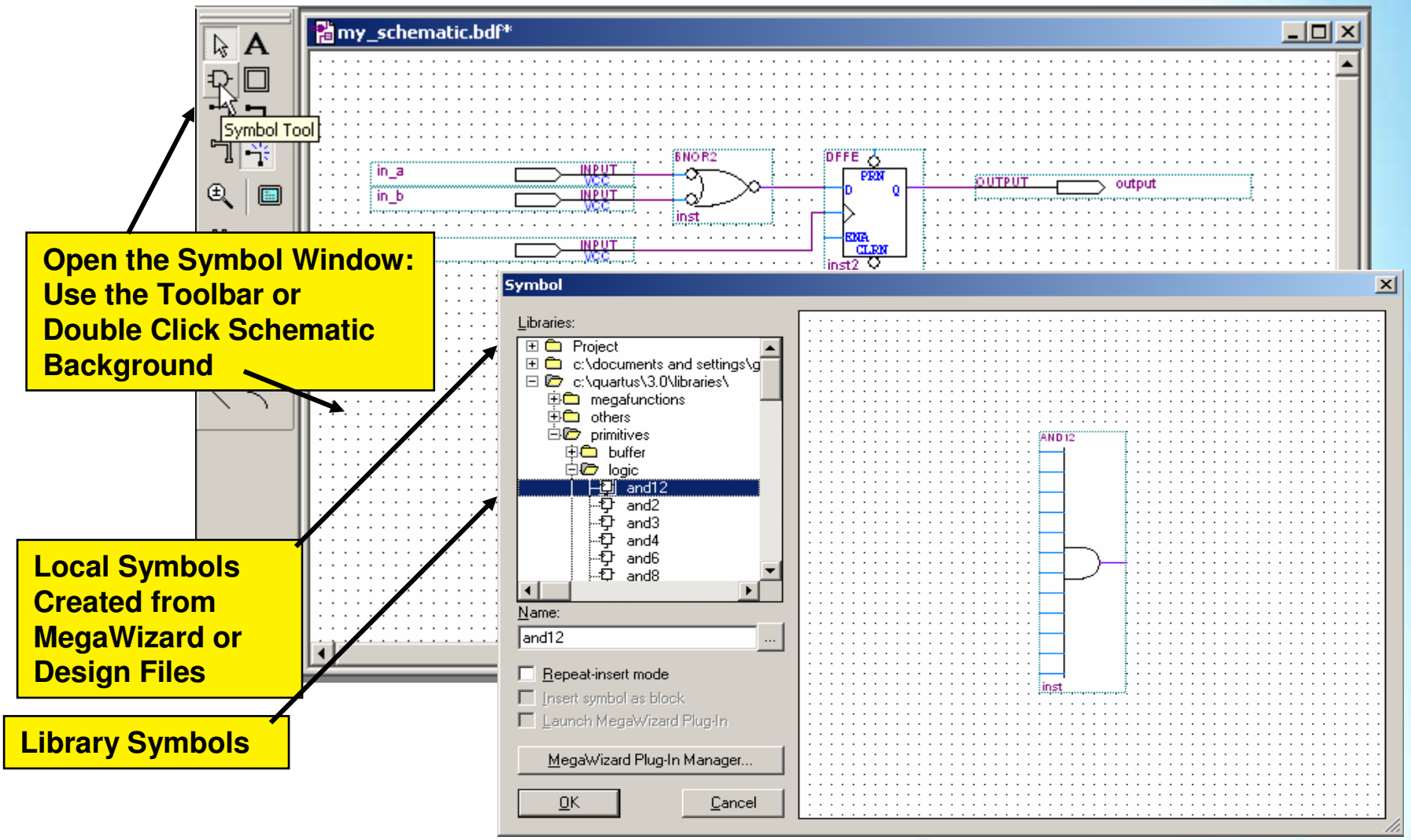
Schematic Design Entry

- Full-featured schematic design capability
- Schematic design creation
  - Draw schematics using library functions (blocks)
    - Gates, flip-flops, pins & other primitives
    - Altera megafunctions & LPMs
  - Create symbols for Verilog, VHDL, or AHDL design files
  - Connect all blocks using wires & busses

Create Schematic

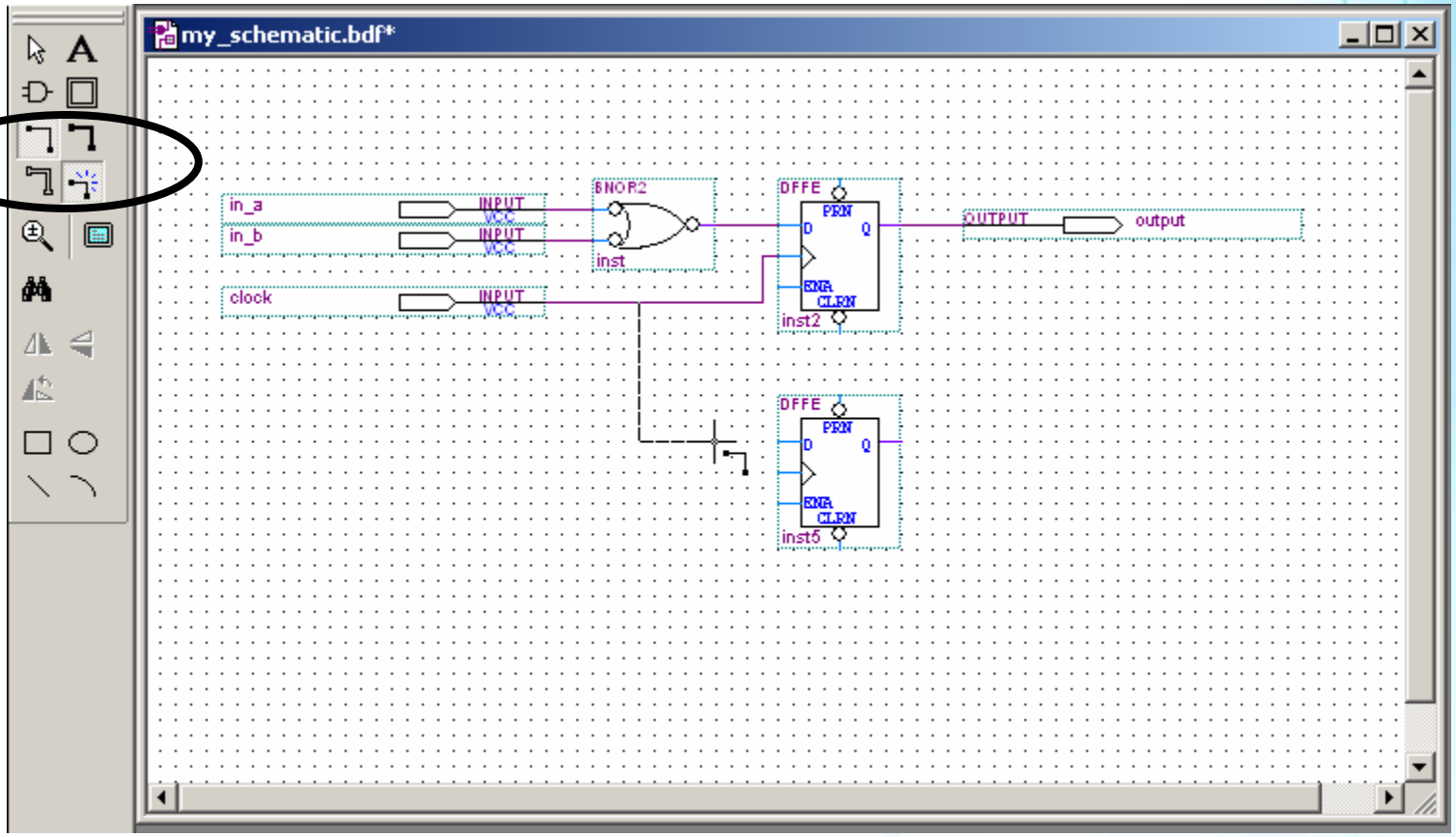


Insert Symbols

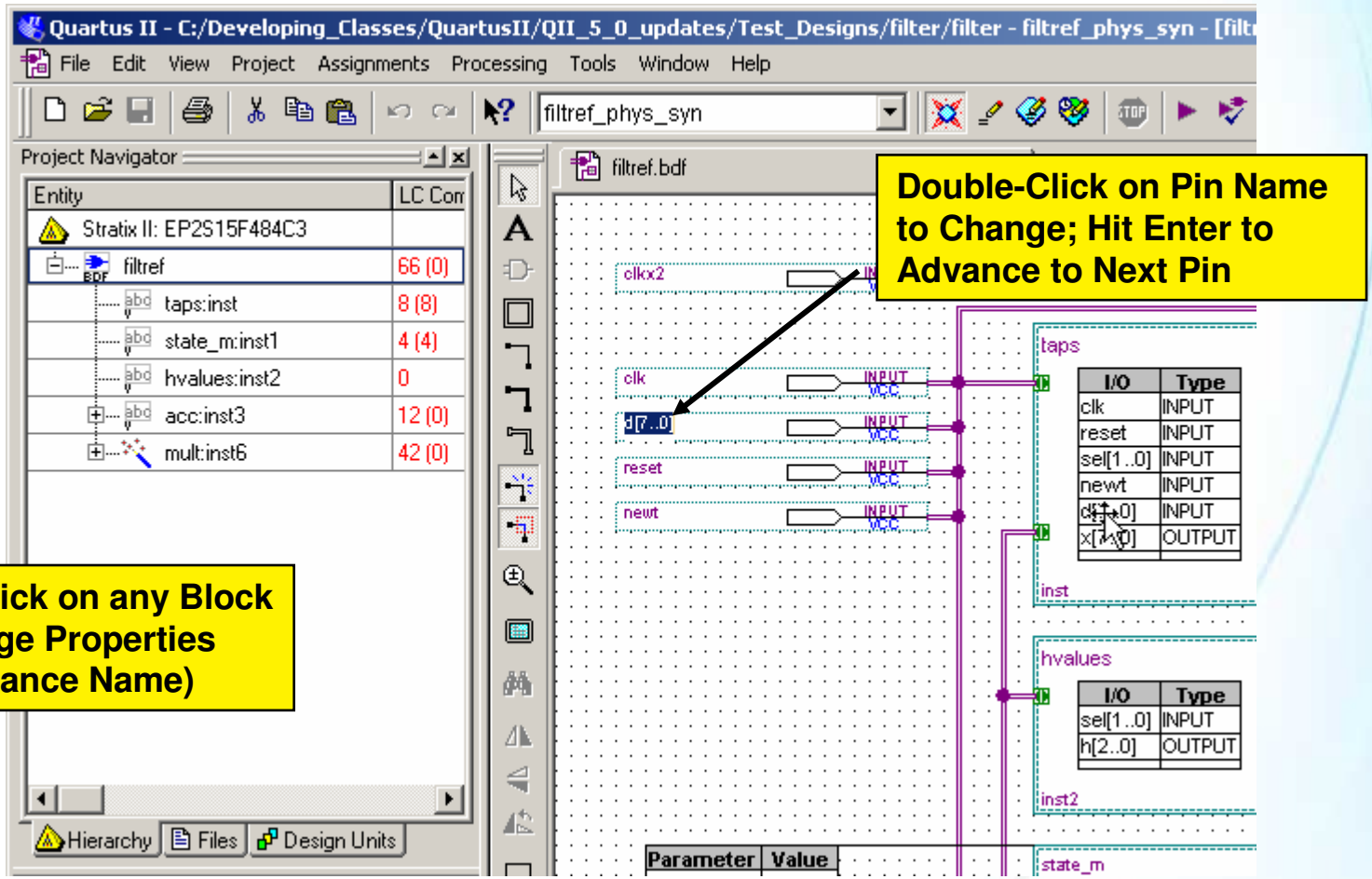


Connect Wires & Buses

Draw Wires, Buses, or Conduit

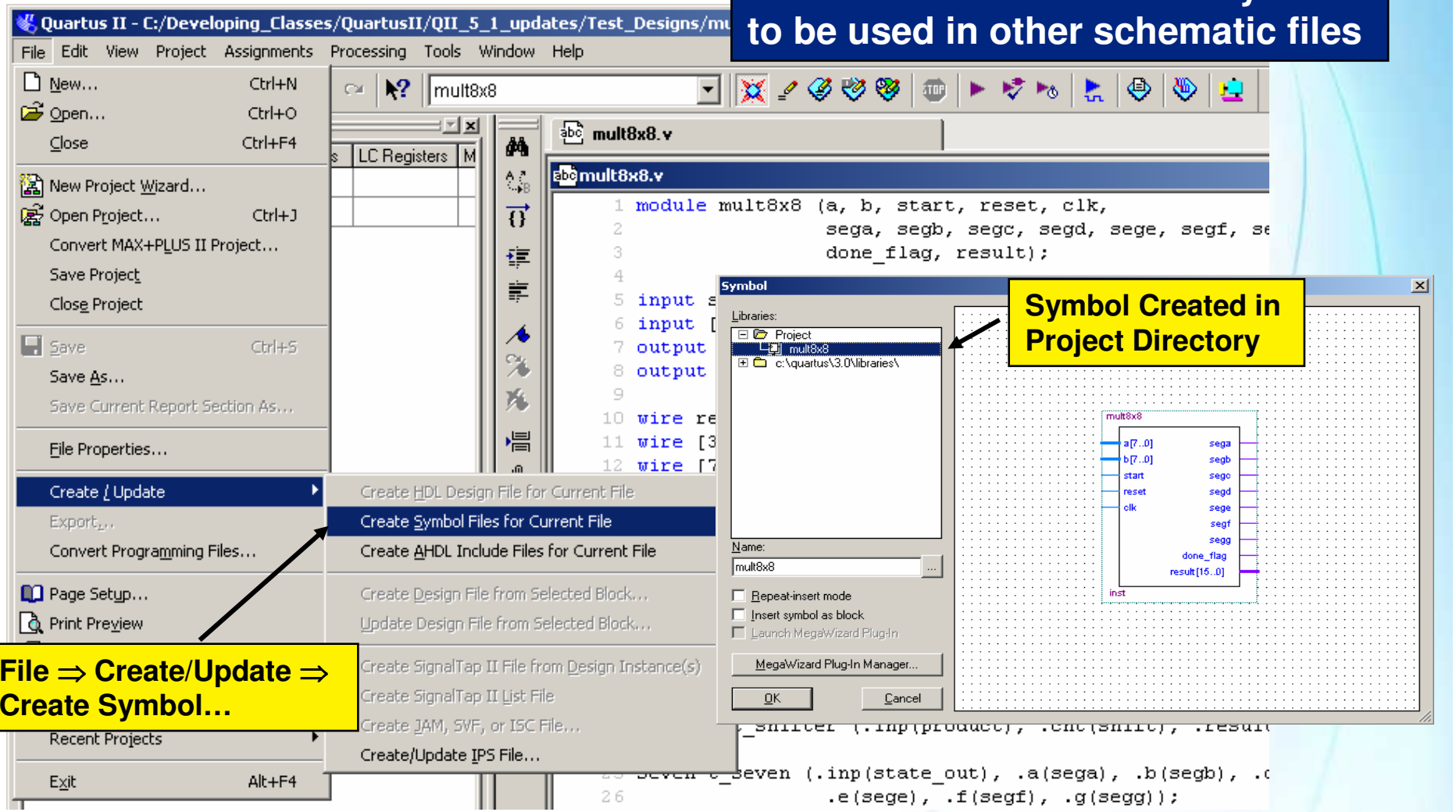


Change Names & Properties

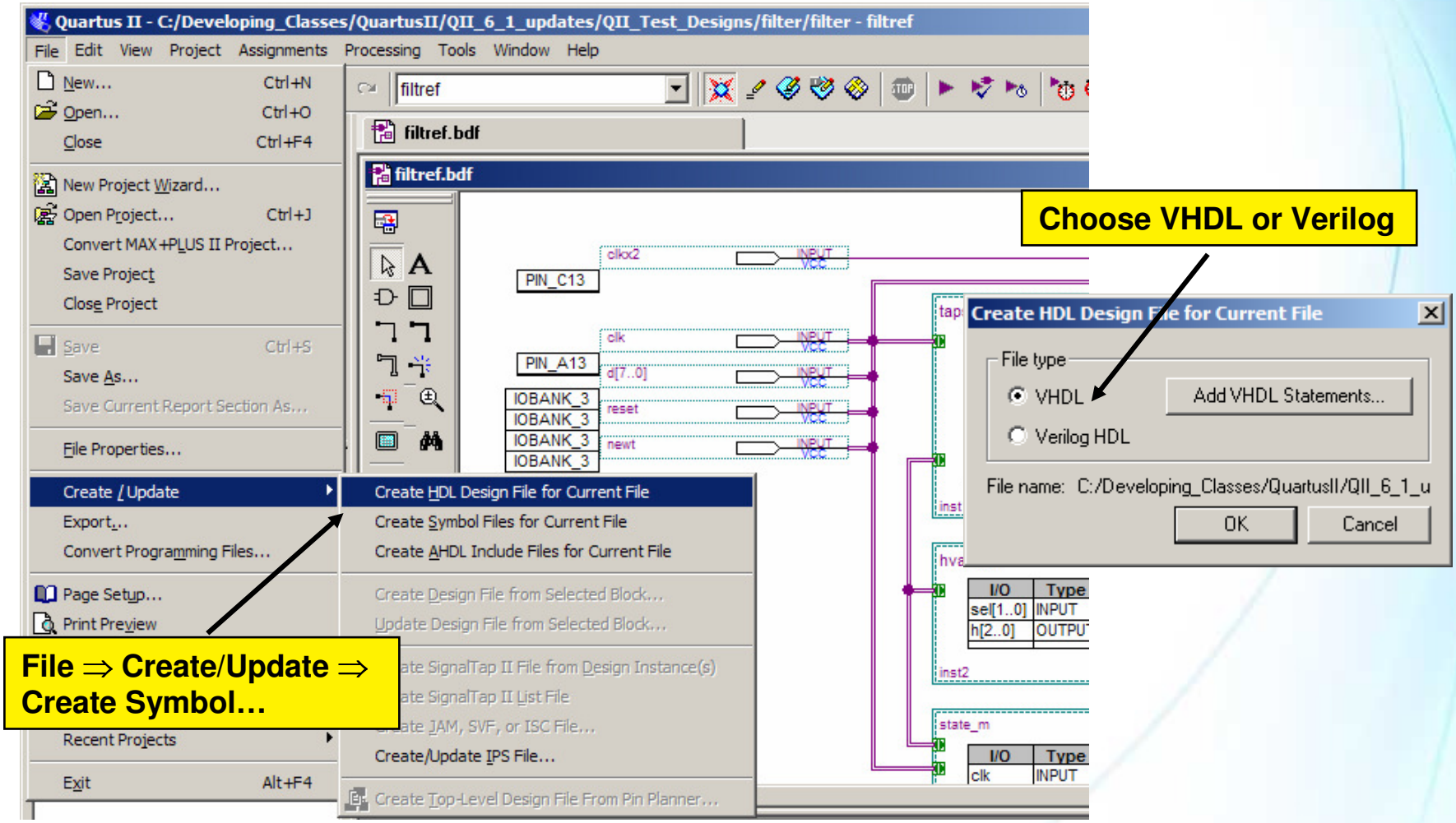


Create Symbols

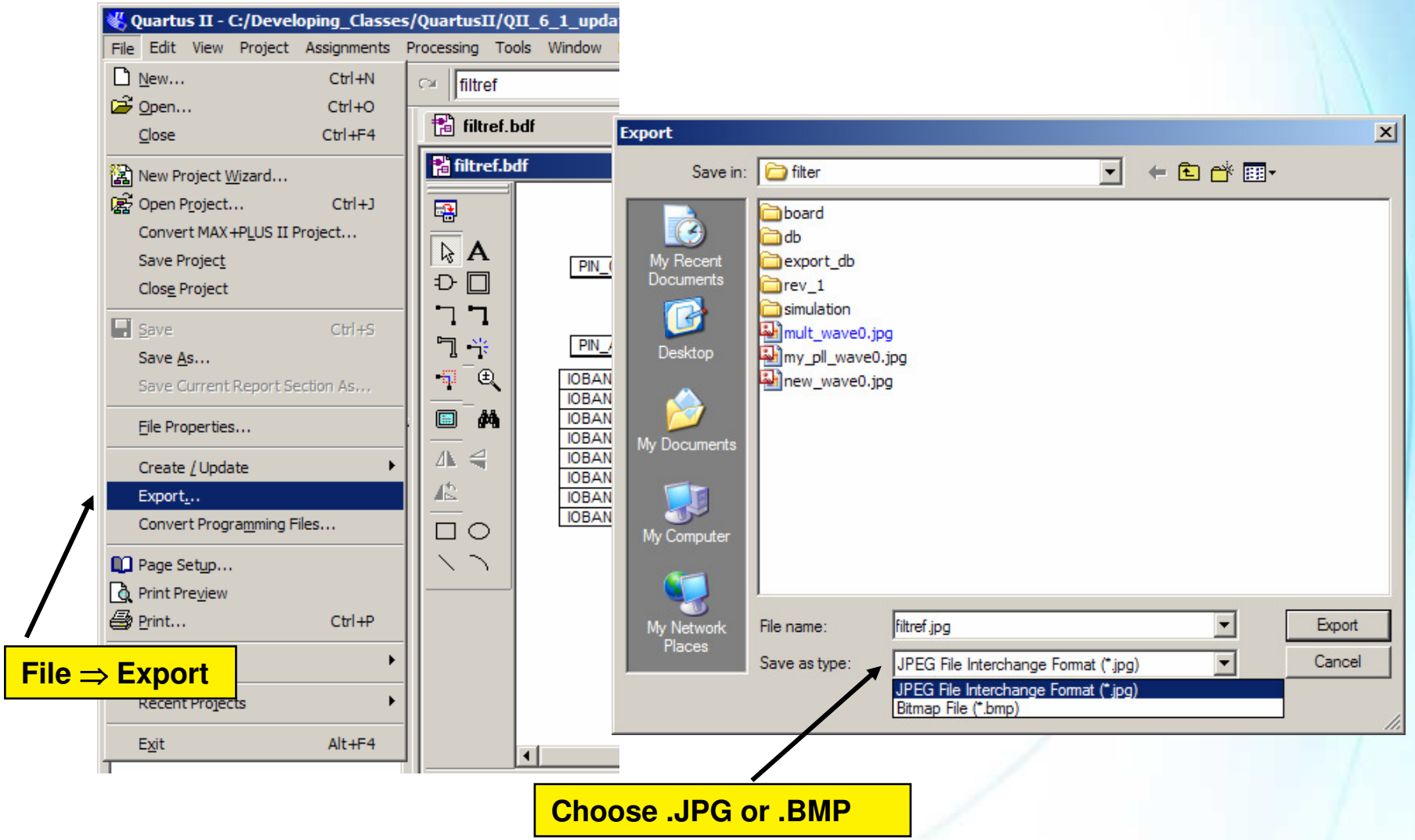
Converted schematic to a Symbol to be used in other schematic files



Convert BDF for HDL



Convert BDF to Image File

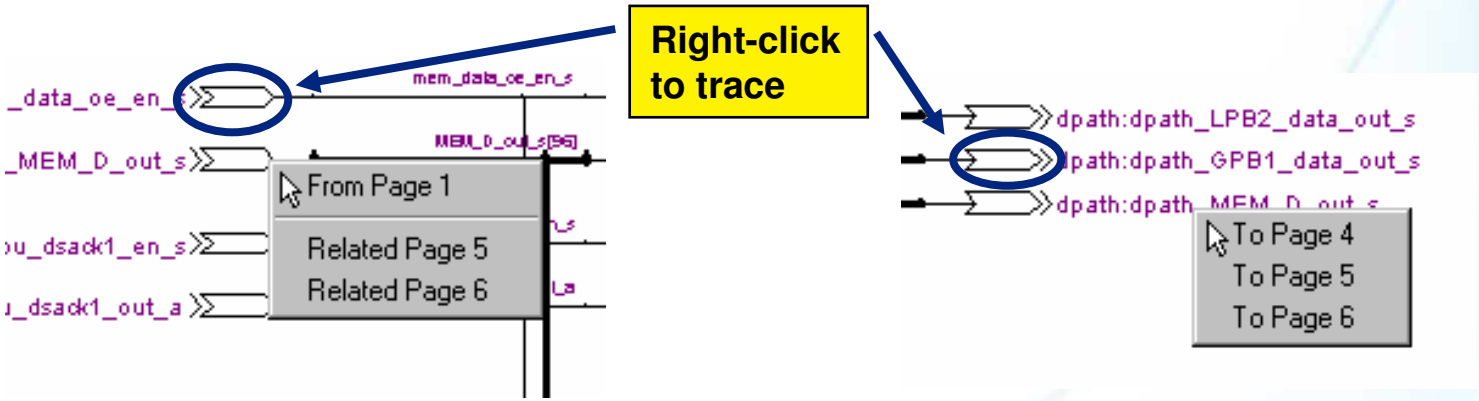


QSF Notes

- Changes to existing assignments updates (i.e. Not moved to end of file)
  - Exceptions (always at end of file)
    - Adding/removing source files
    - Editing members in assignment group
- Sourcing rules
  - Source statements always preserved (not overwritten)
  - Changes to assignments read from source file written back to original file
  - New assignments always written to end of main QSF
  - All assignments copied into new QSF file when new revision created

Page Control

- Hierarchical levels automatically partitioned
  - Control design size per page (tools ⇒ customize ⇒ options)
- Use toolbar to move between pages
- Navigate nets between pages

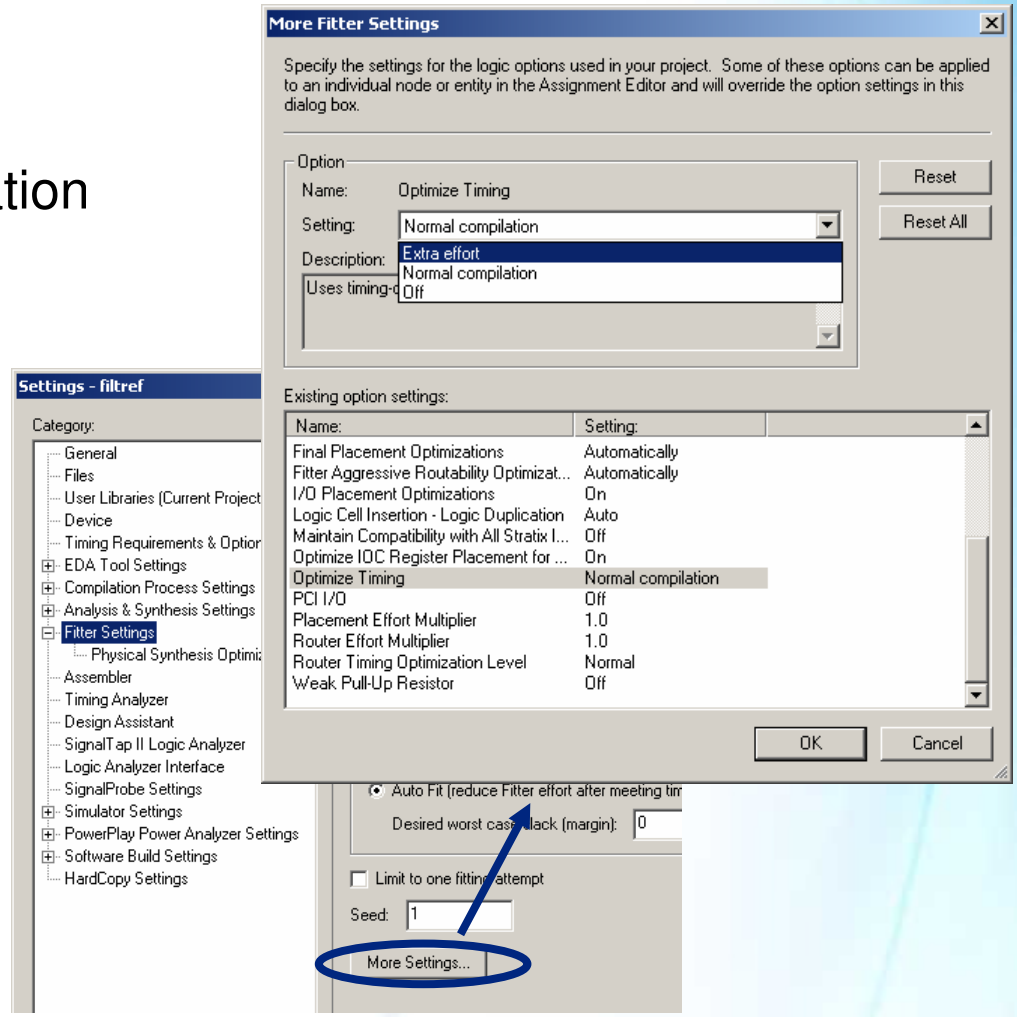


Fitter Settings – More Settings

- Adjust project-wide defaults
- Enable/disable increased/decreased optimization routines
  - May affect compile time

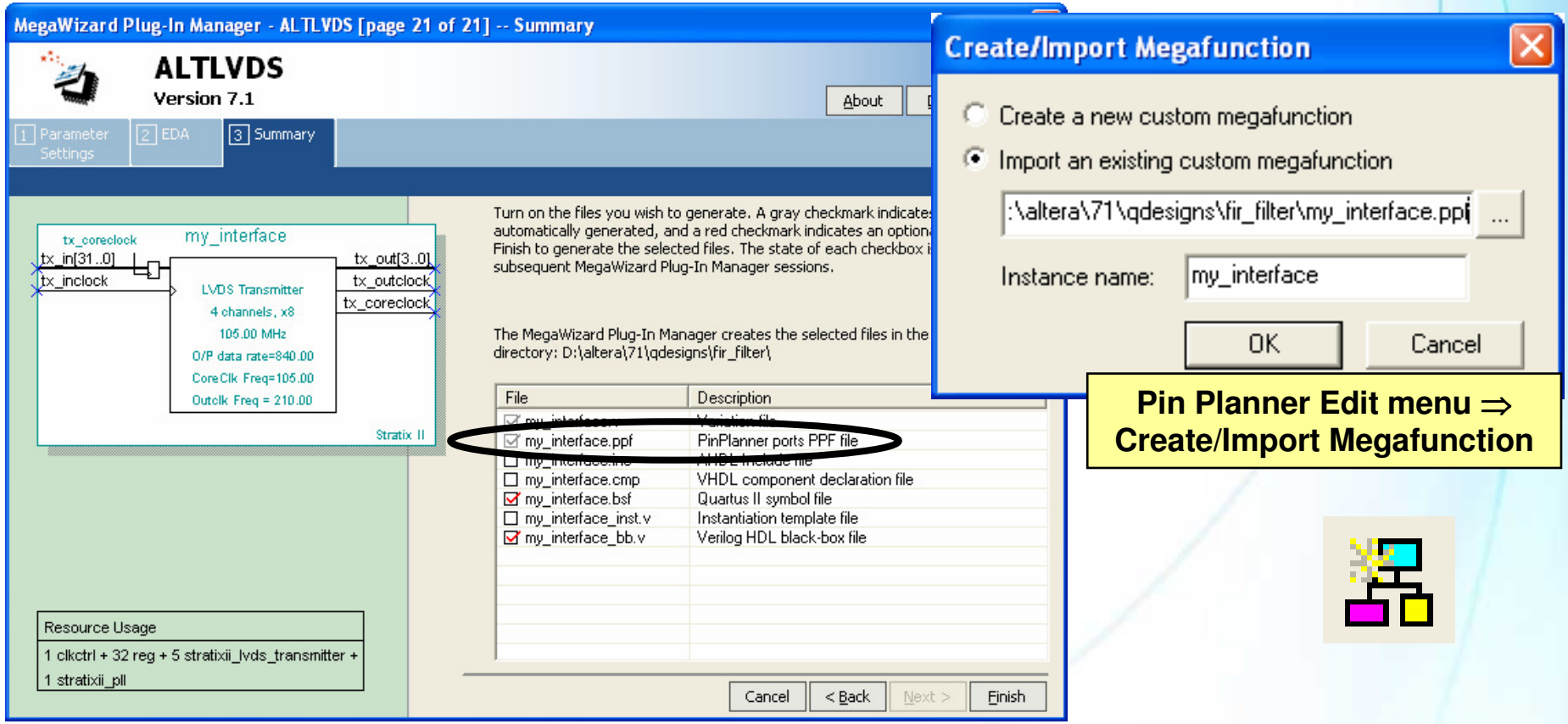
Examples

- **Optimize IOC Register Placement for Timing**
  - **Automatically Uses I/O Cell Registers to Improve I/O Pin Timing**
- **Auto Global SIgnals**
  - **Auto-Promotes High-Fanout Control Signals to Globals**
- **Fitter Aggressive Routability Optimization**
  - **Automatially Performed during Increased Fitting Attempts**



Importing I/O Megafunction

- Select PPF file created when running MegaWizard Plug-In Manager

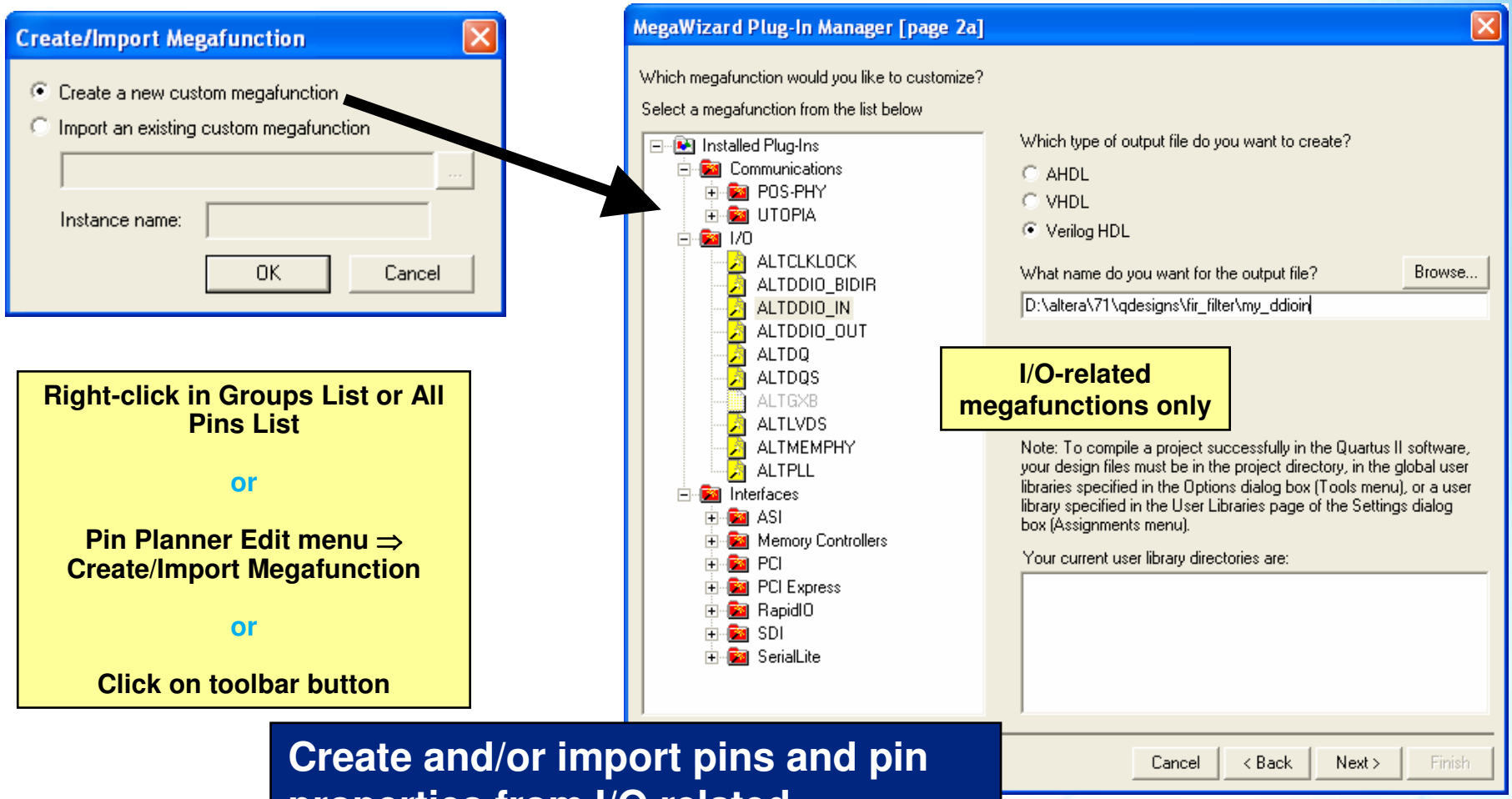


Imported Megafunction

- Megafunction pins appear as a new bus in Groups List
  - Names prefixed with user-defined instance name

Node Name	Direction
d[7..0]	Input Group
yn_out[7..0]	Output Group
my_interface	Bidir Group
my_interface_tx_in[31..0]	Input Group
my_interface_tx_inclock	Input
my_interface_tx_corecl...	Output
my_interface_tx_out[3....	Output Group
my_interface_tx_outclock	Output
<<new node>>	

Creating New I/O Megafunction



Right-click in Groups List or All Pins List

or

Pin Planner Edit menu ⇒ Create/Import Megafunction

or

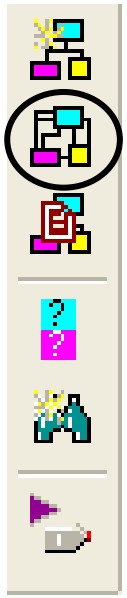
Click on toolbar button

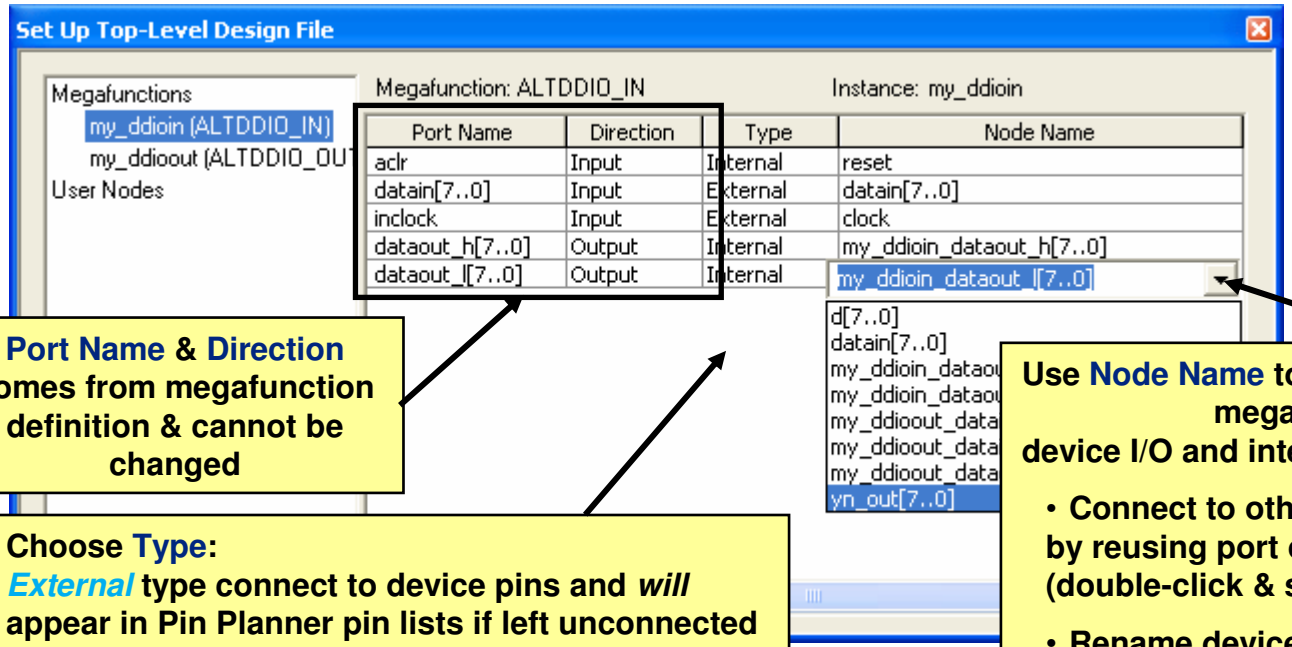
I/O-related megafunctions only

Create and/or import pins and pin properties from I/O related megafunctions & IP (i.e. DDR, LVDS)

Configure & Generate Top-Level HDL (1)

- Choose Set Up Top-Level Design File
  - Right-click, Edit menu, or toolbar
- Configure megafunctions & IP cores by changing values in Type and Node Name columns





Port Name & Direction comes from megafunction definition & cannot be changed

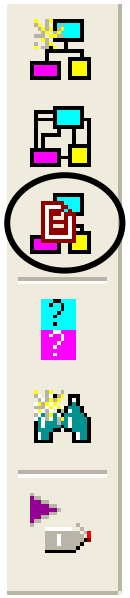
Choose Type:
External **type connect to device pins and *will* appear in Pin Planner pin lists if left unconnected**
Internal **type connect to or become internal nodes and *will not* appear in Pin Planner pin lists**

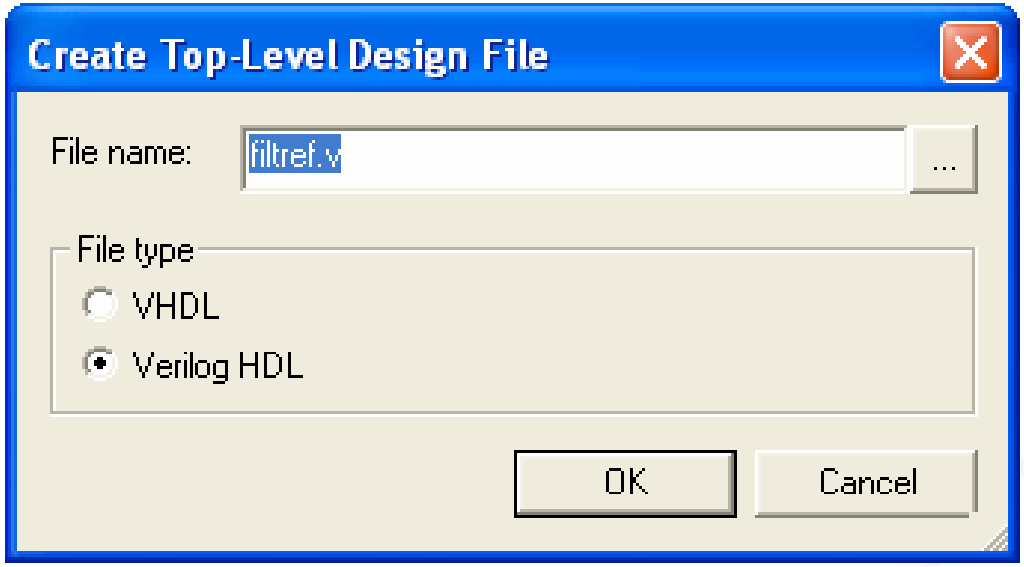
Use Node Name to connect megafunction ports to device I/O and internal node names

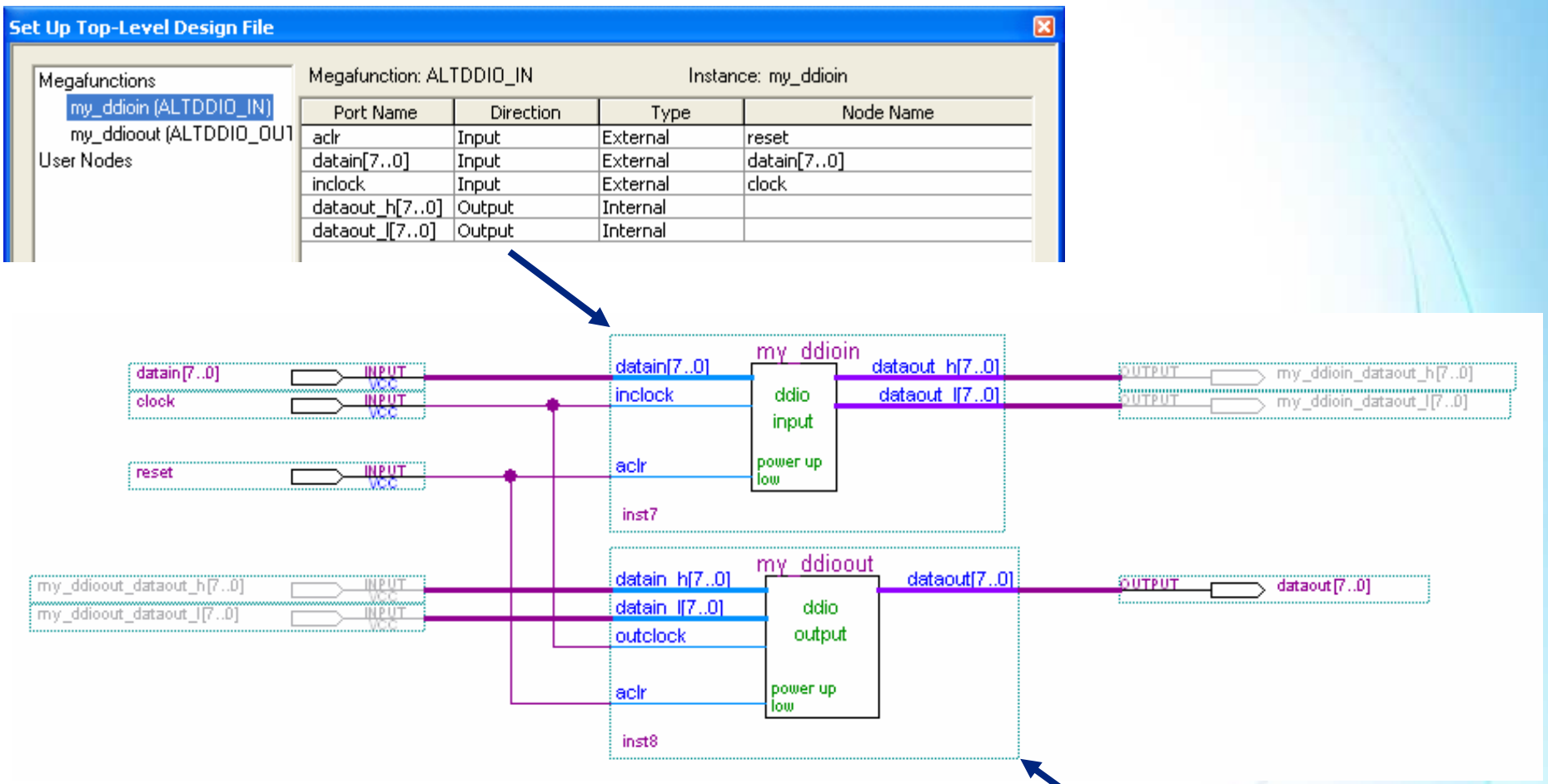
- **Connect to other megafunction ports by reusing port or node names (double-click & select from drop-down)**
- **Rename device I/O & create new node names by typing in new names**

Configure & Generate Top-Level HDL (2)

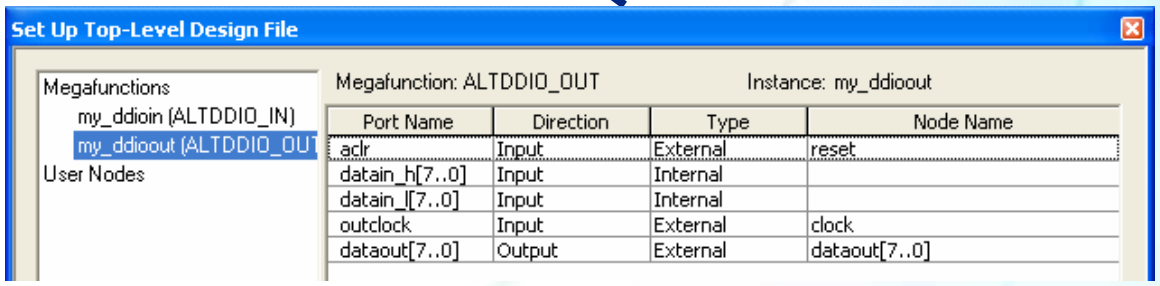
- Choose Create Top-Level Design File
  - Right-click, File menu (Create/Update submenu), or toolbar
- Select a file name and type







Schematic representation of top level file created (schematic not actually created)



Notes on Step 2 (Early I/O Planning Methodology)

- Internal nodes will be commented and set as virtual pins in created top-level file (represented as gray pins on previous slide)
- Make changes to megafunction or IP through the Pin Planner
  - Highlight in Groups List and choose Edit Megafunction (Edit menu or right-click)
  - Must re-import megafunction or IP when changes are made
  - Must update top-level file when changes are made
  - Run I/O Assignment Analysis after any change to validate I/O

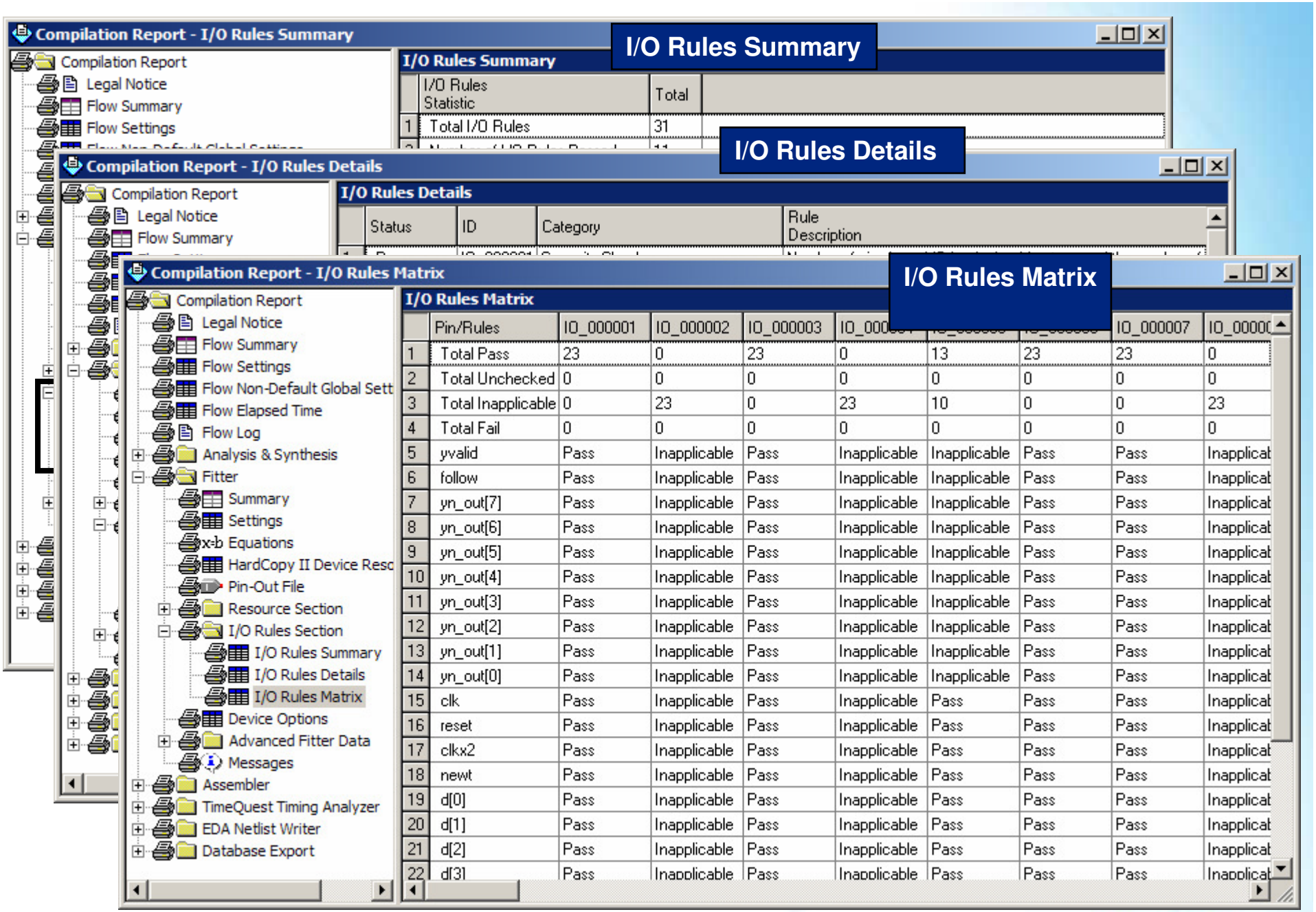
Pin Planner – Error Checking

	Error Checking Description (Examples)
1	Resource availability in I/O bank or Vref group
2	Once half of differential pair is assigned, complement pin not available for assignment
3	No output/bidirectional pins assigned to dedicated input pins (e.g. clock pins)
4	Pin I/O standard must be compatible with the node's I/O standard
5	All nodes in same Vref group must have the same Vref voltage
6	A node assigned to the I/O bank must have an I/O standard compatible with the bank VCCIO

Recommendation: Run I/O Assignment Analysis (Discussed Later) for Complete I/O & I/O-Core Checking

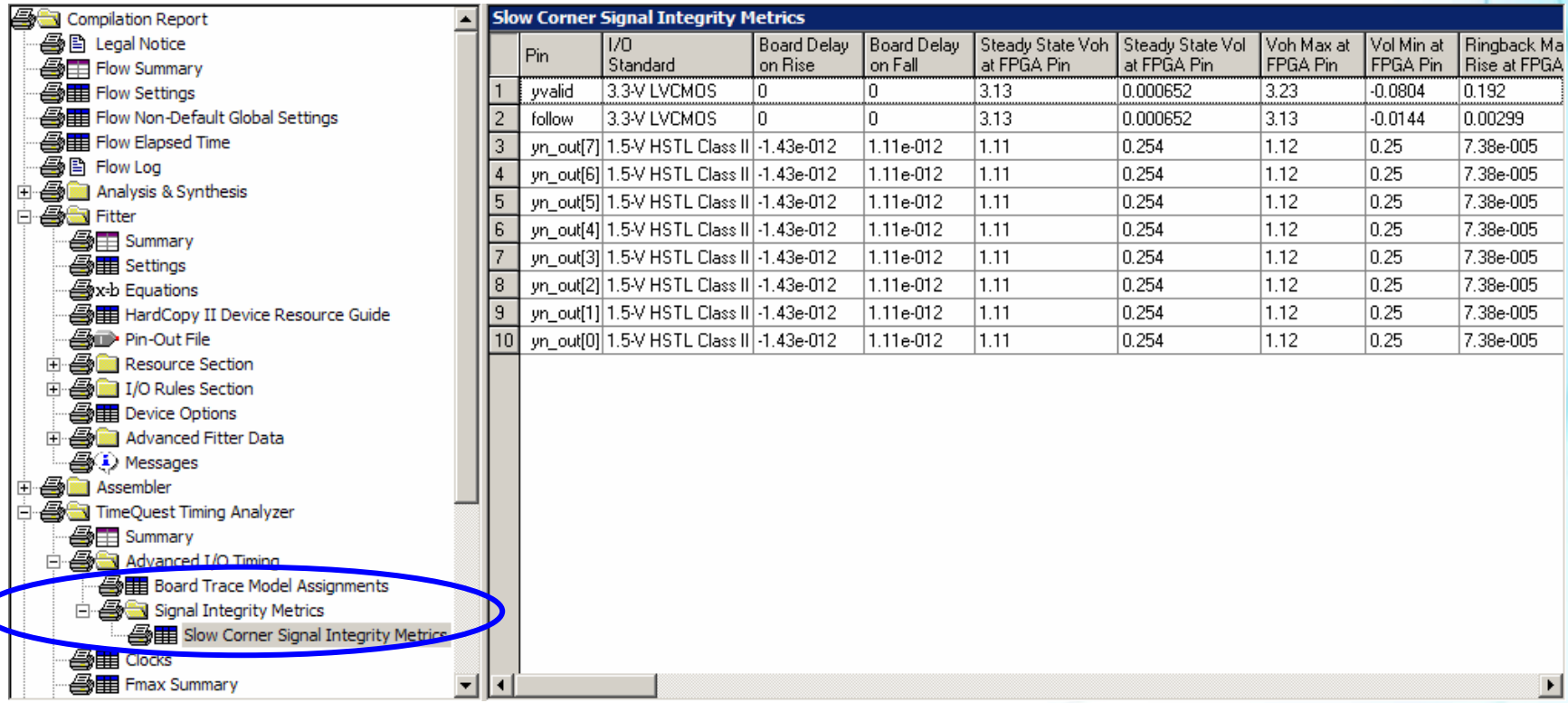
Stratix II I/O Rules Section

- Folder in Compilation Report containing I/O Rules tables
  - Summary with number of I/O Rules checked, passed & failed
  - List of I/O Rules & whether passed or failed
  - Matrix of I/O Rules checked on a per pin basis



Advanced I/O Analysis Output

- Detailed Signal Integrity Metrics appears as in table in timing analysis report



Slow Corner Signal Integrity Metrics

	Pin	I/O Standard	Board Delay on Rise	Board Delay on Fall	Steady State Voh at FPGA Pin	Steady State Vol at FPGA Pin	Voh Max at FPGA Pin	Vol Min at FPGA Pin	Ringback Ma Rise at FPGA
1	yvalid	3.3-V LVCMOS	0	0	3.13	0.000652	3.23	-0.0804	0.192
2	follow	3.3-V LVCMOS	0	0	3.13	0.000652	3.13	-0.0144	0.00299
3	yn_out[7]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
4	yn_out[6]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
5	yn_out[5]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
6	yn_out[4]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
7	yn_out[3]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
8	yn_out[2]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
9	yn_out[1]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005
10	yn_out[0]	1.5-V HSTL Class II	-1.43e-012	1.11e-012	1.11	0.254	1.12	0.25	7.38e-005

Alternative Method to Writing SDC File

- Maintain “golden” SDC input file in SDC File Editor
  - File holds user-entered constraints
  - Add design-specific comments
  - Arrange SDC commands according to design
- Edit “golden” SDC file
  - Edit input SDC file directly using file editor features
  - Copy SDC commands from Console pane (History tab)
  - Copy commands from TimeQuest-generated output SDC file

Quartus II Simulation

- Simulator method & features overview
- Simulator settings
- VWF file creation
- Simulation output

Supported Simulation Methods

- Quartus II
  - VWF (vector waveform file)
    - Primary graphical waveform file
  - CVWF (compressed vector waveform file)
    - Compressed binary version of VWF
  - VCD (value charge dump file)
    - Standardized text-based input file (IEEE-1364)
  - VEC (vector file)
    - Text-based input file
  - Tcl/TK scripting
- 3rd-party simulators
  - Verilog/VHDL testbench

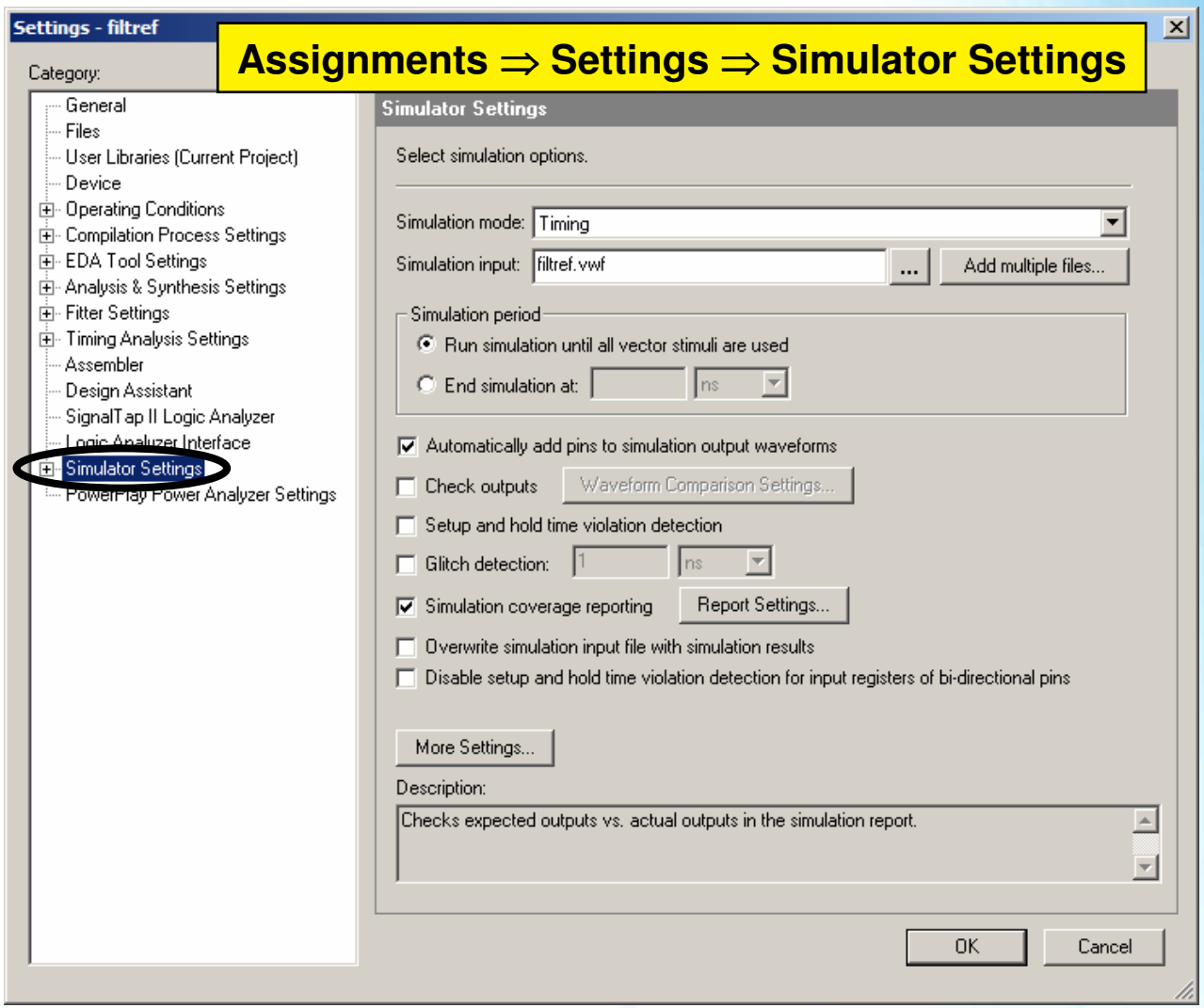
***Note:** The simulator channel file (.SCF) and table file (.TBL) are also supported for backwards compatibility with MAX+PLUS II*

Simulator Features

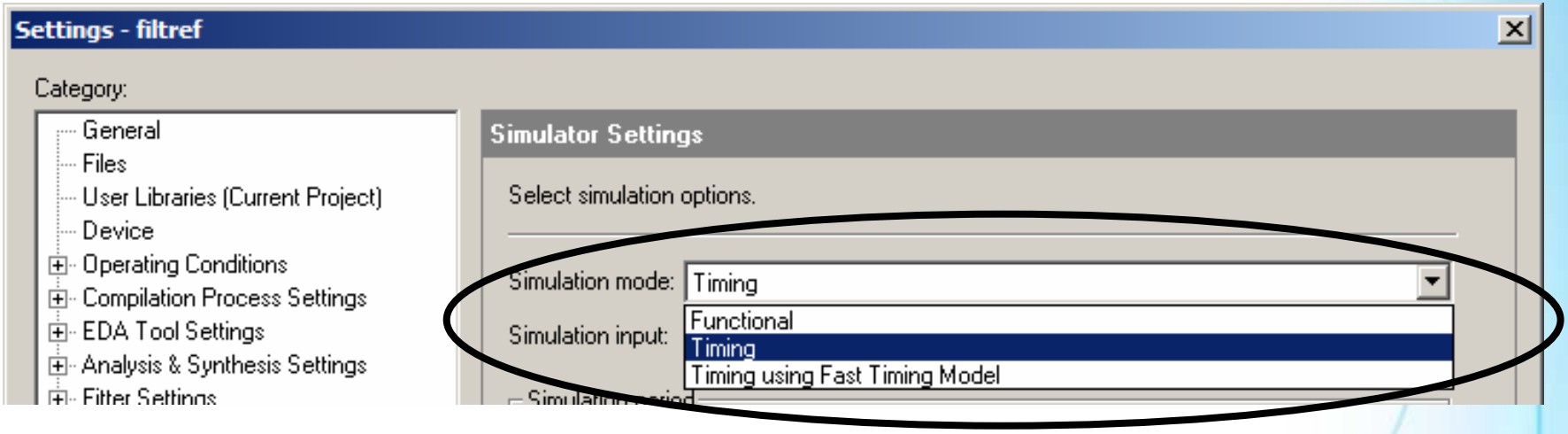
- Converts VWF into HDL testbench
- Generates HDL testbench template
- Supports breakpoints
- Performs automatically
  - Adding output pins to output waveform file
  - Checking outputs at end of simulation

Simulator Settings

- Mode
- Input file
- Period
- Options



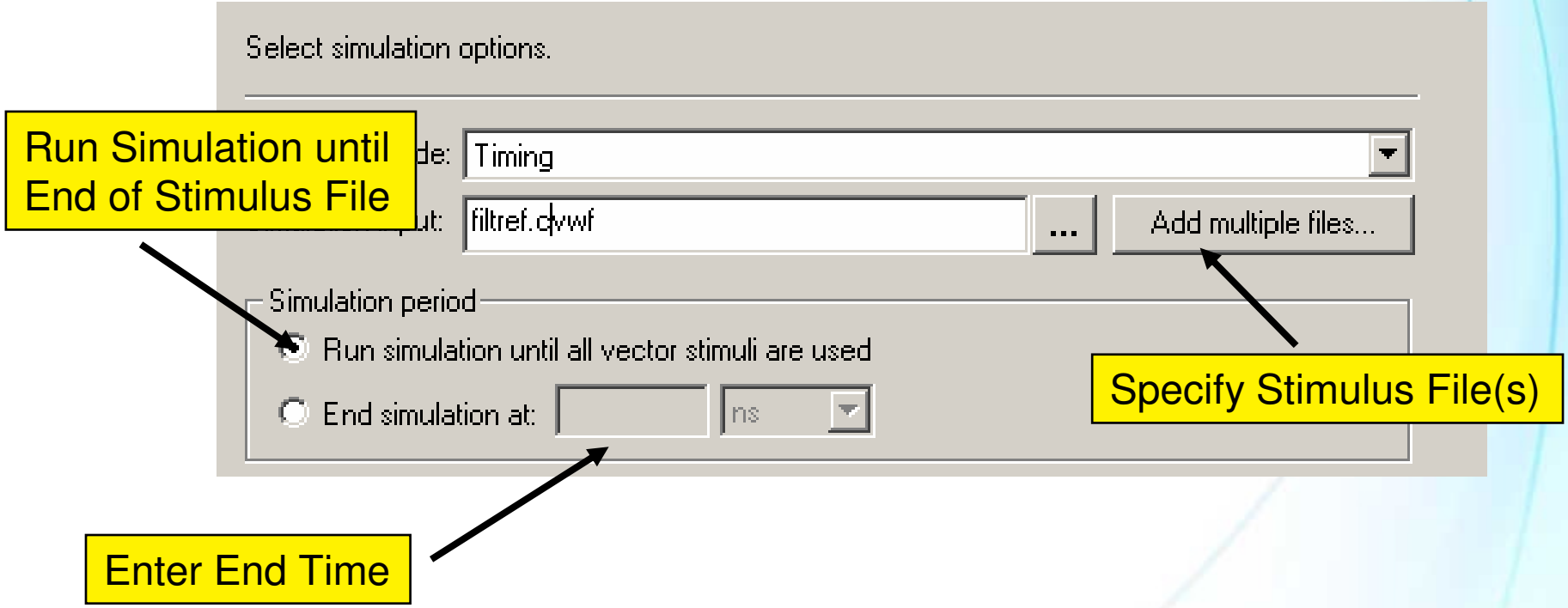
Simulator Modes



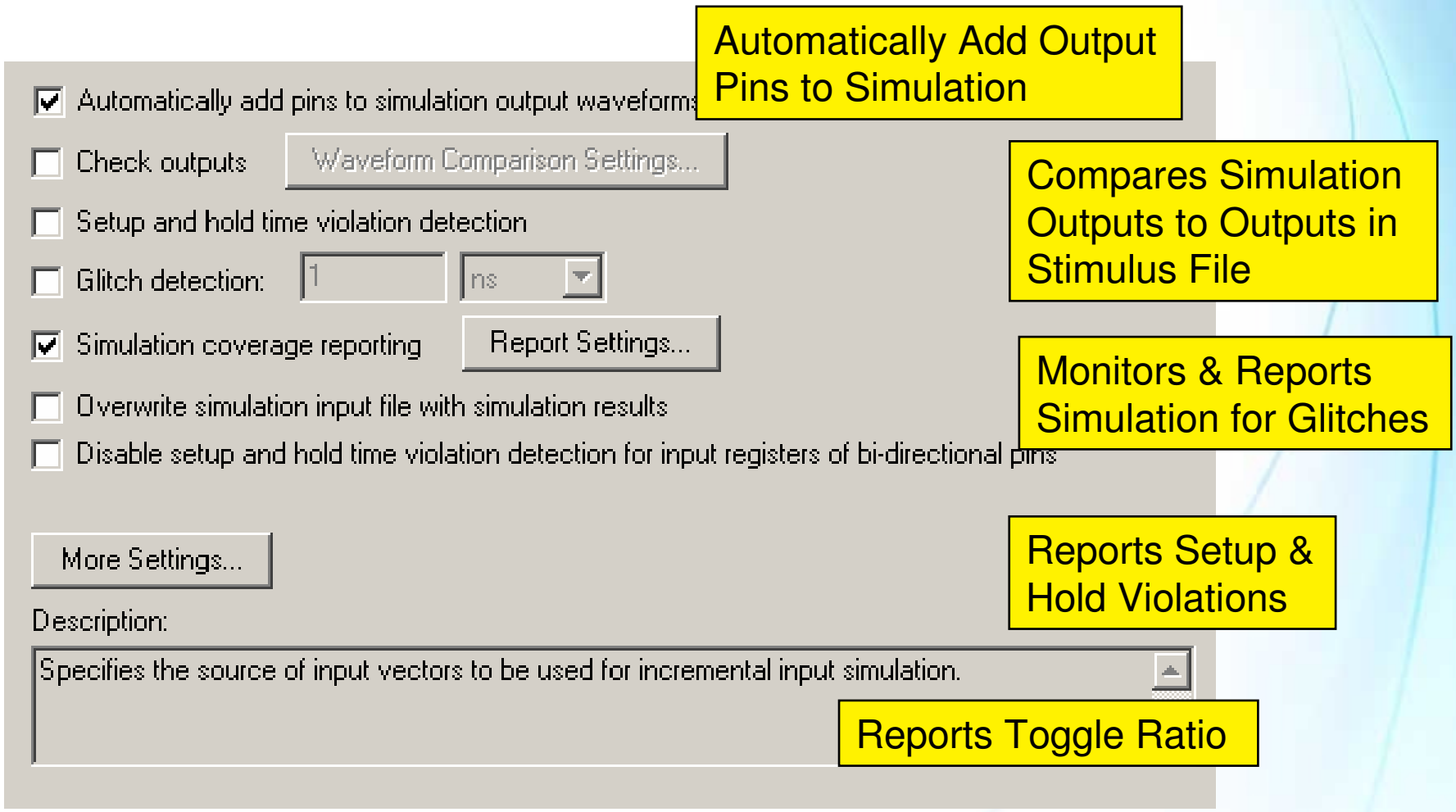
- Functional
  - Type: RTL
  - Uses pre-synthesis netlist
- Timing
  - Type: gate-level or post-place & route
  - Uses fully compiled netlist
  - Uses worst-case timing model
- Timing Using Fast Timing Model
  - Similar to Timing
  - Uses Best-Case Timing Model

Simulator Input & Period

- Specifies stimulus & length of simulation period



Simulator Options



Automatically Add Output Pins to Simulation

Compares Simulation Outputs to Outputs in Stimulus File

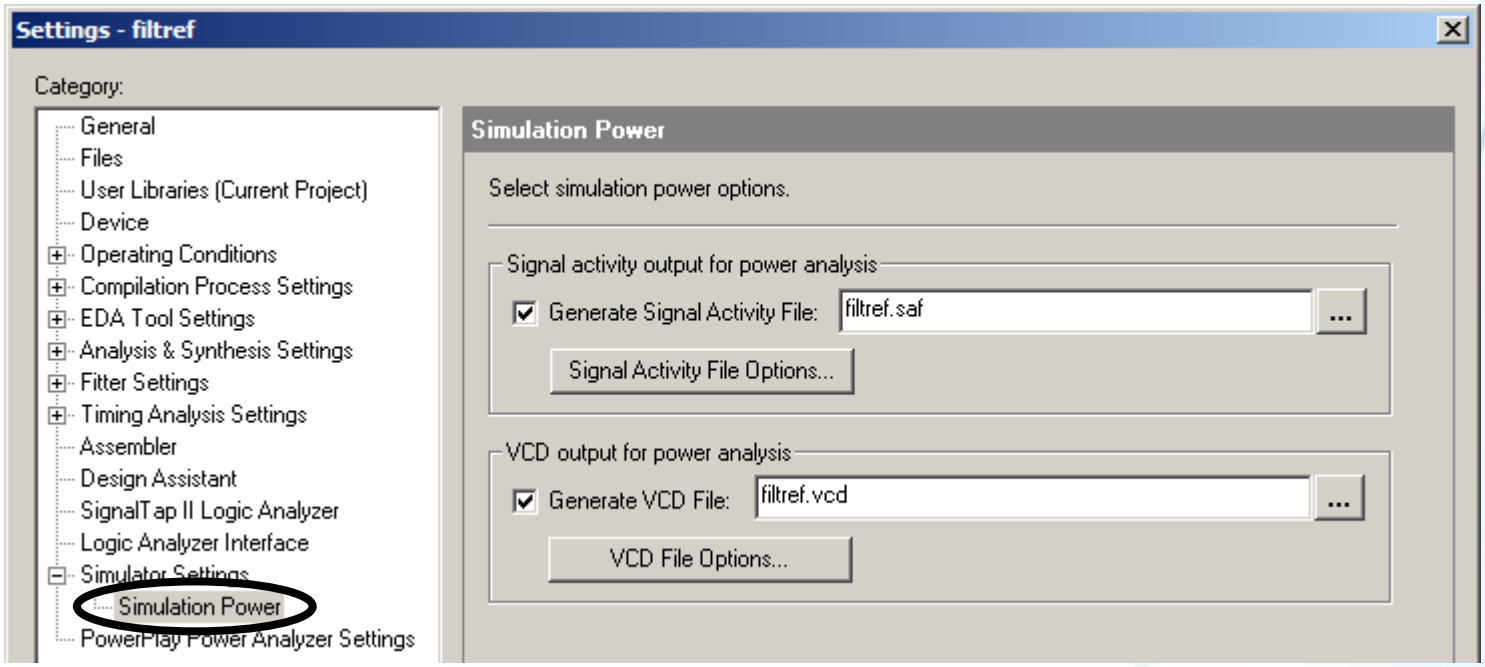
Monitors & Reports Simulation for Glitches

Reports Setup & Hold Violations

Reports Toggle Ratio

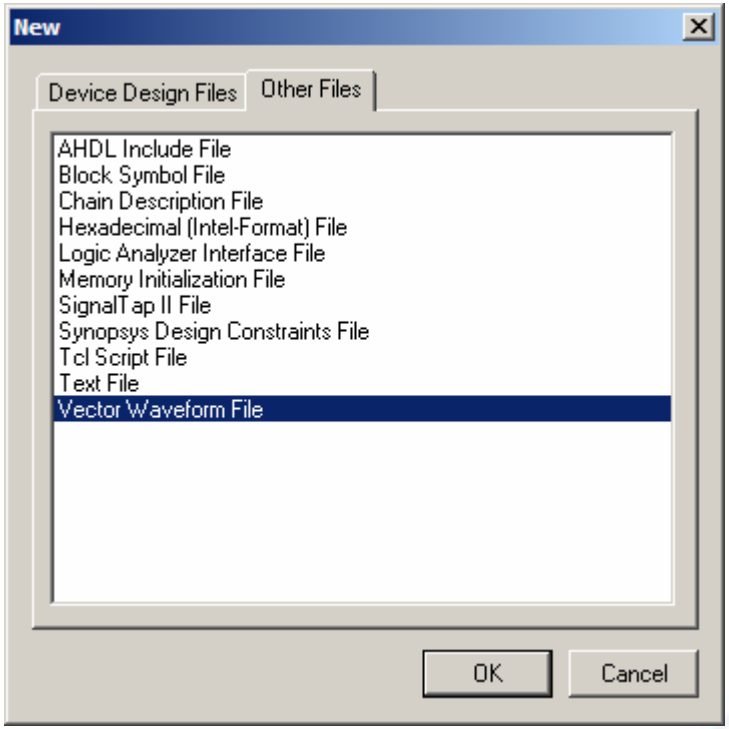
Generate SAF

- Generate SAF or VCD file for power calculation

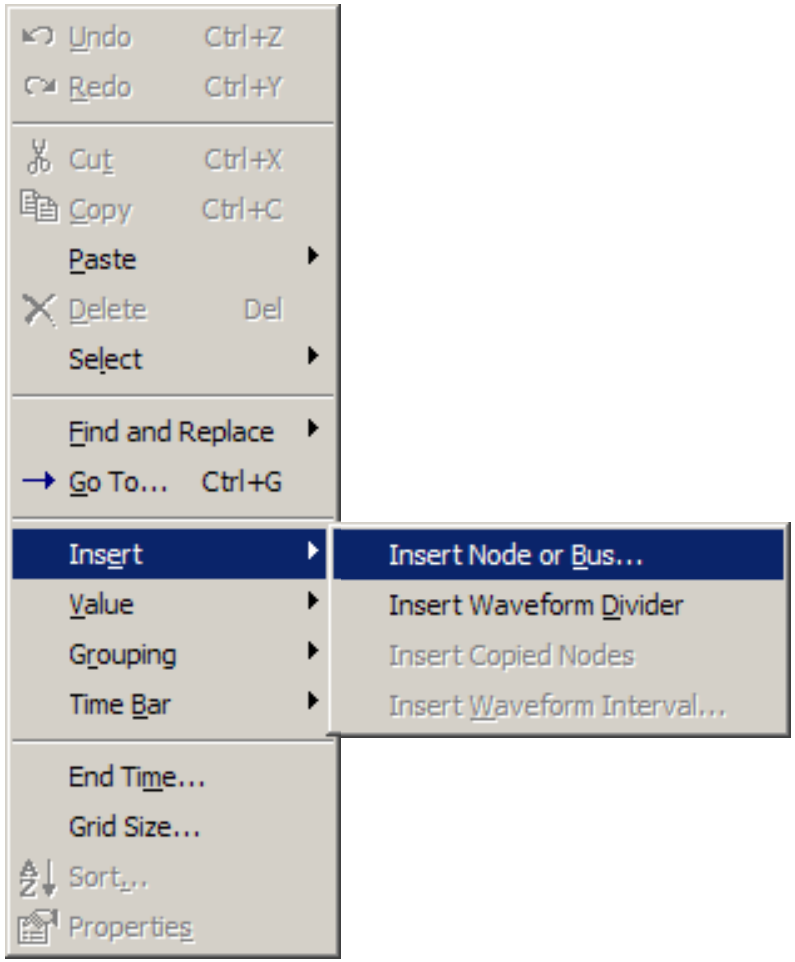


Create New Vector Waveform File

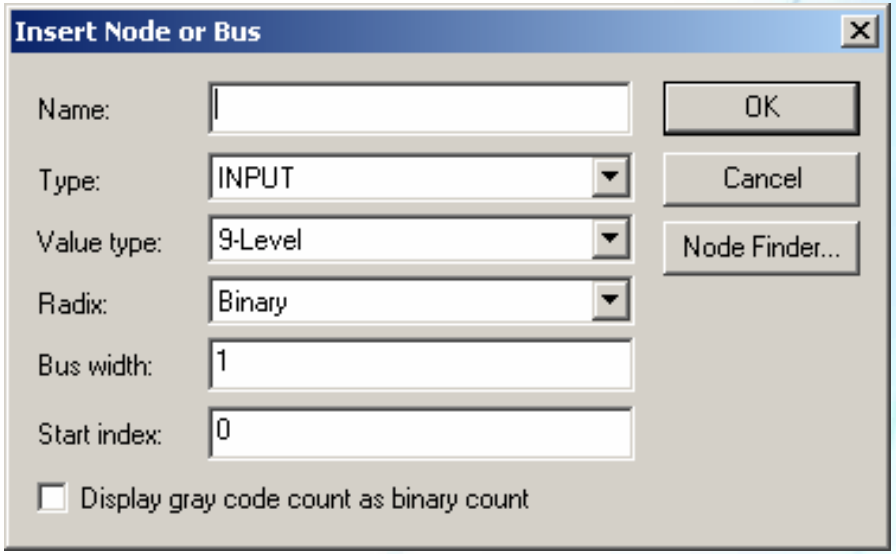
- Select **File** ⇒ **New** ⇒ **Vector Waveform File** (Other Files tab)



Insert Nodes

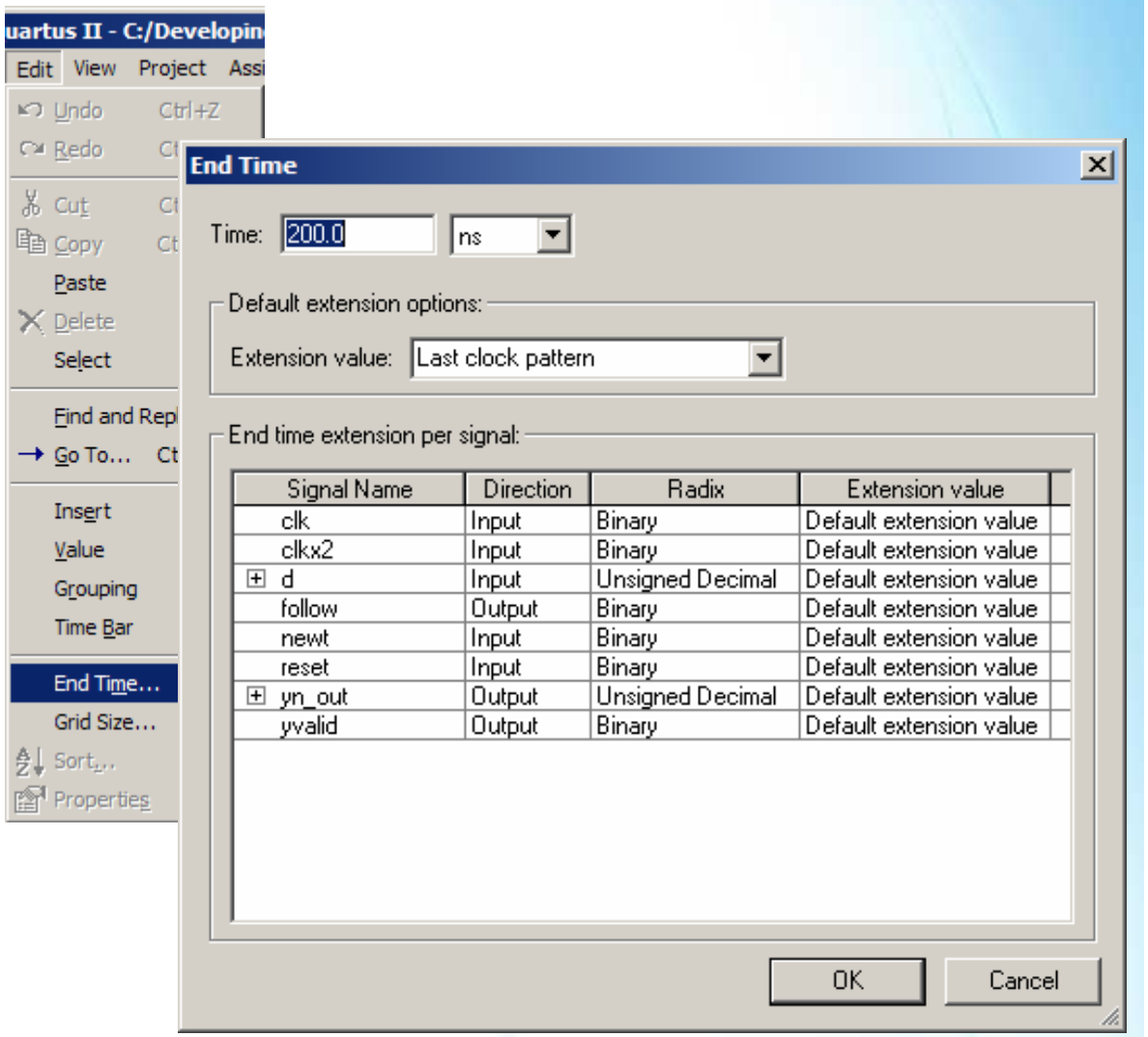


- Select **Insert Node or Bus** (Edit menu)
  - VWF must be open
  - Use node finder



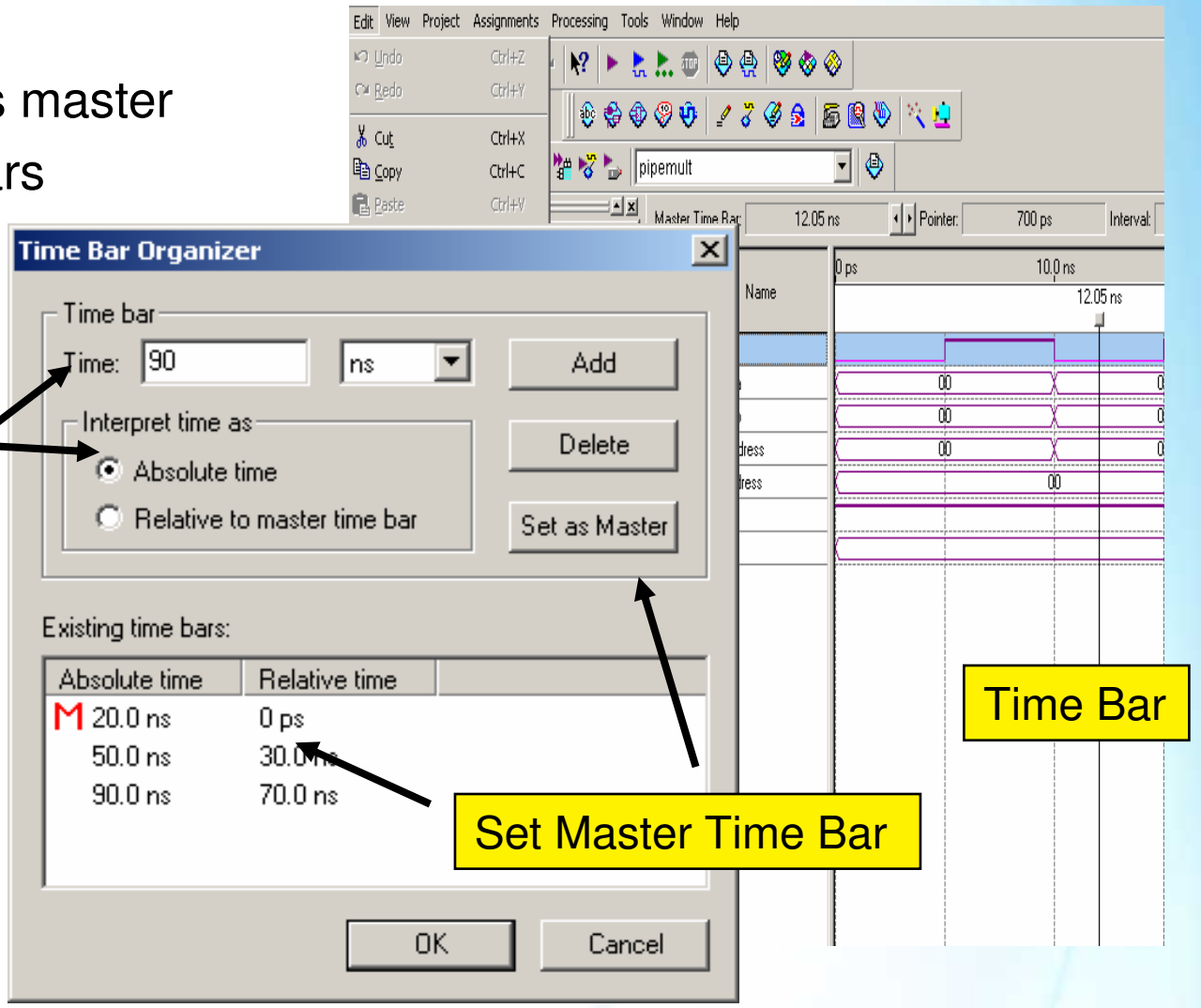
Specify End Time

- Maximum length of simulation time
  - Edit menu
- Specify how to extend signal values



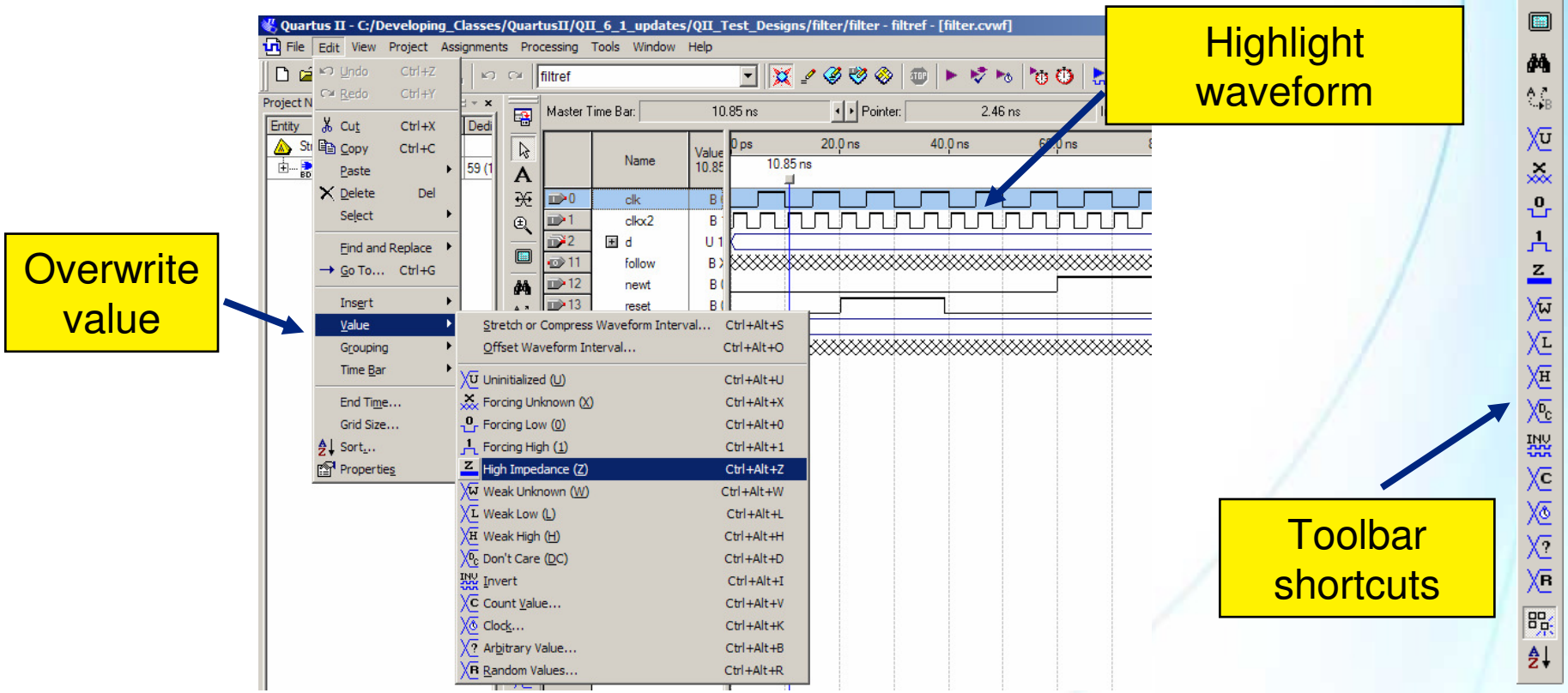
Insert Time Bars

- Set one time bar as master
- Insert other time bars
  - Relative to master
  - Absolute



Draw Stimulus Waveform

- Highlight portion of waveform to change
- Overwrite value with desired value

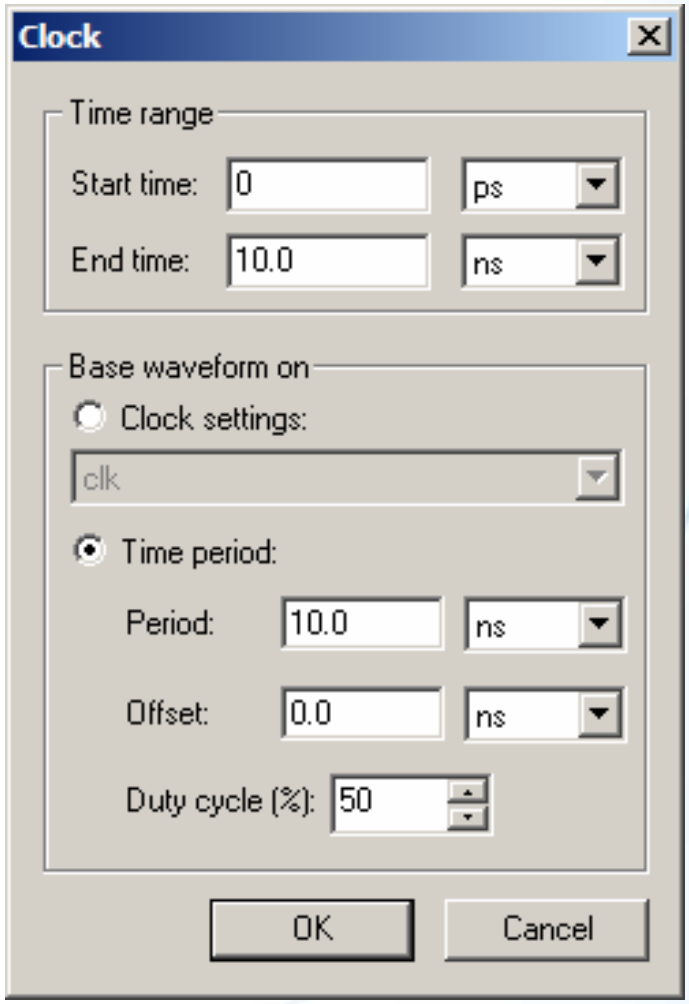


Overwrite Waveform Signal Values

- 1 = forcing ‘1’
- 0 = forcing ‘0’
- X = forcing unknown
- U = uninitialized
- Z = high impedance
- H = weak ‘1’
- L = weak ‘0’
- W = weak unknown
- DC = don’t care

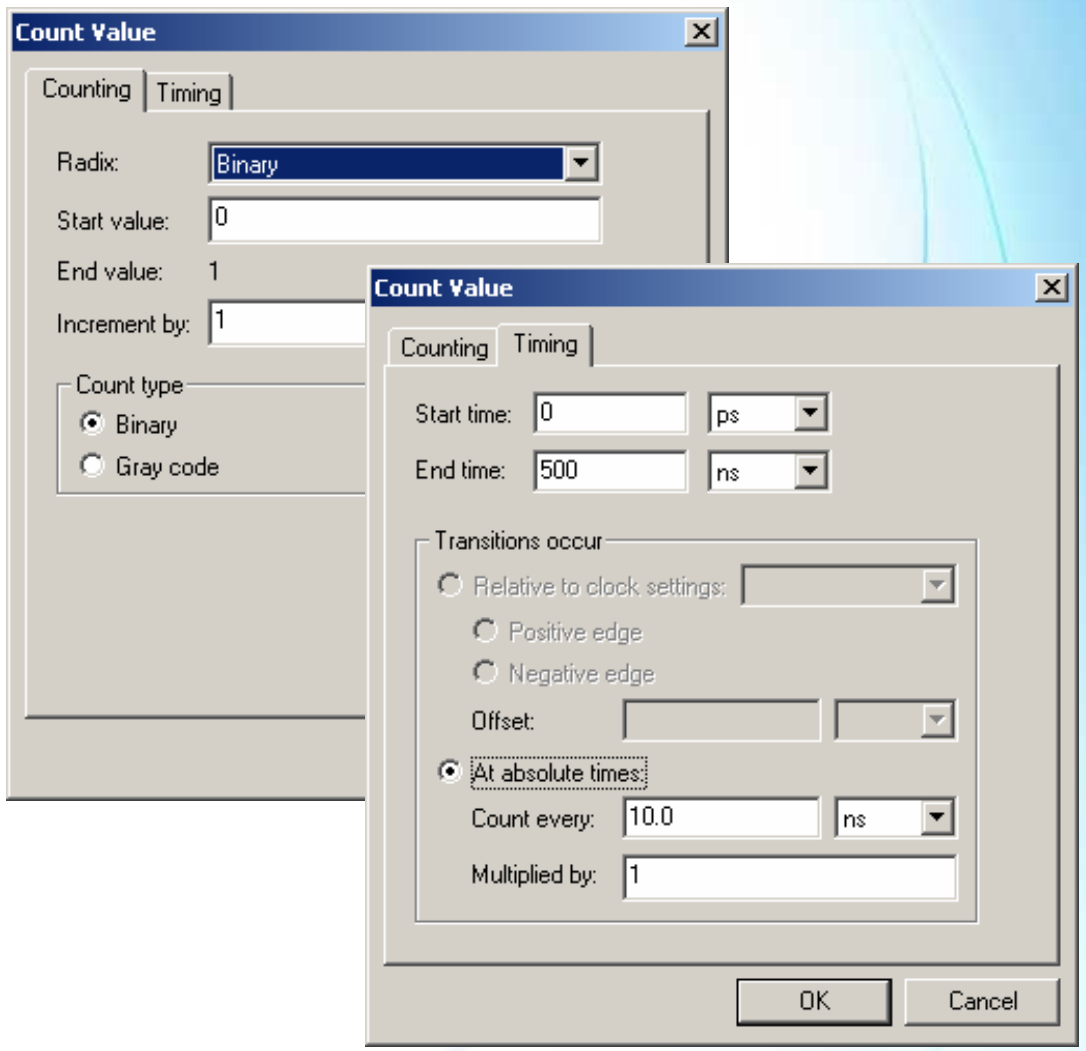
Overwrite Waveform Patterns

- Clock
  - Enter period & duty cycle



Overwrite Waveform Patterns (cont.)

- Counting pattern
  - Enter count timing
  - Enter start value & increment

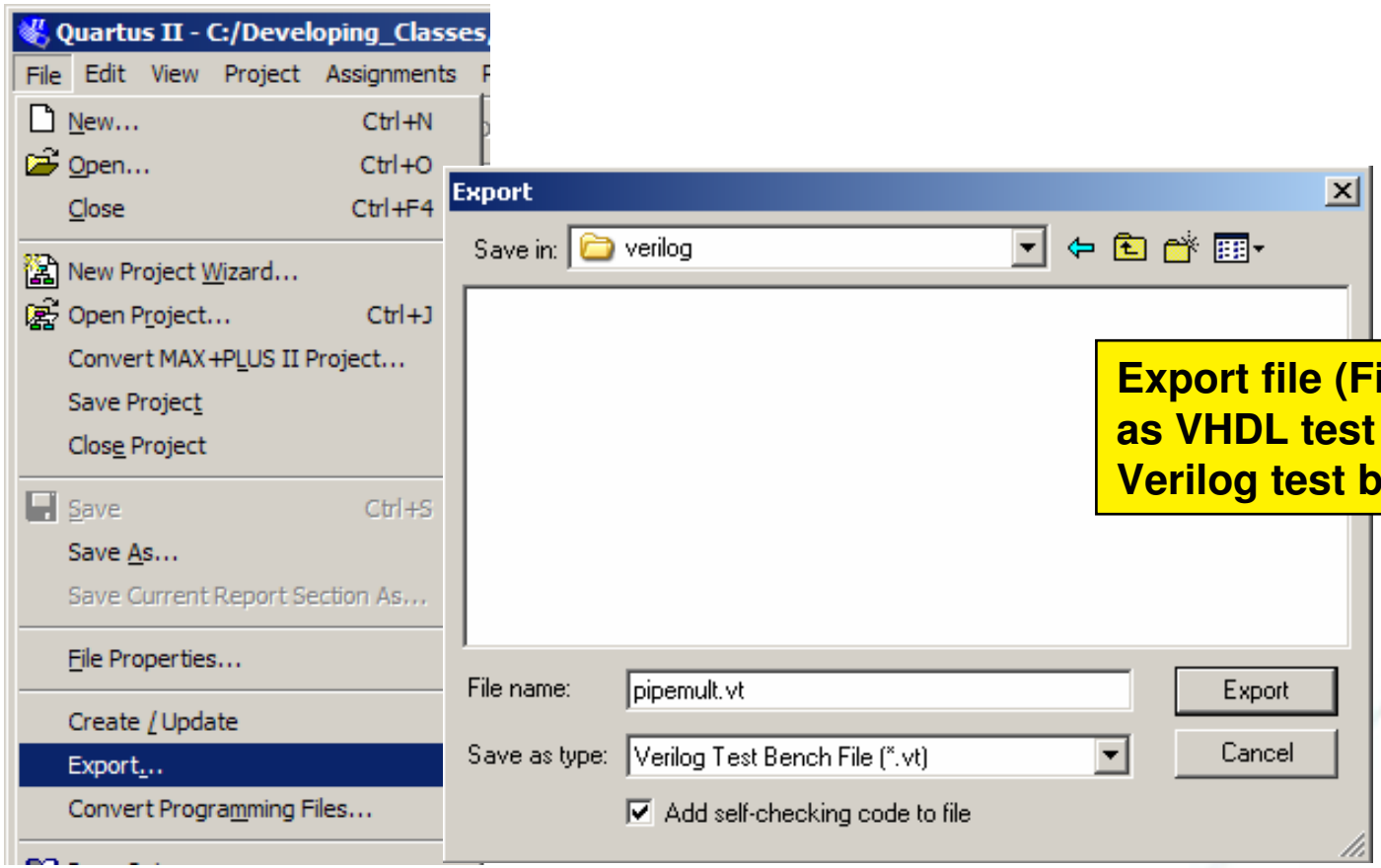


Other Waveform Patterns

- Arbitrary Value
  - Group value for busses
- Random Value

Waveform to Testbench Generator

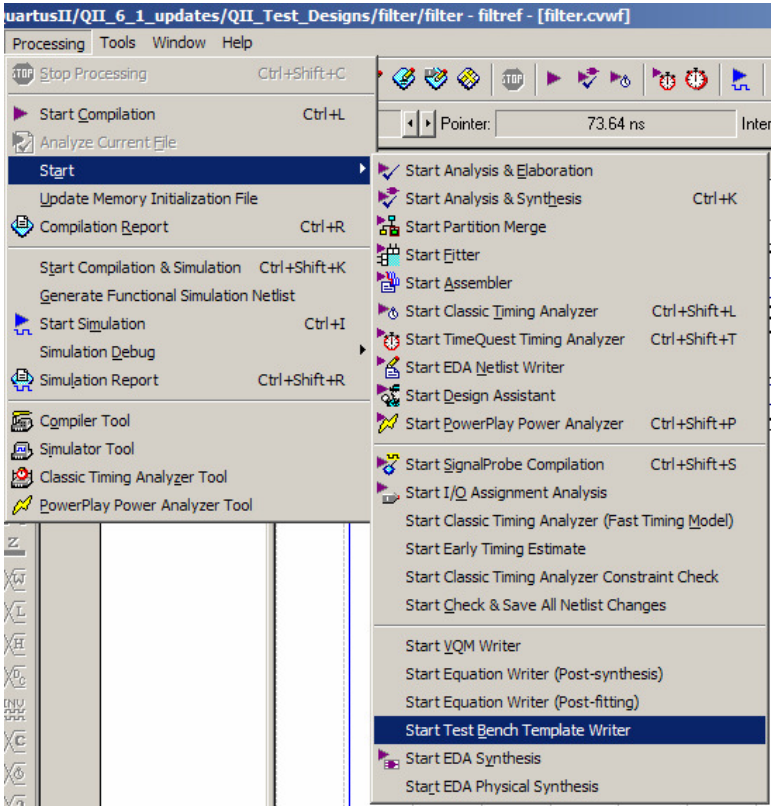
- Converts VWF into HDL testbench

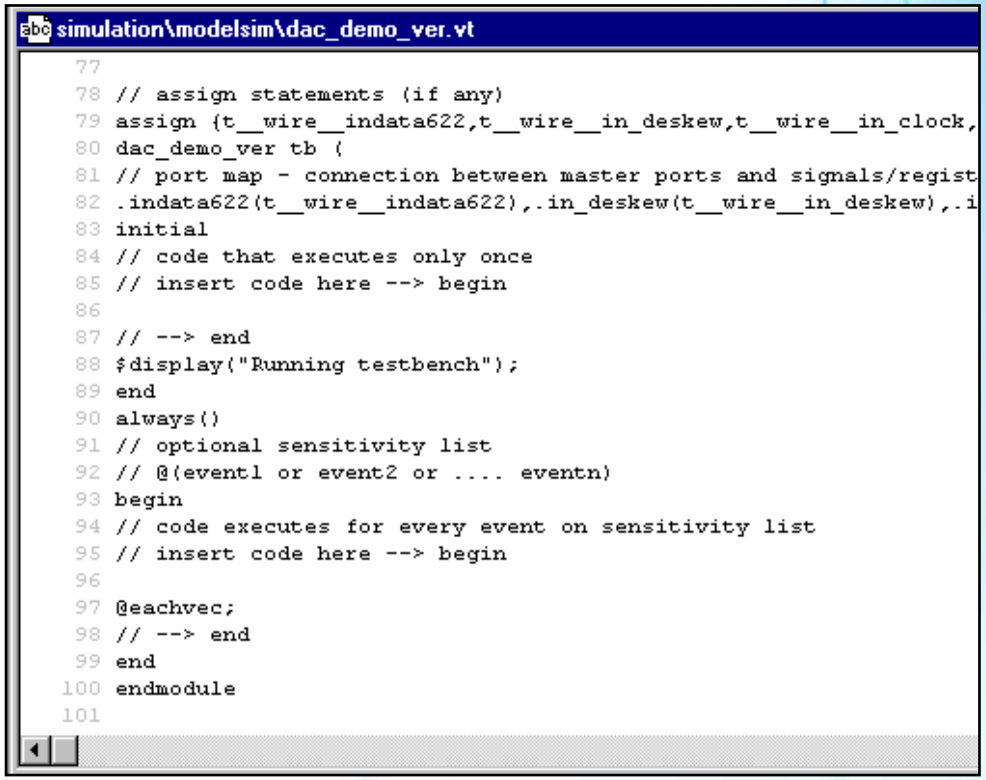


Export file (File menu) and save as VHDL test bench (.VHT) or Verilog test bench (.VT) file

Testbench Template Generator

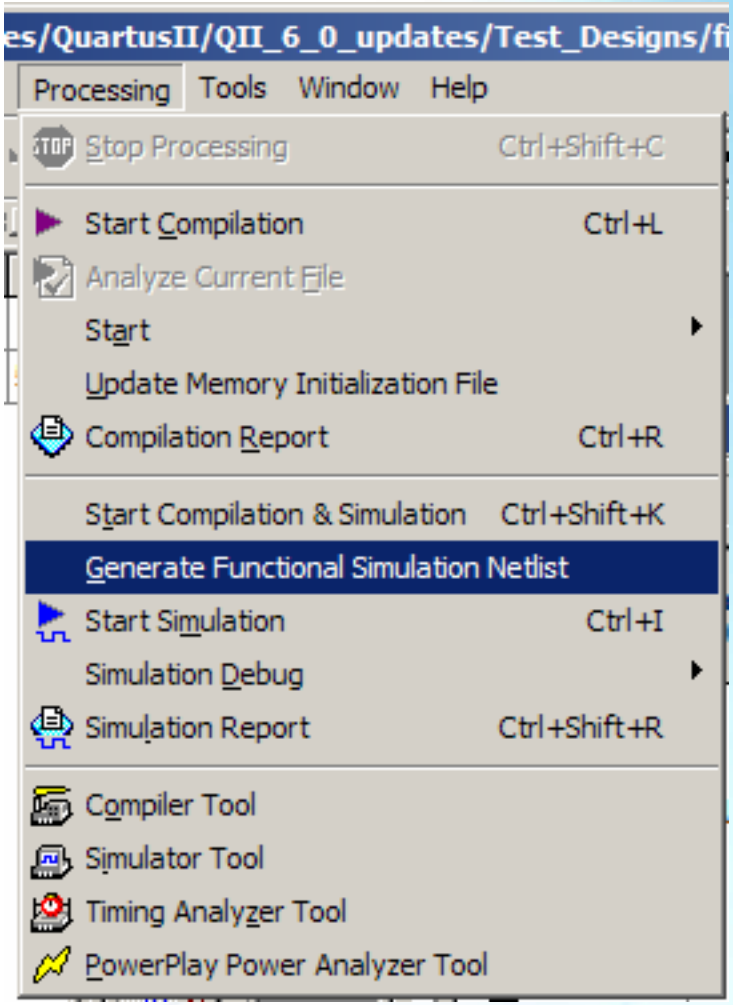
- Generates HDL testbench template
  - User inserts test stimulus





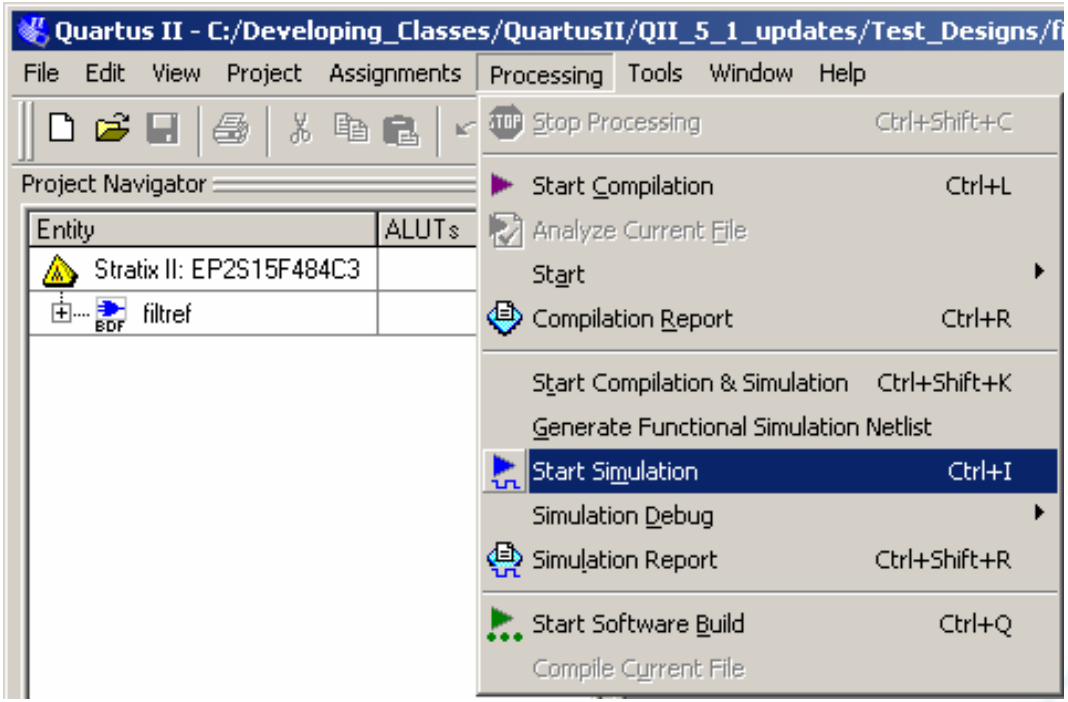
Before Functional Simulation

- Perform **Generate Functional Simulation Netlist** (Processing menu)
  - Creates pre-synthesis netlist
  - Fails simulation if not performed



Starting Simulation

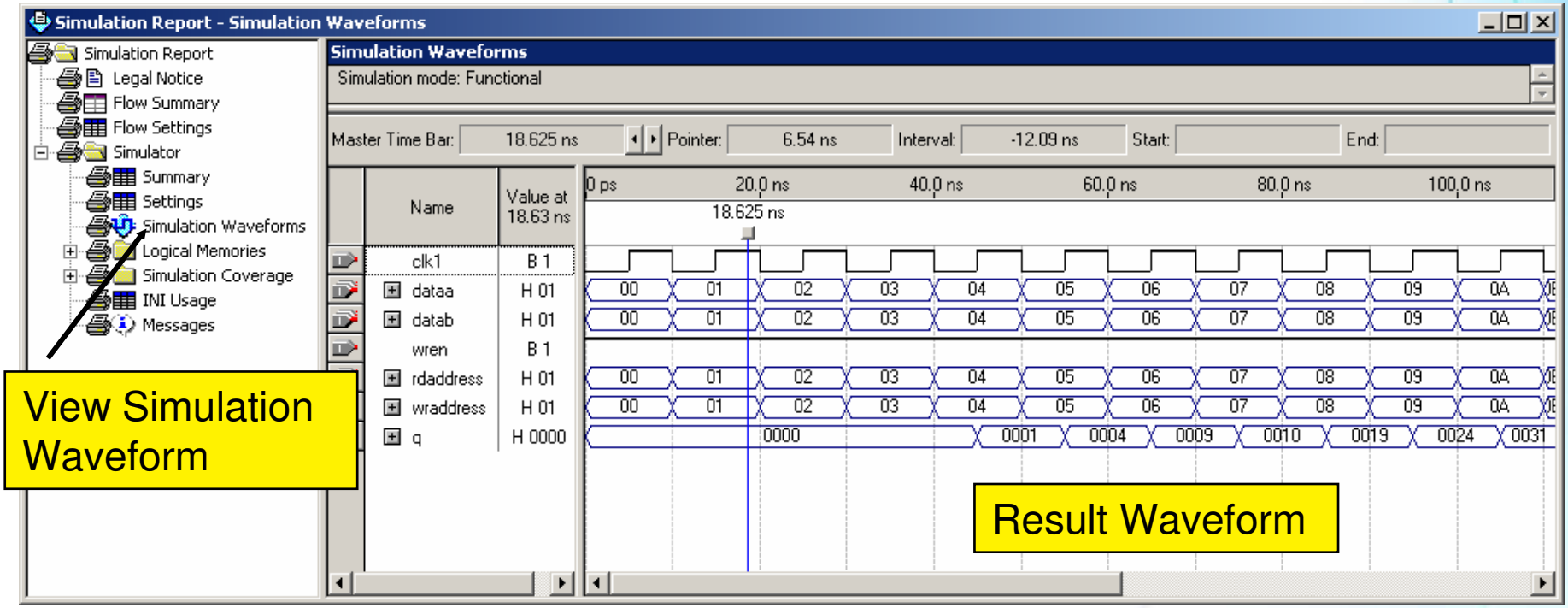
- Processing menu ⇒ Start Simulation



- Scripting

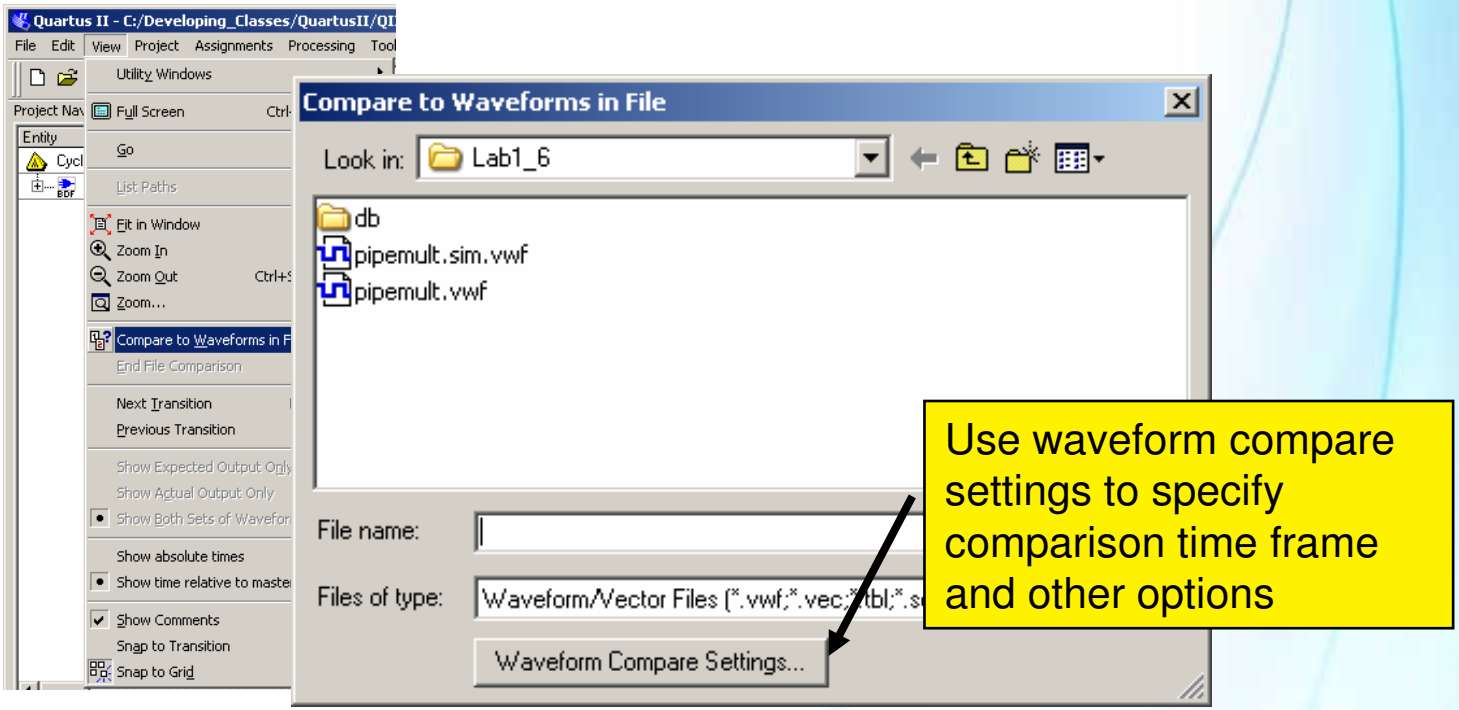
Simulation Report

- Displays simulation result waveform



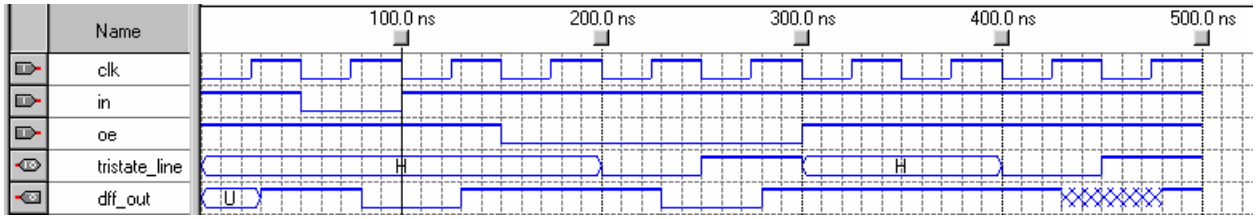
Comparing Waveforms

- Select Compare to Waveforms (View menu)
  - Simulation waveform must be open
- Select VWF comparison file

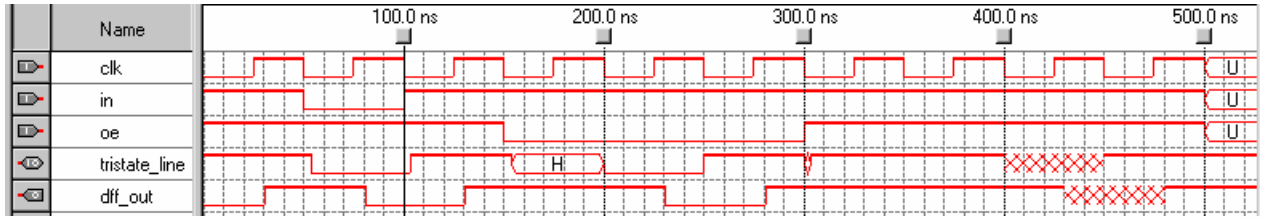


Compared Waveforms (Simulator Report)

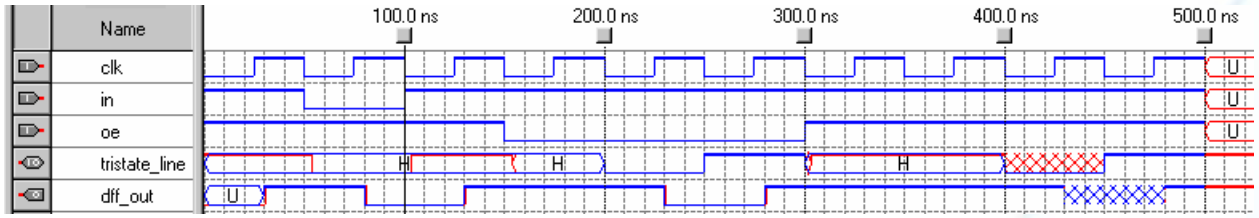
- Original waveforms (ctrl+1)



- Compared file waveforms (ctrl+2)

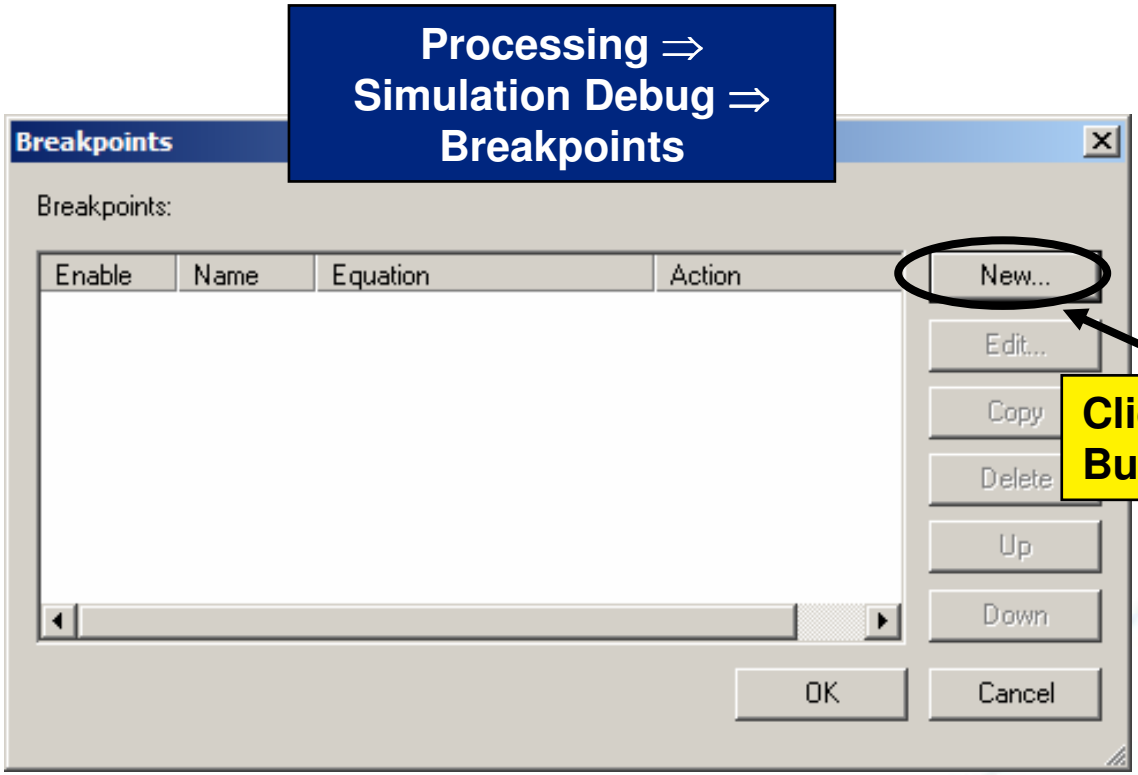


- Both sets of waveforms (ctrl+3)



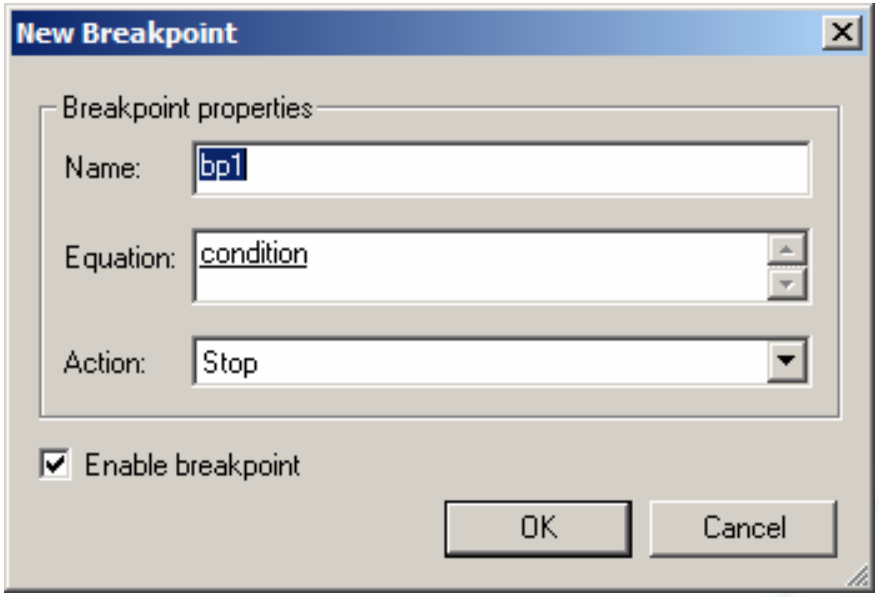
Breakpoints

- Interrupts simulation at specified points



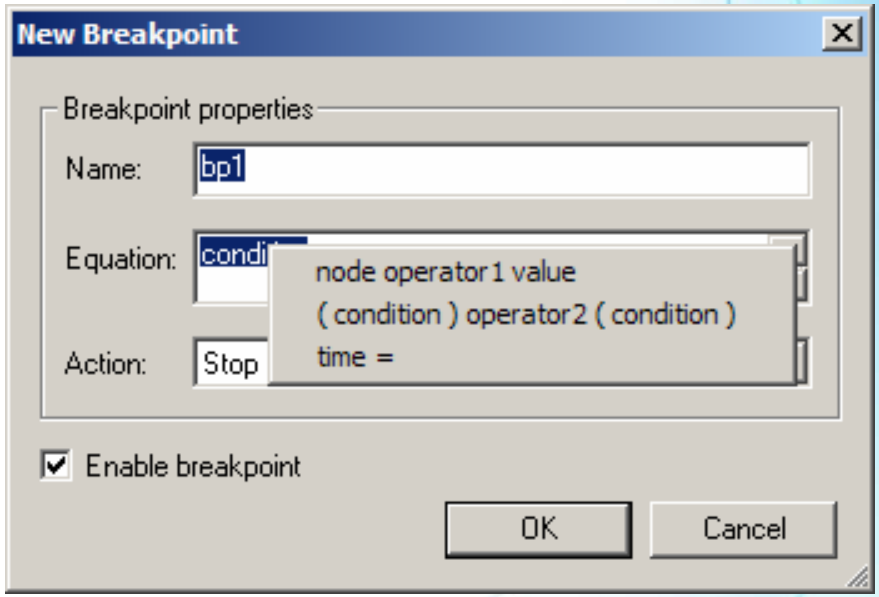
Breakpoints

- Consists of 3 parts
  - Name
  - Equation (condition)
  - Action

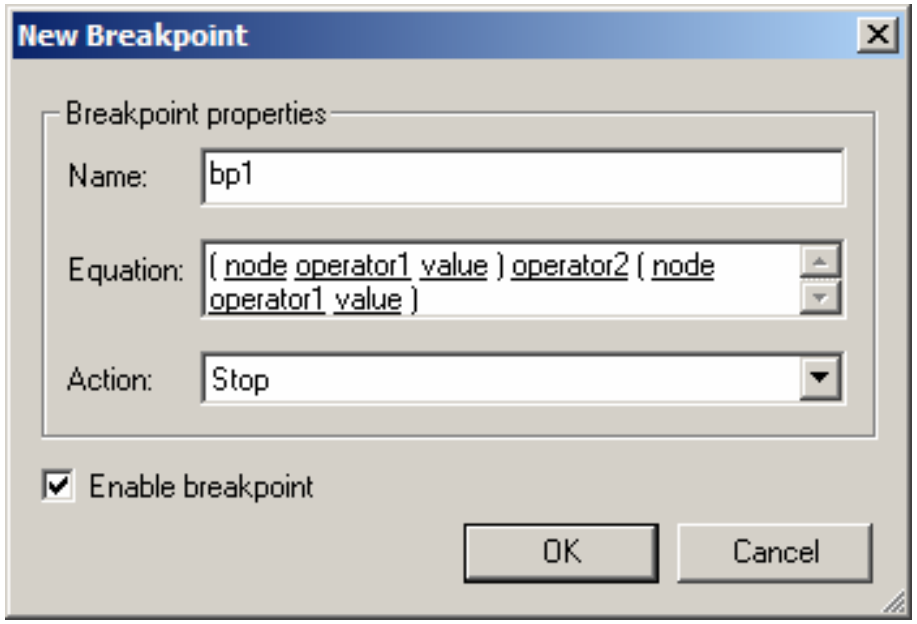


Breakpoint Conditions

- <Node> <operator1> <value>
  - Single condition
  - Ex. Ena = 1
- Time = <value>
  - Single condition
  - Time = 500ns
- <Condition> <operator2> <condition>
  - Complex tests
  - Ena = 1 && time > 500ns



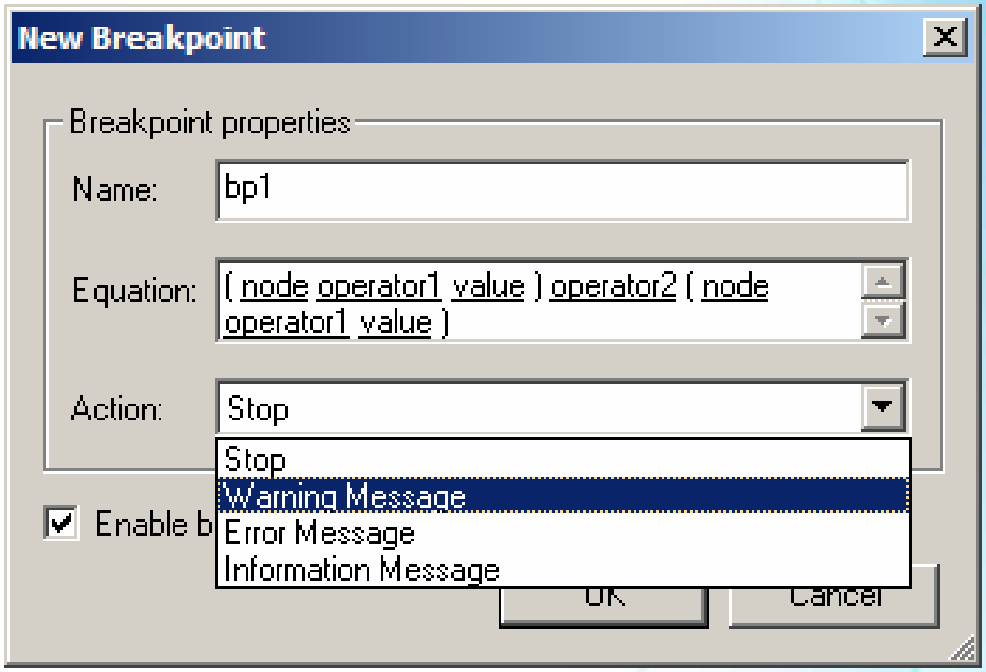
Breakpoint Equations (Cont.)



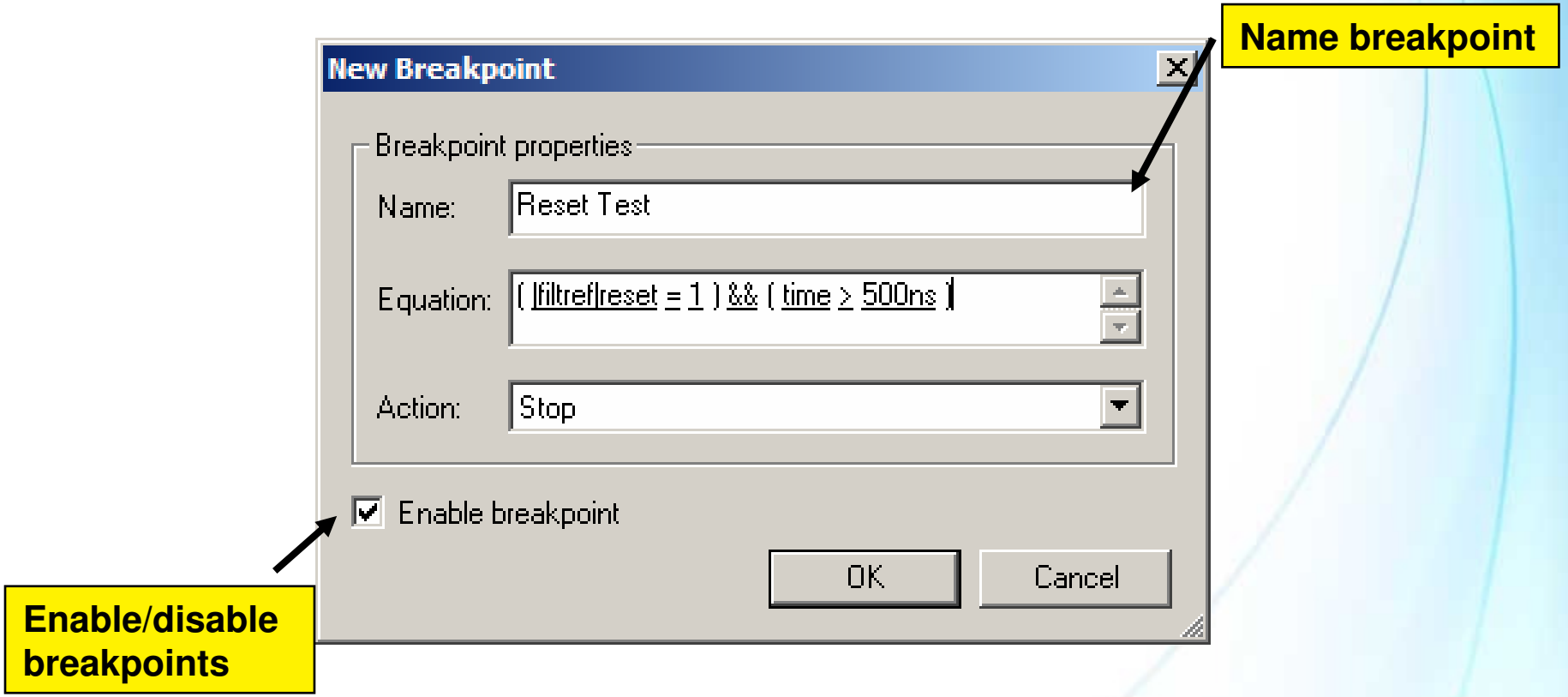
- Clicking on:
  - Node
    - Opens node finder
  - Operator1
    - Allows selection of <, >, =
  - Operator2
    - Allows selection of && (AND) and || (OR) operators

Breakpoint Actions

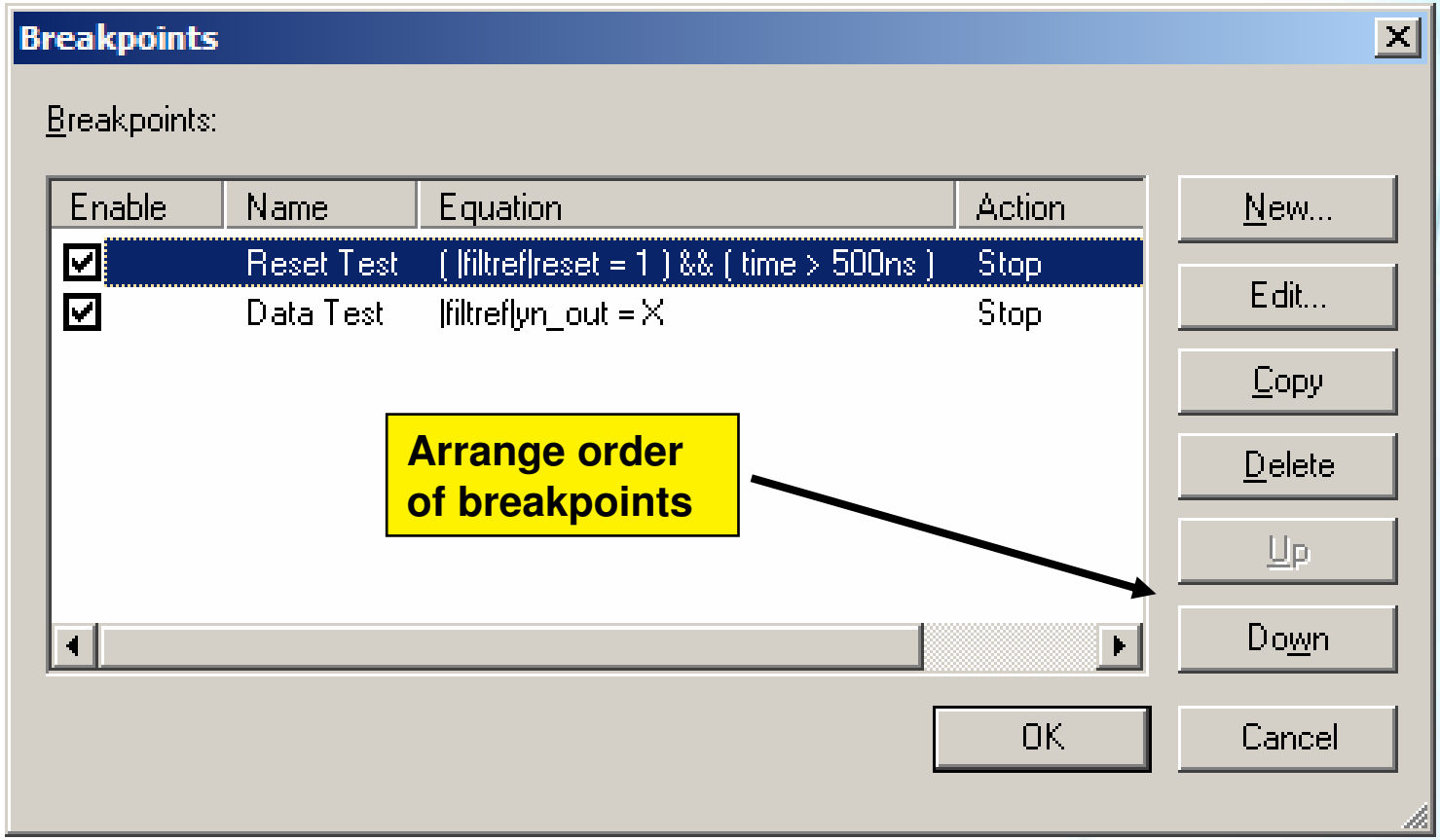
- Stop
- Give error message
- Give warning message
- Give informational message



Example Breakpoint



Organizing Breakpoints



Please go to optional exercise in the Exercise Manual